```
! ******************************************************************             .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@             .
! --------                                                  ---------             .
! -------- CHE2013            version 1.5                   ---------             .
! -------- octobre 2013 - décembre 2014                     ---------             .
! --------                                                  ---------             .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@             .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr     ---------             .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@             .
```

Table des matières

Première partie
données

nombre de programmes principaux : 18

nombre de modules : 34

nombre de subroutines : 148, *publiques* ou *privées*

nombre de lignes de commandes : 23 109

Deuxième partie
compiation

0.1 makefile

```
# ------------------------------------------------------ #
#       Makefile programme CHE (cold-hot-plot)
# ------------------------------------------------------ #
# Makefile produit à partir de makedepf90 (version 2.8.8 -- Erik.Edelmann@iki.fi)
# -> makedepf90 *.f90 '-r $(COMPIL) $(FFLAGS) -c $<' -o aout > makefile
# ------------------------------------------------------ #

# --------------------- Compilos ----------------------- # test gfortran ou ifort

ifeq ($(shell if ifort -v 2> /dev/null ; then echo "OKifort" ; fi),OKifort)
    include macros/ifort.d
else
    ifeq ($(shell if gfortran -v 2> /dev/null ; then echo "OKgfortran" ; fi),OKgfortran)
        include macros/gfortran.d
    else

stop:
                @echo "pas de compilateur fortran !"
                exit 0
    endif
endif

# ----------------- Dependent o-files ------------------- #

FOBJ1=lib/modparam.o lib/mt19937ar.o lib/types.o lib/avancement.o lib/time.o lib/dellipsgc.o lib/subgeiger.o lib/McMC.o lib/intersect.o lib/tirage.o lib/
    tracer_rais.o lib/pbdirect.o lib/misfit.o lib/lectdata.o lib/printmess.o lib/stat.o lib/subparam.o lib/tri.o lib/rechercheinit.o

FOBJ2=lib/mkparamiter.o lib/mkmatricecorrel.o lib/mkfcout.o lib/mkhodo.o lib/mkcoda.o lib/mklatex.o lib/mkmap.o lib/mkres.o lib/mkwada.o lib/mkchatelain.o lib/
    mkmoho_inc.o lib/mkGMT.o lib/mkcarriere.o lib/mkposteriori.o

FOBJ3=lib/mod_sac_io.o lib/modparam.o lib/types.o lib/time.o lib/stat.o

FOBJ4=lib/modparam.o lib/mt19937ar.o lib/types.o lib/avancement.o lib/time.o lib/dellipsgc.o lib/subgeiger.o lib/tracer_rais.o lib/pbdirect.o lib/misfit.o lib/
    lectdata.o lib/stat.o lib/subparam.o lib/tri.o lib/rechercheinit.o

FOBJ5=lib/mod_sac_io.o lib/modparam.o lib/types.o lib/time.o lib/stat.o lib/lectdata.o lib/pbdirect.o lib/dellipsgc.o lib/mt19937ar.o lib/tracer_rais.o lib/
    misfit.o lib/tri.o


# ---------------------- makes ------------------------- #

all: rmv1 mess1 coldruns hotruns plot apriori others otherssac rmv2 mess2

coldruns: che_coldruns_init che_coldruns che_coldruns_syn
hotruns: che_hotruns_init che_hotruns che_hotruns_syn
plot: che_plot
apriori: rmv1 che_apriori
others: rmv1 sac_bin2txt sac_coda sac_readpick sac_spectre sac_stalta_kurtosis verifmediatrice sac_ZNE_2_LQT
otherssac: sac_writepick sac_writepickTheo sac_writepickCata

# ------------------- main programs ------------------- #

# ------------------- coldruns

che_coldruns_init: che_coldruns_init.o
        $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/che_coldruns_init.o $(FOBJ1) lib/mkposteriori.o
```

```
53          mv che_coldruns_init ../BIN/che_coldruns_init.exe
54
55  che_coldruns: che_coldruns.o
56          $(COMPILMPI) -o $@ $(OPTIONC) $(FFLAGS) lib/che_coldruns.o $(FOBJ1)
57          mv che_coldruns ../BIN/che_coldruns.exe
58
59  che_coldruns_syn: che_coldruns_syn.o
60          $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/che_coldruns_syn.o $(FOBJ1)
61          mv che_coldruns_syn ../BIN/che_coldruns_syn.exe
62
63  # ------------------ hotruns
64
65  che_hotruns_init: che_hotruns_init.o
66          $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/che_hotruns_init.o $(FOBJ1)
67          mv che_hotruns_init ../BIN/che_hotruns_init.exe
68
69  che_hotruns: che_hotruns.o
70          $(COMPILMPI) -o $@ $(OPTIONC) $(FFLAGS) lib/che_hotruns.o $(FOBJ1)
71          mv che_hotruns ../BIN/che_hotruns.exe
72
73  che_hotruns_syn: che_hotruns_syn.o
74          $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/che_hotruns_syn.o $(FOBJ1)
75          mv che_hotruns_syn ../BIN/che_hotruns_syn.exe
76
77  # ------------------ plot
78
79  che_plot: che_plot.o
80          $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/che_plot.o $(FOBJ1) $(FOBJ2)
81          mv che_plot ../BIN/che_plot.exe
82
83  # ------------------ apriori
84
85  che_apriori: che_apriori.o
86          $(COMPILMPI) -o $@ $(OPTIONC) $(FFLAGS) lib/che_apriori.o $(FOBJ1)
87          mv che_apriori ../BIN/che_apriori.exe
88
89  # ------------------ others
90
91  sac_bin2txt: sac_bin2txt.o
92          $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_bin2txt.o $(FOBJ3)
93          mv sac_bin2txt ../BIN/sac_bin2txt.exe
94
95  sac_coda: sac_coda.o
96          $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_coda.o $(FOBJ3)
97          mv sac_coda ../BIN/sac_coda.exe
98
99  sac_readpick: sac_readpick.o
100         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_readpick.o $(FOBJ3)
101         mv sac_readpick ../BIN/sac_readpick.exe
102
103 sac_spectre: sac_spectre.o
104         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_spectre.o $(FOBJ3)
105         mv sac_spectre ../BIN/sac_spectre.exe
106
107 sac_ZNE_2_LQT: sac_ZNE_2_LQT.o
108         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_ZNE_2_LQT.o $(FOBJ3) lib/dellipsgc.o
109         mv sac_ZNE_2_LQT ../BIN/sac_ZNE_2_LQT.exe
110
111 sac_stalta_kurtosis: sac_stalta_kurtosis.o
112         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_stalta_kurtosis.o $(FOBJ3)
113         mv sac_stalta_kurtosis ../BIN/sac_stalta_kurtosis.exe
114
115 verifmediatrice : verifmediatrice.o
116         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/verifmediatrice.o $(FOBJ4)
117         mv verifmediatrice ../BIN/verifmediatrice.exe
```

```
118
119 # ------------------ otherssac
120
121 sac_writepick: sac_writepick.o
122         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_writepick.o $(FOBJ3)
123         mv sac_writepick ../BIN/sac_writepick.exe
124
125 sac_writepickTheo: sac_writepickTheo.o
126         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_writepickTheo.o $(FOBJ3)
127         mv sac_writepickTheo ../BIN/sac_writepickTheo.exe
128
129 sac_writepickCata: sac_writepickCata.o
130         $(COMPIL) -o $@ $(OPTIONC) $(FFLAGS) lib/sac_writepickCata.o $(FOBJ5)
131         mv sac_writepickCata ../BIN/sac_writepickCata.exe
132
133 # ------------------------------------------------------ #
134
135 clean : rmv1
136
137 clear : rmv1
138
139 # ---------------------- rules ---------------------------- #
140
141 rmv2 :
142         clear ; rm -rf lib/*.mod lib/*.o
143
144 rmv1 :
145         clear ; rm -rf lib/*.mod lib/*.o ../BIN/*.exe
146
147 mess1 :
148         cat < MES/message1.d
149
150 mess2 :
151         cat < MES/message2.d
152
153 # ------------ rules dependent o-files ------------------- #
154
155 # ---------------------- programmes ----------------------- #
156
157 che_coldruns.o : PROG/coldruns/che_coldruns.f90 misfit.o subparam.o printmess.o time.o pbdirect.o tirage.o McMC.o avancement.o mt19937ar.o types.o modparam.o
158         $(COMPILMPI) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
159 che_coldruns_init.o : PROG/coldruns/che_coldruns_init.f90 rechercheinit.o subparam.o printmess.o lectdata.o mt19937ar.o types.o modparam.o McMC.o intersect.o
    tirage.o mkposteriori.o
160         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
161 che_coldruns_syn.o : PROG/coldruns/che_coldruns_syn.f90 tri.o subparam.o lectdata.o printmess.o types.o modparam.o
162         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
163 che_hotruns.o : PROG/hotruns/che_hotruns.f90 dellipsgc.o misfit.o subparam.o printmess.o pbdirect.o lectdata.o tirage.o McMC.o avancement.o mt19937ar.o time.o
    types.o modparam.o
164         $(COMPILMPI) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
165 che_hotruns_init.o : PROG/hotruns/che_hotruns_init.f90 subparam.o printmess.o time.o tirage.o mt19937ar.o types.o modparam.o McMC.o misfit.o lectdata.o
166         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
167 che_hotruns_syn.o : PROG/hotruns/che_hotruns_syn.f90 subparam.o printmess.o types.o modparam.o
168         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
169 che_plot.o : PROG/che_plot.f90 mt19937ar.o mkGMT.o subparam.o printmess.o lectdata.o mklatex.o types.o modparam.o McMC.o intersect.o tirage.o mkposteriori.o time
    .o
170         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
171 che_apriori.o : PROG/che_apriori.f90 printmess.o mt19937ar.o subparam.o types.o modparam.o McMC.o intersect.o tirage.o misfit.o lectdata.o
172         $(COMPILMPI) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
173 sac_bin2txt.o : PROG/sac_bin2txt.f90 time.o types.o modparam.o mod_sac_io.o stat.o
174         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
175 sac_coda.o : PROG/sac_coda.f90 mod_sac_io.o stat.o time.o types.o modparam.o
176         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
177 sac_readpick.o : PROG/sac_readpick.f90 mod_sac_io.o time.o types.o modparam.o
178         $(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
179 sac_stalta_kurtosis.o : PROG/sac_stalta_kurtosis.f90 mod_sac_io.o time.o types.o modparam.o
```

```
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
verifmediatrice.o : PROG/verifmediatrice.f90 rechercheinit.o lectdata.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
sac_writepick.o : PROG/sac_writepick.f90 mod_sac_io.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
sac_spectre.o : PROG/sac_spectre.f90 mod_sac_io.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
sac_writepickTheo.o : PROG/sac_writepickTheo.f90 mod_sac_io.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
sac_writepickCata.o : PROG/sac_writepickCata.f90 mod_sac_io.o time.o types.o modparam.o lectdata.o pbdirect.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib
sac_ZNE_2_LQT.o : PROG/sac_ZNE_2_LQT.f90 mod_sac_io.o modparam.o dellipsgc.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o lib

# ----------------    for modules    ---------------------- #

McMC.o : MOD/McMC.f90 mt19937ar.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
avancement.o : MOD/avancement.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
modparam.o : MOD/modparam.f90
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
dellipsgc.o : MOD/dellipsgc.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
intersect.o : MOD/intersect.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
lectdata.o : MOD/lectdata.f90 dellipsgc.o misfit.o subparam.o pbdirect.o mt19937ar.o tri.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
misfit.o : MOD/misfit.f90 pbdirect.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkGMT.o : MOD/MOD_GMT/mkGMT.f90 tri.o lectdata.o mkmoho_inc.o mkcoda.o mkhodo.o mkmap.o mkres.o mkfcout.o mkmatricecorrel.o mkparamiter.o mkchatelain.o mkwada.o
     misfit.o dellipsgc.o subparam.o pbdirect.o time.o types.o modparam.o mkcarriere.o mkposteriori.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkchatelain.o : MOD/MOD_GMT/mkchatelain.f90 pbdirect.o time.o avancement.o types.o modparam.o tri.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkcoda.o : MOD/MOD_GMT/mkcoda.f90 pbdirect.o stat.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkfcout.o : MOD/MOD_GMT/mkfcout.f90 mkparamiter.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkhodo.o : MOD/MOD_GMT/mkhodo.f90 pbdirect.o stat.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mklatex.o : MOD/MOD_LaTeX/mklatex.f90 pbdirect.o misfit.o subgeiger.o mkGMT.o subparam.o lectdata.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkmap.o : MOD/MOD_GMT/mkmap.f90 types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkmatricecorrel.o : MOD/MOD_GMT/mkmatricecorrel.f90 stat.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkmoho_inc.o : MOD/MOD_GMT/mkmoho_inc.f90 pbdirect.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkparamiter.o : MOD/MOD_GMT/mkparamiter.f90 types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkres.o : MOD/MOD_GMT/mkres.f90 stat.o subparam.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkposteriori.o : MOD/MOD_GMT/mkposteriori.f90 types.o modparam.o dellipsgc.o tri.o avancement.o tirage.o pbdirect.o misfit.o rechercheinit.o subparam.o time.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkcarriere.o : MOD/MOD_GMT/mkcarriere.f90 modparam.o types.o dellipsgc.o time.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mkwada.o : MOD/MOD_GMT/mkwada.f90 pbdirect.o time.o avancement.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mod_sac_io.o : MOD/MOD_sac/mod_sac_io.f90
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
mt19937ar.o : MOD/MOD_rand/mt19937ar.f90 modparam.o
	$(COMPIL) -c -Ilib $(FFLAGS) $(OPTIOND) $< ; mv *.o *.mod lib
pbdirect.o : MOD/pbdirect.f90 stat.o time.o dellipsgc.o types.o modparam.o tracer_rais.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
```

```
printmess.o : MOD/printmess.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
rechercheinit.o : MOD/rechercheinit.f90 dellipsgc.o mt19937ar.o time.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
stat.o : MOD/stat.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
tracer_rais.o : MOD/tracer_rais.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
subgeiger.o : MOD/MOD_Geiger/subgeiger.f90 dellipsgc.o pbdirect.o time.o mt19937ar.o types.o modparam.o misfit.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
subparam.o : MOD/subparam.f90 rechercheinit.o subgeiger.o mt19937ar.o tri.o avancement.o stat.o time.o types.o modparam.o tri.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
time.o : MOD/time.f90 types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
tirage.o : MOD/tirage.f90 time.o intersect.o mt19937ar.o types.o modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
tri.o : MOD/tri.f90 time.o types.o modparam.o mt19937ar.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib
types.o : MOD/types.f90 modparam.o
	$(COMPIL) -c -Ilib $(OPTIONC) $(FFLAGS) $< ; mv *.o *.mod lib

# ------------------------------------------------------ #
```

0.2 SRC/macros/ifort.d

```
# ------------------------------------------------------ #
#       architecture Makefile programme CHE (cold-hot-plot)
# ------------------------------------------------------ #


# mpimodfile=-I/U.../MPICH2/mpich-3.1.2/mpich-install/include/ ###  sudo find . -name mpi.mod

COMPIL = ifort
COMPILMPI = mpif90 -f90=ifort
OPTIONC = -O3 -fpp -diag-disable 8291 -diag-disable 8290 -assume byterec

# FFLAGS = -check bounds

# ------------------------------------------------------ #
# fpp -> permet de définir _FILE_DIR_ en fonction du compilateur
# ------------------------------------------------------ #
# -diag-disable 8291 -diag-disable 8290  -> format d'écriture dans des fichiers des variables output
# remark #8291: Recommended relationship between field width 'W' and the number of fractional digits 'D' in this edit descriptor is 'W>=D+7'
# remark #8290: Recommended relationship between field width 'W' and the number of fractional digits 'D' in this edit descriptor is 'W>=D+3'
# ------------------------------------------------------ #
# assume byterec -> force to use byte units in access='direct' file
# ------------------------------------------------------ #
```

0.3 SRC/macros/gfortran.d

```
# ------------------------------------------------------ #
#       architecture Makefile programme CHE (cold-hot-plot)
# ------------------------------------------------------ #

COMPIL = gfortran
COMPILMPI = mpif90
OPTIONC = -O3 -cpp
OPTIOND = -fno-range-check # pb in mt19937ar avec gfortran

FFLAGS = -fbounds-check

# -------------------- Options ------------------------- #

# test :
```

```
15
16 #FFLAGS = -fdefault-real-8 -O0 -g -fbounds-check -Wall -ffpe-trap=invalid,zero,overflow,underflow -fbacktrace -ftrapv -fimplicit-none -fno-automatic -ffree-form
       -Wconversion -Wunderflow -Wunreachable-code -g3 -fstack-protector-all
17
18 # ------------------------------------------------------ #
19 # cpp -> permet de définir _FILE_DIR_ en fonction du compilateur
20 # ------------------------------------------------------ #
```

Troisième partie

script bash

0.4 run_it.sh

```
#######################################################################
# Méric Haugmard meric.haugmard@univ-nantes.fr
# !/bin/bash
#######################################################################
# --------                                              --------    .
# --------           CHE2013_coldruns version 2.0       --------    .
# --------           octobre 2013 - décembre 2014       --------    .
# --------                                              --------    .
#######################################################################
# version 1.1 : inversion d'un séisme (jan 2014)                      !
# version 1.2 : inversion de plusieurs séismes (juillet 2014)         !
# version 1.3 : parallelisation OpenMP (septembre 2014)               !
# version 1.4 : parallelisation MPI (octobre 2014)                    !
# version 1.5 : initialisation du prior auto (novembre 2014)          !
# version 1.6 : parallelisation full MPI (décembre 2014)              !
# version 1.7 : ajout d'une notice (décembre 2014)                    !
# version 1.8 : compilation avec ifort et gfortran (janvier 2015)     !
# version 1.9 : moho non tabulaire (fevrier 2015)                     !
# version 2.0 : test 50 séismes (septembre 2015)                      !
# version 2.1 : gestion des carrières, d'après Pascal Guterman (octobre 2015)!
# version 2.2 : ajout de modèle de terre différents pour le problème directe (novembre 2015)!
# version 2.3 : calcules a posteriori (janvier 2016)                  !
#######################################################################
# compile et execute le programme CHE
# permet aussi l'écriture du scripte LOG

# The default process manager is called MPD, which is a ring of daemons on the machines where you will run your MPI programs.
# mpd &

T="$(date +%s)"

chmod +x SRC/run.sh

./SRC/run.sh 2> >(tee stderrlog.d | tee -a alllog.d > /dev/tty ) | tee stdoutlog.d | tee -a alllog.d

mv *log.d OUTPUT/LOG
T="$(($(date +%s)-T))"
echo "execution time (secs) ${T}"

cat OUTPUT/LOG/stderrlog.d

rm -rf .gmtcommands4 .gmtdefaults4 toto.d

#######################################################################
```

0.5 SRC/run.sh

```
#######################################################################
# Méric Haugmard meric.haugmard@univ-nantes.fr
# !/bin/bash
#######################################################################
# --------                                              --------    .
# --------           CHE2013_coldruns version 2.0       --------    .
# --------           octobre 2013 - décembre 2014       --------    .
# --------                                              --------    .
#######################################################################
# mpiexec <- pour executer mpi
```

```
# mpd &
#######################################################################

clear
#######################################################################
cd SRC
#make all
cd ..

#######################################################################
########## # permet la sauvegarde du dernier repertoire OUTPUT
########## # par défaut : no
########## ans="No"
########## # attend 3 secondes une réponse positive
########## read -p "Sauver dernier run [y/n]?" -t 3 ans
########## if [ $ans = y -o $ans = Y -o $ans = yes -o $ans = Yes -o $ans = YES ]
########## then
##########     # on creer un repertoire et sauve dernier run dans OLD
##########     echo yes !
##########     LA_DATE=$(date +%Y"_"%m"_"%d"_"%H"h"%M"_"%S)
##########     mkdir OLD/$LA_DATE ; mv OUTPUT OLD/$LA_DATE/OUTPUT
########## else
##########     # on supprime le repertoire OUTPUT sans sauver le dernier run
##########     echo no !
rm -rf OUTPUT
########## fi
#######################################################################
# (re)création de l'arborescence
mkdir OUTPUT
mkdir OUTPUT/figures
mkdir OUTPUT/files
mkdir OUTPUT/files/Cold
mkdir OUTPUT/files/Hot
mkdir OUTPUT/files/STA
mkdir OUTPUT/files/Plot
mkdir OUTPUT/LOG
mkdir OUTPUT/input
mkdir OUTPUT/GMT
mkdir OUTPUT/LATEX
#######################################################################
cd DATA
ls -f *.dat > seismes.d 2>/dev/null
cd ..

#######################################################################
head -1 PARAM/iteration.d > toto.d
read nbchainecold itercold < toto.d
tail -1 PARAM/iteration.d > toto.d
read nbchainehot iterhot < toto.d
#######################################################################
# programmes principaux
# FORT_FMT_RECL=1000 -> permet d'écrire des fichiers textes de plus de 1000 caracteres par lignes (pour ifort)
#######################################################################
# coldruns
FORT_FMT_RECL=1000 ./BIN/che_coldruns_init.exe || exit  # exécute {exit} uniquement si {/BIN/che_coldruns_init.exe} échoue
    ###################################################################
read nbseisme < OUTPUT/GMT/nbseisme.d
echo '/dev/null' > cmd.exe
    ###################################################################
FORT_FMT_RECL=1000 mpiexec -n $nbchainecold ./BIN/che_coldruns.exe < cmd.exe || exit
FORT_FMT_RECL=1000 ./BIN/che_coldruns_syn.exe || exit
    ###################################################################
# hotruns
FORT_FMT_RECL=1000 ./BIN/che_hotruns_init.exe || exit
FORT_FMT_RECL=1000 mpiexec -n $nbchainehot ./BIN/che_hotruns.exe < cmd.exe || exit
```

```
FORT_FMT_RECL=1000 ./BIN/che_hotruns_syn.exe || exit
    ######################################################################
# plots
FORT_FMT_RECL=1000 ./BIN/che_plot.exe || exit
FORT_FMT_RECL=1000 mpiexec -n $nbseisme ./BIN/che_apriori.exe < cmd.exe || exit
######################################################################
rm -rf .gmtcommands4 .gmtdefaults4 cmd.exe
# supprime les anciennes options par défaut
chmod +x OUTPUT/GMT/script0.sh
# figure recherche_initiale
./OUTPUT/GMT/script0.sh || exit
######################################################################
# diverses copies
cp PARAM/priorIn_HOT.d OUTPUT/input/priorIn_HOT.d
cp PARAM/priorIn_COLD.d OUTPUT/input/priorIn_COLD.d
cp PARAM/paramHypo.d OUTPUT/input/paramHypo.d 2>/dev/null
cp PARAM/paramTerre.d OUTPUT/input/paramTerre.d 2>/dev/null
cp PARAM/iteration.d OUTPUT/input/iteration.d
cp DATA/*.d OUTPUT/input/
######################################################################
# execution des scripts GMT
chmod +x OUTPUT/GMT/script.sh
./OUTPUT/GMT/script.sh || exit
######################################################################
# execution des scripts LaTeX
rm -rf OUTPUT/LOG/gslog.d
cd OUTPUT/LATEX/
grep '*' 2*.tex
ls 2*.tex sta*.tex 2>/dev/null | while read afile
do
  echo $afile
  pdflatex $afile >> ../LOG/afilelog.d
  pdflatex $afile >> ../LOG/afilelog.d
  afilepdf=${afile/tex/pdf}
  gs -dNOPAUSE -dBATCH -sDEVICE=pdfwrite -dCompatibilityLevel=1.4 -dPDFSETTINGS=/prepress -sOutputFile=../$afilepdf $afilepdf >> ../LOG/gslog.d
done
cd ../..
######################################################################

if [ -d DOC/SCRIPT ]
then
    ######################################################################
    rm -rf ./DOC/SCRIPT/*.aux ./DOC/SCRIPT/*.pdf ./DOC/SCRIPT/*.log ./DOC/SCRIPT/*.gz
    rm -rf ./DOC/SCRIPT/prog.txt ./DOC/SCRIPT/pbashacc.txt
    ######################################################################
    # execution d'autres scripts
    chmod +x DOC/SCRIPT/makesumfiles.sh
    ./DOC/SCRIPT/makesumfiles.sh
    echo EDITscripts.tex
    cd DOC/SCRIPT/
    pdflatex EDITscripts.tex > afilelog.d || exit
    pdflatex EDITscripts.tex > afilelog.d || exit
    cd ../..
    ######################################################################
fi
######################################################################
```

Quatrième partie

scripts Fortran

1 programmes principaux

1.1 SRC/PROG/che_coldruns_init.f90

```
! programme principal I
! ************************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                        ---------    .
! -------- CHE2013_coldruns version 2.2                           ---------    .
! -------- octobre 2013 - décembre 2014                           ---------    .
! --------                                                        ---------    .
! -------- Prog. basé uniquement sur des méthodes non linéaires   ---------    .
! --------                                                        ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr           ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope          !
!  that it will be useful however without any warranties.                     !
!  Use it on your own risk. Please, report bugs/improvements/... .           !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! avec la participation de Ianis Gaudot, Éric Beucler et Philippe Cance
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                             !
!         " Une réponse approximative à la bonne question, qui est            !
!            souvent mal posée, est bien meilleure que la réponse             !
!           exacte à une mauvaise question que l'on peut toujours             !
!                       formuler de façon précise ... "                       !
!                                                                             !
!       Tukey, J.W. (1962) : The Future of Data Analysis.                     !
!       The Annals of Mathematical Statistics, Vol. 33, No. 1, p. 1-67.       !
!                                                                             !
!                                                                             !
! ----------------------------------------------------------------------      .
! version 1.1 : inversion d'un séisme (jan 2014)                              !
! version 1.2 : inversion de plusieurs séismes (juillet 2014)                 !
! version 1.3 : parallelisation OpenMP (septembre 2014)                       !
! version 1.4 : parallelisation MPI (octobre 2014)                            !
! version 1.5 : initialisation du prior auto (novembre 2014)                  !
! version 1.6 : parallelisation full MPI (décembre 2014)                      !
! version 1.7 : ajout d'une notice (décembre 2014)                            !
! version 1.8 : compilation avec ifort et gfortran (janvier 2015)             !
! version 1.9 : moho non tabulaire (fevrier 2015)                             !
! version 2.0 : test 50 séismes (septembre 2015)                              !
! version 2.1 : gestion des carrières, d'après Pascal Guterman (octobre 2015)!
! version 2.2 : ajout de modèle de terre différents pour le problème directe (novembre 2015)!
! version 2.3 : calculs a posteriori (janvier 2016)                           !
! ----------------------------------------------------------------------      .
! Initialisation des Coldruns                                                 !
!                                                                             !
program che2013_coldruns_init
  ! ----------------------------------------------------------------      .
  ! --------  modules :                                     ---------     .
  use modparam
  use typetemps
  use mt19937
  use datalecture
  use affiche
  use sub_param
```

```
55  use rechercheepi
56  use time
57  use figure_posteriori
58  ! ------------------------------------------------------------------------     .
59  ! -------- déclaration :                                          --------     .
60  implicit none
61  ! --------                                                       --------     .
62  type(dataall) :: D(nbseismes)                                                 ! données de temps
63  type(parametres), dimension(:), allocatable :: param_init                     ! paramètres d'inv.
64  type(fcout) :: misfit                                                         ! fonction coût
65  type(accept) :: acceptance                                                    ! acceptance
66  type(parametresinv) :: p                                                      ! paramètres d'inv.
67  type(priorEPI) :: pEpis(nbseismes)                                            ! prior
68  type(amoho_centroid) :: acentroid                                             ! si moho non tabulaire
69  type(date_secPS) :: midi20
70  ! --------                                                       --------     .
71  real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)                             ! cercles pond.
72  real(KIND=wr) :: deltaP,deltaS
73  ! --------                                                       --------     .
74  integer(KIND=wi) :: i,j,k
75  integer(KIND=wi) :: mb                                                        ! prior
76  integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                              ! nombre chaînes
77  integer(KIND=wi) :: maxiterhot,maxitercold                                    ! nombre d'itérations
78  integer(KIND=wi) :: nbsta, nbtps(nbseismes)                                   ! nombre station et nombre de données de temps par séismes
79  integer(KIND=wi) :: nbmod
80  ! --------                                                       --------     .
81  character (LEN=5) :: numchaine
82  character (LEN=20) :: nomfichier
83
84  ! ------------------------------------------------------------------------     .
85
86  ! ------------------------------------------------------------------------     .
87  ! -------- initialisation                                        --------     .
88  call print_mess_1
89  call initseed(libre)                                                          ! aléatoire calé sur temps CPU
90  call printnbseismes
91  ! -------- si moho non tabulaire                                 --------     .
92  acentroid%lonC=moho_lon ; acentroid%latC=moho_lat
93  acentroid%NS=moho_NS ; acentroid%EO=moho_EO
94  call alph2vect(acentroid); call vect2alph(acentroid)
95  ! -------- lecture des données                                   --------     . phases and stations list
96  call lectnbdata(nbsta,nbtps)                                                  ! nombre de données
97  do i=1,nbseismes
98    allocate(D(i)%datatps(nbtps(i)))                                            ! alloue par séisme
99  enddo
100 call lectdata(nbsta,nbtps,D)                                                  ! temps d'arrivés par séisme
101 ! -------- cas synthétiques (si besoin)                          --------     .
102 call mksynth(nbtps,D,acentroid)
103 ! -------- réduction du prior pour les paramètres épicentraux    --------     .
104 print*,'initialisation du prior pour les paramètres épicentraux'
105 call zoneRecherche(nbtps,D,pEpis,mb)
106 ! ------------------------------------------------------------------------     .
107 ! --------  nb de chaines de Markov et d'iterations par chaine   --------     .
108 call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold)
109 allocate(param_init(nbChaineMVcold))
110 ! ------------------------------------------------------------------------     .
111
112 do i=1,nbChaineMVcold
113   ! ------------------------------------------------------------------------     .
114   ! -------- initialisation du modèle de terre puis des paramètres hypocentraux
115   call initparam(nbtps,D,param_init(i),pEpis,mb)                               ! par la méthode des hémisphères -> méthode non linéaire
116   ! -------- lecture du prior                                    --------     .
117   call lect_prior(p,param_init(i),"C")                                         ! C -> coldruns
118   ! -------- initialisation des bornes épicentales               --------     .
119   p%centreY(:)=param_init(i)%lat(:)
```

```
    p%centreX(:)=param_init(i)%lon(:)
    p%Rayon=500.0_wr                                                    ! recherche de l'épicentre dans ce rayon, pas au delas
    ! ------------------------------------------------------------------ .
  enddo
  ! -------- recherche les distances de pondération          ---------- .
  call cerclespond(nbtps,D,param_init(1),xmin,xmax,acentroid)
  ! -------- initialisation des autres paramètres           ---------- .
  call init_div(misfit,acceptance)
  ! ------------------------------------------------------------------ . si certains parametres fixes
  call paramfixe(p)
  ! ------------------------------------------------------------------ .


  ! ------------------------------------------------------------------ .
  ! Carriere
  ! ------------------------------------------------------------------ .
  !do i=1,nbseismes
  !  midi20=D(i)%datatps(1)%tpsR
  !  midi20%date%hour=11                                                ! midi TU, pour Vannes en hiver
  !  midi20%date%min=20
  !  call basetime(midi20)
  !  call difftime(deltaP,deltaS,midi20,D(i)%datatps(1)%tpsR)
  !  if ((abs(deltaP).lt.1.00_wr).or.(i==45)) then                  ! une heure
  !    write(*,*)'seisme ',i,' : tire de carriere ?'
  !    write(*,*)'--- > Zhypo max = 6 km'
  !    p%maxi%Zhypo(i) = 6.d0
  !    p%ecartype%Zhypo(i) =  p%ecartype%Zhypo(i)/4.0_wr
  !  endif
  !  ! ------------------------------------------------------------------ .
  !enddo
  ! ------------------------------------------------------------------ .


  ! ------------------------------------------------------------------ .
  ! étude des gradients sur la fonction coût
  ! ------------------------------------------------------------------ .
  if (plotposteriori) then
    nbmod=1
    nomfichier='POST_COLD_i'
    !call PosterioriExploration(p,nbmod,pEpis,nbtps,D,acentroid,xmin,xmax,mb,nomfichier)
  endif
  ! ------------------------------------------------------------------ .


  ! production de fichier
  ! parce que dans la norme MPI-Fortran 2008,
  ! faire un call MPI_BCAST avec des types dérivés, c'est pas prévu !
  ! ---------------------  un fichier par cold runs  ------------------    .
  do k=1,nbChaineMVcold
    write(numchaine(1:5),'(i5)')k
    open(unit=100,file="OUTPUT/files/Cold/In_"//trim(adjustl(numchaine))//".bin", &
         status="replace",form="unformatted",access="sequential")
      write(100)nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold
      write(100)nbsta,nbtps
      write(100)p
      do i=1,nbseismes
        do j=1,nbtps(i)
          write(100)D(i)%datatps(j)
        enddo
      enddo
      write(100)misfit
      write(100)acceptance
      write(100)param_init(k)
      write(100)xmin,xmax
      do i=1,nbseismes
```

```
          write(100)pEpis(i)%nb
          do j=1,pEpis(i)%nb
            write(100)pEpis(i)%pEpi(j)
          enddo
        enddo
        write(100)mb
        write(100)acentroid
      close(100)
    enddo
    ! ---------------------  fin du programme -----------------------------     .
    deallocate(param_init)
    do i=1,nbseismes
      deallocate(D(i)%datatps,pEpis(i)%pEpi)
    enddo
    ! ----------------------------------------------------------------------     .

end program che2013_coldruns_init



! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
```

1.2 SRC/PROG/che_coldruns.f90

```
! programme principal I bis
! **********************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
! -------                                                       --------     .
! ------- CHE2013_coldruns version 1.5                          --------     .
! ------- octobre 2013 - décembre 2014                          --------     .
! -------                                                       --------     .
! ------- Prog. basé uniquement sur des méthodes non linéaires  --------     .
! -------                                                       --------     .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
! ------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
!  This program is distributed for research purposes and in the hope         !
!  that it will be useful however without any warranties.                    !
!  Use it on your own risk. Please, report bugs/improvements/... .           !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
! avec la participation de Ianis Gaudot, Éric Beucler et Philippe Cance
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
!                                                                            !
!      Les voyages initiatiques de Ernesto CHE Gevara de 1951 à 1954 :       !
!       recherche à grande longeur d'onde d'un esprit révolutionnaire.       !
!                                                                            !
! ------------------------------------------------------------------------     .
! Période de rodage                                                          !
!                                                                            !
program che2013_coldruns
  ! ----------------------------------------------------------------------     .
  ! ------- modules :                                           --------     .
  use modparam
  use typetemps
  use mt19937
  use cpt_temps
  use algo_metropolis
  use tirage
  use pb_direct
  use affiche
  use sub_param
  use sub_misfit
  ! ----------------------------------------------------------------------     .
  ! ------- déclaration :                                       --------     .
```

```
implicit none

  include 'mpif.h'

! -------                                                    --------     .
type(dataall) :: D(nbseismes)                                             ! données de temps
type(parametres) :: param_init, param_best                                ! paramètres d'inv.
type(fcout) :: misfit                                                     ! fonction coût
type(accept) :: acceptance                                                ! acceptance
type(parametresinv) :: p                                                  ! paramètres d'inv.
type(priorEPI) :: pEpis(nbseismes)                                        ! prior
type(amoho_centroid) :: acentroid                                         ! si moho non tabulaire
! -------                                                    --------     .
real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)                         ! cercles pond.
! -------                                                    --------     .
integer(KIND=wi) :: i,j,k,l,ok
integer(KIND=wi) :: mb                                                    ! prior
integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                          ! nombre chaînes
integer(KIND=wi) :: maxiterhot,maxitercold                                ! nombre d'itérations
integer(KIND=wi) :: nbsta, nbtps(nbseismes)                               ! nombre station et nombre de données de temps par séismes
! -------                                                    --------     .
logical :: critique                                                       ! si moho trop bas, onde refacté observé mais non prédite (vrai)
logical :: savemod                                                        ! modèle sauvé (vrai)
logical :: accepte                                                        ! modèle accepté (vrai) ou rejeté (faux)
logical :: div                                                            ! chaîne divergente
! -------                                                    --------     .
character (LEN=30) :: chaine
character (LEN=5) :: numbchaine
! -------                                                    --------     . MPI :
integer :: nb_procs,rang,code,err
integer, dimension(:), allocatable :: allseed
integer :: seed
! -------                                                    --------     .
logical, parameter :: plotmisfit=.false.
! ------------------------------------------------------------------     .

! ------------------------------------------------------------------     . MPI_BEGIN
call MPI_INIT(code)
call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
call MPI_COMM_RANK(MPI_COMM_WORLD,rang,code)
! ------------------------------------------------------------------     . MPI_BEGIN
! ------- initialisation de la graine                        --------     .
if (rang==0) then
  call print_mess_2
  call initseed(libre)                                                    ! aléatoire calé sur temps CPU
  ! ------- clacul de nb_procs graines                       --------     .
  allocate(allseed(nb_procs))
  do i=1,nb_procs
    allseed(i)=abs(genrand_int31())+10000
    do while (allseed(i) > 10000000)
      seed = int(10000123.0_wr*genrand_real1())
      allseed(i) = allseed(i) - seed
    end do
  enddo
endif
! ------- partage des graines                                --------     .
call MPI_SCATTER(allseed,1,MPI_INTEGER,seed,1,MPI_INTEGER,0,MPI_COMM_WORLD,code)
call init_genrand (int(seed*(rang+1),wi))                                 ! initialisation des graines pour chaque processus
! write(*,*)'générateur de nombre aléatoire :       ',seed*(int(rang+1))
! ------------------------------------------------------------------     .

! ------------------------------------------------------------------     .
! ------- lecture paramètres et données                      --------     .
write(numbchaine(1:5),'(i5)')rang+1
ok=0
```

```
106  open(unit=100,file="OUTPUT/files/Cold/In_"//trim(adjustl(numbchaine))//".bin", &
107       status="old",form="unformatted",access="sequential",iostat = ok)
108  read(100)nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold
109  read(100)nbsta,nbtps
110  read(100)p
111  do i=1,nbseismes
112    allocate(D(i)%datatps(nbtps(i)))
113    do j=1,nbtps(i)
114      read(100)D(i)%datatps(j)
115    enddo
116  enddo
117  read(100)misfit
118  read(100)acceptance
119  read(100)param_init
120  read(100)xmin,xmax
121  do i=1,nbseismes
122    read(100)pEpis(i)%nb
123    allocate(pEpis(i)%pEpi(pEpis(i)%nb))
124    do j=1,pEpis(i)%nb
125      read(100)pEpis(i)%pEpi(j)
126    enddo
127  enddo
128  read(100)mb
129  read(100)acentroid
130  close(100)
131  ! -------                                                    --------     .
132  if (ok .ne. 0) then
133    write(*,*)"problème dans che2013_coldruns : le fichier ", &
134      "OUTPUT/files/Cold/In_"//trim(adjustl(numbchaine))//".bin n''existe pas "
135    call MPI_ABORT(MPI_COMM_WORLD,err,code)
136  endif
137
138  ! ------------------------------------------------------------------     .
139  ! ------------------------------------------------------------------     .
140  ! -------- début MCMc                                        --------     . COLDRUNS
141  ! ------------------------------------------------------------------     .
142  if(plotmisfit) open(unit=10,file="OUTPUT/files/Cold/mis_"//trim(adjustl(numbchaine))//".txt",status="replace")
143  ! ------------------------------------------------------------------     .
144  i=rang+1
145  ! ------------------------------------------------------------------     .
146  if (rang==0) call print_messchaine(i,nbChaineMVcold)
147
148    ! ------------------------------------------------------------------     .
149    ! -------- début d'une chaîne                                --------     .
150    ! ------------------------------------------------------------------     .
151    savemod=.true.
152    div=.false.
153    unechaine : do j=1,maxitercold
154      ! ------------------------------------------------------------------     . progression en %
155      if (i==1) then
156        if ((j.gt.1000).or.(j.ne.1).or.(j.ne.maxitercold)) then
157          if(mod(j,100)==0) write(chaine(1:30),'(a5,i12,a13)')" 1 - ",j," modèles   "
158        else
159          write(chaine(1:30),'(a5,i12,a13)')" 1 - ",j," modèles     "
160        endif
161        call progress(j,maxitercold,chaine)
162      endif
163      ! ------------------------------------------------------------------     .
164      k=0
165      critique=.true.
166      do while(critique)                                                      ! tant que distance hypocentral << distance critique pour la réfraction
167        k=k+1
168
169          l=int(genrand_real1()*real(nbseismes,wr)+1.0_wr)                    ! aléatoire de 1 à nbseismes, un seul séisme
170          call tirage_H(p,l,all=.true.)                                       ! tirage des tous les paramètres hypocentraux pour le 'l'ième séisme
```

```
171           call tirage_T(p,all=.true.,vpvs=.true.)                    ! tirage des tous les paramètres de terre simultanément
172
173         ! -------- problème direct pour le jeu de paramètre tiré et chaque donnée
174         call tempsTheoDirect(nbtps,p%valNew,D,critique,acentroid)
175         if(k==5) then
176           critique=.false.                                             ! pour sortir apres 5 essais
177           ! -------- une donnée réfractée observée ne peux pas etre inférieur à la distance hypocentrale minimale pour la refraction
178           write(*,*)'problème dans che_coldruns : distance épi + 5 km < distance hypocentrale critique pour la réfraction '
179         endif
180       enddo
181       ! -------- calcul de la fonction coût                  ---------    .
182       call compute_misfit(nbtps,D,misfit%new,xmin,xmax,'C',div)
183       ! ---------------------------------------------------------------    .
184       if (div) then                                                       ! fin de chaine si divergent
185         if (rang==0) write(chaine(1:30),'(a5,i12,a13)')" 1 - ",j," modèles   "
186         if (rang==0) call progress(maxitercold,maxitercold,chaine)
187         exit unechaine
188       endif
189       ! -------- Metropolis (acceptation et rejet des modèles)  ---------    .
190       call metropolis(p,param_best,misfit,acceptance,savemod,accepte)     !  tout le script est là ...
191       ! ---------------------------------------------------------------    .
192       if(plotmisfit) write(10,*)j,misfit%new
193       ! ---------------------------------------------------------------    .
194     enddo unechaine
195     ! ---------------------------------------------------------------    .
196     ! -------- fin d'une chaîne                            ---------    .
197     ! ---------------------------------------------------------------    .
198     call calc_accept(acceptance)
199
200   ! call MPI_Barrier(MPI_COMM_WORLD,err)
201   ! call print_mess_finchainemin(misfit%best,acceptance%val)
202
203   if(plotmisfit) close(10)
204   ! ---------------------------------------------------------------    .
205   ! ---------------------------------------------------------------    .
206   ! -------- fin MCMc                                    ---------    . COLDRUNS
207   ! ---------------------------------------------------------------    .
208   ! ---------------------------------------------------------------    .
209
210   ! ----------------  un fichier OUTPUT par cold runs  ----------------    .
211   open(unit=100,file="OUTPUT/files/Cold/Out_"//trim(adjustl(numbchaine))//".bin", &
212       status="replace",form="unformatted",access="sequential")
213     write(100)nbsta,nbtps
214     write(100)p
215     do i=1,nbseismes
216       do j=1,nbtps(i)
217         write(100)D(i)%datatps(j)
218       enddo
219     enddo
220     write(100)misfit
221     write(100)acceptance
222     write(100)param_init
223     write(100)param_best
224     write(100)xmin,xmax
225     do i=1,nbseismes
226       write(100)pEpis(i)%nb
227       do j=1,pEpis(i)%nb
228         write(100)pEpis(i)%pEpi(j)
229       enddo
230     enddo
231     write(100)mb
232     write(100)acentroid
233   close(100)
234
235   ! ----------------------  fin du programme ---------------------------    .
```

```fortran
  do i=1,nbseismes
    deallocate(D(i)%datatps,pEpis(i)%pEpi)
  enddo
  ! -----------------------------------------------------------------------    . MPI_END
  call MPI_FINALIZE(code)
  ! -----------------------------------------------------------------------    . MPI_END

end program che2013_coldruns



! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
```

1.3 SRC/PROG/che_coldruns_syn.f90

```fortran
! programme principal I ter
! ***********************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------                                                          --------    .
! ------- CHE2013_coldruns version 1.5                             --------    .
! ------- octobre 2013 - décembre 2014                             --------    .
! -------                                                          --------    .
! ------- Prog. basé uniquement sur des méthodes non linéaires     --------    .
! -------                                                          --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! ------- Méric Haugmard meric.haugmard@univ-nantes.fr             --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope           !
!  that it will be useful however without any warranties.                      !
!  Use it on your own risk. Please, report bugs/improvements/... .             !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! avec la participation de Ianis Gaudot, Éric Beucler et Philippe Cance
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                              !
! -----------------------------------------------------------------------    .
! Synthèse des Coldruns                                                        !
!                                                                              !
program che2013_coldruns_init
  ! -----------------------------------------------------------------------    .
  ! ------- modules :                                              --------    .
  use modparam
  use typetemps
  use affiche
  use datalecture
  use sub_param
  use tri
  ! -----------------------------------------------------------------------    .
  ! ------- déclaration :                                          --------    .
  implicit none
  ! -------                                                        --------    .
  type(dataall) :: D(nbseismes)                                                ! données de temps
  type(parametres), dimension(:), allocatable :: param_init, param_best        ! paramètres d'inv.
  type(fcout), dimension(:), allocatable :: misfit                             ! fonction coût
  type(accept), dimension(:), allocatable :: acceptance                        ! acceptance
  type(parametresinv) :: p                                                     ! paramètres d'inv.
  type(coldmoy) :: dc                                                          ! moyennes et écarts-types des modèles du coldrun
  type(priorEPI) :: pEpis(nbseismes)                                           ! prior
  type(amoho_centroid) :: acentroid                                            ! si moho non tabulaire
  ! -------                                                        --------    .
  real(KIND=wr), dimension(:), allocatable :: vec
  real(KIND=wr) :: moy,ec
  real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)                            ! cercles pond.
  ! -------                                                        --------    .
  integer(KIND=wi) :: i,j,k,ok
```

```
integer(KIND=wi) :: mb                                                    ! prior
integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                          ! nombre chaînes
integer(KIND=wi) :: maxiterhot,maxitercold                                ! nombre d'itérations
integer(KIND=wi) :: nbsta, nbtps(nbseismes)                               ! nombre station et nombre de données de temps par séismes]
! -------                                                  --------      .
character (LEN=5) :: chaine
! ------------------------------------------------------------------      .
call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold,chut=.true.)

allocate(param_best(nbChaineMVcold),param_init(nbChaineMVcold))
allocate(misfit(nbChaineMVcold),acceptance(nbChaineMVcold))

do k=1,nbChaineMVcold
  write(chaine(1:5),'(i5)')k
  open(unit=100,file="OUTPUT/files/Cold/Out_"//trim(adjustl(chaine))//".bin", &
       status="old",form="unformatted",access="sequential",iostat = ok)
  read(100)nbsta,nbtps
  read(100)p
  do i=1,nbseismes
    if (.not.(allocated(D(i)%datatps))) allocate(D(i)%datatps(nbtps(i)))
    do j=1,nbtps(i)
      read(100)D(i)%datatps(j)
    enddo
  enddo
  read(100)misfit(k)
  read(100)acceptance(k)
  read(100)param_init(k)
  read(100)param_best(k)
  read(100)xmin,xmax
  do i=1,nbseismes
    read(100)pEpis(i)%nb
    if (.not.(allocated(pEpis(i)%pEpi))) allocate(pEpis(i)%pEpi(pEpis(i)%nb))
    do j=1,pEpis(i)%nb
      read(100)pEpis(i)%pEpi(j)
    enddo
  enddo
  read(100)mb
  read(100)acentroid
  close(100)
enddo
if (ok .ne. 0) then
  write(*,*)"problème dans che2013_coldruns_syn : le fichier ", &
    "OUTPUT/files/Cold/In_"//trim(adjustl(chaine))//".bin n''existe pas "
  stop
endif
! -------- tri des colds runs                                --------      .
call triparam(nbChaineMVcold,misfit,param_best)
! -------- calcul des moyennes pour les coldruns             --------      .
call moycoldruns(nbChaineMVcold,param_best,misfit,nbChaineMVhot,dc)

! ------------------------------  ecriture ---------------------------------      .
allocate(vec(nbChaineMVcold))
do j=1,nbChaineMVcold
 vec(j)=acceptance(j)%val
enddo
call moy_ec (vec,nbChaineMVcold,nbChaineMVcold,moy,ec)
deallocate(vec)
write(*,1111)' acceptance (%)                 : ',moy,' +ou- ',ec
write(*,1111)' fonction coût minimale TOTALE  : ',dc%moytot%mis,' +ou- ',dc%ectot%mis
write(*,1111)' fonction coût minimale SELECT  : ',dc%moyselect%mis,' +ou- ',dc%ecselect%mis

! ----------------------  sauve pour che_hotruns -------------------------      .
open(unit=100,file="OUTPUT/files/passCold2Hot.bin",status="replace",form="unformatted",access="sequential")
  write(100)nbsta,nbtps
  do i=1,nbseismes
```

```
      do j=1,nbtps(i)
        write(100)D(i)%datatps(j)
      enddo
    enddo
    write(100)param_best
    write(100)xmin,xmax
    write(100)dc
    do i=1,nbseismes
        write(100)pEpis(i)%nb
      do j=1,pEpis(i)%nb
        write(100)pEpis(i)%pEpi(j)
      enddo
    enddo
    write(100)mb
    write(100)acentroid
  close(100)

  call print_mess_3
  ! ------------------------  fin du programme ------------------------------    .
  do i=1,nbseismes
    deallocate (D(i)%datatps,pEpis(i)%pEpi)
  enddo
  deallocate(param_best,param_init,misfit,acceptance)
  ! ------------------------------------------------------------------------    .
1111 format(a,f8.3,a,f6.2)
  ! ------------------------------------------------------------------------    .

end program che2013_coldruns_init



! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
```

1.4 SRC/PROG/che_hotruns_init.f90

```
! programme principal II
! ************************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                    ---------    .
! -------- CHE2013_coldruns version 1.5                       ---------    .
! -------- octobre 2013 - décembre 2014                       ---------    .
! --------                                                    ---------    .
! -------- Prog. basé uniquement sur des méthodes non linéaires ---------    .
! --------                                                    ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr       ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope        !
!  that it will be useful however without any warranties.                   !
!  Use it on your own risk. Please, report bugs/improvements/... .          !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                           !
! ------------------------------------------------------------------------    .
! Initialisation des Hotruns                                                !
!                                                                           !
program che2013_hotruns
  ! ------------------------------------------------------------------------    .
  ! -------- modules :                                          ---------    .
  use modparam
  use typetemps
  use mt19937
  use tirage
  use time
  use affiche
  use sub_param
```

```
31  ! ------------------------------------------------------------------------     .
32  ! -------- déclaration :                                          --------     .
33  implicit none
34  ! -------                                                        --------     .
35  type(dataall) :: D(nbseismes)                                                 ! données de temps
36  type(parametres), dimension(:), allocatable :: param_init                     ! paramètres d'inv.
37  type(fcout) :: misfit                                                         ! fonction coût
38  type(accept) :: acceptance                                                    ! acceptance
39  type(parametresinv) :: p                                                      ! paramètres d'inv.
40  type(coldmoy) :: dc                                                           ! modèles du coldrun
41  type(priorEPI) :: pEpis(nbseismes)                                            ! prior
42  type(amoho_centroid) :: acentroid                                             ! si moho non tabulaire
43  ! type(date_secPS) :: midi20
44  ! -------                                                        --------     .
45  real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)                             ! cercles pond.
46  real(KIND=wr) :: val1, val2
47  ! real(KIND=wr) :: deltaP,deltaS
48  ! -------                                                        --------     .
49  integer(KIND=wi) :: mb                                                        ! prior
50  integer(KIND=wi) :: i,j,k,ok
51  integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                              ! nombre chaînes
52  integer(KIND=wi) :: maxiterhot,maxitercold                                    ! nombre d'itérations
53  integer(KIND=wi) :: nbsta,nbtps(nbseismes)                                    ! nombre station possible et reel nombre de données de temps
54  ! -------                                                        --------     .
55  character (LEN=5) :: numberchaine
56  ! ------------------------------------------------------------------------     .
57
58  ! ------------------------------------------------------------------------     .
59  ! --------  nb de chaines de Markov et d'iterations par chaine   --------     .
60  call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold,chut=.true.)
61  allocate(param_init(nbChaineMVcold))
62  ! ------------------------------------------------------------------------     . lecture des coldruns
63  ok = 0
64  open(unit=200,file="OUTPUT/files/passCold2Hot.bin",status="old",form="unformatted",access="sequential",iostat = ok)
65    read(200)nbsta,nbtps
66    do i=1,nbseismes
67      allocate(D(i)%datatps(nbtps(i)))
68      do j=1,nbtps(i)
69        read(200)D(i)%datatps(j)
70      enddo
71    enddo
72    read(200)param_init
73    read(200)xmin,xmax
74    read(200)dc
75    do i=1,nbseismes
76      read(200)pEpis(i)%nb
77      allocate(pEpis(i)%pEpi(pEpis(i)%nb))
78      do j=1,pEpis(i)%nb
79        read(200)pEpis(i)%pEpi(j)
80      enddo
81    enddo
82    read(200)mb
83    read(200)acentroid
84  close(200)
85  if (ok.ne.0) then
86    write(*,*)'problème dans che_hotruns : le fichier OUTPUT/files/passCold2Hot.bin n''existe pas '
87  endif
88  ! ------------------------------------------------------------------------     .
89
90  ! ------------------------------------------------------------------------     .
91  ! ------------------------------------------------------------------------     .
92  ! -------- début MCMc                                            --------     . HOTRUNS
93  ! ------------------------------------------------------------------------     .
94  ! ------------------------------------------------------------------------     .
95
```

```
96      ! -------- lecture du prior                                    ---------     .
97      do i=1,nbChaineMVhot
98        call lect_prior(p,param_init(i),"H")                                      ! H -> hotruns
99      enddo
100
101     ! -------- initialisation des autres paramètres               ---------     .
102     call init_div(misfit,acceptance)
103     ! ------------------------------------------------------------------------  . parametres hypocentraux
104     ! -------- initialisation des bornes temporelles              ---------     .
105     !
106     ! centre du prior correspond à la moyenne des coldruns sélectionnées,
107     ! plus ou moin 3 écart-types (plus une demi seconde)
108     !
109     do j=1,nbseismes
110       p%maxi%Tzero(j) = dc%tempsrefcold(j)                                      ! borne sup.
111       p%maxi%Tzero(j)%sec = dc%moyselect%par%Tzero(j)%sec + max(3.00_wr*dc%ecselect%par%Tzero(j)%sec+0.75_wr,0.75_wr)  ! 3 ecartypes superieurs minimum : 0,75
              seconde
112       p%maxi%Tzero(j)%sec = real(int(p%maxi%Tzero(j)%sec)+1,wr)
113       call basetime(p%maxi%Tzero(j))
114       p%mini%Tzero(j) = dc%tempsrefcold(j)                                    ! borne inf.
115       p%mini%Tzero(j)%sec =  dc%moyselect%par%Tzero(j)%sec - max(3.00_wr*dc%ecselect%par%Tzero(j)%sec+0.75_wr,0.75_wr) ! 3 ecartypes inférieurs minimum : 0,75
              seconde
116       p%mini%Tzero(j)%sec = real(int(p%mini%Tzero(j)%sec),wr)
117       call basetime(p%mini%Tzero(j))
118     enddo
119     ! -------- initialisation des bornes épicentales              ---------     . centre du prior correspond à la moyenne des coldruns sélectionnées, plus ou moin 3
            écart-types
120     p%centreY(:) = dc%moyselect%par%lat(:)
121     p%centreX(:) = dc%moyselect%par%lon(:)
122     ! -------- recherche de l'épicentre dans ce rayon             ---------     . rayon correspond à 3 sigma des coldruns
123     do j=1,nbseismes
124       val1 = 3.00_wr*dc%ecselect%par%lat(j)*(pi*rT)/180.0_wr                    ! en km
125       val2 = 3.00_wr*dc%ecselect%par%lon(j)*pi*(cos(dc%moyselect%par%lat(j)/180.0_wr*pi)*6371.0_wr)/180.0_wr ! en km
126       p%Rayon(j) = max(val1,val2,5.0_wr)                                        ! diamètre minimum : 10,0 km
127     enddo
128     !
129     ! ------------------------------------------------------------------------  . si certains paramètres fixes ...
130     call paramfixe(p)
131     ! ------------------------------------------------------------------------  .
132
133
134     ! ------------------------------------------------------------------------ .
135     ! Carriere
136     ! ------------------------------------------------------------------------ .
137     !do i=1,nbseismes
138     !  midi20=D(i)%datatps(1)%tpsR
139     !  midi20%date%hour=11                                                     ! midi TU, pour Vannes en hiver
140     !  midi20%date%min=20
141     !  call basetime(midi20)
142     !  call difftime(deltaP,deltaS,midi20,D(i)%datatps(1)%tpsR)
143     !  if ((abs(deltaP).lt.1.00_wr).or.(i==45)) then
144     !    write(*,*)'seisme ',i,' : tire de carriere ?'
145     !    write(*,*)'Zhypo max = 6 km'
146     !    p%maxi%Zhypo(i) = 6.d0
147     !    p%ecartype%Zhypo(i) =  p%ecartype%Zhypo(i)/4.0_wr
148     !  endif
149     !  ! ----------------------------------------------------------------------- .
150     !enddo
151     ! ------------------------------------------------------------------------ .
152     ! ------------------------------------------------------------------------ .
153
154     ! production de fichier
155     ! parce que dans la norme MPI-Fortran 2008,
156     ! faire un call MPI_BCAST avec des types dérivés, c'est pas prévu !
157
```

```
158     ! ---------------------  un fichier par hotruns  ------------------------     .
159     do k=1,nbChaineMVhot
160       write(numberchaine(1:5),'(i5)')k
161       open(unit=100,file="OUTPUT/files/Hot/In_"//trim(adjustl(numberchaine))//".bin", &
162           status="replace",form="unformatted",access="sequential")
163         write(100)nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold
164         write(100)nbsta,nbtps
165         write(100)p
166         do i=1,nbseismes
167           do j=1,nbtps(i)
168             write(100)D(i)%datatps(j)
169           enddo
170         enddo
171         write(100)misfit
172         write(100)acceptance
173         write(100)param_init(k)
174         write(100)xmin,xmax
175         write(100)dc
176         do i=1,nbseismes
177           write(100)pEpis(i)%nb
178           do j=1,pEpis(i)%nb
179             write(100)pEpis(i)%pEpi(j)
180           enddo
181         enddo
182         write(100)mb
183         write(100)acentroid
184       close(100)
185     enddo
186   ! ---------------------  fin du programme ------------------------------     .
187   do i=1,nbseismes
188     deallocate (D(i)%datatps,pEpis(i)%pEpi)
189   enddo
190   deallocate(param_init)
191   ! ------------------------------------------------------------------------     .
192
193 end program che2013_hotruns
194
195
196
197 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
```

1.5 SRC/PROG/che_hotruns.f90

```
1  ! programme principal II bis
2  ! ************************************************************************     .mh
3  ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
4  ! --------                                                  ---------     .
5  ! --------  CHE2013_coldruns version 1.5                    ---------     .
6  ! --------  octobre 2013 - décembre 2014                    ---------     .
7  ! --------                                                  ---------     .
8  ! --------  Prog. basé uniquement sur des méthodes non linéaires  ---------     .
9  ! --------                                                  ---------     .
10 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
11 ! --------  Méric Haugmard meric.haugmard@univ-nantes.fr      ---------     .
12 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
13 !  This program is distributed for research purposes and in the hope          !
14 !  that it will be useful however without any warranties.                     !
15 !  Use it on your own risk. Please, report bugs/improvements/... .            !
16 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@     .
17 !                                                                            !
18 !             L'expérience cubaine, puis au Congo ou en Bolivie :            !
19 !                    différentes réalisations concrètes                      !
20 !                mettant à jour le visage de la Révolution.                  !
21 !                                                                            !
22 ! ------------------------------------------------------------------------     .
```

```
23 ! Période des réalisations de densités a posteriori                                    !
24 !                                                                                      !
25 program che2013_hotruns
26   ! ------------------------------------------------------------------------------     .
27   ! -------- modules :                                                  --------     .
28   use modparam
29   use typetemps
30   use time
31   use mt19937
32   use cpt_temps
33   use algo_metropolis
34   use tirage
35   use datalecture
36   use pb_direct
37   use affiche
38   use sub_param
39   use sub_misfit
40   use distance_epi
41   ! ------------------------------------------------------------------------------     .
42   ! -------- déclaration :                                              --------     .
43   implicit none
44
45     include 'mpif.h'
46
47   ! --------                                                           --------     .
48   type(dataall) :: D(nbseismes)                                                      ! données de temps
49   type(parametres) :: param_init,param_best                                          ! paramètres d'inv.
50   type(fcout) :: misfit                                                             ! fonction coût
51   type(accept) :: acceptance                                                        ! acceptance
52   type(parametresinv) :: p                                                          ! paramètres d'inv.
53   type(coldmoy) :: dc                                                               ! modèles du coldrun
54   type(residus), dimension(:), allocatable :: R                                     ! résidus (si FLAGresSTA=.true.)
55   type(priorEPI) :: pEpis(nbseismes)                                                ! prior
56   type(amoho_centroid) :: acentroid                                                 ! si moho non tabulaire
57   ! --------                                                           --------     .
58   real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)                                 ! cercles pond.
59   ! real(KIND=wr) :: VPVSch                                                          ! ratio VpVs défini par le diagramme de Châpelain
60   ! --------                                                           --------     .
61   integer(KIND=wi) :: mb                                                            ! prior
62   integer(KIND=wi) :: i,j,k,l,ok,pourcentage
63   integer(KIND=wi) :: noctet, nbauto
64   integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                                  ! nombre chaînes
65   integer(KIND=wi) :: maxiterhot,maxitercold                                        ! nombre d'itérations
66   integer(KIND=wi) :: nbsta,nbstaR,nbtps(nbseismes)                                 ! nombre station possible, de phases et reel nombre de données de temps
67   integer(KIND=wi) :: nmod                                                          ! nombre modèles sélectionnés par chaine
68   ! --------                                                           --------     .
69   logical :: critique                                                               ! si moho trop bas, onde refacté observé mais non prédite (vrai)
70   logical :: savemod                                                                ! modèle sauvé (vrai)
71   logical :: initauto                                                               ! modèle sauvé (vrai)
72   logical :: accepte                                                                ! modèle accepté (vrai) ou rejeté (faux)
73   ! --------                                                           --------     .
74   character (LEN=5) :: numberchaine
75   character (LEN=30) :: chaine
76   ! --------                                                           --------     . MPI :
77   integer :: nb_procs,rang,code,err
78   integer, dimension(:), allocatable :: allseed
79   integer :: seed
80   ! --------                                                           --------     .
81   logical, parameter :: plotmisfit=.false.
82   ! ------------------------------------------------------------------------------     .
83
84   ! ------------------------------------------------------------------------------     . MPI_BEGIN
85   call MPI_INIT(code)
86   call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
87   call MPI_COMM_RANK(MPI_COMM_WORLD,rang,code)
```

```
88  ! ------------------------------------------------------------------------     . MPI_BEGIN
89  ! -------- initialisation de la graine                            --------     .
90  if (rang==0) then
91    call print_mess_2bis
92    call initseed(libre)                                                        ! aléatoire calé sur temps CPU
93    ! -------- clacul de nb_procs graines                           --------     .
94
95    allocate(allseed(nb_procs))
96    do i=1,nb_procs
97      allseed(i)=abs(genrand_int31())+10000
98      do while (allseed(i) > 10000000)
99        seed = int(10000123.0_wr*genrand_real1())
100       allseed(i) = allseed(i) - seed
101     enddo
102   enddo
103 endif
104 ! -------- partage des graines                                  --------     .
105 call MPI_SCATTER(allseed,1,MPI_INTEGER,seed,1,MPI_INTEGER,0,MPI_COMM_WORLD,code)
106 call init_genrand (int(seed*(rang+1),wi))                                     ! initialisation des graines pour chaque processus
107 !write(*,*)'générateur de nombre aléatoire :       ',seed*(int(rang+1))
108 ! ------------------------------------------------------------------------     .
109
110 ! ------------------------------------------------------------------------     .
111 ! -------- lecture des données                                  --------     .
112 call lectnbdata(nbsta,nbtps)
113 ! ------------------------------------------------------------------------     .
114 ! --------  nb de chaines de Markov et d'iterations par chaine  --------     .
115 call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold,chut=.true.)
116 ! ------------------------------------------------------------------------     . lecture des coldruns
117 ok = 0
118 write(numberchaine(1:5),'(i5)')rang+1
119 open(unit=100,file="OUTPUT/files/Hot/In_"//trim(adjustl(numberchaine))//".bin", &
120   status="old",form="unformatted",access="sequential",iostat = ok)
121 read(100)nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold
122 read(100)nbsta,nbtps
123 read(100)p
124 do i=1,nbseismes
125   allocate(D(i)%datatps(nbtps(i)))
126   do j=1,nbtps(i)
127     read(100)D(i)%datatps(j)
128   enddo
129 enddo
130 read(100)misfit
131 read(100)acceptance
132 read(100)param_init
133 read(100)xmin,xmax
134 read(100)dc
135 do i=1,nbseismes
136   read(100)pEpis(i)%nb
137   allocate(pEpis(i)%pEpi(pEpis(i)%nb))
138   do j=1,pEpis(i)%nb
139     read(100)pEpis(i)%pEpi(j)
140   enddo
141 enddo
142 read(100)mb
143 read(100)acentroid
144 close(100)
145 ! -------                                                      --------     .
146 if (ok.ne.0) then
147   write(*,*)"problème dans che_hotruns : le fichier OUTPUT/files/Hot/In_"//trim(adjustl(numberchaine))//".bin n''existe pas "
148   call MPI_ABORT(MPI_COMM_WORLD,err,code)
149 endif
150 p%valOLd = param_init
151 p%valNew = param_init
152 ! ------------------------------------------------------------------------     . calul résidus aux stations
```

```
153   if (FLAGresSTA) call initR(D,R,maxiterhot,nbtps,nbsta,nbstaR)
154   ! ------------------------------------------------------------------     .
155
156   ! ------------------------------------------------------------------     .
157   ! ------------------------------------------------------------------     .
158   ! -------- début MCMc                                  ---------     . HOTRUNS
159   ! ------------------------------------------------------------------     .
160   if((rang==0).and.(plotmisfit)) open(unit=10,file="OUTPUT/files/Hot/mis_1.txt",status="replace")
161   ! ------------------------------------------------------------------     .
162   i=rang+1
163   ! ------------------------------------------------------------------     .
164     if (rang==0) call print_messchaine(i,nbChaineMVhot)
165     ! -------- initialisation                              ---------     .
166     nmod = 0                                                               ! nombre de modèle sélectionnés pdt une chaîne
167     ! ------------------------------------------------------------------     .
168     ! -------- création d'un fichier par chaîne            ---------     .
169     write(numberchaine(1:5),'(i5)')i
170     inquire ( iolength = noctet ) misfit%new,p%valNew
171     open(unit=205+i,file="OUTPUT/files/"//trim(adjustl(numberchaine))//".bin",status="replace",access='direct',RECL=(noctet))
172     initauto=.false.                                                       ! les premiers modèles ne sont pas sauvés
173     savemod=.false.
174     ! ------------------------------------------------------------------     . calul du ratio VpVs
175     !call Chatelainplot(nbtps,D,vpvs=VPVSch)
176     !write(*,'(a,i3.3)')' VP/VS, Chatelain : 1,',int(VPVSch*1000.0_wr,wi)-1000
177     !call Wadatiplot(nbtps,D,p%valNew,vpvs=VPVSch)
178     !write(*,'(a,i3.3)')' VP/VS, Wadati : 1,',int(VPVSch*1000.0_wr,wi)-1000
179     ! ------------------------------------------------------------------     .
180     ! -------- début d'une chaîne                          ---------     .
181     ! ------------------------------------------------------------------     .
182     unechaine  : do k=1,maxiterhot
183       ! ------------------------------------------------------------------     . progression en %
184       if (i==1) then
185         if ((k.gt.1000).or.(k.ne.1).or.(k.ne.maxiterhot)) then
186           if(mod(k,100)==0) write(chaine(1:30),'(a5,i12,a13)')" 2 - ",k," modèles    "
187         else
188           write(chaine(1:30),'(a5,i12,a13)')" 2 - ",k," modèles     "
189           call progress(k,maxiterhot,chaine)
190         endif
191         call progress(k,maxiterhot,chaine)
192       endif
193       ! ------------------------------------------------------------------     .
194       j=0
195       critique=.true.
196       do while(critique)                                                   ! tant que distance hypocentral << distance critique pour la réfraction
197         j=j+1
198
199         ! -------- tirage au sort des paramètres dans le prior      ---------  .
200         if(k.ne.1) then
201           pourcentage=int(genrand_real1()*100._wr)                         ! aléatoire de 0 à 99
202           l=int(genrand_real1()*real(nbseismes,wr)+1.0_wr)                 ! aléatoire de 1 à nbseismes
203           select case (pourcentage)
204             case (0:24)
205               call tirage_H(p,l,all=.true.)                                ! 25 % : tirage gaussien de de tous les paramètres hypocentraux pour le 'l'ième
                      séisme
206             case (25:49)
207               call tirage_H(p,l,all=.false.)                               ! 25 % : tirage gaussien de 1 des 4 paramètres hypocentraux pour le 'l'ième séisme
208             case (50:74)
209               call tirage_T(p,all=.true.,vpvs=.true.)                      ! 25 % : tirage gaussien de de tous les paramètres de terre (dont VpVs)
210             case (75:99)
211               call tirage_T(p,all=.false.,vpvs=.true.)                     ! 25 % : tirage gaussien de 1 des 4 paramètres de terre (dont VpVs)
212           end select
213         endif
214
215         ! ------------------------------------------------------------------  .
216         ! -------- problème direct pour le jeu de paramètre tiré et chaque donnée
```

```
        call tempsTheoDirect(nbtps,p%valNew,D,critique,acentroid)
        if(j==5) then
          critique=.false.                                          ! pour sortir apres 5 essais
          ! ------- une donnée réfractée observée ne peux pas etre inférieur à la distance hypocentrale minimale pour la refraction
          write(*,*)'problème dans che_hotruns : distance épi + 5 km < distance épi critique pour la réfraction '
        endif
      enddo
      ! ------- calcul de la fonction coût                 --------    .
      call compute_misfit(nbtps,D,misfit%new,xmin,xmax,'H')
      ! ------- modele sauvé ?                             --------    .
      if (.not.initauto) then
        if (k.gt.(maxiterhot/10)) initauto=.true.                   ! on ne garde pas les 10 premier %
        nbauto=(maxiterhot/10)+max(20,int(normal(real(32*nbseismes,wr),real(nbseismes*3,wr)))) ! procchain modèle sauvé
      else
        if(k==nbauto) then
          savemod=.true.                                            ! le modele est sauvé
          nmod=nmod+1
          nbauto=k+max(20,int(normal(real(32*nbseismes,wr),real(nbseismes*3,wr)))) ! procchain modèle sauvé
        else
          savemod=.false.                                           ! le modele n'est pas sauvé
        endif
      endif
      ! ------- Metropolis (acceptation et rejet des modèles)  --------    .
      call metropolis(p,param_best,misfit,acceptance,savemod,accepte)

      if((rang==0).and.(plotmisfit)) then
        if((k.lt.100).or.(mod(k,100)==0)) write(10,*)k+maxitercold,misfit%new
      endif

      ! ------- écriture dans les fichiers                 --------    .
      if(savemod) then
        write(205+i,rec=nmod)misfit%new,p%valNew                    ! écriture du modèle sélectioné dans un fichier files/k.bin
        if (FLAGresSTA) call inR(D,R,nbtps,nbstaR,misfit%new)       ! sauve les résidus dans R -> evaluation des retard aux stations
      endif
      ! ------------------------------------------------------------    .
    enddo unechaine
    call calc_accept(acceptance)

    ! ------------------------------------------------------------    .
    ! ------- fin d'une chaîne                           --------    .
    ! ------------------------------------------------------------    .
    if (FLAGresSTA) call outR(R,nbstaR)                             ! sauve les résidus dans des fichiers .bin
    ! ------------------------------------------------------------    .
    close(205+i)
    ! ------------------------------------------------------------    .
    !call MPI_Barrier(MPI_COMM_WORLD,err)
    !call print_mess_finchainemin(misfit%best,acceptance%val)

  if((rang==0).and.(plotmisfit)) close(10)
  ! ------------------------------------------------------------    .
  ! ------------------------------------------------------------    .
  ! ------- fin MCMc                                   --------    . HOTRUNS
  ! ------------------------------------------------------------    .
  ! ------------------------------------------------------------    .

  ! ----------------  un fichier OUTPUT par cold runs  ------------    .
  open(unit=100,file="OUTPUT/files/Hot/Out_"//trim(adjustl(numberchaine))//".bin", &
     status="replace",form="unformatted",access="sequential")
  write(100)nbsta,nbtps
  write(100)p
  do i=1,nbseismes
    do j=1,nbtps(i)
      write(100)D(i)%datatps(j)
    enddo
  enddo
```

```
  write(100)misfit
  write(100)acceptance
  write(100)dc
  write(100)nmod
  write(100)param_init
  write(100)param_best
  write(100)xmin,xmax
  do i=1,nbseismes
    write(100)pEpis(i)%nb
    do j=1,pEpis(i)%nb
      write(100)pEpis(i)%pEpi(j)
    enddo
  enddo
  write(100)mb
  write(100)acentroid
  close(100)

  ! ------------------------  fin du programme ------------------------------      .
  do i=1,nbseismes
    deallocate (D(i)%datatps,pEpis(i)%pEpi)
  enddo
  if(allocated(R)) deallocate(R)
  ! ---------------------------------------------------------------------      . MPI_END
  call MPI_FINALIZE(code)
  ! ---------------------------------------------------------------------      . MPI_END

end program che2013_hotruns



! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@      .
```

1.6 SRC/PROG/che_hotruns_syn.f90

```
! programme principal II ter
! ************************************************************************     .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@      .
! --------                                                  ---------      .
! -------- CHE2013_coldruns version 1.5                     ---------      .
! -------- octobre 2013 - décembre 2014                     ---------      .
! --------                                                  ---------      .
! -------- Prog. basé uniquement sur des méthodes non linéaires ---------      .
! --------                                                  ---------      .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@      .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr     ---------      .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@      .
!  This program is distributed for research purposes and in the hope         !
!  that it will be useful however without any warranties.                    !
!  Use it on your own risk. Please, report bugs/improvements/... .           !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@      .
!                                                                            !
! ---------------------------------------------------------------------      .
! Synthèse des Hotruns                                                       !
!                                                                            !
program che2013_hotruns
  ! ---------------------------------------------------------------------      .
  ! -------- modules :                                        ---------      .
  use modparam
  use typetemps
  use affiche
  use sub_param
  ! ---------------------------------------------------------------------      .
  ! -------- déclaration :                                    ---------      .
  implicit none
  ! --------                                                  ---------      .
```

```
32  type(dataall) :: D(nbseismes)                                          ! données de temps
33  type(parametres), dimension(:), allocatable :: param_init,param_best   ! paramètres d'inv.
34  type(fcout), dimension(:), allocatable :: misfit                       ! fonction coût
35  type(accept), dimension(:), allocatable :: acceptance                  ! acceptance
36  type(parametresinv) :: p                                               ! paramètres d'inv.
37  type(coldmoy) :: dc                                                    ! modèles du coldrun
38  type(residus), dimension(:), allocatable :: R                          ! résidus (si FLAGresSTA=.true.)
39  type(priorEPI) :: pEpis(nbseismes)                                     ! prior
40  type(amoho_centroid) :: acentroid                                      ! si moho non tabulaire
41  ! -------                                                  ---------    .
42  real(KIND=wr), dimension(:), allocatable :: vec
43  real(KIND=wr) :: moy,ec
44  real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)                      ! cercles pond.
45  ! -------                                                  ---------    .
46  integer(KIND=wi) :: mb                                                 ! prior
47  integer(KIND=wi) :: i,j,k,ok
48  integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                       ! nombre chaînes
49  integer(KIND=wi) :: maxiterhot,maxitercold                             ! nombre d'itérations
50  integer(KIND=wi) :: nbsta,nbtps(nbseismes)                             ! nombre station possible et reel nombre de données de temps
51  integer(KIND=wi), dimension(:), allocatable :: nmod                    ! nombre modèles sélectionnés par chaine
52  ! -------                                                  ---------    .
53  ! -------                                                  ---------    .
54  character (LEN=5) :: numberchaine
55  ! ------------------------------------------------------------------    .
56
57  ! ------------------------------------------------------------------    .
58  ! -------  nb de chaines de Markov et d'iterations par chaine  ---------    .
59  call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold,chut=.true.)
60  allocate(param_best(nbChaineMVhot),param_init(nbChaineMVhot))
61  allocate(misfit(nbChaineMVhot),acceptance(nbChaineMVhot))
62  allocate(nmod(nbChaineMVhot))
63
64  ok=0
65  do k=1,nbChaineMVhot
66    write(numberchaine(1:5),'(i5)')k
67    ! ----------------  un fichier OUTPUT par cold runs  ----------------    .
68    open(unit=100,file="OUTPUT/files/Hot/Out_"//trim(adjustl(numberchaine))//".bin", &
69         status="old",form="unformatted",access="sequential",iostat = ok)
70      read(100)nbsta,nbtps
71      read(100)p
72      do i=1,nbseismes
73        if (.not.(allocated(D(i)%datatps))) allocate(D(i)%datatps(nbtps(i)))
74        do j=1,nbtps(i)
75          read(100)D(i)%datatps(j)
76        enddo
77      enddo
78      read(100)misfit(k)
79      read(100)acceptance(k)
80      read(100)dc
81      read(100)nmod(k)
82      read(100)param_init(k)
83      read(100)param_best(k)
84      read(100)xmin,xmax
85      do i=1,nbseismes
86        read(100)pEpis(i)%nb
87        if (.not.(allocated(pEpis(i)%pEpi))) allocate(pEpis(i)%pEpi(pEpis(i)%nb))
88        do j=1,pEpis(i)%nb
89          read(100)pEpis(i)%pEpi(j)
90        enddo
91      enddo
92      read(100)mb
93      read(100)acentroid
94    close(100)
95  enddo
96  if (ok.ne.0) then
```

```
97      write(*,*)"problème dans che_hotruns : le fichier OUTPUT/files/Hot/Out_"//trim(adjustl(numberchaine))//".bin n''existe pas "
98      stop
99    endif
100
101
102   ! ------------------------------  ecriture ----------------------------------     .
103   allocate(vec(nbChaineMVhot))
104   do j=1,nbChaineMVhot
105     vec(j)=acceptance(j)%val
106    enddo
107   call moy_ec (vec,nbChaineMVhot,nbChaineMVhot,moy,ec)
108   write(*,1111)' acceptance (%)                  : ',moy,' +ou- ',ec
109   ! --------                                                 ---------     .
110   do j=1,nbChaineMVhot
111     vec(j)=misfit(j)%best
112   enddo
113   call moy_ec (vec,nbChaineMVhot,nbChaineMVhot,moy,ec)
114   deallocate(vec)
115   write(*,1111)' fonction coût minimale          : ',moy,' +ou- ',ec
116
117
118   call print_mess_3bis
119
120   ! ------------------------------------------------------------------------     .
121   ! ------------------------------------------------------------------------     .
122   ! --------- fin MCMc                                           ---------     . HOTRUNS
123   ! ------------------------------------------------------------------------     .
124   ! ------------------------------------------------------------------------     .
125
126   ! ----------------------  sauve pour che_plot ----------------------------     .
127   open(unit=204,file="OUTPUT/files/passHot2Plot.bin",status="replace",form="unformatted",access="sequential")
128     do i=1,nbseismes
129       write(204)D(i)%datatps
130     enddo
131     write(204)nmod
132     write(204)p
133     write(204)misfit
134     write(204)param_best
135     write(204)xmin
136     write(204)xmax
137     write(204)acceptance
138     write(204)dc
139     do i=1,nbseismes
140       write(204)pEpis(i)%nb
141       do j=1,pEpis(i)%nb
142         write(204)pEpis(i)%pEpi(j)
143       enddo
144     enddo
145     write(204)mb
146     write(204)acentroid
147   close(204)
148   ! ---------------------  fin du programme ------------------------------     .
149   do i=1,nbseismes
150     deallocate (D(i)%datatps,pEpis(i)%pEpi)
151   enddo
152   deallocate(param_best,param_init,misfit,acceptance,nmod)
153   if(allocated(R)) deallocate(R)
154   ! ------------------------------------------------------------------------     .
155 1111 format(a,f8.3,a,f6.2)
156   ! ------------------------------------------------------------------------     .
157
158 end program che2013_hotruns
159
160
161
```

```
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
```

1.7 SRC/PROG/che_plot.f90

```
! programme principal III
! ****************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                     --------    .
! -------- CHE2013_plots version 1.5                           --------    .
! -------- octobre 2013 - décembre 2014                        --------    .
! --------                                                     --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr        --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope       !
!  that it will be useful however without any warranties.                 !
!  Use it on your own risk. Please, report bugs/improvements/... .         !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                          !
!         Mort de Ernesto CHE Gevara en 1967 (La Higuera, Bolivie),        !
!        il devient alors un symbole de la Révolution bien identifié,      !
!     mise en place d'une analyse a posteriori de la Révolution cubaine.   !
!                                                                          !
! -----------------------------------------------------------------------    .
!                                                                          !
!                     Alea jacta est ; les dés sont jetés.                 !
!                                                                          !
! -----------------------------------------------------------------------    .
! période d'exploitation des résultats                                     !
!                                                                          !
program che2013_plot
  ! ---------------------------------------------------------------------    .
  ! -------- modules :                                         --------    .
  use modparam
  use typetemps
  use latexscript
  use datalecture
  use affiche
  use sub_param
  use figure_GMT
  use mt19937
  use figure_posteriori
  use time
  ! ---------------------------------------------------------------------    .
  ! -------- déclaration :                                     --------    .
  implicit none
  ! --------                                                   --------    .
  type(dataall) :: D(nbseismes)                                             ! données de temps
  type(parametres), dimension(:), allocatable :: param_best                 ! paramètres d'inv.
  type(fcout), dimension(:), allocatable :: misfit                          ! fonction coût
  type(accept), dimension(:), allocatable :: acceptance                     ! acceptance
  type(parametresinv) :: p                                                  ! paramètres d'inv.
  type(densityplot) :: dp
  type(coldmoy) :: dc                                                       ! modèles du coldrun
  type(ellip) :: E(nbseismes)                                               ! éllipses
  type(priorEPI) :: pEpis(nbseismes)                                        ! prior
  type(amoho_centroid) :: acentroid                                         ! si moho non tabulaire
  type(parametres), dimension(:), allocatable :: modelesIN
  ! --------                                                   --------    .
  real(KIND=wr) :: xmin(nbseismes), xmax(nbseismes)            --------    ! cercles pond.
  ! --------                                                   --------    .
  integer(KIND=wi) :: i,j,k,ok
  integer(KIND=wi) :: mb                                                    ! prior
  integer(KIND=wi) :: nbChaineMVhot,nbChaineMVcold                          ! nombre chaînes
  integer(KIND=wi) :: maxiterhot,maxitercold                                ! nombre d'itérations
```

```
integer(KIND=wi) :: nbsta, nbtps(nbseismes)                                  ! nombre station et nombre de données de temps
integer(KIND=wi), dimension(:), allocatable :: nmod                          ! nombre modèles sélectionnés par chaine
integer(KIND=wi) :: nbmod
! -------                                                  --------     .
character (LEN=20) :: nomfichier
character (LEN=5) :: nbdeseismes
character(LEN=4), dimension(:), allocatable :: nomsta
! ------------------------------------------------------------------     .

! ------------------------------------------------------------------     .
call initseed(libre)                                                         ! aléatoire calé sur temps CPU

! ------------------------------------------------------------------     .
! --------  relecture de quelques varaiables               --------     .
! ------------------------------------------------------------------     .
! -------- lecture des données                              --------     .
call lectnbdata(nbsta,nbtps)
do i=1,nbseismes
  allocate(D(i)%datatps(nbtps(i)))
enddo
! --------  nb de chaines de Markov et d'iterations par chaine  --------     .
call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold,chut=.true.)
! -------                                                  --------     .
allocate(param_best(nbChaineMVhot))
allocate(misfit(nbChaineMVhot),acceptance(nbChaineMVhot))
allocate(nmod(nbChaineMVhot))
! ------------------------------------------------------------------     .
! lecture des hotruns
ok=0
open(unit=400,file="OUTPUT/files/passHot2Plot.bin",status="old",form="unformatted",access="sequential",iostat = ok)
do i=1,nbseismes
  read(400)D(i)%datatps
enddo
read(400)nmod
read(400)p
read(400)misfit
read(400)param_best
read(400)xmin
read(400)xmax
read(400)acceptance
read(400)dc
do i=1,nbseismes
  read(400)pEpis(i)%nb
  allocate(pEpis(i)%pEpi(pEpis(i)%nb))
  do j=1,pEpis(i)%nb
    read(400)pEpis(i)%pEpi(j)
  enddo
enddo
read(400)mb
read(400)acentroid
close(400)
if (ok.ne.0) write(*,*)'problème dans che_plot : le fichier OUTPUT/files/passHot2Plot.txt n''existe pas '

! ------------------------------------------------------------------     .
! --------  traitement a posteriori des modèles sélectionnés  --------     .
! ------------------------------------------------------------------     .
call lect_mod_select(p,dp,nbChaineMVhot,misfit,param_best)                   ! relecture des modèles sélectionnés dans les fichier de sortie McMC
! -------- tri des meilleurs modèles                        --------     .
call print_mess_4
dp%deltaxy=150                                                               ! maillage X et Y du diagramme de densité
call moy_mod_select(dp,nbChaineMVhot,param_best,misfit)                      ! calcule les moy, et et mode pour chaque paramètre

! ------------------------------------------------------------------     .
! --------  géneration des densité à priori                  --------     .
! ------------------------------------------------------------------     . ecriture pour che_apriori.90
```

```
do i=1,nbseismes
   write(nbdeseismes(1:5),'(i5)')i
   open(unit=500,file="OUTPUT/files/Plot/apriori_"//trim(adjustl(nbdeseismes))//".bin", &
     status="replace",form="unformatted",access="sequential")
   write(500)i,nbtps,dp%temps_ref,mb
   do k=1,nbseismes
     write(500)D(k)%datatps
   enddo
   do k=1,nbseismes
     write(500)pEpis(k)%nb
     do j=1,pEpis(k)%nb
       write(500)pEpis(k)%pEpi(j)
     enddo
   enddo
   write(500)acentroid
   close(500)
enddo

! ------------------------------------------------------------------------     .
! -------  moyenne 100 meilleurs modèles avec toutes les stations --------     .
! ------------------------------------------------------------------------     .
call mksynthallsta(acentroid,dp)


! ------------------------------------------------------------------------- .
! étude des gradients sur la fonction coût
! ------------------------------------------------------------------------- .
!if (plotposteriori) then
!  nbmod=5000
!  allocate(modelesIN(nbmod))
!  do i=1,nbmod
!    modelesIN(i)%VC=dp%VC%vec10000(i,1)
!    modelesIN(i)%VM=dp%VM%vec10000(i,1)
!    modelesIN(i)%Zmoho=dp%Zmoho%vec10000(i,1)
!    modelesIN(i)%VpVs=dp%VpVs%vec10000(i,1)
!    modelesIN(i)%Lat(:)=dp%Lat(:)%vec10000(i,1)
!    modelesIN(i)%Lon(:)=dp%Lon(:)%vec10000(i,1)
!    modelesIN(i)%Zhypo(:)=dp%Zhypo(:)%vec10000(i,1)
!    modelesIN(i)%Tzero(:)=dp%temps_ref(:)
!    modelesIN(i)%Tzero(:)%sec=modelesIN(i)%Tzero(:)%sec+dp%Tzero(:)%vec10000(i,1)
!    do j=1,nbseismes
!      call basetime(modelesIN(i)%Tzero(j))
!    enddo
!  enddo
!  nomfichier='POST_HOTS_i'
  !call PosterioriExploration(p,nbmod,pEpis,nbtps,D,acentroid,xmin,xmax,mb,nomfichier,modelesIN)
!  deallocate(modelesIN)
!endif
! ------------------------------------------------------------------------- .



! ------------------------------------------------------------------------     .
! --------  géneration des scripts                                ---------     .
! ------------------------------------------------------------------------     .
! -------- diagramme de densité, scripts GMT                      ---------     .
call print_mess_5
call GMTfull(dp,nmod,nbChaineMVhot,xmin,xmax,nbtps,nbsta,D,E,nomsta,acentroid)
! -------- scriptes LaTeX                                         ---------     .
call latexfull(dc,dp,xmin,xmax,nbChaineMVhot,acceptance,param_best,misfit,E,nomsta,acentroid,nbtps,D)
! ------------------------------------------------------------------------     .

! ----------------------  fin du programme ------------------------------     .
deallocate(dp%mis%vec,dp%VC%vec,dp%VM%vec,dp%Zmoho%vec,dp%VpVs%vec)
```

```
  do i=1,nbseismes
    deallocate (D(i)%datatps,pEpis(i)%pEpi)
    deallocate(dp%Lat(i)%vec,dp%Lon(i)%vec,dp%Zhypo(i)%vec,dp%Tzero(i)%vec)
  enddo
  deallocate(param_best,misfit,acceptance,nmod)
  if(allocated(nomsta)) deallocate(nomsta)
  !-------------------------------------------------------------------------.
  call print_line
  !-------------------------------------------------------------------------.

end program che2013_plot



!@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
```

1.8 SRC/PROG/che_apriori.f90

```
! programme principal IV
! ********************************************************************** .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
! --------                                                    ---------    .
! -------- CHE2013_apriori version 1.5                        ---------    .
! -------- octobre 2013 - décembre 2014                       ---------    .
! --------                                                    ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr       ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
!  This program is distributed for research purposes and in the hope          !
!  that it will be useful however without any warranties.                     !
!  Use it on your own risk. Please, report bugs/improvements/... .           !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@.
!                                                                            !
!                                                                            !
!                                                                            !
! -------------------------------------------------------------------------.
! période d'exploitation des distributions a priori                          !
!                                                                            !
program che2013_apriori
  ! -------------------------------------------------------------------------.
  ! -------- modules :                                          ---------    .
  use modparam
  use typetemps
  use sub_param
  use mt19937
  use affiche
  ! -------------------------------------------------------------------------.
  ! -------- déclaration :                                      ---------    .
  implicit none

    include 'mpif.h'

  ! --------                                                    ---------    .
  type(dataall) :: D(nbseismes)                                              ! données de temps
  type(densityplot) :: dp
  type(priorEPI) :: pEpis(nbseismes)                                         ! prior
  type(amoho_centroid) :: acentroid                                          ! si moho non tabulaire
  ! --------                                                    ---------    .
  integer(KIND=wi) :: i,j,k,ok
  integer(KIND=wi) :: mb                                                     ! prior
  integer(KIND=wi) :: nbm_ap                                                 ! nombre de modèles a priori générés
  integer(KIND=wi) :: nbtps(nbseismes)                                       ! nombre station et nombre de données de temps
  ! --------                                                    ---------    .
  character (LEN=5) :: nbdeseismes
  ! -------------------------------------------------------------------------.
```

```
48  ! -------                                                    ---------     . MPI :
49  integer :: nb_procs,rang,code,err
50  integer :: seed
51  integer, dimension(:), allocatable :: allseed
52  ! ------------------------------------------------------------------     . MPI_BEGIN
53  call MPI_INIT(code)
54  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
55  call MPI_COMM_RANK(MPI_COMM_WORLD,rang,code)
56  ! ------------------------------------------------------------------     . MPI_BEGIN
57  ! -------- initialisation de la graine                       ---------     .
58  if (rang==0) then
59    call initseed(libre)                                                   ! aléatoire calé sur temps CPU
60    ! -------- clacul de nb_procs graines                      ---------     .
61    allocate(allseed(nb_procs))
62    do i=1,nb_procs
63      allseed(i)=abs(genrand_int31())+10000
64      do while (allseed(i) > 10000000)
65        seed = int(10000123.0_wr*genrand_real1())
66        allseed(i) = allseed(i) - seed
67      end do
68    enddo
69  endif
70  ! -------- partage des graines                               ---------     .
71  call MPI_SCATTER(allseed,1,MPI_INTEGER,seed,1,MPI_INTEGER,0,MPI_COMM_WORLD,code)
72  call init_genrand (int(seed*(rang+1),wi))                              ! initialisation des graines pour chaque processus
73  ! write(*,*)'générateur de nombre aléatoire :       ',seed*(int(rang+1))
74  ! ------------------------------------------------------------------     .
75  ! ------------------------------------------------------------------     .
76
77  ! ------------------------------------------------------------------     . relecture des données
78  ok=0
79  write(nbdeseismes(1:5),'(i5)')rang+1
80  open(unit=500+rang,file="OUTPUT/files/Plot/apriori_"//trim(adjustl(nbdeseismes))//".bin", &
81    status="old",form="unformatted",access="sequential",iostat = ok)
82  read(500+rang)k,nbtps,dp%temps_ref,mb
83  do i=1,nbseismes
84    allocate(D(i)%datatps(nbtps(i)))
85    read(500+rang)D(i)%datatps
86  enddo
87  do i=1,nbseismes
88    read(500+rang)pEpis(i)%nb
89    allocate(pEpis(i)%pEpi(pEpis(i)%nb))
90    do j=1,pEpis(i)%nb
91      read(500+rang)pEpis(i)%pEpi(j)
92    enddo
93  enddo
94  read(500+rang)acentroid
95  close(500+rang)
96  if (ok.ne.0) then
97    write(*,*)"problème dans che_apriori : le fichier OUTPUT/files/Plot/", &
98      "apriori_"//trim(adjustl(nbdeseismes))//".bin n''existe pas "
99    call MPI_ABORT(MPI_COMM_WORLD,err,code)
100 endif
101 ! ------------------------------------------------------------------     .
102 ! --------  géneration des densité à priori                  ---------     .
103 ! ------------------------------------------------------------------     .
104 nbm_ap=100000
105 call dist_apriori(k,rang,nbtps,D,nbm_ap,dp%temps_ref,pEpis,mb,acentroid)
106
107 ! ------------------------------------------------------------------     .
108 ! ----------------------  fin du programme -------------------------     .
109 do i=1,nbseismes
110   deallocate (D(i)%datatps,pEpis(i)%pEpi)
111 enddo
112 ! ------------------------------------------------------------------     .
```

```
  if (rang==0) call print_mess_fin
  ! -------------------------------------------------------------------------------------------------------------    . MPI_END
  call MPI_FINALIZE(code)
  ! -------------------------------------------------------------------------------------------------------------    . MPI_END

  ! FIN

end program che2013_apriori



! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
```

1.9 SRC/PROG/sac_bin2txt.f90

```
! programme principal sac_bin2txt
! ***************************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                            ---------    .
! -------- sac2txt                                                    ---------    .
! --------                                                            ---------    .
! -------- septembre 2014                                             ---------    .
! --------                                                            ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr              ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope          !
!  that it will be useful however without any warranties.                     !
!  Use it on your own risk. Please, report bugs/improvements/... .            !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                             !
!    transforme un fichier sac en .txt -> hodochrone (dis hypo, km)           !
!                                                                             !
program bin2txt
  ! ------------------------------------------------------------------------------------------------------------    .
  use sac_i_o
  use modparam
  use typetemps
  use time
  ! --------                                                          ---------    .
  implicit none
  ! --------                                                          ---------    . pour SAC
  real(KIND=4), dimension(:), allocatable :: sacfile
  real(KIND=4) :: delta1, b1, e1
  integer(KIND=4) :: NN, ios, npts1
  character(LEN=112) :: file1,ofile
  ! -------------------------------------------------------------------------------------------------------------    . autres
  type(date_sec) :: d,ref,tzero
  real(KIND=wr) :: AmpliTot,AmpliMin,AmpliMax,AmpliNORM, tt
  real(KIND=wr) :: Disthypo
  integer(KIND=wi) :: jourj, j
  ! -------------------------------------------------------------------------------------------------------------    . lecture du fichier sac
  NN = IARGC()
  if ((NN < 2).or.(NN > 2)) THEN
    write(*,'(a)') "usage:  sac2xy sacfile ofile"
    write(*,'(a)') "        sacfile - input sac file"
    write(*,'(a)') "        ofile - output xy file"
    stop
  endif
  call GETARG(1, file1)
  open(UNIT=1,FILE=file1,STATUS='old',IOSTAT=ios)
  if (ios > 0) THEN
    write(*,*) "ERROR - Input file: '", TRIM(adjustl(file1)), "' does not exist ..."
    CLOSE(1)
    stop
```

```
  endif
  close(1)
  call GETARG(2,ofile)
  !-------------------------------------------------------------------------.
  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,          &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacfile)
    if (nvhdr /= 6) then
      write(*,*) "ERROR - File: '", TRIM(adjustl(file1)), "' appears to be of non-native &
      &byte-order or is not a SAC file."
      stop
    endif
  !-------------------------------------------------------------------------. lecture
  read(*,*)tzero,Disthypo,AmpliNORM
  !-------------------------------------------------------------------------. normalisation amplitude
  AmpliMin=1.e9_wr
  AmpliMax=-1.e9_wr
  do j=0,(int(npts1,wi)-1)
    if (AmpliMin.gt.real(sacfile(j+1),wr)) AmpliMin=real(sacfile(j+1),wr)
    if (AmpliMax.lt.real(sacfile(j+1),wr)) AmpliMax=real(sacfile(j+1),wr)
  enddo
  AmpliTot=max(abs(AmpliMin),abs(AmpliMax))
  !-------------------------------------------------------------------------. compute jour julien
  d%date%year=int(nzyear,wi)
  d%date%hour=int(nzhour,wi)
  d%date%min=int(nzmin,wi)
  d%sec=real(nzsec,wr)+real(b1,wr)+real(nzmsec,wr)/1000.0_wr
  ref%date%year=d%date%year
  ref%date%month=int(1,wi)
  jourj=int(nzjday,wi)
  ref%date%day=jourj
  ref%date%hour=int(0,wi)
  ref%date%min=int(0,wi)
  ref%sec=1.0_wr
  call basetime(ref)
  call jDATE(ref%date%jday,ref%date%year,ref%date%month,ref%date%day)
  call GDATE(ref%date%jday,ref%date%year,ref%date%month,ref%date%day)
  call basetime(ref)
  d%date%year=ref%date%year
  d%date%jday=ref%date%jday
  d%date%month=ref%date%month
  d%date%day=ref%date%day
  call basetime(d)
  !-------------------------------------------------------------------------. écriture
  open(UNIT=2,FILE=ofile,STATUS='REPLACE')
  ref=d
  ref%sec = ref%sec - real(delta1,wr)
  do j=0,(int(npts1,wi)-1)
    ref%sec = ref%sec + real(delta1,wr)
    call basetime(ref)
    call difftime(tt,ref,tzero)
    if (tt.gt.-30.0_wr) then
      write(2,'(2f18.6)') tt,Disthypo+real(sacfile(j+1),wr)*AmpliNORM/AmpliTot
    endif
  enddo
  close(2)
```

```
  ! ------------------------------------------------------------------------   .

end program bin2txt


! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! ************************************************************************   .
```

1.10 SRC/PROG/sac_coda.f90

```
! programme principal sac_coda
! ************************************************************************   .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! --------                                                        ---------   .
! -------- calcul de la magnitude Md d'un séisme                  ---------   .
! -------- d'après la durée de la coda (formule de Lee et al., 1972) -----   .
! --------                                                        ---------   .
! -------- septembre 2014                                         ---------   .
! --------                                                        ---------   .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr           ---------   .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
!  This program is distributed for research purposes and in the hope        !
!  that it will be useful however without any warranties.                   !
!  Use it on your own risk. Please, report bugs/improvements/... .          !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
!                                                                           !
program coda
  ! ------------------------------------------------------------------------   .
  use modparam
  use typetemps
  use time
  use statistiques
  use sac_i_o
  ! --------                                                        ---------   .
  implicit none
  ! -------                                                         ---------   .
  type(dataone) :: Adata,pick,ref
  type(date_secPS) :: tps0
  ! ------------------------------------------------------------------------   .
  real(KIND=wr), parameter :: dmax=300.0_wr                                    ! distance maximale de prise en compte
  ! ------------------------------------------------------------------------   . pour SAC
  real(KIND=4), dimension(:), allocatable :: sacfile
  real(KIND=4) :: delta1, b1, e1
  integer(KIND=4) :: NN, ios, npts1, j
  character(LEN=112) :: file1,ofile
  ! ------------------------------------------------------------------------   . autres
  type(date_sec) :: tpsref
  real(KIND=wr) :: ttp,tts,deltatps
  real(KIND=wr) :: AmpliTot,AmpliMin,AmpliMax,AmpliNORM
  real(KIND=wr) :: ML,moyP,ecP,moyS
  integer(KIND=wi) :: i, n, jD, numero,nP, nS
  character (LEN=5) :: nbseisme
  logical :: existe1
  ! ------------------------------------------------------------------------   . lecture du fichier sac
  NN = IARGC()
  if ((NN > 2).or.(NN < 2)) then
    write(*,'(a)') "usage:  sac_coda sacfile"
    write(*,'(a)') "        sacfile - input sac file"
    write(*,'(a)') "        ofile - output xy file"
    stop
  endif
  call GETARG(1, file1)
  open(UNIT=1,FILE=file1,STATUS='OLD',IOSTAT=ios)
  if (ios > 0) then
```

```
    write(*,*) "ERROR - Input file: '", TRIM(adjustl(file1)), "' does not exist ..."
    close(1)
    stop
  endif
  close(1)
  call GETARG(2, ofile)
  ! ------------------------------------------------------------------------     . lecture des données : numéro du séisme, Pg et Sg, d_épi
  read(*,*)numero,pick%tpsR,tpsref,pick%depi,pick%dhypo,AmpliNORM
  tps0%date=tpsref%date
  tps0%secP=tpsref%sec
  tps0%secS=0.0_wr
  call basetime(tps0)
  call basetime(pick%tpsR)
  ! ------------------------------------------------------------------------     .
  if (pick%depi.lt.dmax) then                                                    ! distance maximale
    ! ------------------------------------------------------------------------     .
    open(UNIT=2,FILE=ofile,STATUS='REPLACE',IOSTAT=ios)
    if (ios > 0) then
      write(*,*) "ERROR - Input file: '", TRIM(adjustl(ofile)), "' does not exist ..."
      stop
    endif
    CALL rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,        &
      t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,   &
      resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,   &
      user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2, &
      internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,   &
      unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,  &
      nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,  &
      iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,  &
      imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
      unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
      kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
      kinst,sacfile)
    if (nvhdr /= 6) then
      write(*,*) "ERROR - File: '", TRIM(adjustl(file1)), "' appears to be of non-native &
      &byte-order or is not a SAC file."
      stop
    endif
    pick%sta%staname=(TRIM(adjustl(kstnm))//'000')                               ! nom de la station
    ! ------------------------------------------------------------------------     .
    ! ------------------------------------------------------------------------     . normalisation amplitude
    AmpliMin=1.e9_wr
    AmpliMax=-1.e9_wr
    do j=0,(npts1-1)
      sacfile(j+1)=sqrt(sacfile(j+1))
      if (AmpliMin.gt.real(sacfile(j+1),wr)) AmpliMin=real(sacfile(j+1),wr)
      if (AmpliMax.lt.real(sacfile(j+1),wr)) AmpliMax=real(sacfile(j+1),wr)
    enddo
    AmpliTot=max(abs(AmpliMin),abs(AmpliMax))
    do j=0,(npts1-1)
      sacfile(j+1)=sacfile(j+1)*real(AmpliNORM/AmpliTot,4)
    enddo
    ! --------                                                  ----------     . temps initial
    Adata%tpsR%date%year=int(nzyear,4)
    ! --------                                                ----------     . jour julien
    Adata%tpsR%date%month=1
    Adata%tpsR%date%day=1
    call jDATE (jD,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
    call GDATE (jD+int(nzjday,4)-1,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
    ! --------                                                ----------     .
    Adata%tpsR%date%hour=int(nzhour,4)
    Adata%tpsR%date%min=int(nzmin,4)
    Adata%tpsR%secP=real(nzsec,wr)+real(b1,wr)
    Adata%tpsR%secS=0.0_wr
    call jDATE (Adata%tpsR%date%jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
```

```
121     call basetime(Adata%tpsR)
122     ! ------------------------------------------------------------------     . moyenne du bruit, avant la P
123     moyP=0.0_wr
124     nP=0
125     ref=Adata
126     ref%tpsR%secP = ref%tpsR%secP - real(delta1,wr)
127     do j=0,(npts1-1)
128       ref%tpsR%secP = ref%tpsR%secP + real(delta1,wr)
129       call basetime(ref%tpsR)
130       call difftime(ttp,tts,pick%tpsR,ref%tpsR)
131       if ((j.gt.int(35.0_wr/real(delta1,wr),wi)).and.(ttp.ge.1.0_wr).and.(ttp.le.600.0_wr)) then
132         ! 35 sec apres début signal (taper) ; au moins 1 sec . avant la P (Pg ou Pn) ; moins de 600 secondes avant la P
133         np=nP+1
134         moyP=moyP+real(abs(sacfile(j+1)),wr)
135         endif
136     enddo
137     if (nP.gt.(100)) then
138       moyP=moyP/real(nP,wr)
139       ! ------------------------------------------------------------------     . ecartype sur la moyenne du bruit, avant la P
140       nP=0
141       ecP=0.0_wr
142       ref=Adata
143       ref%tpsR%secP = ref%tpsR%secP - real(delta1,wr)
144       do j=0,(npts1-1)
145         ref%tpsR%secP = ref%tpsR%secP + real(delta1,wr)
146         call basetime(ref%tpsR)
147         call difftime(ttp,tts,pick%tpsR,ref%tpsR)
148         if ((j.gt.int(35.0_wr/real(delta1,wr),wi)).and.(ttp.ge.1.0_wr).and.(ttp.le.600.0_wr)) then
149           ! 35 sec apres début signal (taper) ; au moins 1 sec . avant la P (Pg ou Pn) ; moins de 600 secondes avant la P
150           np=nP+1
151           ecP=ecP+(moyP-real(abs(sacfile(j+1)),wr))**2.0_wr
152         endif
153       enddo
154       ecP=sqrt(ecP/real(nP,wr))
155       ! ------------------------------------------------------------------     . moyenne glissante sur la coda
156       nS=0
157       ref=Adata
158       ref%tpsR%secP = ref%tpsR%secP - real(delta1,wr)
159       boucles : do j=0,(npts1-50)
160         ref%tpsR%secP = ref%tpsR%secP + real(delta1,wr)
161         ref%tpsR%secS = ref%tpsR%secP
162         call basetime(ref%tpsR)
163         call difftime(ttp,tts,pick%tpsR,ref%tpsR)
164         if ((ttp.le.0.0_wr).and.(j.gt.int(35.0_wr/real(delta1,wr),wi)))then ! après la première P et 35 sec apres le début du signal (taper)
165           ns=ns+1
166           moyS=0.0_wr
167           n=0
168           ! moyenne sur les 10 sec d'avant
169           do i=-int(10.0_wr/real(delta1,wr),wi),0
170             n=n+1
171             moyS=moyS+real(abs(sacfile(j+1+i)),wr)
172           enddo
173           moyS=moyS/real(n,wr)
174           call difftime(deltatps,Ml,ref%tpsR,tps0)
175           write(2,*) pick%dhypo+moyS,deltatps
176           if ((tts.le.0.0_wr).and.(moyS.lt.(2.0_wr*(moyP+0.0_wr*ecP)))) then ! après la première S
177             ! magnitude durée (Md), la fin de la coda = la moitié du bruit avant le séisme, cf :
178             ! [Kayal, j.R. (2008) : Microearthquake Seismology and Seismotectonics of South Asia , springer (§3.15)]
179             exit boucles
180           endif
181         endif
182       enddo boucles
183       write(2,*)
184       close(2)
185       ! ------------------------------------------------------------------     .
```

```fortran
      if ((ns.gt.int(1.0_wr/real(delta1,wr),wi))) then                   ! au moins 1.0 sec de signal
        write(nbseisme(1:5),'(i5)')numero
          inquire (file="OUTPUT/files/mag-"//trim(adjustl(nbseisme))//".d",exist=existe1)
          if (existe1) then
            open(UNIT=101,FILE="OUTPUT/files/mag-"//trim(adjustl(nbseisme))//".d", &
            status="old",position="append",action="write")
          else
            open(UNIT=101,FILE="OUTPUT/files/mag-"//trim(adjustl(nbseisme))//".d",STATUS='new')
          endif
          ! formule de Lee et al. (1972)
          Ml = -0.87_wr + 2.0_wr*log10(real(ns,wr)*real(delta1,wr)) + 0.0035_wr*pick%depi
          write(101,10011)kstnm,Ml,real(ns,wr)*real(delta1,wr),pick%depi
          close(101)
      endif
      ! -----------------------------------------------------------------    .
    endif
    ! -------------------------------------------------------------------    .
  endif
  ! ---------------------------------------------------------------------    .
10011 format (a4,3(1x,f19.6))
  ! ---------------------------------------------------------------------    .

end program coda

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! *********************************************************************    .
```

1.11 SRC/PROG/sac_readpick.f90

```fortran
! programme principal sac_readpick
! *********************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                   ---------    .
! -------- readpick : lit les fichiers sac avec les pick.sh   ---------    .
! -------- et ecrit un fichier (phases list) lisible par le programme CHE    .
! --------                                                   ---------    .
! -------- septembre 2014                                    ---------    .
! --------                                                   ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr      ---------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope       !
!  that it will be useful however without any warranties.                  !
!  Use it on your own risk. Please, report bugs/improvements/... .         !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                          !
!
program readpick
  ! ---------------------------------------------------------------------    .
  use modparam
  use typetemps
  use time
  use sac_i_o
  ! --------                                                   ---------    .
  implicit none
  ! ----------------------------------------------------------------------    . pour SAC
  real(KIND=4), dimension(:), allocatable :: sacfile
  real(KIND=4) :: delta1, b1, e1
  integer(KIND=4) :: NN, npts1
  character(LEN=112) :: file1
  ! ----------------------------------------------------------------------    . autres
  integer(KIND=wi) :: JD
  type(dataone) :: Adata
  ! ---------------------------------------------------------------------    . lecture du fichier sac
  NN = IARGC()
```

```
37  if ((NN < 1).or.(NN > 1)) then
38    write(*,'(a)') "usage:  ./readpick.exe sacfile"
39    write(*,'(a)') "        sacfile - input sac file"
40    stop
41  endif
42  call GETARG(1, file1)
43  ! ------------------------------------------------------------------      .
44  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,          &
45    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
46    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
47    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
48    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
49    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
50    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
51    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
52    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
53    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
54    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
55    kinst,sacfile)
56  ! ------------------------------------------------------------------      .
57  if (nvhdr /= 6) then
58    write(*,*) "ERROR - File: '", TRIM(adjustl(file1)), "' appears to be of non-native &
59    &byte-order or is not a SAC file."
60    stop
61  endif
62  ! ------------------------------------------------------------------      .
63  Adata%sta%staname=(TRIM(adjustl(kstnm))//'000')                              ! nom de la station
64  ! ------------------------------------------------------------------      . initialise
65  ! si phase est près du début :  10 sec
66  if (T1.le.(b1+10.0_4)) T1=-12345.0_4
67  if (T2.le.(b1+10.0_4)) T2=-12345.0_4
68  if (T3.le.(b1+10.0_4)) T3=-12345.0_4
69  if (T4.le.(b1+10.0_4)) T4=-12345.0_4
70  if (T5.le.(b1+10.0_4)) T5=-12345.0_4
71  if (T6.le.(b1+10.0_4)) T6=-12345.0_4
72  if (T7.le.(b1+10.0_4)) T7=-12345.0_4
73  if (T8.le.(b1+10.0_4)) T8=-12345.0_4
74  T0=-12345.0_4
75  a=-12345.0_4
76  o=-12345.0_4
77  T9=-12345.0_4
78  ! ------------------------------------------------------------------      .
79  ! ONDES DIRECTES                                                         .
80  ! ------------------------------------------------------------------      .
81  ! T1 : temps arrivée de l'onde Pg
82  ! T2 : incertitudes absolue sur l'onde Pg                                 ! -> calcul du delta après
83  ! ------------------------------------------------------------------      .
84  ! T3 : temps arrivée de l'onde Sg
85  ! T4 : incertitudes absolue sur l'onde Sg
86  ! ------------------------------------------------------------------      .
87  if ((real(T1,wr).ne.-12345.0_wr).and.(real(T2,wr).ne.-12345.0_wr)) then
88    Adata%typeonde='G'
89    Adata%coefP=0 ! modification a posteriori si necessaire
90    Adata%coefS=0
91    ! -------                                                 ---------     . date
92    Adata%tpsR%date%year=int(nzyear,wi)-2000
93    ! -------                                                 ---------     . jour julien
94    Adata%tpsR%date%month=1
95    Adata%tpsR%date%day=1
96    call JDATE (JD,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
97    call GDATE (JD+int(nzjday,wi)-1, Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
98    ! -------                                                 ---------     .
99    Adata%tpsR%date%hour=nzhour
100   Adata%tpsR%date%min=nzmin
101   Adata%tpsR%secP=real(nzsec,wr)+ real(T1,wr) + real(nzmsec,wr)/1000.0_wr
```

```
    ! ------                                                    --------     .
    Adata%sigP=max(abs(real(T1-T2,wr)),real(delta1,wr)/2.0_wr)
    ! ------------------------------------------------------------------     .
    if ((real(T3,wr).ne.-12345.0_wr).and.(real(T4,wr).ne.-12345.0_wr)) then
      Adata%andS='S'
      Adata%tpsR%secS=real(nzsec,wr)+ real(T3,wr) + real(nzmsec,wr)/1000.0_wr
    else
      Adata%andS='X'
      Adata%tpsR%secS=0.0_wr
    endif
    ! ------                                                    --------     .
    Adata%sigS=max(abs(real(T3-T4,wr)),real(delta1,wr)/2.0_wr)
    ! ------                                                    --------     .
    call basetime(Adata%tpsR)                                              ! respect de la base 60, 24, ...
    ! ------                                                    --------     .
    if (Adata%andS=='S') then
      write(*,1000)Adata%sta%staname,'P',Adata%typeonde,Adata%coefP,Adata%tpsR%date%year, &
        Adata%tpsR%date%month,Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min, &
        Adata%tpsR%secP,Adata%tpsR%secS,Adata%andS,Adata%coefS,Adata%sigP,Adata%sigS
    else
      write(*,1001)Adata%sta%staname,'P',Adata%typeonde,Adata%coefP,Adata%tpsR%date%year, &
        Adata%tpsR%date%month,Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min, &
        Adata%tpsR%secP,Adata%sigP
    endif
  endif
  ! ------------------------------------------------------------------     .
  ! ONDES REFRACTEES                                                       .
  ! ------------------------------------------------------------------     .
  ! T5 : temps arrivée de l'onde Pn
  ! T6 : incertitudes absolue sur l'onde Pn
  ! ------------------------------------------------------------------     .
  ! T7 : temps arrivée de l'onde Sn
  ! T8 : incertitudes absolue sur l'onde Sn
  ! ------------------------------------------------------------------     .
  if ((real(T5,wr).ne.-12345.0_wr).and.(real(T6,wr).ne.-12345.0_wr)) then
    Adata%typeonde='N'
    Adata%coefP=0 ! modification a posteriori si necessaire
    Adata%coefS=0
    ! ------                                                    --------     . date
    Adata%tpsR%date%year=int(nzyear,wi)-2000
    ! ------                                                    --------     . jour julien
    Adata%tpsR%date%month=1
    Adata%tpsR%date%day=1
    call JDATE (JD,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
    call GDATE (JD+int(nzjday,wi)-1, Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
    ! ------                                                    --------     .
    Adata%tpsR%date%hour=nzhour
    Adata%tpsR%date%min=nzmin
    Adata%tpsR%secP=real(nzsec,wr)+ real(T5,wr) + real(nzmsec,wr)/1000.0_wr
    ! ------                                                    --------     .
    Adata%sigP=max(abs(real(T5-T6,wr)),real(delta1,wr)/2.0_wr)
    ! ------------------------------------------------------------------     .
    if ((real(T7,wr).ne.-12345.0_wr).and.(real(T8,wr).ne.-12345.0_wr)) then
      Adata%andS='S'
      Adata%tpsR%secS=real(nzsec,wr)+ real(T7,wr) + real(nzmsec,wr)/1000.0_wr
    else
      Adata%andS='X'
      Adata%tpsR%secS=0.0_wr
    endif
    ! ------                                                    --------     .
    Adata%sigS=max(abs(real(T7-T8,wr)),real(delta1,wr)/2.0_wr)
    ! ------                                                    --------     .
    call basetime(Adata%tpsR)                                              ! respect de la base 60, 24, ...
    ! ------                                                    --------     .
    if (Adata%andS=='S') then
```

```
      write(*,1000)Adata%sta%staname,'P',Adata%typeonde,Adata%coefP,Adata%tpsR%date%year, &
        Adata%tpsR%date%month,Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min, &
        Adata%tpsR%secP,Adata%tpsR%secS,Adata%andS,Adata%coefS,Adata%sigP,Adata%sigS
    else
      write(*,1001)Adata%sta%staname,'P',Adata%typeonde,Adata%coefP,Adata%tpsR%date%year, &
        Adata%tpsR%date%month,Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min, &
        Adata%tpsR%secP,Adata%sigP
    endif
  endif
  ! -----------------------------------------------------------------    . format de lecture des données
1000 format(a4,2a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
1001 format(a4,2a1,1x,i1,1x,5i2.2,f6.3,18x,f6.3)
  ! -----------------------------------------------------------------    .

end program readpick

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! *******************************************************************    .
```

1.12 SRC/PROG/sac_writepick.f90

```
! programme principal sac_writepick
! *******************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------                                                   --------    .
! ------- writepick : lit les fichiers (phases list) lisible --------    .
! ------- par le programme CHE et ecrit un fichier           --------    .
! ------- sac avec pick.sh                                   --------    .
! -------                                                   --------    .
! ------- fevrier 2015                                       --------    .
! -------                                                   --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! ------- Méric Haugmard meric.haugmard@univ-nantes.fr       --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope    !
!  that it will be useful however without any warranties.              !
!  Use it on your own risk. Please, report bugs/improvements/... .     !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                        !
!
program writepick
  ! -----------------------------------------------------------------    .
  use modparam
  use typetemps
  use time
  use sac_i_o
  ! -------                                                 --------    .
  implicit none
  ! ------------------------------------------------------------------   . pour SAC
  real(KIND=4), dimension(:), allocatable :: sacfile
  real(KIND=4) :: delta1, b1, e1
  integer(KIND=4) :: NN, npts1
  character(LEN=4) :: astation
  character(LEN=112) :: file1
  character(LEN=112) :: fileppk
  ! ------------------------------------------------------------------   . autres
  real(KIND=wr) :: alpha,beta
  integer(KIND=wi) :: JD, ok
  type(dataone) :: Adata
  ! -----------------------------------------------------------------    . lecture
  NN = IARGC()
  if ((NN < 2).or.(NN > 2)) then
    write(*,'(a)') "usage:  ./readpick.exe sacfile ppkdatafile"
    write(*,'(a)') "        sacfile - input sac file (.bin)"
    write(*,'(a)') "        ppkdatafile - input pick file (.txt)"
```

```
    stop
  endif
  call GETARG(1, file1)
  call GETARG(2, fileppk)
  ! -------                                                    --------    . lecture du fichier sac
  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,          &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacfile)
  if (nvhdr /= 6) then
    write(*,*) "ERROR 1 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
    stop
  endif

  ! ------------------------------------------------------------------    . temps du début de la trace, -> tpsTh <-
  Adata%tpsTh%date%year=int(nzyear,wi)
  ! -------                                                    --------    . jour julien
  Adata%tpsTh%date%month=1
  Adata%tpsTh%date%day=1
  call JDATE (JD,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
  call GDATE (JD+int(nzjday,wi)-1, Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
  call JDATE (Adata%tpsTh%date%Jday,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
  ! -------                                                    --------    .
  Adata%tpsTh%date%hour=nzhour
  Adata%tpsTh%date%min=nzmin
  Adata%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
  Adata%tpsTh%secS=Adata%tpsTh%secP
  call basetime(Adata%tpsTh)
  ! ------------------------------------------------------------------    . initialise
  T0=-12345.0_4
  ! ------------------------------------------------------------------    . lecture du fichier données picks, -> tpsR <-
  ok=0
  open(111, FILE = TRIM(adjustl(fileppk)),status='old',iostat = ok)
  if (ok .ne. 0) then
    write(*,*)'problème : le fichier '//TRIM(adjustl(fileppk))//' n''existe pas '
    stop
  endif
  ! -------                                                    --------    .
  read(111,1000,iostat = ok)Adata%sta%staname,Adata%typeonde,              &
    Adata%coefP,Adata%tpsR%date%year,Adata%tpsR%date%month,                &
    Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min,          &
    Adata%tpsR%secP,Adata%tpsR%secS,Adata%andS,Adata%coefS,                &
    Adata%sigP,Adata%sigS
  if (ok .ne. 0) then
    write(*,*)'fichier '//TRIM(adjustl(fileppk))//' vide ...'
    stop
  endif
  close(111)
  ! -------                                                    --------    .
  if((Adata%sigP.lt.0.0_wr).and.(IsNaN(Adata%sigP))) then
    write(*,*)'problème dans lectdata : les incertitudes sur les données P n''existent pas ',Adata%sigP
    stop
  endif
  if ((Adata%andS.ne."S").and.(Adata%sigS.lt.0.0_wr).and.(IsNaN(Adata%sigS))) then
    write(*,*)'problème dans lectdata : les incertitudes sur les données S n''existent pas ',Adata%sigS
    stop
  endif
```

```
110 ! -------                                                    --------    .
111 Adata%tpsR%date%year=Adata%tpsR%date%year+2000
112 ! -------                                                    --------    . respect du decoupage des années en mois et jours avec prise en compte des années
        bisextiles
113 call basetime(Adata%tpsR)
114 call JDATE(Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
115 call GDATE (Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
116 call basetime(Adata%tpsR)                                                ! respect du decoupage temps dans la base composite 60/12/365 ...
117 ! -------                                                    --------    . verif station
118 astation=TRIM(adjustl(kstnm))//'000'
119 if (Adata%sta%staname.ne.astation) then
120   write(*,*)'problème : les stations sont différentes : ',Adata%sta%staname,(TRIM(adjustl(kstnm))//'000')
121   stop
122 endif
123 ! ---------------------------------------------------------------    .
124 call difftime(alpha,beta,Adata%tpsR,Adata%tpsTh)
125 ! ---------------------------------------------------------------    .
126 ! ONDES DIRECTES                                                    .
127 ! ---------------------------------------------------------------    .
128 ! T1 : temps arrivée de l'onde Pg
129 ! T2 : incertitudes absolue sur l'onde Pg                           ! -> calcul du delta après
130 ! ---------------------------------------------------------------    .
131 ! T3 : temps arrivée de l'onde Sg
132 ! T4 : incertitudes absolue sur l'onde Sg
133 ! ---------------------------------------------------------------    .
134 if (Adata%typeonde.eq.'G') then
135   if(Adata%andS.eq.'S') then
136     T1=real(alpha,4)
137     T2=T1+real(Adata%sigP,4)
138     T3=real(beta,4)
139     T4=T3+real(Adata%sigS,4)
140     if (.not.((T3.gt.(0.0_4+b1)).and.(T4.lt.(real(npts1,4)*delta1+b1)))) then
141       T3=-12345.0_4
142       T4=-12345.0_4
143     endif
144     if (.not.((T4.gt.(0.0_4+b1)).and.(T5.lt.(real(npts1,4)*delta1+b1)))) then
145       T3=-12345.0_4
146       T4=-12345.0_4
147     endif
148   else
149     T1=real(alpha,4)
150     T2=T1+real(Adata%sigP,4)
151   endif
152   if (.not.((T1.gt.(0.0_4+b1)).and.(T2.lt.(real(npts1,4)*delta1+b1)))) then
153     T1=-12345.0_4
154     T2=-12345.0_4
155     T3=-12345.0_4
156     T4=-12345.0_4
157   endif
158   if (.not.((T2.gt.(0.0_4+b1)).and.(T3.lt.(real(npts1,4)*delta1+b1)))) then
159     T1=-12345.0_4
160     T2=-12345.0_4
161     T3=-12345.0_4
162     T4=-12345.0_4
163   endif
164   ! ---------------------------------------------------------------    .
165   ! ONDES REFRACTEES                                                  .
166   ! ---------------------------------------------------------------    .
167   ! T5 : temps arrivée de l'onde Pn
168   ! T6 : incertitudes absolue sur l'onde Pn
169   ! ---------------------------------------------------------------    .
170   ! T7 : temps arrivée de l'onde Sn
171   ! T8 : incertitudes absolue sur l'onde Sn
172   ! ---------------------------------------------------------------    .
173 elseif (Adata%typeonde.eq.'N') then                                    !
```

```fortran
    if(Adata%andS.eq.'S') then
      T5=real(alpha,4)
      T6=T5+real(Adata%sigP,4)
      T7=real(beta,4)
      T8=T7+real(Adata%sigS,4)
      if (.not.((T7.gt.(0.0_4+b1)).and.(T8.lt.(real(npts1,4)*delta1+b1)))) then
        T7=-12345.0_4
        T8=-12345.0_4
      endif
      if (.not.((T8.gt.(0.0_4+b1)).and.(T9.lt.(real(npts1,4)*delta1+b1)))) then
        T8=-12345.0_4
        T7=-12345.0_4
      endif
    else
      T5=real(alpha,4)
      T6=T5+real(Adata%sigP,4)
    endif
    if (.not.((T5.gt.(0.0_4+b1)).and.(T6.lt.(real(npts1,4)*delta1+b1)))) then
      T5=-12345.0_4
      T6=-12345.0_4
      T8=-12345.0_4
      T7=-12345.0_4
    endif
    if (.not.((T6.gt.(0.0_4+b1)).and.(T7.lt.(real(npts1,4)*delta1+b1)))) then
      T6=-12345.0_4
      T5=-12345.0_4
      T8=-12345.0_4
      T7=-12345.0_4
    endif
  ! ------------------------------------------------------------------------      .
  else
    write(*,*)'problème : onde ni directe ni réfractée ... ? ', Adata%typeonde
    stop
  endif
  ! ------------------------------------------------------------------------      .
  ! ------------------------------------------------------------------------      .
  call wbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,        &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,   &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,   &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2, &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,   &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,  &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,  &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,  &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,      &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacfile)
  ! ------------------------------------------------------------------------      .
  if (nvhdr /= 6) then
    write(*,*) "ERROR 2 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
    stop
  endif
  ! ------------------------------------------------------------------------      . format de lecture des données
1000 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
  ! ------------------------------------------------------------------------      .

end program writepick

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@      .
! ************************************************************************      .
```

1.13 SRC/PROG/sac_writepickTheo.f90

```fortran
! programme principal sac_writepick
```

```
2  ! ************************************************************************    .mh
3  ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  .
4  ! --------                                                        ---------   .
5  ! --------  writepick : lit les fichiers (phases list) lisible    ---------   .
6  ! --------  par le programme CHE et ecrit un fichier              ---------   .
7  ! --------  sac avec pick.sh                                      ---------   .
8  ! --------                                                        ---------   .
9  ! --------  fevrier 2015                                          ---------   .
10 ! --------                                                        ---------   .
11 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  .
12 ! --------  Méric Haugmard meric.haugmard@univ-nantes.fr          ---------   .
13 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  .
14 !  This program is distributed for research purposes and in the hope        !
15 !  that it will be useful however without any warranties.                   !
16 !  Use it on your own risk. Please, report bugs/improvements/... .          !
17 ! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@  .
18 !                                                                          !
19 !
20 program writepickTheo
21   ! ------------------------------------------------------------------     .
22   use modparam
23   use typetemps
24   use time
25   use sac_i_o
26   ! --------                                                    ---------   .
27   implicit none
28   ! ------------------------------------------------------------------     . pour SAC
29   real(KIND=4), dimension(:), allocatable :: sacfile
30   real(KIND=4) :: delta1, b1, e1
31   integer(KIND=4) :: NN, npts1
32   character(LEN=4) :: astation
33   character(LEN=112) :: file1
34   character(LEN=112) :: fileppk
35   ! ------------------------------------------------------------------     . autres
36   real(KIND=wr) :: alpha,beta
37   integer(KIND=wi) :: JD, ok
38   type(dataone) :: Adata
39   ! ------------------------------------------------------------------     . lecture
40   NN = IARGC()
41   if ((NN < 2).or.(NN > 2)) then
42     write(*,'(a)') "usage:  ./readpick.exe sacfile ppkdatafile"
43     write(*,'(a)') "        sacfile - input sac file (.bin)"
44     write(*,'(a)') "        ppkdatafile - input pick file (.txt)"
45     stop
46   endif
47   call GETARG(1, file1)
48   call GETARG(2, fileppk)
49   ! --------                                                    ---------   . lecture du fichier sac
50   call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,        &
51     t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,   &
52     resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,   &
53     user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2, &
54     internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,   &
55     unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,  &
56     nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,  &
57     iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,  &
58     imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
59     unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
60     kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
61     kinst,sacfile)
62   if (nvhdr /= 6) then
63     write(*,*) "ERROR 1 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
64     stop
65   endif
66
```

```
! ------------------------------------------------------------------------     . temps du début de la trace, -> tpsTh <-
Adata%tpsTh%date%year=int(nzyear,wi)
! --------                                                  ---------     . jour julien
Adata%tpsTh%date%month=1
Adata%tpsTh%date%day=1
call JDATE (JD,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
call GDATE (JD+int(nzjday,wi)-1, Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
call JDATE (Adata%tpsTh%date%Jday,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
! --------                                                  ---------     .
Adata%tpsTh%date%hour=nzhour
Adata%tpsTh%date%min=nzmin
Adata%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
Adata%tpsTh%secS=Adata%tpsTh%secP
call basetime(Adata%tpsTh)
! ------------------------------------------------------------------------     . initialise
T0=-12345.0_4
a=-12345.0_4
o=-12345.0_4
! ------------------------------------------------------------------------     . lecture du fichier données picks, -> tpsR <-
ok=0
open(111, FILE = TRIM(adjustl(fileppk)),status='old',iostat = ok)
if (ok .ne. 0) then
  write(*,*)'problème : le fichier '//TRIM(adjustl(fileppk))//' n''existe pas '
  stop
endif
! --------                                                  ---------     .
read(111,1000,iostat = ok)Adata%sta%staname,Adata%typeonde,            &
  Adata%coefP,Adata%tpsR%date%year,Adata%tpsR%date%month,              &
  Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min,        &
  Adata%tpsR%secP,Adata%tpsR%secS,Adata%andS,Adata%coefS,              &
  Adata%sigP,Adata%sigS
if (ok .ne. 0) then
  write(*,*)'fichier '//TRIM(adjustl(fileppk))//' vide ...'
  stop
endif
close(111)
! --------                                                  ---------     .
if((Adata%sigP.lt.0.0_wr).and.(IsNaN(Adata%sigP))) then
  write(*,*)'problème dans lectdata : les incertitudes sur les données P n''existent pas ',Adata%sigP
  stop
endif
if ((Adata%andS.ne."S").and.(Adata%sigS.lt.0.0_wr).and.(IsNaN(Adata%sigS))) then
  write(*,*)'problème dans lectdata : les incertitudes sur les données S n''existent pas ',Adata%sigS
  stop
endif
! --------                                                  ---------     .
Adata%tpsR%date%year=Adata%tpsR%date%year+2000
! --------                                                  ---------     . respect du decoupage des années en mois et jours avec prise en compte des années
    bisextiles
call basetime(Adata%tpsR)
call JDATE(Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
call GDATE (Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
call basetime(Adata%tpsR)                                                    ! respect du decoupage temps dans la base composite 60/12/365 ...
! --------                                                  ---------     . verif station
astation=TRIM(adjustl(kstnm))//'000'
if (Adata%sta%staname.ne.astation) then
  write(*,*)'problème : les stations sont différentes : ',Adata%sta%staname,(TRIM(adjustl(kstnm))//'000')
  stop
endif
! ------------------------------------------------------------------------     .
call difftime(alpha,beta,Adata%tpsR,Adata%tpsTh)
! ------------------------------------------------------------------------     .
! ONDES DIRECTES THEORIQUES                                                    .
! ------------------------------------------------------------------------     .
if (Adata%typeonde.eq.'G') then
```

```fortran
    if(Adata%andS.eq.'S') then
      a=real(alpha,4)
      o=real(beta,4)
      if (.not.(o.gt.(0.0_4+b1))) then
        o=-12345.0_4
      endif
    else
      a=real(alpha,4)
    endif
    if (.not.(a.gt.(0.0_4+b1))) then
      a=-12345.0_4
      o=-12345.0_4
    endif
  endif
  ! ----------------------------------------------------------------------    .
  ! ----------------------------------------------------------------------    .
  call wbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,          &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,     &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,     &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,   &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,     &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,    &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,    &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,    &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacfile)
  ! ----------------------------------------------------------------------    .
  if (nvhdr /= 6) then
    write(*,*) "ERROR 2 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
    stop
  endif
  ! ----------------------------------------------------------------------    . format de lecture des données
1000 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
  ! ----------------------------------------------------------------------    .

end program writepickTheo

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! **********************************************************************    .
```

1.14 SRC/PROG/sac_writepickCata.f90

```fortran
! programme principal sac_writepick
! **********************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                      --------    .
! -------- writepick : lit le catalogue                         --------    .
! -------- inscrit les pointés théoriques dans les fichiers sac --------    .
! -------- sac avec pick.sh                                     --------    .
! --------                                                      --------    .
! -------- fevrier 2015                                         --------    .
! --------                                                      --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr         --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!  This program is distributed for research purposes and in the hope        !
!  that it will be useful however without any warranties.                   !
!  Use it on your own risk. Please, report bugs/improvements/... .          !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                           !
!
program writepickCata
  ! ----------------------------------------------------------------------    .
```

```
22  use modparam
23  use typetemps
24  use time
25  use sac_i_o
26  use datalecture
27  use pb_direct
28  ! -------                                                    --------    .
29  implicit none
30  ! ------------------------------------------------------------------    . pour SAC
31  real(KIND=4), dimension(:), allocatable :: sacfile
32  real(KIND=4) :: delta1, b1, e1
33  integer(KIND=4) :: NN, npts1
34  character(LEN=112) :: file1
35  ! ------------------------------------------------------------------    .
36  type(parametre) :: param                                               ! paramètres théoriques du séisme
37  type(seismes) :: theseisme(2)                                          ! paramètres du catalogue
38  integer(KIND=wi) :: find                                               ! ce séisme est présent dans le catalogue, 0-1-2 fois
39  integer(KIND=wi) :: i,j,count,nbsta,nbtps(nbseismes)
40  type(stations), allocatable, dimension(:) :: datasta
41  type(dataone) :: datatemps                                             ! données
42  logical :: critique                                                    ! .true. si distance hypo + 5 km < distance hypo critique pour la réfraction
43  type(amoho_centroid) :: acentroid                                       ! si moho non tabulaire
44  ! ------------------------------------------------------------------    . autres
45  real(KIND=wr) :: alpha,beta
46  integer(KIND=wi) :: JD, ok
47  type(dataone) :: Adata
48  ! ------------------------------------------------------------------    . lecture
49  NN = IARGC()
50  if ((NN < 1).or.(NN > 1)) then
51    write(*,'(a)') "usage:  ./readpick.exe sacfile ppkdatafile"
52    write(*,'(a)') "        sacfile - input sac file (.bin)"
53    stop
54  endif
55  call GETARG(1, file1)
56  ! -------                                                    --------    . lecture du fichier sac
57  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,        &
58    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,   &
59    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,   &
60    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2, &
61    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,   &
62    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,  &
63    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,  &
64    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,  &
65    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
66    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
67    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
68    kinst,sacfile)
69  if (nvhdr /= 6) then
70    write(*,*) "ERROR 1 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
71    stop
72  endif
73  ! -------- si moho non tabulaire                                 --------    .
74  acentroid%lonC=moho_lon ; acentroid%latC=moho_lat
75  acentroid%NS=moho_NS ; acentroid%EO=moho_EO
76  call alph2vect(acentroid); call vect2alph(acentroid)
77  ! ------------------------------------------------------------------    . temps du début de la trace, -> tpsTh <-
78  Adata%tpsTh%date%year=int(nzyear,wi)
79  ! -------                                                    --------    . jour julien
80  Adata%tpsTh%date%month=1
81  Adata%tpsTh%date%day=1
82  call JDATE (JD,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
83  call GDATE (JD+int(nzjday,wi)-1, Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
84  call JDATE (Adata%tpsTh%date%Jday,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
85  ! -------                                                    --------    .
86  Adata%tpsTh%date%hour=nzhour
```

```
87  Adata%tpsTh%date%min=nzmin
88  Adata%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
89  Adata%tpsTh%secS=Adata%tpsTh%secP
90  call basetime(Adata%tpsTh)
91  ! ------------------------------------------------------------------------     . initialise
92  T0=-12345.0_4
93  a=-12345.0_4
94  o=-12345.0_4
95  ! ------------------------------------------------------------------------     . lecture du fichier données picks, -> tpsR <-
96  Adata%sta%staname=TRIM(adjustl(kstnm))//'000'
97  ! ------------------------------------------------------------------------     .
98  ! lecture du catalogue
99  ! ------------------------------------------------------------------------     . d'après la géol :
100 param%VC=6.0_wr
101 param%VM=8.0_wr
102 param%Zmoho=30.0_wr
103 param%VpVs=1.71_wr
104 ! ------------------------------------------------------------------------     . au pif :
105 param%Zhypo=5.0_wr
106 param%lon=-2.0_wr
107 param%lat=47.5_wr
108 ! ------------------------------------------------------------------------     .
109 param%Tzero%date=Adata%tpsTh%date
110 param%Tzero%sec=Adata%tpsTh%secP+60.0_wr                                       ! la trace commence 60 s avant le séisme du catalogue
111 call basetime(param%Tzero)
112 ! -------                                                  ---------    .
113 call catalogue(param,theseisme,find)
114 if (find.lt.1) then                                                          ! deux séismes max !
115   write(*,*)'problème dans writepick : séismes au catalogue'
116   stop
117 elseif (find.gt.2) then
118   write(*,*)'attention dans writepick : plusieurs séismes au catalogue',find
119   write(*,*)theseisme(1)
120   write(*,*)theseisme(2)
121 endif
122 ! -------                                                  ---------    .
123 param%Zhypo=theseisme(1)%pfd
124 param%lon=theseisme(1)%lon
125 param%lat=theseisme(1)%lat
126 param%Tzero%date = theseisme(1)%tps_init%date
127 param%Tzero%sec = theseisme(1)%tps_init%sec
128 call basetime(param%Tzero)
129 ! ------------------------------------------------------------------------     .
130 ! lire la station
131 ! ------------------------------------------------------------------------     .
132 call lectnbdata(nbsta,nbtps)
133 allocate(datasta(nbsta))
134 ! -------                                                  ---------    .
135 open(503, FILE = 'DATA/sta.d',status='old',iostat = ok)
136 if (ok .ne. 0) then
137   open(503, FILE = 'sta.d',status='old',iostat = ok)
138   if (ok .ne. 0) then
139     write(*,*)'problème : le fichier DATA/sta.d n''existe pas '
140     stop
141   endif
142 endif
143 ! -------                                                  ---------    .
144 do i=1,nbsta
145   read(503,*,iostat = ok) datasta(i)
146 enddo
147 close(503)
148 ! -------                                                  ---------    .
149 count=0
150 do j=1,nbsta
151   if(Adata%sta%staname.eq.datasta(j)%staname) then
```

```
      datatemps%sta=datasta(j)                                          ! attribution d'une station
      count=count+1
    endif
  enddo
  ! -------                                                  --------    .
  if(count.gt.1) then                                                   ! vérification de l'absence de doublons
    write(*,*)'problème : station ',datatemps%sta%staname,' en double in file : DATA/sta.d'
    stop
  elseif(count.eq.0) then
    write(*,*)'problème : station ',datatemps%sta%staname,' non répertoriée'
    stop
  endif
  ! -------                                                  --------    .
  deallocate(datasta)
  ! ------------------------------------------------------------------    .
  ! distance épi, problème direct
  ! ------------------------------------------------------------------    .
  datatemps%typeonde='G'
  datatemps%andS='S'
  call tempsTheoDirectone(param,datatemps,critique,acentroid)
  ! ------------------------------------------------------------------    .
  ! tps arrivée, tpsR = Tzero (catalogue) + parcours
  ! ------------------------------------------------------------------    .
  Adata%tpsR%date=param%Tzero%date
  Adata%tpsR%secP=param%Tzero%sec+datatemps%tpsparcP
  Adata%tpsR%secS=param%Tzero%sec+datatemps%tpsparcS
  ! -------                                                  --------    . respect du decoupage des années en mois et jours avec prise en compte des années
      bisextiles
  call basetime(Adata%tpsR)
  call JDATE(Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
  call GDATE (Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
  call basetime(Adata%tpsR)                                             ! respect du decoupage temps dans la base composite 60/12/365 ...
  ! ------------------------------------------------------------------    .
  ! ondes P et S directes
  ! ------------------------------------------------------------------    .
  call difftime(alpha,beta,Adata%tpsR,Adata%tpsTh)
  ! ------------------------------------------------------------------    .
  ! ONDES DIRECTES THEORIQUES                                           .
  ! ------------------------------------------------------------------    .
  a=real(alpha,4)
  o=real(beta,4)
  if (.not.(o.gt.(0.0_4+b1))) then
    o=-12345.0_4
  endif
  ! -------                                                  --------    .
  if (.not.(a.gt.(0.0_4+b1))) then
    a=-12345.0_4
    o=-12345.0_4
  endif
  ! ------------------------------------------------------------------    .
  ! ------------------------------------------------------------------    .
  call wbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,          &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,     &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacfile)
  ! ------------------------------------------------------------------    .
  if (nvhdr /= 6) then
```

```fortran
    write(*,*) "ERROR 2 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
    stop
  endif
  ! ------------------------------------------------------------------   . format de lecture des données
1000 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
  ! ------------------------------------------------------------------   .

end program writepickCata

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! *********************************************************************   .
```

1.15 SRC/PROG/sac_spectre.f90

```fortran
! programme principal sac_spectre
! *********************************************************************   .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! --------                                                     --------   .
! -------- spectre : lit les fichiers (phases list) lisible    --------   .
! -------- par le programme CHE et trace les spectres          --------   .
! -------- autour des arrivées de Pg et Sg                     --------   .
! --------                                                     --------   .
! -------- septembre/octobre 2015                              --------   .
! --------                                                     --------   .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr        --------   .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
!  This program is distributed for research purposes and in the hope      !
!  that it will be useful however without any warranties.                 !
!  Use it on your own risk. Please, report bugs/improvements/... .        !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
!                                                                         !
!
program spectre
  ! ------------------------------------------------------------------   .
  use modparam
  use typetemps
  use time
  use sac_i_o
  ! --------                                                  --------   .
  implicit none
  ! -------------------------------------------------------------------   . pour SAC
  real(KIND=4), dimension(:), allocatable :: sacfile
  real(KIND=4) :: delta1, b1, e1
  integer(KIND=4) :: NN, npts1
  character(LEN=4) :: astation
  character(LEN=112) :: file1
  character(LEN=112) :: fileppk
  ! -------------------------------------------------------------------   . autres
  real(KIND=wr) :: alpha,beta
  integer(KIND=wi) :: JD, ok
  type(dataone) :: Adata
  ! -------------------------------------------------------------------   . fft
  integer :: i,j,iter,t,nn1,nn2
  double precision, allocatable :: spdata(:),temps(:),freq(:)
  double precision :: thedelta, amin, amax
  ! ------------------------------------------------------------------   .
  double precision :: plusoumoins=0.75
  logical :: plot=.false.
  ! ------------------------------------------------------------------   . lecture
  NN = IARGC()
  if ((NN < 2).or.(NN > 2)) then
    write(*,'(a)') "usage:  ./readpick.exe sacfile ppkdatafile"
    write(*,'(a)') "        sacfile - input sac file (.bin)"
    write(*,'(a)') "        ppkdatafile - input pick file (.txt)"
```

```
    stop
  endif
  call GETARG(1, file1)
  call GETARG(2, fileppk)
  ! -------                                                    --------    . lecture du fichier sac
  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,          &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacfile)
  if (nvhdr /= 6) then
    write(*,*) "ERROR 1 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
    stop
  endif
  ! ------------------------------------------------------------------    .
  thedelta=real(delta1)
  ! ------------------------------------------------------------------    . temps du début de la trace, -> tpsTh <-
  Adata%tpsTh%date%year=int(nzyear,wi)
  ! -------                                                    --------    . jour julien
  Adata%tpsTh%date%month=1
  Adata%tpsTh%date%day=1
  call JDATE (JD,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
  call GDATE (JD+int(nzjday,wi)-1, Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
  call JDATE (Adata%tpsTh%date%Jday,Adata%tpsTh%date%year,Adata%tpsTh%date%month,Adata%tpsTh%date%day)
  ! -------                                                    --------    .
  Adata%tpsTh%date%hour=nzhour
  Adata%tpsTh%date%min=nzmin
  Adata%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
  Adata%tpsTh%secS=Adata%tpsTh%secP
  call basetime(Adata%tpsTh)
  ! ------------------------------------------------------------------    . initialise
  T0=-12345.0_4
  ! ------------------------------------------------------------------    . lecture du fichier données picks, -> tpsR <-
  ok=0
  open(111, FILE = TRIM(adjustl(fileppk)),status='old',iostat = ok)
  if (ok .ne. 0) then
    write(*,*)'problème : le fichier '//TRIM(adjustl(fileppk))//' n''existe pas '
    stop
  endif
  ! -------                                                    --------    .
  read(111,1000,iostat = ok)Adata%sta%staname,Adata%typeonde,            &
    Adata%coefP,Adata%tpsR%date%year,Adata%tpsR%date%month,              &
    Adata%tpsR%date%day,Adata%tpsR%date%hour,Adata%tpsR%date%min,        &
    Adata%tpsR%secP,Adata%tpsR%secS,Adata%andS,Adata%coefS,              &
    Adata%sigP,Adata%sigS
  if (ok .ne. 0) then
    write(*,*)'fichier '//TRIM(adjustl(fileppk))//' vide ...'
    stop
  endif
  ! -------                                                    --------    .
  if((Adata%sigP.lt.0.0_wr).and.(IsNaN(Adata%sigP))) then
    write(*,*)'problème dans lectdata : les incertitudes sur les données P n''existent pas ',Adata%sigP
    stop
  endif
  if ((Adata%andS.ne."S").and.(Adata%sigS.lt.0.0_wr).and.(IsNaN(Adata%sigS))) then
    write(*,*)'problème dans lectdata : les incertitudes sur les données S n''existent pas ',Adata%sigS
    stop
  endif
```

```
117 ! -------                                                    ---------    .
118 Adata%tpsR%date%year=Adata%tpsR%date%year+2000
119 ! -------                                                    ---------    . respect du decoupage des années en mois et jours avec prise en compte des années
        bisextiles
120 call basetime(Adata%tpsR)
121 call JDATE(Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
122 call GDATE (Adata%tpsR%date%Jday,Adata%tpsR%date%year,Adata%tpsR%date%month,Adata%tpsR%date%day)
123 call basetime(Adata%tpsR)                                                 ! respect du decoupage temps dans la base composite 60/12/365 ...
124 ! -------                                                    ---------    . verif station
125 astation=TRIM(adjustl(kstnm))//'000'
126 if (Adata%sta%staname.ne.astation) then
127   write(*,*)'problème : les stations sont différentes : ',Adata%sta%staname,(TRIM(adjustl(kstnm))//'000')
128   stop
129 endif
130 ! ---------------------------------------------------------------    .
131 call difftime(alpha,beta,Adata%tpsR,Adata%tpsTh)
132 ! ---------------------------------------------------------------    .
133 ! ONDES DIRECTES                                                    .
134 ! ---------------------------------------------------------------    .
135 ! T1 : temps arrivée de l'onde Pg
136 ! T2 : incertitudes absolue sur l'onde Pg                           ! -> calcul du delta après
137 ! ---------------------------------------------------------------    .
138 ! T3 : temps arrivée de l'onde Sg
139 ! T4 : incertitudes absolue sur l'onde Sg
140 ! ---------------------------------------------------------------    .
141 if (Adata%typeonde.eq.'G') then
142   if(Adata%andS.eq.'S') then
143     T1=real(alpha,4)
144     T2=T1+real(Adata%sigP,4)
145     T3=real(beta,4)
146     T4=T3+real(Adata%sigS,4)
147     if (.not.((T3.gt.(0.0_4+b1)).and.(T4.lt.(real(npts1,4)*delta1+b1)))) then
148       T3=-12345.0_4
149       T4=-12345.0_4
150     endif
151     if (.not.((T4.gt.(0.0_4+b1)).and.(T5.lt.(real(npts1,4)*delta1+b1)))) then
152       T3=-12345.0_4
153       T4=-12345.0_4
154     endif
155   else
156     T1=real(alpha,4)
157     T2=T1+real(Adata%sigP,4)
158   endif
159   if (.not.((T1.gt.(0.0_4+b1)).and.(T2.lt.(real(npts1,4)*delta1+b1)))) then
160     T1=-12345.0_4
161     T2=-12345.0_4
162     T3=-12345.0_4
163     T4=-12345.0_4
164   endif
165   if (.not.((T2.gt.(0.0_4+b1)).and.(T3.lt.(real(npts1,4)*delta1+b1)))) then
166     T1=-12345.0_4
167     T2=-12345.0_4
168     T3=-12345.0_4
169     T4=-12345.0_4
170   endif
171   ! ---------------------------------------------------------------    .
172   ! ONDES REFRACTEES                                                  .
173   ! ---------------------------------------------------------------    .
174   ! T5 : temps arrivée de l'onde Pn
175   ! T6 : incertitudes absolue sur l'onde Pn
176   ! ---------------------------------------------------------------    .
177   ! T7 : temps arrivée de l'onde Sn
178   ! T8 : incertitudes absolue sur l'onde Sn
179   ! ---------------------------------------------------------------    .
180 elseif (Adata%typeonde.eq.'N') then                                    !
```

```
    if(Adata%andS.eq.'S') then
      T5=real(alpha,4)
      T6=T5+real(Adata%sigP,4)
      T7=real(beta,4)
      T8=T7+real(Adata%sigS,4)
      if (.not.((T7.gt.(0.0_4+b1)).and.(T8.lt.(real(npts1,4)*delta1+b1)))) then
        T7=-12345.0_4
        T8=-12345.0_4
      endif
      if (.not.((T8.gt.(0.0_4+b1)).and.(T9.lt.(real(npts1,4)*delta1+b1)))) then
        T8=-12345.0_4
        T7=-12345.0_4
      endif
    else
      T5=real(alpha,4)
      T6=T5+real(Adata%sigP,4)
    endif
    if (.not.((T5.gt.(0.0_4+b1)).and.(T6.lt.(real(npts1,4)*delta1+b1)))) then
      T5=-12345.0_4
      T6=-12345.0_4
      T8=-12345.0_4
      T7=-12345.0_4
    endif
    if (.not.((T6.gt.(0.0_4+b1)).and.(T7.lt.(real(npts1,4)*delta1+b1)))) then
      T6=-12345.0_4
      T5=-12345.0_4
      T8=-12345.0_4
      T7=-12345.0_4
    endif
  ! ------------------------------------------------------------------     .
  else
    write(*,*)'problème : onde ni directe ni réfractée ... ? ', Adata%typeonde
    stop
  endif
  ! ------------------------------------------------------------------     .
  if (nvhdr /= 6) then
    write(*,*) "ERROR 2 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
    stop
  endif
  ! ------------------------------------------------------------------     .


  t=int(plusoumoins/thedelta)

  ! ------------------------------------------------------------------     .
  ! ONDE Pg :
  ! ------------------------------------------------------------------     .
  do i=1,npts1
    if (real(real(i)*thedelta).lt.(real(T1))) then
      iter=i
    endif
  enddo
  ! ------------------------------------------------------------------     .
  nn1=min(iter+t,npts1)-max(iter-t,1)
  nn2=2**max0(int(dlog(dble(nn1))/dlog(2.0d0))+1,3)                         ! prochaine puissance de 2 (fft)
  ! ------------------------------------------------------------------     .
  allocate(spdata(2*nn2),temps(2*nn2),freq(2*nn2))
  ! ------------------------------------------------------------------     .
  freq(1)=0.0d0
  freq(2)=0.0d0
  do i=3,nn2-1,2
    freq(i)=real((i+1)/2-1)/real(nn2*thedelta)
    freq(i+1)=freq(i)
  enddo
  freq(nn2+1)=1.0/real(2*thedelta)
```

```
246   freq(nn2+2)=freq(nn2+1)
247   do i=nn2+2,2*nn2-2,2
248     freq(i)=real((-(i+1)+nn2)/2-1)/real(nn2*thedelta)
249     freq(i+1)=freq(i)
250   enddo
251   freq(2*nn2-1)=-1.0/real(nn2*thedelta)
252   freq(2*nn2)=freq(2*nn2-1)
253   ! ------------------------------------------------------------------      .
254   spdata(:)=0.0d0
255   temps(:)=0.0d0
256   ! ------------------------------------------------------------------      .
257   j=max(iter-t,1)
258   do i=1,2*nn2-2,2
259     if(j<min(iter+t,npts1)) then
260       spdata(i)=sacfile(j)
261     else
262       spdata(i)=0.0d0
263     endif
264     j=j+1
265   enddo
266   ! ------------------------------------------------------------------      .
267   do i=3,2*nn2-2,2
268     temps(i)=temps(i-1)+thedelta
269     temps(i+1)=temps(i-1)+thedelta
270   enddo
271   ! ------------------------------------------------------------------      .
272   if (plot) then
273   open(100, FILE = 'data.xy1',status='replace')
274   do i=1,2*nn1,2
275     write(100,*)temps(i),spdata(i)
276   enddo
277   close(100)
278   endif
279   ! ------------------------------------------------------------------      .
280   amin=999.999d0
281   amax=-999.999d0
282   do i=3,nn2-2,2
283     if (log10(sqrt(spdata(i)**2.0d0+spdata(i+1)**2.0d0))<amin) amin=log10(sqrt(spdata(i)**2.0d0+spdata(i+1)**2.0d0))
284     if (log10(sqrt(spdata(i)**2.0d0+spdata(i+1)**2.0d0))>amax) amax=log10(sqrt(spdata(i)**2.0d0+spdata(i+1)**2.0d0))
285   enddo
286   ! ------------------------------------------------------------------      .
287   call four1(spdata,nn2,1)
288   open(102, FILE = 'OUTPUT/GMT/spectre_'//TRIM(adjustl(kstnm))//'_'//TRIM(adjustl(kcmpnm))//'.ph',status='replace')
289   open(101, FILE = 'OUTPUT/GMT/spectre_'//TRIM(adjustl(kstnm))//'_'//TRIM(adjustl(kcmpnm))//'.am',status='replace')
290   do i=3,nn2-2,2
291     write(101,*)freq(i),log10(sqrt(spdata(i)**2.0d0+spdata(i+1)**2.0d0))/(amax-amin)*1000.d0-amin+0.1d0
292     write(102,*)freq(i),atan2(spdata(i+1),spdata(i))
293   enddo
294   close(101)
295   close(102)
296   ! ------------------------------------------------------------------      .
297   if (plot) then
298   call four1(spdata,nn2,-1)
299   open(103, FILE = 'data.xy2',status='replace')
300   do i=1,2*nn1,2
301     spdata(i)=spdata(i)/real(nn2)
302     write(103,*)temps(i),spdata(i)
303   enddo
304   close(103)
305   endif
306   ! ------------------------------------------------------------------      .
307   deallocate(spdata,temps,freq)
308
309   ! ------------------------------------------------------------------      . format de lecture des données
310   1000 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
```

```
311 ! ------------------------------------------------------------------------      .
312
313
314 ! ------------------------------------------------------------------------      .
315 ! ------------------------------------------------------------------------      .
316 CONTAINS
317 ! ------------------------------------------------------------------------      .
318 ! ------------------------------------------------------------------------      .
319
320
321 subroutine four1(adata,nn,isign)
322   ! ----------------------------------------------------------------------      .
323   implicit none
324   ! ----------------------------------------------------------------------      .
325   ! FFT routine :
326   ! Press W. H., Teukolsky S.A., Vetterling W.T.
327   ! Numerical Recipes in Fortran 77 --- The Art of Scientific Computing --
328   ! Second Edition - Volume 1 of Fortran Numerical Recipes1
329   ! ----------------------------------------------------------------------      .
330   integer, intent (in) :: isign,nn
331   double precision, intent (inout) :: adata(2*nn)
332   ! -------                                                  --------      .
333   integer i,istep,j,m,mmax,n
334   double precision tempi,tempr,theta,wi,wpi,wpr,wr,wtemp
335   ! ----------------------------------------------------------------------      .
336   n=2*nn
337   j=1
338   do i=1,n,2                                                               ! This is the bit-reversal section of the routine.
339     if(j.gt.i)then
340       tempr=adata(j)                                                       ! Exchange the two complex numbers.
341       tempi=adata(j+1)
342       adata(j)=adata(i)
343       adata(j+1)=adata(i+1)
344       adata(i)=tempr
345       adata(i+1)=tempi
346     endif
347     m=n/2
348     10001 if ((m.ge.2).and.(j.gt.m)) then
349       j=j-m
350       m=m/2
351       goto 10001
352     endif
353     j=j+m
354   enddo
355   ! ----------------------------------------------------------------------      .
356   mmax=2                                                                   ! Here begins the Danielson-Lanczos section of the routine.
357   ! -------                                                  --------      .
358   10002 if (n.gt.mmax) then                                                ! Outer loop executed log2 nn times.
359     istep=2*mmax
360     theta=6.28318530717959d0/real(isign*mmax)                              ! Initialize for the trigonometric recurrence
361     wpr=-2.d0*sin(0.5d0*theta)**2
362     wpi=sin(theta)
363     wr=1.d0
364     wi=0.d0
365     do m=1,mmax,2                                                          ! Here are the two nested inner loops.
366       do i=m,n,istep
367         j=i+mmax                                                           ! This is the Danielson-Lanczos formula:
368         tempr=sngl(wr)*adata(j)-sngl(wi)*adata(j+1)
369         tempi=sngl(wr)*adata(j+1)+sngl(wi)*adata(j)
370         adata(j)=adata(i)-tempr
371         adata(j+1)=adata(i+1)-tempi
372         adata(i)=adata(i)+tempr
373         adata(i+1)=adata(i+1)+tempi
374       enddo
375       wtemp=wr                                                             ! Trigonometric recurrence.
```

```fortran
        wr=wr*wpr-wi*wpi+wr
        wi=wi*wpr+wtemp*wpi+wi
      enddo
      mmax=istep
      goto 10002                                              ! Not yet done.
    endif                                                     ! All done.
    ! -------                                       -------   .
  end subroutine four1
  ! -----------------------------------------------------------   .


end program spectre

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! *****************************************************************   .
```

1.16 SRC/PROG/sac_stalta_kurtosis.f90

```fortran
! programme principal sac_stalta_kurtosis
! *****************************************************************   .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! -------                                                  -------   .
! ------- stalta_kurtosis :                                -------   .
! -------                                                  -------   .
! ------- fevrier 2015                                     -------   .
! -------                                                  -------   .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
! ------- Méric Haugmard meric.haugmard@univ-nantes.fr     -------   .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
!  This program is distributed for research purposes and in the hope   !
!  that it will be useful however without any warranties.              !
!  Use it on your own risk. Please, report bugs/improvements/... .     !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@   .
!                                                                      !
!
! CONTAINS : subroutine Moment
!
program stalta_kurtosis
  ! -----------------------------------------------------------   .
  use modparam
  use typetemps
  use time
  use sac_i_o
  ! -------                                              -------   .
  implicit none
  ! ------------------------------------------------------------   . pour SAC
  real(k), dimension(:), allocatable :: sacfile1,sacfile2,sacfile3
  real(k), dimension(:), allocatable :: sacfilestalta, sacfilekurt, sacfilekurtdervivcentre
  real(k), dimension(:), allocatable :: F2,F3,F4
  real(k) :: delta1, b1, e1
  real(k) :: delta2, b2, e2
  real(k) :: delta3, b3, e3
  integer(k) :: NN, npts1, npts2, npts3
  character(LEN=112) :: file1,file2,file3
  character(LEN=112) :: outfile
  integer(k) :: anbpts
  ! ------------------------------------------------------------   . autres
  integer(KIND=wi) :: JD, inc, i,j
  integer(KIND=wi) :: sta, lta
  real(KIND = wr) :: nsec, ave, sdev, skew, curt1, curt2, curt3, nextmax
  real(KIND = wr), dimension(:), allocatable :: Asta, Alta, FC
  real(KIND=wr) :: deltaP1,deltaS1,deltaP2,deltaS2
  type(dataone) :: Adata1,Adata2,Adata3
  logical :: existe1
  ! -----------------------------------------------------------   . lecture
```

```
48  NN = IARGC()
49  if ((NN < 1).or.(NN > 3).or.(NN == 2)) then
50    write(*,'(a)') "usage:  ./sac_stalta_kurtosis sacfile (1 or 3 files : Z, N , E)"
51    write(*,'(a)') "        sacfile - input sac file (.bin)",NN
52    stop
53  endif
54  call GETARG(1, file1)
55  inquire (file=file1,exist=existe1) ! option différente selon compilo !
56  if (existe1) then
57  ! -------                                                    --------    . lecture du fichier sac
58  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,         &
59    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
60    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
61    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
62    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
63    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
64    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
65    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
66    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
67    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
68    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
69    kinst,sacfile1)
70  if (nvhdr /= 6) then
71    write(*,*) "ERROR 1 - File 1 : '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
72    stop
73  endif
74  allocate(FC(npts1))
75  ! ------------------------------------------------------------------    . temps du début de la trace, -> tpsTh <-
76  Adata1%tpsTh%date%year=int(nzyear,wi)
77  ! -------                                                    --------    . jour julien
78  Adata1%tpsTh%date%month=1
79  Adata1%tpsTh%date%day=1
80  call JDATE (JD,Adata1%tpsTh%date%year,Adata1%tpsTh%date%month,Adata1%tpsTh%date%day)
81  call GDATE (JD+int(nzjday,wi)-1, Adata1%tpsTh%date%year,Adata1%tpsTh%date%month,Adata1%tpsTh%date%day)
82  call JDATE (Adata1%tpsTh%date%Jday,Adata1%tpsTh%date%year,Adata1%tpsTh%date%month,Adata1%tpsTh%date%day)
83  ! -------                                                    --------    .
84  Adata1%tpsTh%date%hour=nzhour
85  Adata1%tpsTh%date%min=nzmin
86  Adata1%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
87  Adata1%tpsTh%secS=Adata1%tpsTh%secP
88  call basetime(Adata1%tpsTh)
89  ! ------------------------------------------------------------------    .
90
91  ! ------------------------------------------------------------------    .
92  if (NN==3) then
93    call GETARG(2,file2)
94    call GETARG(3,file3)
95    ! -------                                                  --------    . lecture du fichier sac
96    call rbsac(file2,delta2,depmin,depmax,scale,odelta,b2,e2,o,a,internal1,       &
97      t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,  &
98      resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,  &
99      user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,&
100     internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,  &
101     unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin, &
102     nzsec,nzmsec,nvhdr,norid,nevid,npts2,internal4,nwfid,nxsize,nysize,unused8, &
103     iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp, &
104     imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,      &
105     unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
106     kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
107     kinst,sacfile2)
108   if (nvhdr /= 6) then
109     write(*,*) "ERROR 1 - File 2 : '", TRIM(adjustl(file2)), "' appears to be of non-native byte-order or is not a SAC file."
110     stop
111   endif
112   ! ------------------------------------------------------------------    . temps du début de la trace, -> tpsTh <-
```

```
    Adata2%tpsTh%date%year=int(nzyear,wi)
    ! -------                                                    --------    . jour julien
    Adata2%tpsTh%date%month=1
    Adata2%tpsTh%date%day=1
    call JDATE (JD,Adata2%tpsTh%date%year,Adata2%tpsTh%date%month,Adata2%tpsTh%date%day)
    call GDATE (JD+int(nzjday,wi)-1, Adata2%tpsTh%date%year,Adata2%tpsTh%date%month,Adata2%tpsTh%date%day)
    call JDATE (Adata2%tpsTh%date%Jday,Adata2%tpsTh%date%year,Adata2%tpsTh%date%month,Adata2%tpsTh%date%day)
    ! -------                                                    --------    .
    Adata2%tpsTh%date%hour=nzhour
    Adata2%tpsTh%date%min=nzmin
    Adata2%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
    Adata2%tpsTh%secS=Adata2%tpsTh%secP
    call basetime(Adata2%tpsTh)
    ! ------------------------------------------------------------------    .

    ! -------                                                    --------    . lecture du fichier sac
    call rbsac(file3,delta3,depmin,depmax,scale,odelta,b3,e3,o,a,internal1,         &
      t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
      resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
      user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
      internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
      unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
      nzsec,nzmsec,nvhdr,norid,nevid,npts3,internal4,nwfid,nxsize,nysize,unused8,   &
      iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
      imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
      unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
      kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
      kinst,sacfile3)
    if (nvhdr /= 6) then
      write(*,*) "ERROR 1 - File 3 : '", TRIM(adjustl(file3)), "' appears to be of non-native byte-order or is not a SAC file."
      stop
    endif
    ! ------------------------------------------------------------------    . temps du début de la trace, -> tpsTh <-
    Adata3%tpsTh%date%year=int(nzyear,wi)
    ! -------                                                    --------    . jour julien
    Adata3%tpsTh%date%month=1
    Adata3%tpsTh%date%day=1
    call JDATE (JD,Adata3%tpsTh%date%year,Adata3%tpsTh%date%month,Adata3%tpsTh%date%day)
    call GDATE (JD+int(nzjday,wi)-1, Adata3%tpsTh%date%year,Adata3%tpsTh%date%month,Adata3%tpsTh%date%day)
    call JDATE (Adata3%tpsTh%date%Jday,Adata3%tpsTh%date%year,Adata3%tpsTh%date%month,Adata3%tpsTh%date%day)
    ! -------                                                    --------    .
    Adata3%tpsTh%date%hour=nzhour
    Adata3%tpsTh%date%min=nzmin
    Adata3%tpsTh%secP=real(nzsec,wr) + real(nzmsec,wr)/1000.0_wr
    Adata3%tpsTh%secS=Adata3%tpsTh%secP
    call basetime(Adata3%tpsTh)
    ! ------------------------------------------------------------------    . verification 1
    call difftime(deltaP1,deltaS1,Adata1%tpsTh,Adata2%tpsTh)
    call difftime(deltaP2,deltaS2,Adata1%tpsTh,Adata3%tpsTh)
    if ((deltaP1+deltaS2).gt.(real(delta1,wr)*4.0_wr)) then
      write(*,*)'problème dans stalta_kurtosis, fichiers non homogènes en temps '
      write(*,*)Adata1%tpsTh
      write(*,*)Adata2%tpsTh,deltaP1
      write(*,*)Adata3%tpsTh,deltaP2
      write(*,*)delta1
      stop
    endif
    ! ------------------------------------------------------------------    . verification 2
    if ((npts1.ne.npts2).or.(npts1.ne.npts3)) then
      write(*,*)'problème dans stalta_kurtosis, fichiers non homogènes en durée '
      write(*,*)npts1,npts2,npts3
      stop
    endif
    ! ------------------------------------------------------------------    . verification 3
    if ((delta1.ne.delta2).or.(delta1.ne.delta3)) then
```

```
      write(*,*)'problème dans stalta_kurtosis, fichiers non homogènes en taux échantillonnage'
      write(*,*)delta1,delta2,delta3
      stop
    endif
  endif
  ! ---------------------------------------------------------------------    .



  ! ---------------------------------------------------------------------    .
  ! KURTOSIS
  ! cf : Baillard, W. C. Crawford, V. Ballu, C. Hibert, & A. Mangeney (2014) :
  !      An Automatic Kurtosis-Based P- and S-Phase Picker Designed for Local Seismic Networks.
  !      BSSA, Vol. 104, No. 1, 16p.
  ! ---------------------------------------------------------------------    .
  allocate(sacfilekurt(npts1))
  allocate(sacfilekurtdervivcentre(npts1))
  allocate(F2(npts1),F3(npts1),F4(npts1))
  nsec=2.5_wr                                                               ! 1.0 secondes
  inc=int(nsec/real(delta1,wr)+1.0_wr,wi)
  sacfilekurt=real(0.0,k)
  anbpts=int(inc,k)
  ! -------                                                  ---------    .
  ! The central moment of order d at sample k can be written as
  ! -------                                                  ---------    .
  do i=anbpts,npts1
    call Moment(sacfile1(i-anbpts+1:i),anbpts,ave,sdev,skew,curt1)
    if (NN==3) then
      call Moment(sacfile2(i-anbpts+1:i),anbpts,ave,sdev,skew,curt2)
      call Moment(sacfile3(i-anbpts+1:i),anbpts,ave,sdev,skew,curt3)
      sacfilekurt(i)=real((curt1+curt2+curt3)/3.0_wr,k)
    else
      sacfilekurt(i)=real(curt1,k)
    endif
  enddo
  ! -------                                                  ---------    .
  ! The first transformation essentially cleans the initial CF
  ! of all strictly negative gradients (Fig. 2c), because only
  ! positive gradients characterize the transition from noise to
  ! a coherent signal.
  ! -------                                                  ---------    .
  do i=1,npts1-1
   if ((sacfilekurt(i+1)-sacfilekurt(i)).gt.0.0_k) then
    F2(i+1)=F2(i)+(sacfilekurt(i+1)-sacfilekurt(i))
   else
    F2(i+1)=F2(i)
   endif
  enddo
  F2(npts1)=F2(npts1-1)
  ! -------                                                  ---------    .
  ! The second transformation removes a linear trend from
  ! F2, so that the first and last values equal zero.
  ! In this way, the onsets become local minima.
  ! -------                                                  ---------    .
  do i=1+1,npts1
    F3(i)=F2(i)-((F2(npts1)-F2(1))/(real(npts1-1,k))*real(i-1,k)+F2(1))
  enddo
  ! -------                                                  ---------    .
  ! The final transformation makes the amplitude of the
  ! minima amplitude scale with the total change in the kurtosis
  ! that follows, so that the greatest minima correspond to the
  ! greatest onset strengths
  ! -------                                                  ---------    .
  do i=1,npts1-2
```

```
    ! prochain maxima
    do j=i+1,npts1
      if (F3(j+1).lt.F3(j)) then
        nextmax=F3(j)
        exit
      endif
    enddo
    if ((F3(i)-nextmax).lt.0.0_k) then
      F4(i)=F3(i)-real(nextmax,k)
    else
      F4(i)=0.0_k
    endif
  enddo
  F4(npts1)=0.0_k
  F4(npts1-1)=0.0_k
  F4(npts1-2)=0.0_k
  ! ------                                                    --------    .
  !Dérivée centré d'ordre 2
  !do i=2,npts1-1
  !sacfilekurtdervivcentre(i)=(sacfilekurt(i+1)-sacfilekurt(i-1))/(2.0_k*delta1)
  !enddo
  !sacfilekurtdervivcentre(1)=sacfilekurtdervivcentre(2)
  !sacfilekurtdervivcentre(npts1)=sacfilekurtdervivcentre(npts1)
  ! ------------------------------------------------------------------------    .
  deallocate(F2,F3)
  ! ------------------------------------------------------------------------    .



  ! ------------------------------------------------------------------------    .
  ! STA / LTA                                                                   .
  ! ------------------------------------------------------------------------    .
  if (NN==3) then
    ! -------                                                 --------    . fonction caractéristique
    FC=(real(sacfile1,wr)**2.0_wr+real(sacfile2,wr)**2.0_wr+real(sacfile3,wr)**2.0_wr)/3.0_wr
  else
    ! -------                                                 --------    . fonction caractéristique : energie
    FC=real(sacfile1,wr)**2.0_wr
   endif
  ! ------------------------------------------------------------------------    .
  allocate(sacfilestalta(npts1))
  allocate(Asta(npts1))
  allocate(Alta(npts1))
  Asta=0.1e-9_wr
  Alta=0.1e-9_wr
  nsec=3.0_wr                                                                 ! 2.5 secondes pour sta
  sta=int(nsec/real(delta1,wr)+1.0_wr,wi)
  nsec=30.0_wr                                                                ! 25 secondes pour lta
  lta=int(nsec/real(delta1,wr)+1.0_wr,wi)
  sacfilestalta=real(0.0,k)
  ! ------------------------------------------------------------------------    . initialisation STA
  Asta(lta+sta+1)=0.00001_wr
  do i=1,lta
   Alta(lta+sta+1)=Alta(lta+sta+1)+FC(i)
  enddo
  Alta(lta+sta+1)=Alta(lta+sta+1)/real(lta,wr)
  ! ------------------------------------------------------------------------    . initialisation LTA
  Alta(lta+sta+1)=0.00001_wr
  do i=lta+1,lta+sta
    Asta(lta+sta+1)=Asta(lta+sta+1)+FC(i)
  enddo
  Asta(lta+sta+1)=Asta(lta+sta+1)/real(sta,wr)
  ! ------------------------------------------------------------------------    . STA - LTA
  do i=lta+sta+sta,npts1-1
    ! démarre à + 2 sta sinon instable
```

```
308       Alta(i)= FC(i-sta-1)/real(lta,wr) + (1.0_wr-1.0_wr/real(lta,wr))*Alta(i-1)
309       Asta(i)= FC(i-1)/real(sta,wr) + (1.0_wr-1.0_wr/real(sta,wr))*Asta(i-1)
310       sacfilestalta(i)=real(Asta(i)/Alta(i),k)
311     enddo
312
313     deallocate(Asta,Alta,FC)
314     !-------------------------------------------------------------------      .
315
316     !-------------------------------------------------------------------      .
317     ! ecriture des résultats                                                  .
318     !-------------------------------------------------------------------      .
319     outfile=TRIM(adjustl(file1))//'.kurt'
320     !-------------------------------------------------------------------      .
321     call wbsac(outfile,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,       &
322       t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
323       resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
324       user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
325       internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
326       unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
327       nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
328       iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
329       imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
330       unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
331       kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
332       kinst,F4)
333
334     !-------------------------------------------------------------------      .
335     !-------------------------------------------------------------------      .
336     outfile=TRIM(adjustl(file1))//'.stalta'
337     !-------------------------------------------------------------------      .
338     call wbsac(outfile,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,       &
339       t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
340       resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
341       user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
342       internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
343       unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
344       nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
345       iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
346       imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
347       unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
348       kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
349       kinst,sacfilestalta)
350     !-------------------------------------------------------------------      .
351     if (nvhdr /= 6) then
352       write(*,*) "ERROR 2 - File: '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
353       stop
354     endif
355     !-------------------------------------------------------------------      .
356     deallocate(sacfilestalta,sacfilekurt,sacfilekurtdervivcentre,F4)
357     else
358       write(*,*) "NO FILE ... ",file1
359     endif
360     !-------------------------------------------------------------------      .
361     !-------------------------------------------------------------------      .
362
363   CONTAINS
364
365     !-------------------------------------------------------------------      .
366     !-------------------------------------------------------------------      .
367
368     subroutine Moment(Adata1,n,ave,sdev,skew,curt)
369       !-------------------------------------------------------------------      .
370       ! calcul de la moyenne (ave), de l'écart-type (sdev)
371       ! du coefficient de dissymétrie (skew) et
372       ! du coefficient d'aplatissement de Pearson (curt pour kurtosis)
```

```
    ! ---------------------------------------------------------------------    .
    implicit none
    ! ---------------------------------------------------------------------    .
    integer(k), intent (in) :: n
    real(KIND=k), intent (in) :: Adata1(n)
    real(KIND=wr), intent (out) ::   ave,sdev,skew,curt
    ! --------                                                      --------    .
    integer(KIND=wi) :: j
    real(KIND=wr) :: var,p,s
    ! ---------------------------------------------------------------------    .
    if(n.le.1) then
      write(*,*)'problème dans Moment : n < 2 ! ',n
      stop
    end if
    s=0.0_wr
    do j=1,n
      s=s+real(Adata1(j),wr)
    end do
    ave=s/real(n,wr)                                                          ! moyenne
    ! ---------------------------------------------------------------------    .
    var=0.0_wr
    skew=0.0_wr
    curt=0.0_wr
    do j=1,n
      s=real(Adata1(j),wr)-ave
      p=s*s
      var=var+p
      p=p*s
      skew=skew+p
      p=p*s
      curt=curt+p
    end do
    ! ---------------------------------------------------------------------    .
    !var=var/real(n-1,wr)                                                     ! variance echantillon
    var=var/real(n,wr)                                                        ! variance population
    sdev=sqrt(var)                                                            ! ecart-type
    if(var.ne.0.0_wr) then
      skew=skew/(real(n,wr)*sdev**3.0_wr)                                     ! coefficient de dissymétrie
      curt=curt/(real(n,wr)*var**2.0_wr)-3.0_wr                               ! coefficient d'aplatissement de Pearson
    else
      skew=-1.0e9_wr
      curt=-1.0e9_wr
    end if
    ! ---------------------------------------------------------------------    .
    ! write(*,*) n,ave,sdev,skew,curt
    ! ---------------------------------------------------------------------    .
  end subroutine Moment

  ! -----------------------------------------------------------------------    .
  ! -----------------------------------------------------------------------    .

end program stalta_kurtosis

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! ************************************************************************    .
```

1.17 SRC/PROG/verifmediatrice.f90

```
! programme principal sac_verifmediatrice
! ************************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                       --------    .
! --------                                                       --------    .
! -------- avril 2015                                            --------    .
! --------                                                       --------    .
```

```fortran
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!   This program is distributed for research purposes and in the hope       !
!   that it will be useful however without any warranties.                  !
!   Use it on your own risk. Please, report bugs/improvements/... .         !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                           !
!
program verifmediatrice
  ! ----------------------------------------------------------------------    .
  ! carte de recherches epicentrales a priori
  ! ----------------------------------------------------------------------    .
  use modparam
  use typetemps
  use datalecture
  use rechercheepi
  ! --------                                                      --------    .
  implicit none
  ! --------                                                      --------    .
  type(dataall) :: D(nbseismes)                                               ! données de temps
  type(priorEPI) :: pEpis(nbseismes)                                          ! prior
  integer(KIND=wi) :: i, mb, nbsta, nbtps(nbseismes)
  ! ----------------------------------------------------------------------    .
  ! existe OUTPUT/GMT/ et  OUTPUT/figures/
  ! existe DATA/sta.d et DATA/sta.d et DATA/seismes.d DATA/xxxx.xx.xx.xx.xx.d
  ! --------                                                      --------    .
  ! -------- lecture des données                                 --------    . phases and stations list
  call lectnbdata(nbsta,nbtps)                                                ! nombre de données
  do i=1,nbseismes
    allocate(D(i)%datatps(nbtps(i)))                                          ! alloue par séisme
  enddo
  call lectdata(nbsta,nbtps,D)
  ! --------                                                      --------    .
  print*,'initialisation du prior pour les paramètres épicentraux'
  call zoneRecherche(nbtps,D,pEpis,mb)
  ! --------                                                      --------    .
  write(*,*)'chmod +x OUTPUT/GMT/script*.sh ; ./OUTPUT/GMT/script*.sh '       ! chmod +x ; plot it
  ! --------                                                      --------    .
  ! ----------------------------------------------------------------------    .
end program verifmediatrice

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! *********************************************************************    .
```

1.18 SRC/PROG/sac_ZNE_2_LQT.f90

```fortran
! programme principal sac_ZNE_2_LQT
! *********************************************************************    .mh
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! --------                                                        --------    .
! --------                                                        --------    .
! -------- mars 2015                                              --------    .
! --------                                                        --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!   This program is distributed for research purposes and in the hope       !
!   that it will be useful however without any warranties.                  !
!   Use it on your own risk. Please, report bugs/improvements/... .         !
! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
!                                                                           !
!
program ZNE_2_LQT
  ! ----------------------------------------------------------------------    .
```

```
19  use modparam
20  use distance_epi
21  use sac_i_o
22  ! --------                                                    --------    .
23  implicit none
24  real (kind=wr) :: dlat1,dlat2,dlon1,dlon2,d,alti,pfd
25  real (kind=wr) :: a_baz                                                 ! backazimuth measured clockwise from north
26  real (kind=wr) :: alpha                                                 ! angle of incidence, measured from vertical
27  real (kind=wr) :: M3D(3,3), val(7)
28  integer(KIND=wi) :: ok
29  character(LEN=4) :: Asta
30  logical :: found
31  ! ------------------------------------------------------------------    . pour SAC
32  real(k), dimension(:), allocatable :: sacZ, sacN, sacE, sacL, sacQ, sacT
33   CHARACTER(LEN=8) :: kstnm1,kstnm2,kstnm3
34  real(k) :: delta1, b1, e1
35  real(k) :: delta2, b2, e2
36  real(k) :: delta3, b3, e3
37  integer(k) :: NN, npts1, npts2, npts3
38  character(LEN=112) :: file1,file2,file3
39  character(LEN=112) :: outfile
40  ! ------------------------------------------------------------------    .
41  ! Plesinger, A., M. Hellweg and D. Seidl (1986) : Interactive high-resolution polarization analysis of broadband seismograms. J. Geophysics, 59, p. 129-139.
42  ! (http://service.iris.edu/irisws/rotation/docs/1/help/)
43  ! ------------------------------------------------------------------    .
44  ! Z , E , and N represents the 3 seismograms with original orientations
45  ! ------------------------------------------------------------------    .
46  ! L, Q, and T represent the three seismograms that are output as below.
47  ! L - Aligned in direction of P wave propagation
48  ! Q - Aligned in the direction of the SV phase movement
49  ! T - Aligned in the direction of the SH phase movement
50  ! ------------------------------------------------------------------    .
51  ! -------- Lecture des données                             --------    .
52  ! ------------------------------------------------------------------    .
53  NN = IARGC()
54  if ((NN > 3).or.(NN < 3)) then
55    write(*,'(a)') "usage:  ./ZNE_2_LQT.exe sacfileZ sacfileN sacfileE latEvent LonEvent pdfEvent"
56    write(*,'(a)') "        sacfile - input sac file (.bin)"
57    stop
58  endif
59  call GETARG(1,file1)
60  call GETARG(2,file2)
61  call GETARG(3,file3)
62  ! ------------------------------------------------------------------    . lecture de l'hypocentre
63  write(*,*)"hypocentre : lat(deg),lon(deg),pfd(m)"
64  read(*,*)dlat2,dlon2,pfd
65
66  !  dlat2=47.657_wr fixe ?
67  !  dlon2=-2.8_wr
68  !  pfd=4.0_wr
69
70  ! ------------------------------------------------------------------    . lecture du fichier sac Z
71  call rbsac(file1,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,         &
72    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
73    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
74    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
75    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
76    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
77    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
78    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
79    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
80    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm1,khole,ko,ka,kt0,kt1,&
81    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
82    kinst,sacZ)
83  ! --------                                                    --------    .
```

```
84  if (nvhdr /= 6) then
85    write(*,*) "ERROR 1 - File Z : '", TRIM(adjustl(file1)), "' appears to be of non-native byte-order or is not a SAC file."
86    stop
87  endif
88  ! -------                                                    --------    . verification de la composante
89  if((kcmpnm(3:3).ne."Z").and.(kcmpnm(3:3).ne."1")) then
90    write(*,*)'problème mauvaise composante -> Z,1'
91    write(*,*)kcmpnm
92    stop
93  endif
94  ! ------------------------------------------------------------------    . lecture du fichier sac N
95  call rbsac(file2,delta2,depmin,depmax,scale,odelta,b2,e2,o,a,internal1,          &
96    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
97    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
98    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
99    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
100   unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
101   nzsec,nzmsec,nvhdr,norid,nevid,npts2,internal4,nwfid,nxsize,nysize,unused8,   &
102   iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
103   imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
104   unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm2,khole,ko,ka,kt0,kt1,&
105   kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
106   kinst,sacN)
107 ! -------                                                    --------    .
108 if (nvhdr /= 6) then
109   write(*,*) "ERROR 1 - File N : '", TRIM(adjustl(file2)), "' appears to be of non-native byte-order or is not a SAC file."
110   stop
111 endif
112 ! -------                                                    --------    . verification de la composante
113 if((kcmpnm(3:3).ne."N").and.(kcmpnm(3:3).ne."2")) then
114   write(*,*)'problème mauvaise composante -> N,2'
115   write(*,*)kcmpnm
116   stop
117 endif
118 ! ------------------------------------------------------------------    .  lecture du fichier sac E
119 call rbsac(file3,delta3,depmin,depmax,scale,odelta,b3,e3,o,a,internal1,          &
120   t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
121   resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
122   user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
123   internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
124   unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
125   nzsec,nzmsec,nvhdr,norid,nevid,npts3,internal4,nwfid,nxsize,nysize,unused8,   &
126   iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
127   imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
128   unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm3,khole,ko,ka,kt0,kt1,&
129   kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
130   kinst,sacE)
131 ! -------                                                    --------    .
132 if (nvhdr /= 6) then
133   write(*,*) "ERROR 1 - File E : '", TRIM(adjustl(file3)), "' appears to be of non-native byte-order or is not a SAC file."
134   stop
135 endif
136 ! -------                                                    --------    . verification de la composante
137 if((kcmpnm(3:3).ne."E").and.(kcmpnm(3:3).ne."3")) then
138   write(*,*)'problème mauvaise composante -> E,3'
139   write(*,*)kcmpnm
140   stop
141 endif
142 ! ------------------------------------------------------------------    . verification 1
143 if ((npts1.ne.npts2).or.(npts1.ne.npts3)) then
144   write(*,*)'problème dans stalta_ZNE_2_LQT, fichiers non homogènes en durée '
145   write(*,*)npts1,npts2,npts3
146   stop
147 endif
148 ! ------------------------------------------------------------------    . verification 2
```

```
149  if ((delta1.ne.delta2).or.(delta1.ne.delta3)) then
150    write(*,*)'problème dans stalta_ZNE_2_LQT, fichiers non homogènes en taux échantillonnage'
151    write(*,*)delta1,delta2,delta3
152    stop
153  endif
154  !-------------------------------------------------------------------     . verification 3
155  if ((kstnm1.ne.kstnm2).or.(kstnm1.ne.kstnm3)) then
156    write(*,*)'problème dans stalta_ZNE_2_LQT, differentes stations'
157    write(*,*)kstnm1,kstnm2,kstnm3
158    stop
159  endif
160  !-------------------------------------------------------------------     .
161  !-------------------------------------------------------------------     .
162
163  !-------------------------------------------------------------------     .
164  !-------- station                                  ---------     .
165  !-------------------------------------------------------------------     .
166  found=.false.
167  open(503, FILE = 'sta.d',status='old',iostat = ok)
168  if (ok .ne. 0) then
169    write(*,*)'problème : le fichier sta.d n''existe pas '
170    stop
171  endif
172  !-------------------------------------------------------------------     .
173  do while(ok.eq.0)
174    read(503,*,iostat = ok) Asta,val
175
176    if ((TRIM(adjustl(Asta))==TRIM(adjustl(kstnm1(1:4)))).or.(TRIM(adjustl(Asta))==TRIM(adjustl(kstnm1(1:4)))//"0")) then ! station en trois letters + "0"
177      if(found) then
178        write(*,*)'problème : la station '//TRIM(adjustl(kstnm1))//' est présente deux fois dans sta.d'
179        stop
180      else
181        dlat1=val(1)
182        dlon1=val(2)
183        alti=val(3)
184        found=.true.
185      endif
186    endif
187  enddo
188  close(503)
189  if (.not.found) then
190    write(*,*)'problème : la station '//TRIM(adjustl(kstnm1))//' n''existe pas '
191    stop
192  endif
193
194  !-------------------------------------------------------------------     .
195  !-------- rotation                                 ---------     .
196  !-------------------------------------------------------------------     .
197  call dellipsgc(dlat2,dlon2,dlat1,dlon1,d,a_baz)
198  ! write(*,*)d,a_baz,alpha/pi*180.0_wr
199  !-------------------------------------------------------------------     .
200  ! incidence is the angle from vertical at which an incoming ray arrives.
201  ! A ray arriving from directly below the station would have an incidence of 0 deg.
202  alpha=atan(d/(pfd+alti/1000.0_wr))
203  !-------------------------------------------------------------------     .
204
205  !-------------------------------------------------------------------     .
206  M3D(1,1)=cos(alpha)
207  M3D(1,2)=-sin(alpha)*sin(a_baz)
208  M3D(1,3)=-sin(alpha)*cos(a_baz)
209  M3D(2,1)=sin(alpha)
210  M3D(2,2)=cos(alpha)*sin(a_baz)
211  M3D(2,3)=cos(alpha)*cos(a_baz)
212  M3D(3,1)=0.0_wr
213  M3D(3,2)=-cos(a_baz)
```

```fortran
  M3D(3,3)=sin(a_baz)
  ! ------------------------------------------------------------------------    .
  allocate(sacL(npts1),sacQ(npts1),sacT(npts1))

  sacL = M3D(1,1)*sacZ + M3D(1,2)*sacE + M3D(1,3)*sacN
  sacQ = M3D(2,1)*sacZ + M3D(2,2)*sacE + M3D(2,3)*sacN
  sacT = M3D(3,1)*sacZ + M3D(3,2)*sacE + M3D(3,3)*sacN

  ! ------------------------------------------------------------------------    .
  ! -------- Écriture des données                                   ---------    .
  ! ------------------------------------------------------------------------    .
  outfile=TRIM(adjustl(file1))//'.L'
  kcmpnm='HHL'
  call wbsac(outfile,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,       &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm1,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacL)
  ! ------------------------------------------------------------------------    .
  outfile=TRIM(adjustl(file2))//'.Q'
  kcmpnm='HHQ'
  call wbsac(outfile,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,       &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm2,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacQ)
  ! ------------------------------------------------------------------------    .
  outfile=TRIM(adjustl(file3))//'.T'
  kcmpnm='HHT'
  call wbsac(outfile,delta1,depmin,depmax,scale,odelta,b1,e1,o,a,internal1,       &
    t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
    resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
    user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
    internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
    unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
    nzsec,nzmsec,nvhdr,norid,nevid,npts1,internal4,nwfid,nxsize,nysize,unused8,   &
    iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
    imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
    unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm3,khole,ko,ka,kt0,kt1,&
    kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
    kinst,sacT)
  ! ------------------------------------------------------------------------    .
  deallocate(sacZ,sacN,sacE,sacL,sacQ,sacT)
  ! ------------------------------------------------------------------------    .
end program ZNE_2_LQT

! @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@    .
! ************************************************************************    .
```

2 modules

2.1 SRC/MOD/MOD_GMT/mkGMT.f90

```
! permet la création des fichiers et autres scripts GMT en vue de la production des figures
! mars 2014
! ************************************************************************     .
! ------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------     .
! ************************************************************************     .
! ------------------------------------------------------------------------     .

MODULE figure_GMT

    use modparam

    implicit none

    private

    public  :: GMTfull
    public  :: affiche_temps_ref

    ! ------------------------------------------------------------------     .
    ! on défini _FILE_DIR_ en fonction du compilateur -> test existance dossier

#ifdef __INTEL_COMPILER
#define _FILE_DIR_ DIRECTORY
#elif __GFORTRAN__
#define _FILE_DIR_ FILE
#endif

    ! ------------------------------------------------------------------     .

CONTAINS

! ------------------------------------------------------------------     .

  subroutine GMTfull(dp,nmod,nbChaineMV,xmin,xmax,nbtps,nbsta,D,Ellips,nomsta,acentroid)
    ! -------                                                 --------     .mh
    ! production d'un large script bash (souvent > 10 000 lignes)
    ! pour produire l'ensemble des figures sous G.M.T.
    ! ------------------------------------------------------------------     .
    use typetemps
    use time
    use pb_direct
    use sub_param
    use distance_epi
    use sub_misfit
    ! -------                                                 --------     .
    use figure_GMTwada
    use figure_GMTchat
    use figure_GMTpar
    use figure_GMTmCorr
    use figure_GMTfc
    use figure_GMTres
    use figure_GMTmap
    use figure_GMThodo
    use figure_GMTcoda
    use figure_GMTmoho_inc
    use figure_GMTcarriere
    use figure_posteriori
    ! -------                                                 --------     .
    implicit none
    ! -------                                                 --------     .
    integer(KIND=wi), intent(in) :: nbtps(nbseismes),nbsta                  ! nombre de données de temps et de station
```

```
    integer(KIND=wi), intent (in) :: nbChaineMV                                 ! nombre de chaîne de Markov
    integer(KIND=wi), intent (in) :: nmod(nbChaineMV)                           ! nombre de modèles retenus par chaîne de Markov
    type(densityplot), intent (inout) :: dp                                     ! modèles retenus par McMC
    type(dataall), intent(inout) :: D(nbseismes)                                ! données
    real(KIND=wr), intent(in) :: xmin(nbseismes),xmax(nbseismes)                ! cercles de pondération
    type(ellip), intent(out) :: Ellips(nbseismes)                               ! ellipse
    character(LEN=4), dimension(:), allocatable, intent(out) :: nomsta
    type(amoho_centroid), intent (in) :: acentroid
    ! -------                                                 ---------     .
    type(parametres) :: param_best
    type(parametre) :: aparambest
    type(pond) :: w
    integer(KIND=wi) :: i,j
    logical :: critique
    real(KIND=wr) :: delta
    integer(KIND=wi) :: triseismes(nbseismes)
    type(date_sec) :: datetriseismes(nbseismes)
    character (LEN=5) :: numberfile
    logical :: existe1
    logical :: plot=.true.
    ! ------------------------------------------------------------------     .
    open(unit=600,file="OUTPUT/GMT/script.sh",STATUS="replace")
    open(unit=601,file="OUTPUT/GMT/files.txt",STATUS="replace")
    write(600,'(a)')"echo ' '"
    write(600,'(a)')"echo '-------------------------------------------------------'"
    write(600,'(a)')"echo '--------------------  script GMT ----------------------'"
    write(600,'(a)')"echo '-------------------------------------------------------'"
    write(600,'(a)')"echo ' '"
    write(600,'(a)')"gmtset LABEL_FONT_SIZE 15"                                 ! nouvelles options GMT
    write(600,'(a)')'gmtset HEADER_FONT_SIZE 15'
    write(600,'(a)')"gmtset ANNOT_FONT_PRIMARY Times-Roman"
    write(600,'(a)')"gmtset ANNOT_FONT_SECONDARY Times-Roman"
    write(600,'(a)')"gmtset PAPER_MEDIA A3"
    write(600,'(a)')"gmtset TIME_LANGUAGE FR"
    write(600,'(a)')"gmtset TRANSPARENCY 50"
    write(600,'(a)')"gmtset PLOT_DEGREE_FORMAT dddmm"
    write(600,'(a)')"gmtset BASEMAP_TYPE fancy"
    write(600,'(a)')"gmtset CHAR_ENCODING ISOLatin1+"
    ! ------------------------------------------------------------------     . tables couleurs pour GMT
    write(600,'(a)')"echo '0     0       0       0       5       255     0       255' > OUTPUT/GMT/colorpal1.cpt"
    !write(600,'(a)')"echo '0    0       0       0       .5      1       254     255' >> OUTPUT/GMT/colorpal1.cpt"
    !write(600,'(a)')"makecpt -Ccool -T.5/5/0.01 > OUTPUT/GMT/colorpal1.cpt"  ! pour la fonction coût
    write(600,'(a)')"makecpt -Crainbow -T5/33/.01 -N -Z >> OUTPUT/GMT/colorpal1.cpt"
    write(600,'(a)')"echo '33    0       1       1       100     0       1       1' >> OUTPUT/GMT/colorpal1.cpt"
    ! -------                                                 ---------     . pour le diagramme de densité
    write(600,'(a)')"makecpt -Cno_green -T1.0/97.5/.1 -N > OUTPUT/GMT/colorpal2.cpt"
    write(600,'(a)')"echo '97.5 255 0 0 100 255 0 255' >> OUTPUT/GMT/colorpal2.cpt"
    write(600,'(a)')"echo 'B 0 0 255' >> OUTPUT/GMT/colorpal2.cpt"
    write(600,'(a)')"echo 'F 255 0 0' >> OUTPUT/GMT/colorpal2.cpt"
    ! -------                                                 ---------     . pour la pondération
    write(600,'(a)')"makecpt -Cno_green -T0/1/.01 -N > OUTPUT/GMT/colorpal3.cpt"
    write(600,'(a)')"pp=0/171/235"                                              ! couleur cyan
    write(600,'(a)')"ss=255/0/255"                                              ! couleur fushia
    ! -------                                                 ---------     . pour le diagramme de densité du gif
    write(600,'(a)')"makecpt -Cno_green -T1/95/.1 -N > OUTPUT/GMT/colorpal5.cpt"
    write(600,'(a)')"echo 'N 255 255 255' >> OUTPUT/GMT/colorpal5.cpt"
    write(600,'(a)')"echo 'B 255 255 255' >> OUTPUT/GMT/colorpal5.cpt"
    write(600,'(a)')"echo 'F 255 0 0' >> OUTPUT/GMT/colorpal5.cpt"
    ! ------------------------------------------------------------------     . pour les tirs de carrières
    write(600,'(a)')"makecpt -Chot -T-3/3/1 > OUTPUT/GMT/colortir.cpt"
    ! ------------------------------------------------------------------     .
    ! tri des séismes dans l'ordre :
    do i=1,nbseismes
      datetriseismes(i)=dp%temps_ref(i)
    enddo
```

```
127     do j=1,nbseismes
128       triseismes(j)=1
129       do i=1,nbseismes
130         call difftime(delta,datetriseismes(i),datetriseismes(j))
131         if (delta.lt.0.0_wr) triseismes(j)=triseismes(j)+1
132       enddo
133     enddo
134     j=-1
135     do i=1,nbseismes
136       if (triseismes(i)==1)j=i
137     enddo
138     ! ------------------------------------------------------------------     . modèle de référence : best modèle
139     !param_best%VC=dp%VC%best
140     !param_best%VM=dp%VM%best
141     !param_best%Zmoho=dp%Zmoho%best
142     !param_best%VpVs=dp%VpVs%best
143     !do i=1,nbseismes
144     !   param_best%Zhypo(i)=dp%Zhypo(i)%best
145     !   param_best%lon(i)=dp%lon(i)%best
146     !   param_best%lat(i)=dp%lat(i)%best
147     !   param_best%Tzero(i) = dp%temps_ref(i)
148     !   param_best%Tzero(i)%sec = dp%Tzero(i)%best
149     !   call basetime(param_best%Tzero(i))
150     !enddo
151     ! (à modifier aussi dans mkwada.f90 et mklatex.f90)
152     ! --------                                                 ---------     . modèle de référence : 100 best modèle
153     param_best%VC=dp%VC%moy_100
154     param_best%VM=dp%VM%moy_100
155     param_best%Zmoho=dp%Zmoho%moy_100
156     param_best%VpVs=dp%VpVs%moy_100
157     do i=1,nbseismes
158       param_best%Zhypo(i)=dp%Zhypo(i)%moy_100
159       param_best%lon(i)=dp%lon(i)%moy_100
160       param_best%lat(i)=dp%lat(i)%moy_100
161       param_best%Tzero(i) = dp%temps_ref(i)
162       param_best%Tzero(i)%sec = dp%Tzero(i)%moy_100
163       call basetime(param_best%Tzero(i))
164     enddo
165     ! --------                                                 ---------     . modèle de référence : un modèle fixe
166     !param_best%VC=6.0_wr
167     !param_best%VM=8.0_wr
168     !param_best%Zmoho=30.0_wr
169     !param_best%VpVs=1.71_wr
170     !do i=1,nbseismes
171     !   param_best%Zhypo(i)=15.0_wr
172     !   param_best%lon(i)=-2.25_wr
173     !   param_best%lat(i)=48.25_wr
174     !   param_best%Tzero(i) = dp%temps_ref(i)
175     !   param_best%Tzero(i)%sec = 30.0_wr
176     !   call basetime(param_best%Tzero(i))
177     !enddo ! (à modifier aussi dans mkwada.f90 et mklatex.f90)
178     ! ------------------------------------------------------------------     .
179     !write(*,*)param_best
180     ! ------------------------------------------------------------------     .
181     call tempsTheoDirect(nbtps,param_best,D,critique,acentroid)
182     call mvPall_2_P1(aparambest,param_best,1)
183     ! ------------------------------------------------------------------     . production des scripts pour chaque couple de parametres
184     i=0
185     ! ------------------------------------------------------------------     . pour VC versus VpVs
186     if(plot) call GMT_2paramplot(i,dp%VC,dp%VpVs,dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
187     ! ------------------------------------------------------------------     . pour VM versus Zmoho
188     if(plot) call GMT_2paramplot(i,dp%VM,dp%Zmoho,dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
189     ! ------------------------------------------------------------------     .
190     do i=1,nbseismes
191       call  mvPall_2_P1(aparambest,param_best,i)
```

```
    ! ------------------------------------------------------------------ . pour lon versus lat (format carte)
    if(plot) call GMT_2paramplot(i,dp%lon(i),dp%lat(i),dp%mis,dp%deltaxy,dp%nbparam, &
      nbChaineMV,dp%temps_ref,nmod,aparambest,t=triseismes,E=Ellips(i))
    ! ------------------------------------------------------------------ . pour Zhypo versus Tzero
    if(plot) call GMT_2paramplot(i,dp%Zhypo(i),dp%Tzero(i),dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
    ! ------------------------------------------------------------------ .
    !if(plot) call GMT_2paramplot(i,dp%lon(i),dp%Zhypo(i),dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
    !if(plot) call GMT_2paramplot(i,dp%Zhypo(i),dp%lat(i),dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
    !if(plot) call GMT_2paramplot(i,dp%Zhypo(i),dp%Zmoho,dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
    !if(plot) call GMT_2paramplot(i,dp%Tzero(i),dp%Zmoho,dp%mis,dp%deltaxy,dp%nbparam,nbChaineMV,dp%temps_ref,nmod,aparambest)
    ! ------------------------------------------------------------------ .
  enddo
  ! ------------------------------------------------------------------ . production d'autres scripts
  if((plot).and.(FLAGresSTA)) call GMT_resSTA(nbsta,nbtps,D,nomsta)
  ! ------- ---------- .
  do i=1,nbseismes
    write(numberfile(1:5),'(i5)')i
    call mvPall_2_P1(aparambest,param_best,i)
    call ponderation(nbtps(i),D(i)%datatps,xmin(i),xmax(i),w)
    ! ------- ---------- .
    if((plot).and.(plotposteriori)) then
      call GMT_posteriori_lonlat(i,dp)
      call GMT_posteriori(i,dp,dp%VC)
      call GMT_posteriori(i,dp,dp%VM)
      call GMT_posteriori(i,dp,dp%VpVs)
      call GMT_posteriori(i,dp,dp%Zmoho)
      call GMT_posteriori(i,dp,dp%Zhypo(i))
      call GMT_posteriori(i,dp,dp%Tzero(i))
    endif
    ! ------- ---------- .
    ! if((plot).and.(plotposteriori)) call GMT_posteriori ... atres plots
    ! ------- ---------- .
    if(plot) call GMT_map(i,xmin(i),xmax(i),dp,nbtps(i),D(i)%datatps)
    ! ------- ---------- .
    if((plot).and.(FLAG_non_tabulaire)) call GMT_moho(acentroid,i,nbtps(i),xmax(i),D(i)%datatps,aparambest)
    ! ------- ---------- .
    if(plot) call GMT_res(i,xmax(i),dp,nbtps(i),D(i)%datatps)
    ! ------- ---------- .
    if(plot) call GMT_Hodochrone(i,nbtps(i),D(i)%datatps,aparambest,xmax(i),acentroid)
    ! ------- ---------- .
    if(plot) then
      inquire (_FILE_DIR_="DATA/sac-"//trim(adjustl(numberfile)),exist=existe1) ! option différente selon compilo !
      if ((existe1).and.(tracessac)) call GMT_coda(i,nbtps(i),D(i)%datatps, &
      aparambest,xmax(i),dp%lon(i)%vec10000(1,1),dp%lat(i)%vec10000(1,1),acentroid)
    endif
    ! ------- ---------- .
    if(plot) call GMT_carriere(i,dp%lon(i)%vec10000(1,1),dp%lat(i)%vec10000(1,1),dp%temps_ref(i))
    ! ------- ---------- .
  enddo
  ! ------- ---------- .
  if(plot) call GMT_chatelain(nbtps,nbsta,D,dp)
  ! ------- ---------- .
  if(plot) call GMT_wadati(nbtps,D,param_best,dp)
  ! ------- ---------- .
  if((plot).and.(plotgraph)) call GMT_fc(dp,nbChaineMV,nmod)
  ! ------- ---------- .
  if((plot).and.(plotgraph)) call GMT_param(dp,nbChaineMV,nmod)
  ! ------- ---------- .
  if(plot) call GMT_mCorr(dp)
  ! ------------------------------------------------------------------ .
  write(600,'(a8)')"echo ' '"
  write(600,'(a58)')"echo '-----------------------------------------------------------'"
  write(600,'(a58)')"echo '-------------------  fin script GMT -----------------------'"
  write(600,'(a58)')"echo '-----------------------------------------------------------'"
  write(600,'(a8)')"echo ' '"
```

```
257     write(600,'(a)')"rm -rf OUTPUT/GMT/*.ps OUTPUT/GMT/*.eps"
258     close(600)
259     close(601)
260     ! ------------------------------------------------------------------     .
261
262     CONTAINS
263
264       ! ------------------------------------------------------------------     .
265
266     subroutine GMT_2paramplot(mm,param1,param2,mis,deltaxy,nbparam,nbChaineMV,temps_ref,nmod,param_best,t,E)
267       ! -------                                                 ---------     .mh
268       ! production d'une partie du script GMT pour le diagramme de densité,
269       ! les distributions de probabilités marginalles et la représentation 2D de la fonction coût,
270       ! d'un couple de paramètres (param1,param2)
271       ! script différent si param1 = lon et param2 = lat -> carte
272       ! ------------------------------------------------------------------     .
273       use typetemps
274       use time
275       use datalecture
276       ! -------                                                 ---------     .
277       implicit none
278       ! -------                                                 ---------     .
279       integer(KIND=wi), intent (in) :: mm
280       type(densityplot_one), intent (inout) :: param1, param2, mis          ! les deux paramètres
281       type(date_sec), intent (in) :: temps_ref(nbseismes)                   ! temps de réference (si param1 = Tzéro ou param2 = Tzéro)
282       integer(KIND=wi), intent (in) :: deltaxy                              ! nombre de discrétisations pour le mode et le diagramme de densité
283       integer(KIND=wi), intent (in) :: nbparam                              ! nombre de modèles
284       integer(KIND=wi), intent (in) :: nbChaineMV                           ! nombre de chaînes
285       integer(KIND=wi), intent (in) :: nmod(nbChaineMV)                     ! nombre de modèles par chaîne
286       ! -------                                                 ---------     .
287       type(ellip), intent(out), optional :: E                               ! ellipse
288       integer(KIND=wi), intent (in), optional :: t(nbseismes)
289       ! -------                                                 ---------     .
290       integer(KIND=wi) :: i,j,k,l,m,ok
291       ! -------                                                 ---------     .
292       integer(KIND=wi), parameter :: deltaxymis = 1000                      ! nombre de discrétisations pour la représentation 2D de la fonction coût
293       type(stations) :: datasta                                             ! propriétés d'une une station
294       ! -------                                                 ---------     .
295       real(KIND=wr) :: diff1,diff2,d_diff                                   ! pour les échelles
296       real(KIND=wr) :: themax,themin,minmax1,minmax2,val,val1,val2,X,Y
297       real(KIND=wr) :: delta_1,delta_2                                      ! pas de discrétisation pour la représentation 2D de la fonction coût
298       ! -------                                                 ---------     . quelques chaînes de caractères
299       real(KIND=wr) :: moyBAZ,sumBAZ,baz,p_a,p_b,p_c,dist,bazV(360),bazVbis(360)
300       real(KIND=wr) :: tl
301       ! -------                                                 ---------     .
302       integer(KIND=wi) :: Noldtime, Nnewtime, ratetime
303       integer(KIND=wi) :: find                                              ! séisme trouvé dans le catalogue (find=1 ou 2)
304       ! -------                                                 ---------     .
305       character(LEN=30) :: char_0
306       character(LEN=13) :: char_1,char_2,char_3,char_4
307       character(LEN=10) :: char_map(4)
308       character(LEN=5) :: char_5
309       character(LEN=2) :: char_6
310       character(LEN=7) :: filename                                          ! base du nom des fichiers de sorties
311       type(parametre) :: param_best
312       logical :: existe1, existe2, existe3, existe4, findtest               ! a priori dispo (existe), séisme trouvé dans le catalogue
313       type(seismes) :: refseisme(2)
314       character (LEN=5) :: numberfile
315       ! ------------------------------------------------------------------     . initialisation
316       findtest=.false.
317       ! ------------------------------------------------------------------     .
318       m=mm
319       filename=param1%name//"_"//param2%name                                 ! nom des fichiers de sorties
320       if (m==0) then
321       m=1
```

```
322           write(numberfile(1:5),'(i5)')m
323           do i=1,nbseismes
324             write(601,'(i6,1x,a)')i,filename//"-"//trim(adjustl(numberfile))//".pdf"
325           enddo
326         else
327         write(numberfile(1:5),'(i5)')m
328         write(601,'(i6,1x,a)')m,filename//"-"//trim(adjustl(numberfile))//".pdf"
329         endif
330         ! ------------------------------------------------------------------      .
331         call mkdensityplot(m,param1,param2,deltaxy,nbparam,filename)          ! créer la grille de densité
332         ! ------------------------------------------------------------------      .
333         ! identifie les points permettant la représentation de la fonction coût
334         call mkfcoutplot(m,param1,param2,mis,deltaxymis,nbparam,filename,delta_1,delta_2)
335         ! ------------------------------------------------------------------      .
336         ! -------- ecriture dans un fichier du vecteur 1, 2 et misfit  ---      .
337         open(unit=603,file="OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_v12.bin", &
338              STATUS="replace",access='direct',RECL=24)
339         do i=1,10000
340           write(603,rec=i)real(param1%vec10000(i,1),8),real(param2%vec10000(i,1),8),real(param1%vec10000(i,2),8)
341         enddo
342         close(603)
343         ! -------- ecriture dans un fichier du vecteur 1, 2 et misfit  ---      .
344         open(unit=604,file="OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_tot.bin", &
345              STATUS="replace",access='direct',RECL=24)
346         do i=1,nbparam
347           write(604,rec=i)real(param1%vec(i),8),real(param2%vec(i),8),real(mis%vec(i),8)
348         enddo
349         close(604)
350         ! -------- ecriture du script GMT                          ---------      .
351         write(*,'(2a)')" ecriture du script GMT pour ",filename
352         write(600,'(a)')"BEFORE=$SECONDS"
353         call system_clock(Noldtime)
354         write(600,'(a)')"#######################################"
355         write(600,'(a)')"#######################################"
356         write(600,*)
357         write(600,'(2a)')"echo 'execution du script GMT pour '",filename
358         write(600,'(a)')"###########################################"
359         write(600,'(3a)')"#####     density plot : ",filename,"     ####"
360         write(600,'(a)')"###########################################"
361         write(numberfile(1:5),'(i5)')m
362         write(600,'(a)')"file=OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".ps"
363
364         ! ------------------------------------------------------------------      .
365         ! ------------------------------------------------------------------      .
366         ! -------- production du script si différent d'une carte  ---------      .
367         ! ------------------------------------------------------------------      .
368         ! ------------------------------------------------------------------      .
369
370         if (.not.((param1%name.eq."lon").and.(param2%name.eq."lat"))) then
371           write(600,'(a)')"geoproj=-JX5i/5i"                                  ! système de projection
372           write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",param1%themin,"/",param1%themax,"/", &
373           param2%themin,"/",param2%themax                                     ! bornes minimales et maximales
374           write(600,'(a)')"############## xyz -> grid #############"
375           ! -------- grille pour le diagramme de densité             ---------      .
376           if ((param1%name.eq."lon").and.(param2%name.eq."_zh")) then
377             write(600,'(a19,E13.7,a1,E13.7,a2)')"xyz2grd $geozone -I",param1%delta*1.01_wr,"/",param2%delta*1.01_wr," \"
378             write(600,'(a)')" OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".bin -bi3d -Nnan \"
379             write(600,'(a)')" -GOUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd "
380           elseif ((param1%name.eq."_zh").and.(param2%name.eq."lat")) then
381             write(600,'(a19,E13.7,a1,E13.7,a2)')"xyz2grd $geozone -I",param1%delta*1.01_wr,"/",param2%delta*1.01_wr," \"
382             write(600,'(a)')" OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".bin -bi3d -Nnan \"
383             write(600,'(a)')" -GOUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd "
384           else
385             write(600,'(a19,E13.7,a1,E13.7,a2)')"xyz2grd $geozone -I",param1%delta*1.01_wr,"/",param2%delta*1.01_wr," \"
386             write(600,'(a)')" OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".bin -bi3d -Nnan \"
```

```
387           write(600,'(a)')" -GOUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd "
388         endif
389         ! -------- grille pour la fonction coût                    --------     .
390         write(600,'(a19,E13.7,a1,E13.7,a2)')"xyz2grd $geozone -I",delta_1*1.5_wr,"/",delta_2*1.5_wr," \"
391         write(600,'(a)')" OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"mis.bin -bi3d -Nnan \"
392         write(600,'(a)')" -GOUTPUT/GMT/topo2_"//trim(adjustl(numberfile))//".grd "
393         ! ---------------------------------------------------------     .
394         ! -------- choix des échelles et incréments                --------     .
395         d_diff=(param1%themax-param1%themin)/4.5_wr
396         if (d_diff.gt.0.0001_wr) diff1=real(int(d_diff*100000.0_wr,wi),wr)/100000.0_wr
397         if (d_diff.gt.0.001_wr) diff1=real(int(d_diff*10000.0_wr,wi),wr)/10000.0_wr
398         if (d_diff.gt.0.001_wr) diff1=real(int(d_diff*1000.0_wr,wi),wr)/1000.0_wr
399         if (d_diff.gt.0.01_wr) diff1=real(int(d_diff*100.0_wr,wi),wr)/100.0_wr
400         if (d_diff.gt.0.1_wr) diff1=real(int(d_diff*10.0_wr,wi),wr)/10.0_wr
401         if (d_diff.gt.1.0_wr) diff1=real(int(d_diff*1.0_wr,wi),wr)/1.0_wr
402         if (d_diff.gt.10.0_wr) diff1=real(int(d_diff*0.1_wr,wi),wr)/0.1_wr
403         if (d_diff.gt.100.0_wr) diff1=real(int(d_diff*0.01_wr,wi),wr)/0.01_wr
404         write(char_1,'(E13.7)')diff1
405         !write(*,*)d_diff,dp%st_1,diff1
406         ! --------                                                --------     .
407         d_diff=(param2%themax-param2%themin)/4.5_wr
408         if (d_diff.gt.0.0001_wr) diff2=real(int(d_diff*100000.0_wr,wi),wr)/100000.0_wr
409         if (d_diff.gt.0.001_wr) diff2=real(int(d_diff*10000.0_wr,wi),wr)/10000.0_wr
410         if (d_diff.gt.0.001_wr) diff2=real(int(d_diff*1000.0_wr,wi),wr)/1000.0_wr
411         if (d_diff.gt.0.01_wr) diff2=real(int(d_diff*100.0_wr,wi),wr)/100.0_wr
412         if (d_diff.gt.0.1_wr) diff2=real(int(d_diff*10.0_wr,wi),wr)/10.0_wr
413         if (d_diff.gt.1.0_wr) diff2=real(int(d_diff*1.0_wr,wi),wr)/1.0_wr
414         if (d_diff.gt.10.0_wr) diff2=real(int(d_diff*0.1_wr,wi),wr)/0.1_wr
415         if (d_diff.gt.100.0_wr) diff2=real(int(d_diff*0.01_wr,wi),wr)/0.01_wr
416         write(char_2,'(E13.7)')diff2
417         ! ---------------------------------------------------------     .
418         write(char_3,'(E13.7)')param1%delta
419         write(char_4,'(E13.7)')param2%delta
420         ! ---------------------------------------------------------     .
421         write(600,'(a)')"#########################################"
422         write(600,'(a)')"############ diag. fct coût #############"
423         write(600,'(a)')"#########################################"
424         write(600,'(2a)')"psbasemap $geozone $geoproj -Ba"//char_1//":""""//param1%char//"""":/a"//char_2//":""""// &
425         param2%char//"""":WenS -K -X2.5i -Yc >  $file"
426         write(600,'(a)')"########### affiche les points ##########"
427         write(600,'(2a)')"psxy $geozone $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"mis.bin -bi3d", &
428         " -Sc0.015i -COUTPUT/GMT/colorpal1.cpt -O -K >> $file"
429         write(600,'(a)')"############ contour densité ############"
430         ! ---------------------------------------------------------     .
431         if ((param1%name.eq."lon").and.(param2%name.eq."_zh")) then
432           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd $geozone ", &
433           "$geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
434           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd $geozone ", &
435           "$geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
436           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd $geozone ", &
437           "$geoproj -Ba0 -C25 -L24/25 -A+s15 -W2 -O -K >> $file"
438         elseif ((param1%name.eq."_zh").and.(param2%name.eq."lat")) then
439           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd $geozone ", &
440           "$geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
441           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd $geozone ", &
442           "$geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
443           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd $geozone ", &
444           "$geoproj -Ba0 -C25 -L24/25 -A+s15 -W2 -O -K >> $file"
445         else
446           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd $geozone ", &
447           "$geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
448           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd $geozone ", &
449           "$geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
450           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd $geozone ", &
451           "$geoproj -Ba0 -C25 -L24/25 -A+s15 -W2 -O -K >> $file"
```

```
452         endif
453         ! ------------------------------------------------------------     .
454         write(600,'(a)')"###### moy modèles des chaînes  1 #######"
455         ! -------- affiche la moyennes de l'ensemble du meilleur modèle de chaque chaîne
456         minmax1 = param1%moy_bestchaine - param1%ec_bestchaine
457         minmax2 = param1%moy_bestchaine + param1%ec_bestchaine
458         themin = param2%themin  + 0.035_wr * (param2%themax - param2%themin)
459         themax = param2%themax  - 0.035_wr * (param2%themax - param2%themin)
460         write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%moy_bestchaine,themin," \n ",param1%moy_bestchaine,themax, &
461         """ | psxy $geozone $geoproj -W3,grey -O -K >> $file"
462         minmax1 = param2%moy_bestchaine - param2%ec_bestchaine
463         minmax2 = param2%moy_bestchaine + param2%ec_bestchaine
464         themin = param1%themin  + 0.035_wr * (param1%themax - param1%themin)
465         themax = param1%themax  - 0.035_wr * (param1%themax - param1%themin)
466         write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,themin,param2%moy_bestchaine," \n ",themax,param2%moy_bestchaine, &
467         """ | psxy $geozone $geoproj -W3,grey -O -K >> $file"
468         write(600,'(a)')"######### 10000 meilleurs modèles #######"
469         ! -------- affiche les 10000 meilleurs modèles          --------     .
470         ! write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_v12.bin -Sc0.001i -O -K -bi3d >> $file"
471         write(600,'(a)')"############ barre de couleur ############"
472         ! -------- affiche la barre de couleur                  --------     .
473         write(600,'(2a)')"psscale -D1/-1/0.50E+01/0.25ch -B25.0:""fonction co\373t"": -S -I ", &
474         "-COUTPUT/GMT/colorpal1.cpt -O -K >> $file"
475         write(600,'(a)')"########## 10 meilleurs modèles ##########"
476         ! -------- affiche les 10 meilleurs modèles (tous différents) sous la forme d'étoiles jaunes.
477         i=1
478         l=1
479         write(600,'(a,2f15.5,a)')"echo ",param1%vec10000(i,1),param2%vec10000(i,1), &
480         "| psxy $geozone $geoproj  -Sa0.1i -Gyellow -Wthinnest,black -O -K >> $file"
481         do while(l.lt.10)
482           i=i+1
483           if (mis%vec10000(i,1).ne.mis%vec10000(i-1,1)) then
484             l=l+1
485             write(600,'(a,2f15.5,a)')"echo ",param1%vec10000(i,1),param2%vec10000(i,1), &
486             "| psxy $geozone $geoproj  -Sa0.1i -Gyellow -Wthinnest,black -O -K >> $file"
487           endif
488         enddo
489         ! --------                                                --------     .
490         write(600,'(a)')"psbasemap $geozone $geoproj -Ba0 -K -O >>  $file"
491         write(600,'(a)')"###### moy des meilleurs modèles ########"
492         ! -------- affiche la moyenne +ou- un ecart-type des 10000, 1000 et 100 meilleurs modèles, avec des flêches sur les côtés
493         minmax1 = 0.015_wr * (param1%themax - param1%themin)
494         minmax2 = 0.015_wr * (param2%themax - param2%themin)
495         ! --------                                                --------     .
496         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_10000+param1%ec_10000, &
497         param2%themin+minmax2,param1%moy_10000-param1%ec_10000,param2%themin+minmax2, &
498         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
499         write(600,'(a,4f15.5,a)')"echo -e """,param1%themin+minmax1, &
500         param2%moy_10000+param2%ec_10000,param1%themin+minmax1,param2%moy_10000-param2%ec_10000, &
501         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
502         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_1000+param1%ec_1000, &
503         param2%themin+minmax2,param1%moy_1000-param1%ec_1000,param2%themin+minmax2, &
504         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
505         write(600,'(a,4f15.5,a)')"echo -e """,param1%themin+minmax1, &
506         param2%moy_1000+param2%ec_1000,param1%themin+minmax1,param2%moy_1000-param2%ec_1000, &
507         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
508         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_100+param1%ec_100, &
509         param2%themin+minmax2,param1%moy_100-param1%ec_100,param2%themin+minmax2, &
510         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
511         write(600,'(a,4f15.5,a)')"echo -e """,param1%themin+minmax1, &
512         param2%moy_100+param2%ec_100,param1%themin+minmax1,param2%moy_100-param2%ec_100, &
513         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
514         ! --------                                                --------     .
515         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_10000+param1%ec_10000, &
516         param2%themax-minmax2,param1%moy_10000-param1%ec_10000,param2%themax-minmax2, &
```

```
517         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
518         write(600,'(a,4f15.5,a)')"echo -e """,param1%themax-minmax1, &
519         param2%moy_10000+param2%ec_10000,param1%themax-minmax1, param2%moy_10000-param2%ec_10000, &
520         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
521         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_1000+param1%ec_1000, &
522         param2%themax-minmax2,param1%moy_1000-param1%ec_1000,param2%themax-minmax2, &
523         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
524         write(600,'(a,4f15.5,a)')"echo -e """,param1%themax-minmax1, &
525         param2%moy_1000+param2%ec_1000,param1%themax-minmax1,param2%moy_1000-param2%ec_1000, &
526         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
527         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_100+param1%ec_100, &
528         param2%themax-minmax2,param1%moy_100-param1%ec_100,param2%themax-minmax2, &
529         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
530         write(600,'(a,4f15.5,a)')"echo -e """,param1%themax-minmax1, &
531         param2%moy_100+param2%ec_100,param1%themax-minmax1,param2%moy_100-param2%ec_100, &
532         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
533         ! -------- affiche la moyennes de l'ensemble du meilleur modèle de chaque chaîne
534         write(600,'(a)')"###### moy modèles des chaînes 2 #######"
535         minmax1 = param1%moy_bestchaine - param1%ec_bestchaine
536         minmax2 = param1%moy_bestchaine + param1%ec_bestchaine
537         themin = param2%themin  + 0.035_wr * (param2%themax - param2%themin)
538         themax = param2%themax  - 0.035_wr * (param2%themax - param2%themin)
539         write(600,'(a,2f15.5,a)')"echo ",minmax1,themin," 0 0.1i | psxy $geozone $geoproj \"
540         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
541         write(600,'(a,2f15.5,a)')"echo ",minmax2,themin," 0 0.1i | psxy $geozone $geoproj \"
542         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
543         write(600,'(a,2f15.5,a)')"echo ",param1%moy_bestchaine,themin," 0 0.1i | psxy $geozone $geoproj \"
544         write(600,'(a)')"-SV0.04i/0.05i/0.04i -Gred -Wthinnest,black -O -K >> $file"
545         write(600,'(a,2f15.5,a)')"echo ",minmax1,themax," 180 0.1i | psxy $geozone $geoproj \"
546         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
547         write(600,'(a,2f15.5,a)')"echo ",minmax2,themax," 180 0.1i | psxy $geozone $geoproj \"
548         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
549         write(600,'(a,2f15.5,a)')"echo ",param1%moy_bestchaine,themax," 180 0.1i | psxy $geozone $geoproj \"
550         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gred -Wthinnest,black -O -K >> $file"
551         minmax1 = param2%moy_bestchaine - param2%ec_bestchaine
552         minmax2 = param2%moy_bestchaine + param2%ec_bestchaine
553         themin = param1%themin  + 0.035_wr * (param1%themax - param1%themin)
554         themax = param1%themax  - 0.035_wr * (param1%themax - param1%themin)
555         write(600,'(a,2f15.5,a)')"echo ",themin,minmax1," 90 0.1i | psxy $geozone $geoproj \"
556         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
557         write(600,'(a,2f15.5,a)')"echo ",themin,minmax2," 90 0.1i | psxy $geozone $geoproj \"
558         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
559         write(600,'(a,2f15.5,a)')"echo ",themin,param2%moy_bestchaine," 90 0.1i | psxy $geozone $geoproj \"
560         write(600,'(a)')"-SV0.04i/0.05i/0.04i -Gred -Wthinnest,black -O -K >> $file"
561         write(600,'(a,2f15.5,a)')"echo ",themax,minmax1," 270 0.1i | psxy $geozone $geoproj \"
562         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
563         write(600,'(a,2f15.5,a)')"echo ",themax,minmax2," 270 0.1i | psxy $geozone $geoproj \"
564         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
565         write(600,'(a,2f15.5,a)')"echo ",themax,param2%moy_bestchaine," 270 0.1i | psxy $geozone $geoproj \"
566         write(600,'(a)')"-SV0.04i/0.06i/0.04i -Gred -Wthinnest,black -O -K >> $file"
567         ! -------- affiche le temps de réference                 --------    .
568         if((param1%name.eq."_to").or.(param2%name.eq."_to")) then
569           write(600,'(a)')"############ temps réference ############"
570           call affiche_temps_ref(temps_ref(m),char_0,-1)
571           X = param1%themin + 0.3_wr*(param1%themax-param1%themin)
572           Y = param2%themin + 0.05_wr*(param2%themax-param2%themin)
573           write(600,'(a,2f15.5,a)')"echo """,X,Y," 15 0 5 6 "//char_0//""" | pstext $geozone $geoproj -O -K >> $file"
574         endif
575         write(600,'(a)')"#########################################"
576         write(600,'(a)')"############# diag. densité #############"
577         write(600,'(a)')"#########################################"
578         write(600,'(2a)')"psbasemap $geozone $geoproj -Ba"//char_1//":""""//param1%char//"""":/a"//char_2//":""""// &
579         param2%char//"""":WenS -K -O -X6.25i >>  $file"
580         write(600,'(a)')"################ grid image ##############"
581         ! --------                                              --------    .
```

```
582         if ((param1%name.eq."lon").and.(param2%name.eq."_zh")) then
583           write(600,'(2a)')"grdimage $geozone $geoproj OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd ",&
584           "-Qnan -COUTPUT/GMT/colorpal2.cpt -B0 -O -K -Sn >> $file"
585           write(600,'(a)')"############### grid contour ##############"
586           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd ", &
587           "$geozone $geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
588           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd ", &
589           "$geozone $geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
590           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".lon_zh.grd ", &
591           "$geozone $geoproj -Ba0 -C25 -L24/25 -A+s15 -W2 -O -K >> $file"
592         elseif ((param1%name.eq."_zh").and.(param2%name.eq."lat")) then
593           write(600,'(2a)')"grdimage $geozone $geoproj OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd ",&
594           "-Qnan -COUTPUT/GMT/colorpal2.cpt -B0 -O -K -Sn >> $file"
595           write(600,'(2a)')"############### grid contour ##############"
596           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd ", &
597           "$geozone $geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
598           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd ", &
599           "$geozone $geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
600           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".zh_lat.grd ", &
601           "$geozone $geoproj -Ba0 -C25 -L24/25 -A+s15 -W2 -O -K >> $file"
602         else
603           write(600,'(2a)')"grdimage $geozone $geoproj OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd ",&
604           "-Qnan -COUTPUT/GMT/colorpal2.cpt -B0 -O -K -Sn >> $file"
605           write(600,'(a)')"############### grid contour ##############"
606           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd ", &
607           "$geozone $geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
608           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd ", &
609           "$geozone $geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
610           write(600,'(2a)')"grdcontour OUTPUT/GMT/topo1_"//trim(adjustl(numberfile))//".grd ", &
611           "$geozone $geoproj -Ba0 -C25 -L24/25 -A+s15 -W2 -O -K >> $file"
612         endif
613         ! _______                                                 ________     .
614         write(600,'(a)')"psbasemap $geozone $geoproj -Ba0 -K -O >>  $file"
615         write(600,'(a)')"psscale -D1/-1/0.50E+01/0.25ch -B25.0:""densit\351"": -S -I -COUTPUT/GMT/colorpal2.cpt -O -K >> $file"
616         call catalogue(param_best,refseisme,find)
617         ! _______                                                 ________     . 1 séisme
618         if ((find==1).or.(find==2)) then
619           findtest=.false.
620           write(600,'(a)')"################### catalogue 1 #################"
621           if (param1%name.eq."lon") then
622             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,refseisme(1)%lon,param2%themin+0.001_wr," \n ", &
623             refseisme(1)%lon,param2%themax-0.001_wr," "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
624             findtest=.true.
625           endif
626           if(param2%name.eq."lon") then
627             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,refseisme(1)%lon," \n ", &
628             param1%themax-0.001_wr,refseisme(1)%lon, " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
629             findtest=.true.
630           endif
631           if(param1%name.eq."lat") then
632             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,refseisme(1)%lat,param2%themin+0.001_wr," \n ",refseisme(1)%lat, &
633             param2%themax-0.001_wr, " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
634             findtest=.true.
635           endif
636           if(param2%name.eq."lat") then
637             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,refseisme(1)%lat," \n ",param1%themax-0.001_wr, &
638             refseisme(1)%lat, " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
639             findtest=.true.
640           endif
641           if(param1%name.eq."_zh") then
642             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,refseisme(1)%pfd,param2%themin+0.001_wr," \n ",refseisme(1)%pfd, &
643             param2%themax-0.001_wr, " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
644             findtest=.true.
645           endif
646           if(param2%name.eq."_zh")then
```

```
647             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,refseisme(1)%pfd," \n ",param1%themax-0.001_wr, &
648             refseisme(1)%pfd, " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
649             findtest=.true.
650           endif
651           if(param1%name.eq."_to") then
652             call difftime(val,refseisme(1)%tps_init,temps_ref(m))
653             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,val,param2%themin+0.001_wr," \n ",val,param2%themax-0.001_wr, &
654             " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
655             findtest=.true.
656           endif
657           if(param2%name.eq."_to") then
658             call difftime(val,refseisme(1)%tps_init,temps_ref(m))
659             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,val," \n ",param1%themax-0.001_wr,val, &
660             " "" | psxy $geozone $geoproj -W4,orange,- -O -K -N >> $file"
661             findtest=.true.
662           endif
663           write(600,'(a)')"#########################################"
664         endif
665         ! _______                                                 ________     . 2 séismes
666         if (find==2) then
667           findtest=.false.
668           write(600,'(a)')"################## catalogue 2 ################"
669           if (param1%name.eq."lon") then
670             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,refseisme(2)%lon,param2%themin+0.001_wr," \n ", &
671             refseisme(2)%lon,param2%themax-0.001_wr, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
672             findtest=.true.
673           endif
674           if(param2%name.eq."lon") then
675             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,refseisme(2)%lon," \n ", &
676             param1%themax-0.001_wr,refseisme(2)%lon, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
677             findtest=.true.
678           endif
679           if(param1%name.eq."lat") then
680             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,refseisme(2)%lat,param2%themin+0.001_wr," \n ", &
681             refseisme(2)%lat,param2%themax-0.001_wr, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
682             findtest=.true.
683           endif
684           if(param2%name.eq."lat") then
685             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,refseisme(2)%lat," \n ", &
686             param1%themax-0.001_wr,refseisme(2)%lat, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
687             findtest=.true.
688           endif
689           if(param1%name.eq."_zh") then
690             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,refseisme(2)%pfd,param2%themin+0.001_wr," \n ", &
691             refseisme(2)%pfd,param2%themax-0.001_wr, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
692             findtest=.true.
693           endif
694           if(param2%name.eq."_zh")then
695             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,refseisme(2)%pfd," \n ", &
696             param1%themax-0.001_wr,refseisme(2)%pfd, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
697             findtest=.true.
698           endif
699           if(param1%name.eq."_to") then
700             call difftime(val,refseisme(2)%tps_init,temps_ref(m))
701             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,val,param2%themin+0.001_wr," \n ",val, &
702             param2%themax-0.001_wr, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
703             findtest=.true.
704           endif
705           if(param2%name.eq."_to") then
706             call difftime(val,refseisme(2)%tps_init,temps_ref(m))
707             write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%themin+0.001_wr,val," \n ", &
708             param1%themax-0.001_wr,val, " "" | psxy $geozone $geoproj -W4,orange,-.- -O -K -N >> $file"
709             findtest=.true.
710           endif
711           write(600,'(a)')"#########################################"
```

```
712         endif
713       ! ------------------------------------------------------------------      .
714       ! ------------------------------------------------------------------      .
715
716       else
717
718       ! ------------------------------------------------------------------      .
719       ! ------------------------------------------------------------------      .
720       ! -------- carte pour les paramètres lon et lat            ----------     .
721       ! ------------------------------------------------------------------      .
722       ! ------------------------------------------------------------------      .
723
724         ! ------------------------------------------------------------------    .
725         ! ------ calcul de l'ellipse des 1000 meilleurs modèles ----------     .
726         ! ------------------------------------------------------------------    . baz moyen [0:360]
727         bazV(:)=0.0_wr
728         open(unit=101,file="OUTPUT/GMT/ellipse-"//trim(adjustl(numberfile))//".txt")
729         do i=1,1000
730           write(101,*)param1%vec10000(i,1),param2%vec10000(i,1),param1%vec10000(i,2)
731           call dellipsgc(param2%moy_1000,param1%moy_1000,param2%vec10000(i,1),param1%vec10000(i,1),dist,baz)
732             do j=1,360
733                 if ((baz.gt.real(j-1,wr)).and.(baz.le.real(j,wr))) bazV(j)=bazV(j)+dist**2
734             enddo
735         enddo
736         close(101)
737         ! ------------------------------------------------------------------    . lissage baz
738         do i=1,100
739           ! ------------------------------------------------------------------  .
740           bazVbis(1)=bazV(1)+(bazV(360)+bazV(2))/2.0_wr
741           do j=2,359
742             bazVbis(j)=bazV(j)+(bazV(j-1)+bazV(j+1))/2.0_wr
743           enddo
744           bazVbis(360)=bazV(360)+(bazV(359)+bazV(1))/2.0_wr
745           ! ------------------------------------------------------------------  .
746           bazV(1)=bazVbis(1)+(bazVbis(360)+bazVbis(2))/2.0_wr
747           do j=2,359
748             bazV(j)=bazVbis(j)+(bazVbis(j-1)+bazVbis(j+1))/2.0_wr
749           enddo
750           bazV(360)=bazVbis(360)+(bazVbis(359)+bazVbis(1))/2.0_wr
751           ! ------------------------------------------------------------------  .
752         enddo
753         ! ------------------------------------------------------------------    . sélection baz
754         sumBAZ=-1.0_wr
755         do j=1,360
756           if (bazV(j).gt.sumBAZ) then
757             moyBAZ=real(j,wr)-0.5_wr
758             sumBAZ=bazV(j)
759           endif
760         enddo
761         ! ------------------------------------------------------------------    .
762         ! ------------------------------------------------------------------    . axes a et b, demi axes [km], plus o moins 1 sigma
763         p_a=0.0_wr
764         p_b=0.0_wr
765         do i=1,1000
766           call dellipsgc(param2%moy_1000,param1%moy_1000,param2%vec10000(i,1),param1%vec10000(i,1),dist,baz)
767           p_c = min( abs(mod(moyBAZ-BAZ,90.0_wr)), &
768                      abs(mod(mod(moyBAZ,180.0_wr)-BAZ,90.0_wr)), &
769                      abs(mod(moyBAZ-mod(BAZ,180.0_wr),90.0_wr)), &
770                      abs(mod(mod(moyBAZ,180.0_wr)-mod(BAZ,180.0_wr),90.0_wr)))
771           p_a = p_a + (cos(p_c/180.0_wr*pi)*dist)**2.0_wr                    ! ecartype de l'axe a, moyenne nulle [km]
772           p_b = p_b + (sin(p_c/180.0_wr*pi)*dist)**2.0_wr                    ! ecartype de l'axe b, moyenne nulle [km]
773         enddo
774         p_a=sqrt(p_a/real(1000,wr))
775         p_b=sqrt(p_b/real(1000,wr))
776         ! ------------------------------------------------------------------    . axe a > b
```

```
777         if (p_b.gt.p_a) then
778           p_c=p_b
779           p_b=p_a
780           p_a=p_c
781         endif
782         ! ------------------------------------------------------------     . sauve baz, axes a et b
783         E%ang=moyBAZ
784         E%axeA=p_a
785         E%axeB=p_b
786         ! ------------------------------------------------------------     . futures gif plots
787         open(unit=100,file="OUTPUT/GMT/doc-"//trim(adjustl(numberfile))//".txt")
788         write(100,*) t(m)
789         write(100,*) temps_ref(m)
790         write(100,*) param1%moy_1000,param2%moy_1000,E%ang,E%axeA*2.0_wr,E%axeB*2.0_wr
791         write(100,*) param1%moy_1000,param2%moy_1000,E%ang,E%axeA*4.0_wr,E%axeB*4.0_wr
792         write(100,*) param1%moy_1000,param2%moy_1000,E%ang,E%axeA*6.0_wr,E%axeB*6.0_wr
793         close(100)
794         ! ------------------------------------------------------------     .
795         write(600,'(a)')"geoproj=-JQ5i"
796         ! -------- rééquilible min /max -> carte de au moins 5 x 5 km --     .
797         ! -------- km / degree en latitude                 ---------     .
798         val = 2.0_wr * pi * rT * sin((90.0_wr-param2%vec10000(1,1))/180.0_wr*pi) /360.0_wr
799         do while(((param1%themax-param1%themin)*val).lt.8.0_wr)
800           param1%themax = param1%themax + (param1%themax-param1%themin)/100.0_wr
801           param1%themin = param1%themin - (param1%themax-param1%themin)/100.0_wr
802         enddo
803         ! -------- km / degree en longitude                ---------     .
804         val = 2.0_wr * pi * rT / 360.0_wr
805         do while(((param2%themax-param2%themin)*val).lt.8.0_wr)
806           param2%themax = param2%themax + (param2%themax-param2%themin)/100.0_wr
807           param2%themin = param2%themin - (param2%themax-param2%themin)/100.0_wr
808         enddo
809         ! -------- rééquilible min /max -> map carrée      ---------     .
810         if ((param1%themax-param1%themin).gt.(param2%themax-param2%themin)) then
811           val= (param2%themax-param2%themin)/2.0_wr
812           param2%themax = param2%themax - val + (param1%themax-param1%themin)/2.0_wr
813           param2%themin = param2%themin + val - (param1%themax-param1%themin)/2.0_wr
814         else
815           val= (param1%themax-param1%themin)/2.0_wr
816           param1%themax = param1%themax - val + (param2%themax-param2%themin)/2.0_wr
817           param1%themin = param1%themin + val - (param2%themax-param2%themin)/2.0_wr
818         endif
819         write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",param1%themin, &
820         "/",param1%themax,"/",param2%themin,"/",param2%themax
821         write(600,'(a)')"############### xyz -> grid #############"
822         write(600,'(a19,E13.7,a1,E13.7,2a)')"xyz2grd $geozone -I",param1%delta*1.01_wr,"/",param2%delta*1.01_wr, &
823         " OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".bin -bi3d -Nnan ",&
824         " -GOUTPUT/GMT/topo0_"//trim(adjustl(numberfile))//".grd "
825         write(600,'(a19,E13.7,a1,E13.7,2a)')"xyz2grd $geozone -I",delta_1*1.5_wr,"/",delta_2*1.5_wr, &
826         " OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"mis.bin -bi3d -Nnan ", &
827         " -GOUTPUT/GMT/topo2_"//trim(adjustl(numberfile))//".grd "
828         ! ------------------------------------------------------------     .
829         ! -------- choix des échelles et incréments        ---------     .
830         d_diff=(param1%themax-param1%themin)/4.0_wr
831         if (d_diff.gt.0.0001_wr) diff1=real(int(d_diff*100000.0_wr,wi),wr)/100000.0_wr
832         if (d_diff.gt.0.001_wr) diff1=real(int(d_diff*10000.0_wr,wi),wr)/10000.0_wr
833         if (d_diff.gt.0.001_wr) diff1=real(int(d_diff*1000.0_wr,wi),wr)/1000.0_wr
834         if (d_diff.gt.0.01_wr) diff1=real(int(d_diff*100.0_wr,wi),wr)/100.0_wr
835         if (d_diff.gt.0.1_wr) diff1=real(int(d_diff*10.0_wr,wi),wr)/10.0_wr
836         if (d_diff.gt.1.0_wr) diff1=real(int(d_diff*1.0_wr,wi),wr)/1.0_wr
837         if (d_diff.gt.10.0_wr) diff1=real(int(d_diff*0.1_wr,wi),wr)/0.1_wr
838         if (d_diff.gt.100.0_wr) diff1=real(int(d_diff*0.01_wr,wi),wr)/0.01_wr
839         write(char_1,'(E13.7)')diff1
840         ! --------                                          ---------     .
841         d_diff=(param2%themax-param2%themin)/4.0_wr
```

```
        if (d_diff.gt.0.0001_wr) diff2=real(int(d_diff*100000.0_wr,wi),wr)/100000.0_wr
        if (d_diff.gt.0.001_wr) diff2=real(int(d_diff*10000.0_wr,wi),wr)/10000.0_wr
        if (d_diff.gt.0.001_wr) diff2=real(int(d_diff*1000.0_wr,wi),wr)/1000.0_wr
        if (d_diff.gt.0.01_wr) diff2=real(int(d_diff*100.0_wr,wi),wr)/100.0_wr
        if (d_diff.gt.0.1_wr) diff2=real(int(d_diff*10.0_wr,wi),wr)/10.0_wr
        if (d_diff.gt.1.0_wr) diff2=real(int(d_diff*1.0_wr,wi),wr)/1.0_wr
        if (d_diff.gt.10.0_wr) diff2=real(int(d_diff*0.1_wr,wi),wr)/0.1_wr
        if (d_diff.gt.100.0_wr) diff2=real(int(d_diff*0.01_wr,wi),wr)/0.01_wr
        write(char_2,'(E13.7)')diff2
        ! ------------------------------------------------------------    .
        write(char_3,'(E13.7)')param1%delta
        write(char_4,'(E13.7)')param2%delta
        ! ------------------------------------------------------------    .
        write(600,'(a)')"#########################################"
        write(600,'(a)')"############ diag. fct coût #############"
        write(600,'(a)')"#########################################"
        write(600,'(2a)')"psbasemap $geozone $geoproj -Ba"//char_1//":""""//param1%char//"""":/a"//char_2//":""""// &
        param2%char//"""":WenS -K -X2.5i -Yc >  $file"
        write(600,'(a)')"pscoast $geozone $geoproj -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -O -K >>  $file"
        write(600,'(a)')"########## affiche les points ##########"
        write(600,'(2a)')"psxy $geozone $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"mis.bin -bi3d", &
        " -Sc0.015i -COUTPUT/GMT/colorpal1.cpt -O -K >> $file"
        write(600,'(a)')"############ contour densité ############"
        write(600,'(2a)')"grdcontour OUTPUT/GMT/topo0_"//trim(adjustl(numberfile))//".grd ", &
        "$geozone $geoproj -Ba0 -C75 -L74/75 -W2 -O -K >> $file"
        write(600,'(2a)')"grdcontour OUTPUT/GMT/topo0_"//trim(adjustl(numberfile))//".grd ", &
        "$geozone $geoproj -Ba0 -C50 -L49/50 -W2 -O -K >> $file"
        write(600,'(a)')"###### moy modèles des chaînes  1 #######"
        ! ------- affiche la moyennes de l'ensemble du meilleur modèle de chaque chaîne
        minmax1 = param1%moy_bestchaine - param1%ec_bestchaine
        minmax2 = param1%moy_bestchaine + param1%ec_bestchaine
        themin = param2%themin  + 0.035_wr * (param2%themax - param2%themin)
        themax = param2%themax  - 0.035_wr * (param2%themax - param2%themin)
        write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,param1%moy_bestchaine,themin," \n ",param1%moy_bestchaine,themax, &
        " "" | psxy $geozone $geoproj -W3,grey -O -K >> $file"
        minmax1 = param2%moy_bestchaine - param2%ec_bestchaine
        minmax2 = param2%moy_bestchaine + param2%ec_bestchaine
        themin = param1%themin  + 0.035_wr * (param1%themax - param1%themin)
        themax = param1%themax  - 0.035_wr * (param1%themax - param1%themin)
        write(600,'(a,2f15.5,a,2f15.5,a)')"echo -e """,themin,param2%moy_bestchaine," \n ",themax,param2%moy_bestchaine, &
        " "" | psxy $geozone $geoproj -W3,grey -O -K >> $file"
        write(600,'(a)')"######### 10000 meilleurs modèles #######"
        ! ------- affiche les 10000 meilleurs modèles          -------    .
        ! write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_v12.bin -Sc0.001i -O -K -bi3d >> $file"
        write(600,'(a)')"########### barre de couleur ############"
        ! ------- affiche la barre de couleur                  -------    .
        write(600,'(2a)')"psscale -D1/-1/0.50E+01/0.25ch -B25.0:""fonction co\373t"": -S -I  ", &
        "-COUTPUT/GMT/colorpal1.cpt -O -K >> $file"
        write(600,'(a)')"######### 10 meilleurs modèles ##########"
        ! ------- affiche les 10 meilleurs modèles (tous différents) sous la forme d'étoiles jaunes
        i=1
        l=1
        write(600,'(a,2f15.5,a)')"echo ",param1%vec10000(i,1),param2%vec10000(i,1), &
        "| psxy $geozone $geoproj  -Sa0.1i -Gyellow -Wthinnest,black -O -K >> $file"
        do while(l.lt.10)
          i=i+1
          if (mis%vec10000(i,1).ne.mis%vec10000(i-1,1)) then
            l=l+1
            write(600,'(a,2f15.5,a)')"echo ",param1%vec10000(i,1),param2%vec10000(i,1), &
            "| psxy $geozone $geoproj  -Sa0.1i -Gyellow -Wthinnest,black -O -K >> $file"
          endif
        enddo
        ! -------                                              -------    .
        write(600,'(a)')"psbasemap $geozone $geoproj -Ba0 -K -O >>  $file"
        write(600,'(a)')"###### moy des meilleurs modèles ########"
```

```
907         ! -------- affiche la moyenne +ou- un ecart-type des 10000, 1000 et 100 meilleurs modèles, avec des flêches sur les côtés
908         minmax1 = 0.015_wr * (param1%themax - param1%themin)
909         minmax2 = 0.015_wr * (param2%themax - param2%themin)
910         ! --------                                          --------     .
911         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_10000+param1%ec_10000, &
912         param2%themin+minmax2,param1%moy_10000-param1%ec_10000,param2%themin+minmax2, &
913         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
914         write(600,'(a,4f15.5,a)')"echo -e """,param1%themin+minmax1,param2%moy_10000+param2%ec_10000, &
915         param1%themin+minmax1,param2%moy_10000-param2%ec_10000, &
916         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
917         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_1000+param1%ec_1000,param2%themin+minmax2, &
918         param1%moy_1000-param1%ec_1000,param2%themin+minmax2, &
919         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
920         write(600,'(a,4f15.5,a)')"echo -e """,param1%themin+minmax1,param2%moy_1000+param2%ec_1000, &
921         param1%themin+minmax1,param2%moy_1000-param2%ec_1000, &
922         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
923         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_100+param1%ec_100,param2%themin+minmax2, &
924         param1%moy_100-param1%ec_100,param2%themin+minmax2, &
925         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i  -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
926         write(600,'(a,4f15.5,a)')"echo -e """,param1%themin+minmax1,param2%moy_100+param2%ec_100, &
927         param1%themin+minmax1,param2%moy_100-param2%ec_100, &
928         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
929         ! --------                                          --------     .
930         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_10000+param1%ec_10000,param2%themax-minmax2, &
931         param1%moy_10000-param1%ec_10000,param2%themax-minmax2, &
932         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
933         write(600,'(a,4f15.5,a)')"echo -e """,param1%themax-minmax1,param2%moy_10000+param2%ec_10000, &
934         param1%themax-minmax1,param2%moy_10000-param2%ec_10000, &
935         " "" | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
936         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_1000+param1%ec_1000,param2%themax-minmax2, &
937         param1%moy_1000-param1%ec_1000,param2%themax-minmax2, &
938         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
939         write(600,'(a,4f15.5,a)')"echo -e """,param1%themax-minmax1,param2%moy_1000+param2%ec_1000, &
940         param1%themax-minmax1,param2%moy_1000-param2%ec_1000, &
941         " "" | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i  -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
942         write(600,'(a,4f15.5,a)')"echo -e """,param1%moy_100+param1%ec_100,param2%themax-minmax2, &
943         param1%moy_100-param1%ec_100,param2%themax-minmax2, &
944         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i  -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
945         write(600,'(a,4f15.5,a)')"echo -e """,param1%themax-minmax1,param2%moy_100+param2%ec_100, &
946         param1%themax-minmax1,param2%moy_100-param2%ec_100, &
947         " "" | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
948         ! -------- affiche la moyennes de l'ensemble du meilleur modèle de chaque chaîne
949         write(600,'(a)')"###### moy modèles des chaînes 2 #######"
950         minmax1 = param1%moy_bestchaine - param1%ec_bestchaine
951         minmax2 = param1%moy_bestchaine + param1%ec_bestchaine
952         themin = param2%themin + 0.035_wr * (param2%themax - param2%themin)
953         themax = param2%themax - 0.035_wr * (param2%themax - param2%themin)
954         write(600,*)"echo ",minmax1,themin," 0 0.1i | psxy $geozone $geoproj ", "\"
955         write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
956         write(600,*)"echo ",minmax2,themin," 0 0.1i | psxy $geozone $geoproj ", "\"
957         write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
958         write(600,*)"echo ",param1%moy_bestchaine,themin," 0 0.1i | psxy $geozone $geoproj ", "\"
959         write(600,*)"-SV0.04i/0.05i/0.04i -Gred -Wthinnest,black -O -K >> $file"
960         write(600,*)"echo ",minmax1,themax," 180 0.1i | psxy $geozone $geoproj ", "\"
961         write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
962         write(600,*)"echo ",minmax2,themax," 180 0.1i | psxy $geozone $geoproj ", "\"
963         write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
964         write(600,*)"echo ",param1%moy_bestchaine,themax," 180 0.1i | psxy $geozone $geoproj ", "\"
965         write(600,*)"-SV0.04i/0.06i/0.04i -Gred -Wthinnest,black -O -K >> $file"
966         minmax1 = param2%moy_bestchaine - param2%ec_bestchaine
967         minmax2 = param2%moy_bestchaine + param2%ec_bestchaine
968         themin = param1%themin  + 0.035_wr * (param1%themax - param1%themin)
969         themax = param1%themax  - 0.035_wr * (param1%themax - param1%themin)
970         write(600,*)"echo ",themin,minmax1," 90 0.1i | psxy $geozone $geoproj ", "\"
971         write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
```

```
      write(600,*)"echo ",themin,minmax2," 90 0.1i | psxy $geozone $geoproj ", "\"
      write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
      write(600,*)"echo ",themin,param2%moy_bestchaine," 90 0.1i | psxy $geozone $geoproj ", "\"
      write(600,*)"-SV0.04i/0.05i/0.04i -Gred -Wthinnest,black -O -K >> $file"
      write(600,*)"echo ",themax,minmax1," 270 0.1i | psxy $geozone $geoproj ", "\"
      write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
      write(600,*)"echo ",themax,minmax2," 270 0.1i | psxy $geozone $geoproj ", "\"
      write(600,*)"-SV0.04i/0.06i/0.04i -Gorange -Wthinnest,black -O -K >> $file"
      write(600,*)"echo ",themax,param2%moy_bestchaine," 270 0.1i | psxy $geozone $geoproj ", "\"
      write(600,*)"-SV0.04i/0.06i/0.04i -Gred -Wthinnest,black -O -K >> $file"
      ! ------------------------------------------------------------      .
      write(600,'(a)')"######### affiche les stations ##########"
      ok = 0
      open(605, FILE = 'DATA/sta.d',status='old',iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans GMT_2paramplot : le fichier data/sta.d n''existe pas '
        stop
      endif
      do while(ok .eq. 0)
        read(605,*,iostat = ok)datasta
        if ((datasta%lon.gt.param1%themin).and.(datasta%lon.lt.param1%themax) &
              .and.(datasta%lat.gt.param2%themin).and.(datasta%lat.lt.param2%themax)) then
          write(600,*)" echo ",datasta%lon,datasta%lat," | psxy $geoproj $geozone -St0.1i -Gred  ", &
            "-Lk -Wthinnest -O -K >> $file"
        endif
      enddo
      close(605)
      ! ------------------------------------------------------------      .
      ! -------- valeurs pour placer la rose des vents et l'échelle --      .
      val = param2%themax - (param2%themax-param2%themin)*0.175_wr        ! LAT max rose
      if ((val.ge.-90.0_wr).and.(val.le.-10.0_wr)) then
        write(char_map(1),'(f10.6)')  val
      endif
      if ((val.gt.-10.0_wr).and.(val.lt.0.0_wr)) then
        write(char_map(1),'(f10.7)')  val
      endif
      if ((val.ge.0.0_wr).and.(val.lt.10.0_wr)) then
        write(char_map(1),'(f10.8)')  val
      endif
      if ((val.ge.10.0_wr).and.(val.le.90.0_wr)) then
        write(char_map(1),'(f10.7)')  val
      endif
      val = param2%themin + (param2%themax-param2%themin)*0.075_wr        ! LAT min barre
      if ((val.ge.-90.0_wr).and.(val.le.-10.0_wr)) then
        write(char_map(3),'(f10.6)')  val
      endif
      if ((val.gt.-10.0_wr).and.(val.lt.0.0_wr)) then
        write(char_map(3),'(f10.7)')  val
      endif
      if ((val.ge.0.0_wr).and.(val.lt.10.0_wr)) then
        write(char_map(3),'(f10.8)')  val
      endif
      if ((val.ge.10.0_wr).and.(val.le.90.0_wr)) then
        write(char_map(3),'(f10.7)')  val
      endif
      val = param1%themin + (param1%themax-param1%themin)*0.2_wr          ! LON min barre
      if ((val.ge.-180.0_wr).and.(val.le.-100.0_wr)) then
        write(char_map(2),'(f10.5)')  val
      endif
      if ((val.gt.-100.0_wr).and.(val.le.-10.0_wr)) then
        write(char_map(2),'(f10.6)')  val
      endif
      if ((val.gt.-10.0_wr).and.(val.lt.0.0_wr)) then
        write(char_map(2),'(f10.7)')  val
      endif
```

```
        if ((val.ge.0.0_wr).and.(val.lt.10.0_wr)) then
          write(char_map(2),'(f10.8)')  val
        endif
        if ((val.ge.10.0_wr).and.(val.lt.100.0_wr)) then
          write(char_map(2),'(f10.7)')  val
        endif
        if ((val.ge.100.0_wr).and.(val.le.180.0_wr)) then
          write(char_map(2),'(f10.6)')  val
        endif
          val = param1%themin + (param1%themax-param1%themin)*0.125_wr      ! LON min rose
        if ((val.ge.-180.0_wr).and.(val.le.-100.0_wr)) then
          write(char_map(4),'(f10.5)')  val
        endif
        if ((val.gt.-100.0_wr).and.(val.le.-10.0_wr)) then
          write(char_map(4),'(f10.6)')  val
        endif
        if ((val.gt.-10.0_wr).and.(val.lt.0.0_wr)) then
          write(char_map(4),'(f10.7)')  val
        endif
        if ((val.ge.0.0_wr).and.(val.lt.10.0_wr)) then
          write(char_map(4),'(f10.8)')  val
        endif
        if ((val.ge.10.0_wr).and.(val.lt.100.0_wr)) then
          write(char_map(4),'(f10.7)')  val
        endif
        if ((val.ge.100.0_wr).and.(val.le.180.0_wr)) then
          write(char_map(4),'(f10.6)')  val
        endif
        ! ------------------------------------------------------------     .
        ! -------- taille de l'échelle (en km)                 ---------     .
        i=int((param1%themax-param1%themin)/8.0_wr/360._wr*2._wr*pi*rT)
        write(char_6,'(i2.2)')  i
        write(600,'(a)')"######### rose vents et échelle #########"
        write(600,*) "psbasemap $geozone $geoproj -Ba0g0.06", &
        " -Lf"//char_map(2)//"/"//char_map(3)//"/"//char_map(3)//"/"//char_6//"k+l+jl", &
        " -Tf"//char_map(4)//"/"//char_map(1)//"/0.75i/3:@,@,@,@-N@-:", &
        " -O -K >>  $file"
        write(600,'(a)')"##########################################"
        write(600,'(a)')"############## diag. densité ##############"
        write(600,'(a)')"##########################################"
        write(600,*)"psbasemap $geozone $geoproj -Ba"//char_1//":"""//param1%char//""":/a"//char_2//":"""// &
        param2%char//""":WenS -K -O -X6.25i >>  $file"
        write(600,*)"pscoast $geozone $geoproj -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -O -K >>  $file"
        write(600,'(a)')"################ grid image ##############"
        write(600,*)"grdimage $geozone $geoproj OUTPUT/GMT/topo0_"//trim(adjustl(numberfile))//".grd  ", &
        "-QNaN -COUTPUT/GMT/colorpal2.cpt -B0 -O -K -Sn >> $file"
        write(600,'(a)')"############### grid contour ##############"
        write(600,*)"grdcontour OUTPUT/GMT/topo0_"//trim(adjustl(numberfile))//".grd $geozone $geoproj  ", &
        "-Ba0 -C75 -L74/75 -W2 -O -K >> $file"
        write(600,*)"grdcontour OUTPUT/GMT/topo0_"//trim(adjustl(numberfile))//".grd $geozone $geoproj  ", &
        "-Ba0 -C50 -L49/50 -W2 -O -K >> $file"
        write(600,'(a)')"######### cercles concentriques #########"
        do i=2,100,2
          write(600,*)"echo "" ",param1%vec10000(1,1),param2%vec10000(1,1),"0",i,i," "" \"
          write(600,*)" | psxy $geozone $geoproj -SE -W3,grey,-- -O -K >> $file"
        enddo
        ! ------------------------------------------------------------     . ellipses
        write(numberfile(1:5),'(i5)')m
        write(600,*)"echo ",param1%moy_1000,param2%moy_1000,E%ang,E%axeA*2.0_wr,E%axeB*2.0_wr, &
             " > OUTPUT/GMT/ellipse-"//trim(adjustl(numberfile))//".txt"
        write(600,*)"echo ",param1%moy_1000,param2%moy_1000,E%ang,E%axeA*4.0_wr,E%axeB*4.0_wr, &
             " >> OUTPUT/GMT/ellipse-"//trim(adjustl(numberfile))//".txt"
        write(600,*)"echo ",param1%moy_1000,param2%moy_1000,E%ang,E%axeA*6.0_wr,E%axeB*6.0_wr, &
             " >> OUTPUT/GMT/ellipse-"//trim(adjustl(numberfile))//".txt"
        write(600,*)"psxy $geozone $geoproj -SE -W2,red,-- OUTPUT/GMT/ellipse-"//trim(adjustl(numberfile))//".txt -O -K -N >> $file"
```

```
! ------------------------------------------------------------      .
write(600,'(a)')"######### rose vents et échelle #########"
write(600,*) "psbasemap $geozone $geoproj -Ba0g0.06", &
" -Lf"//char_map(2)//"/"//char_map(3)//"/"//char_map(3)//"/"//char_6//"k+l+jl", &
" -Tf"//char_map(4)//"/"//char_map(1)//"/0.75i/3:@,@,@,@-N@-:", &
" -O -K >>  $file"
write(600,*)"psscale -D1/-1/0.50E+01/0.25ch -B25.0:""densit\351"": -S -I -COUTPUT/GMT/colorpal2.cpt -O -K >> $file"
write(600,'(a)')"################### catalogue 1 ###################"
call catalogue(param_best,refseisme,find)
if ((find==1).or.(find==2)) then
  findtest=.true.
  write(600,*)"echo -e """,refseisme(1)%lon,param2%themin+0.001_wr," \n ",refseisme(1)%lon,param2%themax-0.001_wr," \"
  write(600,*)" "" | psxy $geozone $geoproj -W4,orange,- -O -K >> $file"
  write(600,*)"echo -e """,param1%themin+0.001_wr,refseisme(1)%lat," \n ",param1%themax-0.001_wr,refseisme(1)%lat," \"
  write(600,*)" "" | psxy $geozone $geoproj -W4,orange,- -O -K >> $file"
  write(600,'(a)')"#############################################"
endif
if (find==2) then
  findtest=.true.
  write(600,*)"echo -e """,refseisme(2)%lon,param2%themin+0.001_wr," \n ",refseisme(2)%lon,param2%themax-0.001_wr," \"
  write(600,*)" "" | psxy $geozone $geoproj -W4,orange,-.- -O -K >> $file"
  write(600,*)"echo -e """,param1%themin+0.001_wr,refseisme(2)%lat," \n ",param1%themax-0.001_wr,refseisme(2)%lat," \"
  write(600,*)" "" | psxy $geozone $geoproj -W4,orange,-.- -O -K >> $file"
  write(600,'(a)')"#############################################"
endif

! ------------------------------------------------------------      . geiger avec Modèle Terre Arroucau

write(600,'(2a)')"psxy $geoproj $geozone -Sa0.15i -Gred -Wthinnest ", & ! étoile rouge
  " OUTPUT/GMT/ArrALL-"//trim(adjustl(numberfile))//".txt -O -K -N >> $file"
write(600,'(2a)')"head -n 1 OUTPUT/GMT/ArrALLt-"//trim(adjustl(numberfile))//".txt ", &
  " | pstext $geoproj $geozone -D.5/.5v1 -O -K -N >> $file"
write(600,'(2a)')"tail -n 1 OUTPUT/GMT/ArrALLt-"//trim(adjustl(numberfile))//".txt ", &
  " | pstext $geoproj $geozone -D.5/.5v1 -O -K -N >> $file"

! ------------------------------------------------------------      . geiger avec Modèle Terre Si-Hex

write(600,'(2a)')"psxy $geoproj $geozone -Sa0.15i -Gblue -Wthinnest ", & ! étoile bleue
  "  OUTPUT/GMT/SiHexALL-"//trim(adjustl(numberfile))//".txt -O -K -N >> $file"
write(600,'(2a)')"head -n 1 OUTPUT/GMT/SiHexALLt-"//trim(adjustl(numberfile))//".txt ", &
  " | pstext $geoproj $geozone -D.5/.5v1 -O -K -N >> $file"
write(600,'(2a)')"tail -n 1 OUTPUT/GMT/SiHexALLt-"//trim(adjustl(numberfile))//".txt ", &
  " | pstext $geoproj $geozone -D.5/.5v1 -O -K -N >> $file"

! ------------------------------------------------------------      . geiger avec Modèle Terre CÉA

write(600,'(2a)')"psxy $geoproj $geozone -Sa0.15i -Ggreen -Wthinnest ", & ! étoile green
  "  OUTPUT/GMT/CEAALL-"//trim(adjustl(numberfile))//".txt -O -K -N >> $file"
write(600,'(2a)')"head -n 1 OUTPUT/GMT/CEAALLt-"//trim(adjustl(numberfile))//".txt ", &
  " | pstext $geoproj $geozone -D.5/.5v1 -O -K -N >> $file"
write(600,'(2a)')"tail -n 1 OUTPUT/GMT/CEAALLt-"//trim(adjustl(numberfile))//".txt ", &
  " | pstext $geoproj $geozone -D.5/.5v1 -O -K -N >> $file"

! ------------------------------------------------------------      .
write(600,'(a)')"############## légende n0 ###############"
val = param2%themax-(param2%themax-param2%themin)*0.035_wr
val1 = param1%themin+(param1%themax-param1%themin)*0.275_wr
val2 = param1%themin+(param1%themax-param1%themin)*0.325_wr
write(600,*)"echo  -e '",val1,val," \n ",val2,val,"' | psxy $geozone $geoproj -W2,red,-- -O -K >> $file "
val1 = param1%themin+(param1%themax-param1%themin)*0.35_wr
write(600,*)"echo '",val1,val," 12 0 4 LM ellipse @:10:\050\2611, 2 et 3@~\163@~\051@:: ", &
"des 1000 meilleurs mod\350les ' | pstext $geozone $geoproj -Gred -O -K >> $file"
! ------------------------------------------------------------      .
! plots                                                     -      .
! ------------------------------------------------------------      .
```

```
1167
1168        ! ------------------------------------------------------------------      .
1169        ! ------------------------------------------------------------------      .
1170
1171        endif ! (param lon-lat ou autres ?)
1172
1173        ! ------------------------------------------------------------------      .
1174        ! ------------------------------------------------------------------      .
1175
1176        write(600,'(a)')"#########################################"
1177        write(600,'(a)')"################ histo 1 ################"
1178        write(600,'(a)')"#########################################"
1179        write(numberfile(1:5),'(i5)')m
1180        write(600,*)"geoproj=-JX5i/1.5i"
1181        write(600,'(a10,E13.7,a1,E13.7,a5)')"geozone=-R",param1%themin,"/",param1%themax,"/0/15"
1182        write(600,*)"psbasemap $geozone $geoproj -Ba"//char_1//"/a10:""probabilit\351 (%)"":sWen -K -O -Y5.25i  >>  $file"
1183        write(600,'(a)')"################## mode ##################"
1184        write(600,*)"echo -e """,param1%mode,"0 \n",param1%mode," 100 "" | psxy $geozone $geoproj -W2,red -O -K >> $file"
1185        write(600,*)"echo -e """,param1%mediane,"0 \n",param1%mediane," 100 "" | psxy $geozone $geoproj -W2,blue -O -K >> $file"
1186        write(600,*)"echo -e """,param1%moy_tot,"0 \n",param1%moy_tot," 100 "" | psxy $geozone $geoproj -Wgreen -O -K >> $file"
1187        write(600,*)"echo -e """,param1%moy_tot+param1%ec_tot,"0 \n",param1%moy_tot+param1%ec_tot, &
1188        " 100 "" | psxy $geozone $geoproj -Wgreen,:-: -O -K >> $file"
1189        write(600,*)"echo -e """,param1%moy_tot-param1%ec_tot,"0 \n",param1%moy_tot-param1%ec_tot, &
1190        " 100 "" | psxy $geozone $geoproj -Wgreen,:-: -O -K >> $file"
1191        write(600,'(a)')"############ chaque chaîne 1 ############"
1192        ! ------- distribution de probabilités marginales a posteriori pour chaque chaîne
1193        k=0
1194        do i=1,nbChaineMV                                                    ! fond gris, puis barres noires
1195          write(char_5,'(i5)')10000+i
1196          open(unit=650+i,file="OUTPUT/GMT/"//filename//char_5//"-"//trim(adjustl(numberfile))//".bin" &
1197          ,STATUS="replace",access='direct',RECL=16)
1198          do j=1,nmod(i)
1199            k=k+1
1200            write(650+i,rec=j)real(param1%vec(k),8),real(param2%vec(k),8)
1201          enddo
1202          close(650+i)
1203          if (nmod(i).gt.100) then
1204            write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/"//filename//char_5//"-"//trim(adjustl(numberfile))//".bin", &
1205            " -bi2d -T0 -W"//char_3//" -Ggray -Ba0/a0 -O -K -Z1 >>  $file"
1206          endif
1207        enddo
1208        write(600,'(a)')"################ geiger ################"
1209        existe3=.true.
1210        if(param1%name.eq."_vc") existe3=.false.
1211        if(param1%name.eq."_vm") existe3=.false.
1212        if(param1%name.eq."_zm") existe3=.false.
1213        if(param1%name.eq."vps") existe3=.false.
1214        if (FLAGhypofixe) then
1215          if(param1%name.eq."lon") existe3=.false.
1216          if(param1%name.eq."lat") existe3=.false.
1217          if(param1%name.eq."_zh") existe3=.false.
1218          if(param1%name.eq."_to") existe3=.false.
1219        endif
1220        if(existe3) then
1221          ! ------- distribution de probabilités marginales a priori ------     .
1222          write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/geiger"//param1%name//"-"//trim(adjustl(numberfile))//".bin", &
1223          " -bi1d -W"//char_3//" -Gred -Ba0/a0 -O -K -Z1 >>  $file"
1224          write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/geiger"//param1%name//"-"//trim(adjustl(numberfile))//".bin", &
1225          " -bi1d -W"//char_3//" -S -L0/0 -Ba0/a0 -O -K -Z1 >>  $file"
1226        endif
1227        write(600,'(a)')"################ apriori ################"
1228        existe1=.true.
1229        if (FLAGterrefixe) then
1230          if(param1%name.eq."_vc") existe1=.false.
1231          if(param1%name.eq."_vm") existe1=.false.
```

```
1232          if(param1%name.eq."_zm") existe1=.false.
1233          if(param1%name.eq."vps") existe1=.false.
1234        endif
1235        if (FLAGhypofixe) then
1236          if(param1%name.eq."lon") existe1=.false.
1237          if(param1%name.eq."lat") existe1=.false.
1238          if(param1%name.eq."_zh") existe1=.false.
1239          if(param1%name.eq."_to") existe1=.false.
1240        endif
1241        if(existe1) then
1242        ! ------- distribution de probabilités marginales a priori ------     .
1243          if((param1%name.eq."_vc").or.(param1%name.eq."_vm").or.(param1%name.eq."_zm").or.(param1%name.eq."vps"))then
1244            write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/aprio"//param1%name//"-1.bin", &
1245              " -bi1d -W"//char_3//" -Gorange -Ba0/a0 -O -K -Z1 >>  $file"
1246            write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/aprio"//param1%name//"-1.bin", &
1247              " -bi1d -W"//char_3//" -S -L0/0 -Ba0/a0 -O -K -Z1 >>  $file"
1248          else
1249            write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/aprio"//param1%name//"-"//trim(adjustl(numberfile))//".bin", &
1250              " -bi1d -W"//char_3//" -Gorange -Ba0/a0 -O -K -Z1 >>  $file"
1251            write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/aprio"//param1%name//"-"//trim(adjustl(numberfile))//".bin", &
1252              " -bi1d -W"//char_3//" -S -L0/0 -Ba0/a0 -O -K -Z1 >>  $file"
1253          endif
1254        endif
1255        write(600,'(a)')"################ totale ################"
1256        ! ------- distribution de probabilités marginales a posteriori TOTALE  !!! -Gp300/1:BblueF-
1257        write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_tot.bin", &
1258        " -bi3d -T0  -W"//char_3//" -Gblue -Ba0/a0 -O -K -Z1 >>  $file" !
1259        write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_tot.bin", &
1260        " -bi3d -T0  -W"//char_3//" -L1/0 -S -Ba0/a0 -O -K -Z1 >>  $file" !
1261        write(600,'(a)')"############ chaque chaîne 2 ############"
1262        do i=1,nbChaineMV
1263          write(char_5,'(i5)')10000+i
1264          if (nmod(i).gt.100) then
1265            write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/"//filename//char_5//"-"//trim(adjustl(numberfile))//".bin", &
1266            " -bi2d -T0  -W"//char_3//" -L0/0 -Ba0/a0 -O -K -Z1 -S >>  $file"
1267          endif
1268        enddo
1269
1270        if(existe3) then
1271          ! ------- distribution de probabilités marginales a priori ----     .
1272          write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/geiger"//param1%name//"-"//trim(adjustl(numberfile))//".bin", &
1273            " -bi1d -W"//char_3//" -Sred -L1/1 -Ba0/a0 -O -K -Z1 >>  $file"
1274        endif
1275
1276        write(600,*)"psbasemap $geozone $geoproj -Ba0 -K -O >>  $file"
1277        ! ------------------------------------------------------     .
1278        write(600,'(a)')"#########################################"
1279        write(600,'(a)')"################ histo 2 ################"
1280        write(600,'(a)')"#########################################"
1281        write(600,*)"geoproj=-JX1.5i/5i"
1282        write(600,'(a15,E13.7,a1,E13.7)')"geozone=-R0/15/",param2%themin,"/",param2%themax
1283        write(600,'(a13,E13.7,a1,E13.7,a5)')"geozonebis=-R",param2%themin,"/",param2%themax,"/0/15"
1284        write(600,*)"psbasemap $geozone $geoproj -Ba10:""probabilit\351 (%)"":/a"//char_2//"Swne -K -O -Y-5.25i -X5.25i  >>  $file"
1285        write(600,'(a)')"################ mode ################"
1286        write(600,*)"echo -e "" 0",param2%mode,"\n 100 ",param2%mode,""" | psxy $geozone $geoproj -W2,red -O -K >> $file"
1287        write(600,*)"echo -e "" 0",param2%mediane,"\n 100 ",param2%mediane,""" | psxy $geozone $geoproj -W2,blue -O -K >> $file"
1288        write(600,*)"echo -e "" 0",param2%moy_tot," \n 100 ",param2%moy_tot,""" | psxy $geozone $geoproj -Wgreen -O -K >> $file"
1289        write(600,*)"echo -e "" 0",param2%moy_tot+param2%ec_tot," \n 100 ",param2%moy_tot+param2%ec_tot, &
1290        """ | psxy $geozone $geoproj -Wgreen,:-: -O -K >> $file"
1291        write(600,*)"echo -e "" 0",param2%moy_tot-param2%ec_tot," \n 100 ",param2%moy_tot-param2%ec_tot, &
1292        """ | psxy $geozone $geoproj -Wgreen,:-: -O -K >> $file"
1293        write(600,'(a)')"############ chaque chaîne 1 ############"
1294        ! ------- distribution de probabilités marginales a posteriori pour chaque chaîne
1295        do i=1,nbChaineMV                                                    ! fond gris, puis barres noires
1296          write(char_5,'(i5)')10000+i
```

```
      if (nmod(i).gt.100) then
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/"//filename//char_5//"-"//trim(adjustl(numberfile))//".bin", &
        " -bi2d -T1 -W"//char_4//" -Ggray -Ba0/a0 -K -O -Z1 -A0 >> $file" ! -bi3d
      endif
    enddo
    write(600,'(a)')"################## geiger #################"
    existe4=.true.
    if(param2%name.eq."_vc") existe4=.false.
    if(param2%name.eq."_vm") existe4=.false.
    if(param2%name.eq."_zm") existe4=.false.
    if(param2%name.eq."vps") existe4=.false.
    if (FLAGhypofixe) then
      if(param2%name.eq."lon") existe4=.false.
      if(param2%name.eq."lat") existe4=.false.
      if(param2%name.eq."_zh") existe4=.false.
      if(param2%name.eq."_to") existe4=.false.
    endif
    if(existe4) then
      ! -------- distribution de probabilités marginales a priori ------    .
      write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/geiger"//param2%name//"-"//trim(adjustl(numberfile))//".bin", &
      " -bi1d -W"//char_4//" -Gred -Ba0/a0 -O -K -Z1 -A0 >> $file"
      write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/geiger"//param2%name//"-"//trim(adjustl(numberfile))//".bin", &
      " -bi1d -W"//char_4//" -S -L0/0 -Ba0/a0 -O -K -Z1 -A0 >> $file"
    endif
    write(600,'(a)')"################## apriori #################"
    existe2=.true.
    if (FLAGterrefixe) then
      if(param2%name.eq."_vc") existe2=.false.
      if(param2%name.eq."_vm") existe2=.false.
      if(param2%name.eq."_zm") existe2=.false.
      if(param2%name.eq."vps") existe2=.false.
    endif
      if (FLAGhypofixe) then
      if(param2%name.eq."lon") existe2=.false.
      if(param2%name.eq."lat") existe2=.false.
      if(param2%name.eq."_zh") existe2=.false.
      if(param2%name.eq."_to") existe2=.false.
    endif
    if(existe2) then
      ! -------- distribution de probabilités marginales a priori ----    .
      if((param2%name.eq."_vc").or.(param2%name.eq."_vm").or.(param2%name.eq."_zm").or.(param2%name.eq."vps"))then
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/aprio"//param2%name//"-1.bin", &
          " -bi1d -W"//char_4//" -Gorange -Ba0/a0 -O -K -Z1 -A0 >> $file"
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/aprio"//param2%name//"-1.bin", &
          " -bi1d -W"//char_4//" -S -L0/0 -Ba0/a0 -O -K -Z1 -A0 >> $file"
      else
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/aprio"//param2%name//"-"//trim(adjustl(numberfile))//".bin", &
          " -bi1d -W"//char_4//" -Gorange -Ba0/a0 -O -K -Z1 -A0 >> $file"
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/aprio"//param2%name//"-"//trim(adjustl(numberfile))//".bin", &
          " -bi1d -W"//char_4//" -S -L0/0 -Ba0/a0 -O -K -Z1 -A0 >> $file"
      endif
    endif
    write(600,'(a)')"################## totale #################"
    ! -------- distribution de probabilités marginales a posteriori TOTALE  !!! -Gp300/1:BblueF-
    write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_tot.bin -T1", &
    " -bi3d -W"//char_4//" -Gblue -Ba0/a0 -O -K -Z1 -A0 >> $file"
    write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"_tot.bin -T1", &
    " -bi3d -W"//char_4//" -L1/0 -S -Ba0/a0 -O -K -Z1 -A0 >> $file"
    write(600,'(a)')"############# chaque chaîne 2 #############"
    do i=1,nbChaineMV
      write(char_5,'(i5)')10000+i
      if (nmod(i).gt.100) then
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/"//filename//char_5//"-"//trim(adjustl(numberfile))//".bin", &
        " -bi2d -T1 -W"//char_4//" -L0/0 -Ba0/a0 -K -O -Z1 -A0 -S >> $file" ! -bi3d
      endif
```

```
      enddo
      if(existe4) then
        ! -------- distribution de probabilités marginales a priori ----     .
        write(600,*)"pshistogram $geozonebis $geoproj OUTPUT/GMT/geiger"//param2%name//"-"//trim(adjustl(numberfile))//".bin", &
          " -bi1d -W"//char_4//" -Sred -L1/1 -Ba0/a0 -O -K -Z1 -A0 >>  $file"
      endif

      write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
      ! ------------------------------------------------------------     .
      write(600,'(a)')"##########################################"
      write(600,'(a)')"############## légende n1 ################"
      write(600,'(a)')"##########################################"
      write(600,*)"geoproj=-JX1.5i/1.5i"
      write(600,*)"geozone=-R0/1/0/1"
      write(600,*)"echo -e '0.0 .10 \n 0.125 .10' | psxy $geozone $geoproj -Wgreen -O -K -Y5.25i >> $file"
      write(600,*)"echo '0.15 .10 15 0 4 LM moyenne @:10: \050\2611@~\163@~\051 @::' | pstext $geozone $geoproj -O -K >> $file"
      write(600,*)"echo -e '0.0 .12 \n 0.125 .12' | psxy $geozone $geoproj -Wgreen,:-: -O -K >> $file"
      write(600,*)"echo -e '0.0 .08 \n 0.125 .08' | psxy $geozone $geoproj -Wgreen,:-: -O -K >> $file"
      write(600,*)"echo -e '0.0 .25 \n 0.125 .25' | psxy $geozone $geoproj -W2,red  -O -K  >> $file"
      write(600,*)"echo -e '0.5 .25 \n 0.625 .25' | psxy $geozone $geoproj -W2,blue  -O -K  >> $file"
      write(600,*)"echo '0.15 .25 15 0 4 LM mode' | pstext $geozone $geoproj -O -K >> $file"
      write(600,*)"echo '0.65 .25 15 0 4 LM m\351diane' | pstext $geozone $geoproj -O -K -N >> $file"

       if (findtest) then
          write(600,*)"echo -e '0.01 .4 \n 0.125 .4' | psxy $geozone $geoproj -W4,orange,- -O -K  >> $file"
          write(600,*)"echo '0.15 .4 15 0 4 LM catalogue' | pstext $geozone $geoproj -O -K >> $file"
      endif

      write(600,*)"echo '0.02 .875 0.35 0.25' | psxy $geozone $geoproj -Ggray -Sr -O -K -L -N >> $file"
      write(600,*)"echo '0.02 .875 0.35 0.25' | psxy $geozone $geoproj -W1,black -Sr -O -K -L -N >> $file"
      write(600,*)"echo '0.02 .735 0.35 0.25' | psxy $geozone $geoproj -Gblue -Sr -O -K -L -N >> $file"
      write(600,*)"echo '0.02 .735 0.35 0.25' | psxy $geozone $geoproj -W1,black -Sr -O -K -L -N >> $file"

      if (existe1.or.existe2) write(600,*)"echo '0.02 .58 0.35 0.25' | psxy $geozone $geoproj -Gorange -Sr -O -K -L -N >> $file"
      if (existe1.or.existe2) write(600,*)"echo '0.02 .58 0.35 0.25' | psxy $geozone $geoproj -W1,black -Sr -O -K -L -N >> $file"
      if (existe3.or.existe4) write(600,*)"echo '0.635 .58 0.35 0.25' | psxy $geozone $geoproj -Gred -Sr -O -K -L -N >> $file"
      if (existe3.or.existe4) write(600,*)"echo '0.635 .58 0.35 0.25' | psxy $geozone $geoproj -W1,black -Sr -O -K -L -N >> $file"
      if (existe1.or.existe2) write(600,*)"echo '0.15 .60 15 0 4 LM a priori' | pstext $geozone $geoproj -O -K >> $file"
      if (existe3.or.existe4) write(600,*)"echo '0.70 .60 15 0 4 LM Geiger' | pstext $geozone $geoproj -O -K -N >> $file"

      write(600,*)"echo '0.15 .90 15 0 4 LM \050par cha\356ne\051' | pstext $geozone $geoproj -O -K >> $file"
      write(600,*)"echo '0.15 .75 15 0 4 LM a posteriori' | pstext $geozone $geoproj -O -K >> $file"
      ! ------------------------------------------------------------     .
      write(600,'(a)')"##########################################"
      write(600,'(a)')"############## légende n2 ################"
      write(600,'(a)')"##########################################"
      write(600,*)"geoproj=-JX5i/1.5i"
      write(600,*)"geozone=-R0/1/0/1"
      write(600,'(a)')"################ figure #################"
      write(600,*)"echo '0.075 .05 0 0.1i' | psxy $geozone $geoproj", &
        " -SV0.04i/0.05i/0.04i -Gred -Wthinnest,black  -O -K -X-11.5i -UTL/0/1.5i  >> $file"
      write(600,*)"echo '0.100 .05 0 0.1i' | psxy $geozone $geoproj", &
        " -SV0.04i/0.05i/0.04i -Gorange -Wthinnest,black  -O -K >> $file"
      write(600,*)"echo '0.050 .05 0 0.1i' | psxy $geozone $geoproj", &
        " -SV0.04i/0.05i/0.04i -Gorange -Wthinnest,black  -O -K >> $file"
      write(600,*)"echo '0.01 .25 0.15 .25' | psxy $geozone $geoproj -SVS0.08i/0.12i/0.08i", &
        " -Gp300/73:Bgray45F- -Wthinnest,black -O -K >> $file"
      write(600,*)"echo '0.01 .40 0.15 .40' | psxy $geozone $geoproj -SVS0.06i/0.10i/0.06i", &
        " -Gp300/73:Bgray80F- -Wthinnest,black -O -K >> $file"
      write(600,*)"echo '0.01 .55 0.15 .55' | psxy $geozone $geoproj -SVS0.04i/0.08i/0.04i", &
        " -Gp300/73:Bgray100F- -Wthinnest,black -O -K >> $file"
      write(600,*)"echo '0.075 .7' | psxy $geozone $geoproj -Sa0.1i -Gyellow -Wthinnest,black -O -K >> $file"
      write(600,'(a)')"################ texte #################"
      write(600,*)"echo '0.175 .10 15 0 4 LM moyenne des meilleurs mod\350les de chaque cha\356ne'", &
        " | pstext $geozone $geoproj -O -K >> $file"
```

```
      write(600,*)"echo '0.175 .25 15 0 4 LM moyenne @:10: \050\2611@~\163@~\051 @:: des 10000 meilleurs mod\350les'", &
        " | pstext $geozone $geoproj -O -K >> $file"
      write(600,*)"echo '0.175 .40 15 0 4 LM moyenne @:10: \050\2611@~\163@~\051 @:: des 1000 meilleurs mod\350les'", &
        " | pstext $geozone $geoproj -O -K >> $file"
      write(600,*)"echo '0.175 .55 15 0 4 LM moyenne @:10: \050\2611@~\163@~\051 @:: des 100 meilleurs mod\350les'", &
        " | pstext $geozone $geoproj -O -K >> $file"
      write(600,*)"echo '0.174 .70  15 0 4 LM 10 meilleurs mod\350les diff\351rents' | pstext $geozone $geoproj -O >> $file"
      ! ------------------------------------------------------------------      .
      write(600,'(a)')"#########################################"
      write(600,'(a)')"############## conversions ###############"
      write(600,'(a)')"#########################################"
      write(600,*)"ps2raster OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
      write(600,'(2a)')"mv OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".pdf ", &
        "OUTPUT/figures/"//filename//"-"//trim(adjustl(numberfile))//".pdf"
      write(600,'(a)')"#########################################"
      write(600,*)
      write(600,*)"#######################################"
      write(600,*)"#######################################"
      write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
      write(600,*)" echo $ELAPSED secondes"
      call system_clock(Nnewtime,ratetime)
      tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
      write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)') temps : ',int(tl/3600.0_wr,wi), &
      int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
      ! ------------------------------------------------------------------      .

    end subroutine GMT_2paramplot
    ! ------------------------------------------------------------------      .
  end subroutine GMTfull

  ! ------------------------------------------------------------------      .

  subroutine affiche_temps_GMT(temps_ref,char_1)
    ! -------                                                 ---------      .mh
    ! conversion du temps de référence -> GMT, pas utilisée mais peut être utile
    ! ------------------------------------------------------------------      .
    use typetemps
    ! -------                                                 ---------      .
    implicit none
    ! -------                                                 ---------      .
    character(LEN=30), intent (out) :: char_1
    type(date_sec), intent (in) :: temps_ref
    ! ------------------------------------------------------------------      .
    write(char_1,"(i4.4,4(1a,i2.2))") temps_ref%date%year,"-",temps_ref%date%month, &
    "-",temps_ref%date%day,"T",temps_ref%date%hour,":",temps_ref%date%min
    ! ------------------------------------------------------------------      .
  end subroutine affiche_temps_GMT

  ! ------------------------------------------------------------------      .

  subroutine affiche_temps_ref(temps_ref,char_1,ok)
    ! -------                                                 ---------      .mh
    ! affiche le temps de référence sur les figures
    ! -------                                                 ---------      .
    use typetemps
    ! -------                                                 ---------      .
    implicit none
    ! -------                                                 ---------      .
    character(LEN=30), intent (out) :: char_1
    type(date_sec), intent (in) :: temps_ref
    integer(KIND=wi), intent (in)  :: ok
    ! -------                                                 ---------      .
    character(LEN=30) :: lemois
    character(LEN=2) :: char_2
    ! ------------------------------------------------------------------      .
```

```
1492    if (ok == 1) then                                                    ! style LaTeX pour les accents
1493      if(temps_ref%date%month.eq.1) lemois="janvier"
1494      if(temps_ref%date%month.eq.2) lemois="f\'evrier"
1495      if(temps_ref%date%month.eq.3) lemois="mars"
1496      if(temps_ref%date%month.eq.4) lemois="avril"
1497      if(temps_ref%date%month.eq.5) lemois="mai"
1498      if(temps_ref%date%month.eq.6) lemois="juin"
1499      if(temps_ref%date%month.eq.7) lemois="juillet"
1500      if(temps_ref%date%month.eq.8) lemois="ao\^ut"
1501      if(temps_ref%date%month.eq.9) lemois="septembre"
1502      if(temps_ref%date%month.eq.10) lemois="octobre"
1503      if(temps_ref%date%month.eq.11) lemois="novembre"
1504      if(temps_ref%date%month.eq.12) lemois="d\'ecembre"
1505    else                                                                 ! style GMT pour les accents
1506      if(temps_ref%date%month.eq.1) lemois="janvier"
1507      if(temps_ref%date%month.eq.2) lemois="f\351vrier"
1508      if(temps_ref%date%month.eq.3) lemois="mars"
1509      if(temps_ref%date%month.eq.4) lemois="avril"
1510      if(temps_ref%date%month.eq.5) lemois="mai"
1511      if(temps_ref%date%month.eq.6) lemois="juin"
1512      if(temps_ref%date%month.eq.7) lemois="juillet"
1513      if(temps_ref%date%month.eq.8) lemois="ao\373t"
1514      if(temps_ref%date%month.eq.9) lemois="septembre"
1515      if(temps_ref%date%month.eq.10) lemois="octobre"
1516      if(temps_ref%date%month.eq.11) lemois="novembre"
1517      if(temps_ref%date%month.eq.12) lemois="d\351cembre"
1518    endif
1519    ! _______                                                 ________     .
1520    if ((temps_ref%date%day.gt.0).and.(temps_ref%date%day.lt.32)) then
1521      write(char_2,'(i2.2)')len(trim(lemois))
1522      write(char_1,"(i2.2,1x,a"//char_2//",1x,i4.4,1x,i2.2,a1,i2.2)")temps_ref%date%day,lemois,temps_ref%date%year, &
1523      temps_ref%date%hour,":",temps_ref%date%min
1524    else
1525      write(char_1,'(a30)')"problème in affiche_temps_ref"
1526    endif
1527    ! ------------------------------------------------------------------     .
1528  end subroutine affiche_temps_ref
1529
1530  ! ------------------------------------------------------------------     .
1531
1532  subroutine mkdensityplot(l,param1,param2,deltaxy,nbparam,filename)
1533    ! _______                                                 ________     .mh
1534    ! Calcul du diagramme de densité pour deux paramètres (param1,param2) et
1535    ! stock une grille xyz en .bin de deltaxy x deltaxy arguments
1536    ! _______                                                 ________     .
1537    use typetemps
1538    ! _______                                                 ________     .
1539    implicit none
1540    ! _______                                                 ________     .
1541    integer(KIND=wi), intent (in) :: l
1542    type(densityplot_one), intent (inout) :: param1, param2                ! les deux paramètres
1543    integer(KIND=wi), intent (in) :: deltaxy                               ! pas de discrétisation pour le mode et le diagramme de densité
1544    integer(KIND=wi), intent (in) :: nbparam                               ! nombre de modèles
1545    character(LEN=7), intent (in) :: filename                              ! base du nom des fichiers de sorties
1546    ! _______                                                 ________     .
1547    integer(KIND=wi) :: i,j,k
1548    ! _______                                                 ________     .
1549    real(KIND=wr) :: themax
1550    real(KIND=wr), dimension(:,:), allocatable :: tab                      ! grille pour le diagramme de densité
1551    character (LEN=5) :: numberfile
1552    ! ------------------------------------------------------------------     .
1553    write(numberfile(1:5),'(i5)')l
1554    allocate(tab(deltaxy,deltaxy))
1555    do i=1,deltaxy
1556      do j=1,deltaxy
```

```
1557          tab(i,j)=0.0_wr                                                    ! initialisation
1558        enddo
1559      enddo
1560      ! ------- delta de pas                                  --------     .
1561      param1%delta=(param1%themax-param1%themin)/real(deltaxy,wr)
1562      param2%delta=(param2%themax-param2%themin)/real(deltaxy,wr)
1563      ! ------------------------------------------------------------------     . calcul de densité
1564      do k=1,nbparam
1565        i=int((param1%vec(k)-param1%themin)/param1%delta)+1
1566        j=int((param2%vec(k)-param2%themin)/param2%delta)+1
1567        if ((i.ge.1).and.(i.le.deltaxy)) then
1568          if ((j.ge.1).and.(j.le.deltaxy)) then
1569            tab(i,j)=tab(i,j)+1.0_wr
1570          else
1571            write(*,*)'problème dans mkdensityplot : calcul de densité, incrément incorrect, boucle 1',i,i,k
1572          endif
1573        else
1574          write(*,*)'problème dans mkdensityplot : calcul de densité, incrément incorrect, boucle 2',i,i,k
1575        endif
1576      enddo
1577      ! ------- normalisation (min max entre 0 et 100%)          --------     .
1578      themax=-1.0_wr
1579      do i=1,deltaxy
1580        do j=1,deltaxy
1581          if (themax.lt.tab(i,j)) themax=tab(i,j)                            ! garde le maximum
1582        enddo
1583      enddo
1584      do i=1,deltaxy
1585        do j=1,deltaxy
1586          tab(i,j)= 100.0_wr/(themax)*tab(i,j)
1587        enddo
1588      enddo
1589      ! ------------------------------------------------------------------     .
1590      ! ------- ecriture dans un fichier de la grille de densité --------     .
1591      open(unit=606,file="OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//".bin",STATUS="replace",access='direct',RECL=24)
1592      k=1
1593      do i=1,deltaxy
1594        do j=1,deltaxy
1595          if (tab(i,j).gt.0.5_wr) then                                       ! au moins 0.5 %
1596            write(606,rec=k)real(param1%themin+(real(i,wr)-.5_wr)*param1%delta,8) , &
1597            real(param2%themin +(real(j,wr)-.5_wr)*param2%delta,8),real(tab(i,j),8)
1598            k=k+1
1599          endif
1600        enddo
1601      enddo
1602      close(606)
1603      ! ------------------------------------------------------------------     .
1604      deallocate(tab)
1605      ! ------------------------------------------------------------------     .
1606    end subroutine mkdensityplot
1607
1608    ! ------------------------------------------------------------------     .
1609
1610    subroutine mkfcoutplot(l,param1,param2,mis,deltaxymis,nbparam,filename,delta_1,delta_2)
1611      ! -------                                                  --------     .mh
1612      ! Identifie les points représentatifs pour la représentation de la fonction coût
1613      ! Un grille échantillonant les plus petits misfits pour les deux paramètres (param1,param2)
1614      ! puis stock les points triés en xyz en .bin
1615      ! -> permet de faire des figures avec moins de recouvrement de point (donc plus légères)
1616      ! -> permet le tri des modèles selon la fonction coût
1617      ! -------                                                  --------     .
1618      use typetemps
1619      use tri
1620      ! -------                                                  --------     .
1621      implicit none
```

```
1622    ! -------                                                  --------    .
1623    integer(KIND=wi), intent (in) :: l
1624    type(densityplot_one), intent (inout) :: param1, param2, mis          ! les deux paramètres
1625    integer(KIND=wi), intent (in) :: deltaxymis                           ! nombre de discrétisation (entier)
1626    integer(KIND=wi), intent (in) :: nbparam                              ! nombre de modèles
1627    character(LEN=7), intent (in) :: filename                             ! base du nom des fichiers de sorties
1628    real(KIND=wr), intent (out) :: delta_1, delta_2                       ! pas de disrétisation (réels)
1629    ! -------                                                  --------    .
1630    integer(KIND=wi) :: i,j,k,n
1631    ! -------                                                  --------    .
1632    real(KIND=wr) :: themax,themin
1633    real(KIND=wr), dimension(:,:,:), allocatable :: tabmis                ! grille pour la représentation 2D de la fonction coût
1634    real(KIND=wr), dimension(:,:), allocatable :: tabmistri,tabmistribis  ! points sélectionnés et triés
1635    character (LEN=5) :: numberfile
1636    ! ------------------------------------------------------------------  .
1637    write(numberfile(1:5),'(i5)')l
1638    allocate(tabmis(deltaxymis,deltaxymis,3))
1639    do i=1,deltaxymis
1640      do j=1,deltaxymis
1641        tabmis(i,j,3)=100000.0_wr                                         ! initialisation de grille de la fonction coût
1642      enddo
1643    enddo
1644    ! ------- delta de pas                                     --------    .
1645    delta_1=(param1%themax-param1%themin)/real(deltaxymis,wr)
1646    delta_2=(param2%themax-param2%themin)/real(deltaxymis,wr)
1647    ! ------------------------------------------------------------------  .
1648    ! ------- garde le meilleur modèle dans chaque case        --------    .
1649    do k=1,nbparam
1650      i=int((param1%vec(k)-param1%themin)/delta_1)+1
1651      j=int((param2%vec(k)-param2%themin)/delta_2)+1
1652      if ((i.ge.1).and.(i.le.deltaxymis)) then
1653        if ((j.ge.1).and.(j.le.deltaxymis)) then
1654          if (tabmis(i,j,3).gt.mis%vec(k)) then
1655            tabmis(i,j,1)=param1%vec(k)
1656            tabmis(i,j,2)=param2%vec(k)
1657            tabmis(i,j,3)=mis%vec(k)
1658          endif
1659        else
1660          write(*,*)'problème dans mkfcoutplot : calcul de densité, incrément incorrect, boucle 1',i,i,k
1661        endif
1662      else
1663        write(*,*)'problème dans mkfcoutplot : calcul de densité, incrément incorrect, boucle 2',i,i,k
1664      endif
1665    enddo
1666    ! ------- nombre de cases retenues                          --------    .
1667    n=0
1668    do i=1,deltaxymis
1669      do j=1,deltaxymis
1670        if ((tabmis(i,j,3).lt.10000.0_wr).and.(tabmis(i,j,3).gt.0.0_wr)) n=n+1
1671      enddo
1672    enddo
1673    ! ------- transforme la grille en points                   --------    .
1674    allocate(tabmistri(n,3))                                              ! non triés
1675    allocate(tabmistribis(n,3))                                           ! triés
1676    k=0
1677    do i=1,deltaxymis
1678      do j=1,deltaxymis
1679        if ((tabmis(i,j,3).lt.10000.0_wr).and.(tabmis(i,j,3).gt.0.0_wr)) then
1680          k=k+1
1681          tabmistri(k,1)=tabmis(i,j,1)
1682          tabmistri(k,2)=tabmis(i,j,2)
1683          tabmistri(k,3)=tabmis(i,j,3)
1684        endif
1685      enddo
1686    enddo
```

```
    ! -------- normalisation (min max entre 0 et 100%)          ---------     .
    themax=-1.0_wr
    themin=100000._wr
    do i=1,n
      if (themax.lt.tabmistri(i,3)) themax=tabmistri(i,3)
      if (themin.gt.tabmistri(i,3)) themin=tabmistri(i,3)
    enddo
    do i=1,n
      tabmistri(i,3)= 100.0_wr/(themax-themin)*tabmistri(i,3) - 100.0_wr/(themax-themin)*themin
    enddo
    ! -------- tri                                              ---------     .
    call tri_bulle(tabmistri,n,tabmistribis)
    ! -------- ecriture dans un fichier des points              ---------     .
    open(unit=607,file="OUTPUT/GMT/"//filename//"-"//trim(adjustl(numberfile))//"mis.bin",STATUS="replace",access='direct',RECL=24)
    k=0
    do i=n,1,-1                                                                ! tri décroissant
      k=k+1
      write(607,rec=k)real(tabmistribis(i,1),8),real(tabmistribis(i,2),8),real(tabmistribis(i,3),8)
    enddo
    close(607)
    ! ---------------------------------------------------------------------     .
    deallocate(tabmis,tabmistri,tabmistribis)
    ! ---------------------------------------------------------------------     .

  end subroutine mkfcoutplot

END MODULE figure_GMT



! ************************************************************************     .
! ************************************************************************     .
```


2.2 SRC/MOD/MOD_GMT/mkchatelain.f90

```
! permet la création des scripts GMT pour le diagramme de Wadati modifié (châtelain, 1978)
! mars 2014
! ************************************************************************     .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------     .
! ************************************************************************     .
! ---------------------------------------------------------------------     .

MODULE figure_GMTchat

    use modparam

    implicit none

    private

    public  :: GMT_chatelain


CONTAINS

! ---------------------------------------------------------------------     .

  subroutine GMT_chatelain(nbtps,nbsta,D,dp)
    ! --------                                                 ---------     .mh
    ! production d'une partie du script GMT pour le diagramme de Châtelain (Châtelain, 1978)
    ! le diagramme de Châtelain (ou Wadati modifié) est indépendant des parametres
    ! et peux ainsi identifier une erreur de pointé sur les ondes          !
    ! --------                                                 ---------     .
    use typetemps
    use cpt_temps
```

```
31      use time
32      use pb_direct
33      use tri
34      ! ------                                                  --------    .
35      implicit none
36      integer(KIND=wi), intent(in) :: nbtps(nbseismes),nbsta
37      type(dataall), intent(in) :: D(nbseismes)
38      type(densityplot), intent (in) :: dp
39      ! ------                                                  --------    .
40      real (KIND=wr) :: aREF, R2REF                                       ! coeficient directeur de la régression
41      real (KIND=wr) :: aDIR, R2DIR                                       ! Chi2
42      real (KIND=wr) :: X, Y, tl
43      real (KIND=wr) :: Xmaxi, Ymaxi
44      ! ------                                                  --------    .
45      integer (KIND=wi), parameter :: taille=5000
46      ! ------                                                  --------    .
47      type(date_sec) :: one_tps_1,one_tps_2
48      integer(KIND=wi) :: i, j, k, l
49      integer(KIND=wi) :: n, nREF, nDIR
50      integer(KIND=wi) :: Noldtime, Nnewtime, ratetime
51      integer(KIND=wi) :: test1,test2
52      real (KIND=wr) :: XY(taille,3),XYREF(taille,3),XYDIR(taille,3)
53      real (KIND=wr) :: sDIR(taille,2),sREF(taille,2)
54      real (KIND=wr) :: x1, y1, sta_dist(nbsta+2)
55      real (KIND=wr) :: a, R2                                             ! coeficient directeur et Chi2 de la régression
56      real (KIND=wr) :: nsta(nbsta+2)
57      character(len=4) :: nomstaDIR(taille,2),nomstaREF(taille,2)
58      character(LEN=4) :: sta(nbsta+2)
59      character(LEN=7) :: char1, char2
60      character (LEN=5) :: char_nbseismes
61      ! ------------------------------------------------------------    . diagramme de chatelainplot pour toutes les ondes
62      call chatelainplot(nbtps,D,a=a,R2=R2,XY=XY,nb=n)
63      ! ------                                                  --------    . diagramme de chatelainplot pour ondes réfractées
64      call chatelainplot(nbtps,D,a=aREF,R2=R2REF,XY=XYREF,nb=nREF,sig=sREF,atype='N',nom_sta=nomstaREF)
65      ! ------                                                  --------    . diagramme de chatelainplot pour ondes directes
66      call chatelainplot(nbtps,D,a=aDIR,R2=R2DIR,XY=XYDIR,nb=nDIR,sig=sDIR,atype='G',nom_sta=nomstaDIR)
67      ! ------------------------------------------------------------    . minmax
68      Xmaxi=1.0_wr
69      Ymaxi=1.0_wr
70      do i=1,n
71        if (Xmaxi.lt.XY(i,1)) Xmaxi=XY(i,1)
72        if (Ymaxi.lt.XY(i,2)) Ymaxi=XY(i,2)
73      enddo
74      Xmaxi=1.1_wr*Xmaxi
75      Ymaxi=1.1_wr*Ymaxi
76      ! ------------------------------------------------------------    .
77      ! ------------------------------------------------------------    .
78      ! ecart des stations à la courbe théorique
79      ! ------------------------------------------------------------    .
80      ! ------------------------------------------------------------    . DIR :
81      ! ------------------------------------------------------------    .
82      sta_dist(:)=0.0_wr
83      sta(:)='    '
84      nsta(:)=0.0_wr
85      l=1
86      ! ------                                                  --------    .
87      do i=1,n
88        if (nomstaDIR(i,1).ne.nomstaDIR(i,2)) then
89          ! ------                                              --------    .
90          test1=0
91          test2=0
92          ! ------                                              --------    .
93          do k=1,l
94            ! ------                                            --------    . si la station existe déjà
95            if (sta(k)==nomstaDIR(i,1)) then
```

```
            call distancePlot(aDIR,XYDIR(i,1),XYDIR(i,2),sta_dist(k))
            nsta(k)=nsta(k)+1.0_wr
            test1=test1+1
          endif
          ! -------                                                  ---------     .
          if (sta(k)==nomstaDIR(i,2)) then
            call distancePlot(aDIR,XYDIR(i,1),XYDIR(i,2),sta_dist(k))
            nsta(k)=nsta(k)+1.0_wr
            test2=test2+1
          endif
          ! -------                                                  ---------     .
        enddo
        ! -------                                                  ---------     . si la station n'existe pas encore
        if (test1==0) then
          sta(l)=nomstaDIR(i,1)
          call distancePlot(aDIR,XYDIR(i,1),XYDIR(i,2),sta_dist(l))
          nsta(l)=1.0_wr
          l=l+1
        elseif(test1.gt.1) then
          write(*,*)'problème dans GMT_chatelain 1 : test1 =',test1
          stop
        endif
        ! -------                                                  ---------     .
        if (test2==0) then
          sta(l)=nomstaDIR(i,2)
          call distancePlot(aDIR,XYDIR(i,1),XYDIR(i,2),sta_dist(l))
          nsta(l)=1.0_wr
          l=l+1
        elseif(test1.gt.1) then
          write(*,*)'problème dans GMT_chatelain 1 : test2 =',test2
          stop
        endif
        ! -------                                                  ---------     .
        if (l.gt.nbsta+2) then
            write(*,*)'problème dans GMT_chatelain 1 : l > nbsta : ',l,nbsta
            stop
        endif
        ! -------                                                  ---------     .
      endif
    enddo
    ! -------                                                    ---------     . moyenne
    do i=1,nbsta
      if (nsta(i).gt.0.0_wr) sta_dist(i)=sta_dist(i)/nsta(i)
    enddo
    ! -------                                                    ---------     . tri
    call tri_bulle(sta_dist,sta,nbsta+2)
    ! -------                                                    ---------     .
    X = Xmaxi * 0.75_wr
    Y = Ymaxi * 0.075_wr
    open(unit=33,file="OUTPUT/GMT/chat-stationsDIR.d",status='replace')
    do i=1,nbsta
      Y = Y + Ymaxi * 0.05_wr
      if ((sta_dist(i).gt.0.5_wr).and.(i.le.5)) then
        write(33,'(2f10.5,a,f8.2,a)')X,Y," 15 0 5 6 @~\104@~ "//sta(i)//" g = ",sta_dist(i)," s"
      else
        write(33,'(a,f6.2,a)')"-1000 -1000 15 0 5 6 @~\104@~ "//sta(i)//" g = ",sta_dist(i)," s"
      endif
    enddo
    close(33)
    ! ------------------------------------------------------------------     .
    ! ------------------------------------------------------------------     . REF :
    ! ------------------------------------------------------------------     .
    sta_dist(:)=0.0_wr
    sta(:)='    '
    nsta(:)=0.0_wr
```

```
161     l=1
162     ! --------                                                  ---------     .
163     do i=1,n
164       if (nomstaREF(i,1).ne.nomstaREF(i,2)) then
165         ! --------                                              ---------     .
166         test1=0
167         test2=0
168         ! --------                                              ---------     .
169         do k=1,l
170           ! --------                                            ---------     . si la station existe déjà
171           if (sta(k)==nomstaREF(i,1)) then
172             call distancePlot(aREF,XYREF(i,1),XYREF(i,2),sta_dist(k))
173             nsta(k)=nsta(k)+1.0_wr
174             test1=test1+1
175           endif
176           ! --------                                            ---------     .
177           if (sta(k)==nomstaREF(i,2)) then
178             call distancePlot(aREF,XYREF(i,1),XYREF(i,2),sta_dist(k))
179              nsta(k)=nsta(k)+1.0_wr
180             test2=test2+1
181           endif
182           ! --------                                            ---------     .
183         enddo
184         ! --------                                              ---------     . si la station n'existe pas encore
185         if (test1==0) then
186           sta(l)=nomstaREF(i,1)
187           call distancePlot(aREF,XYREF(i,1),XYREF(i,2),sta_dist(l))
188           nsta(l)=1.0_wr
189           l=l+1
190         elseif(test1.gt.1) then
191           write(*,*)'problème dans GMT_chatelain 2 : test1 =',test1
192           stop
193         endif
194         ! --------                                              ---------     .
195         if (test2==0) then
196           sta(l)=nomstaREF(i,2)
197           call distancePlot(aREF,XYREF(i,1),XYREF(i,2),sta_dist(l))
198           nsta(l)=1.0_wr
199           l=l+1
200         elseif(test1.gt.1) then
201           write(*,*)'problème dans GMT_chatelain 2 : test2 =',test2
202           stop
203         endif
204         ! --------                                              ---------     .
205         if (l.gt.nbsta+2) then
206           write(*,*)'problème dans GMT_chatelain 2 : l > nbsta : ',l,nbsta
207           stop
208         endif
209         ! --------                                              ---------     .
210       endif
211     enddo
212     ! --------                                                  ---------     . moyenne
213     do i=1,nbsta
214       if (nsta(i).gt.0.0_wr) sta_dist(i)=sta_dist(i)/nsta(i)
215     enddo
216     ! --------                                                  ---------     . tri
217     call tri_bulle(sta_dist,sta,nbsta+2)
218     ! --------                                                  ---------     .
219     X = Xmaxi * 0.9_wr
220     Y = Ymaxi * 0.075_wr
221     open(unit=33,file="OUTPUT/GMT/chat-stationsREF.d",status='replace')
222     do i=1,nbsta
223       Y = Y + Ymaxi * 0.05_wr
224       if ((sta_dist(i).gt.0.5_wr).and.(i.le.5)) then
225         write(33,'(2f10.5,a,f8.2,a)')X,Y," 15 0 5 6 @~\104@~ "//sta(i)//" n = ",sta_dist(i)," s"
```

```
      else
        write(33,'(a,f6.2,a)')"-1000 -1000 15 0 5 6 @~\104@~ "//sta(i)//" n = ",sta_dist(i)," s"
      endif
    enddo
    close(33)
    ! ------------------------------------------------------------------     .
    ! ------------------------------------------------------------------     . SCRIPT GMT :
    ! ------------------------------------------------------------------     .
    write(600,*)"BEFORE=$SECONDS"
    call system_clock(Noldtime)
    write(600,*)"echo 'execution du script GMT chatelainplot'"
    write(*,*)"ecriture du script GMT chatelainplot "
    ! ------------------------------------------------------------------     .
    do j=1,nbseismes
      write(char_nbseismes(1:5),'(i5)')j
      write(600,*)"##########################################"
      write(600,*)"############## chatelainplot #############"
      ! ------------------------------------------------------------------     .
      write(600,*)"geoproj=-JX13i/8i"                                         ! système de projection
      write(600,'(a12,E13.7,a3,E13.7)')"geozone=-R0/",Xmaxi,"/0/",Ymaxi
      write(600,*)"file=OUTPUT/GMT/chatelainplot"//trim(adjustl(char_nbseismes))//".ps"
      if (Xmaxi.gt.60.0_wr) then
        write(600,*)"psbasemap $geozone $geoproj -Ba10f5:""T@-P1@--T@-P2@- (s)"":",&
        "/a10f5:""T@-S1@--T@-S2@- (s)"":WenS -Xc -Yc -K >  $file"
      else
        write(600,*)"psbasemap $geozone $geoproj -Ba2f.5:""T@-P1@--T@-P2@- (s)"":",&
        "/a2f.5:""T@-S1@--T@-S2@- (s)"":WenS -Xc -Yc -K >  $file"
      endif
      ! -------                                                 --------     .
      do i=1,int(Xmaxi,wi)+1
        ! -------                                               --------     . traits théoriquea min et max
        write(600,*)"echo -e '",i-1,(dp%VpVs%themax)*real(i-1,wr),"\n",i,(dp%VpVs%themax)*real(i,wr), &
            "' | psxy $geozone $geoproj -W1,red,- -O -K >>  $file"
        write(600,*)"echo -e '",i-1,(dp%VpVs%themin)*real(i-1,wr),"\n",i,(dp%VpVs%themin)*real(i,wr), &
            "' | psxy $geozone $geoproj -W1,red,- -O -K >>  $file"
        ! -------                                               --------     . traits théoriquea (droite de regression) TOTAL
        write(600,*)"echo -e "" ",i-1,a*real(i-1,wr),"\n",i,a*real(i,wr), " \"
        write(600,*)" "" | psxy $geozone $geoproj -W5 -O -K >>  $file"
        ! -------                                               --------     . traits théoriquea (droite de regression) DIR
        write(600,*)"echo -e "" ",i-1,aDIR*real(i-1,wr),"\n",i,aDIR*real(i,wr), " \"
        write(600,*)" "" | psxy $geozone $geoproj -W5,gray,-.-. -O -K >>  $file"
        ! -------                                               --------     . traits théoriquea (droite de regression) REF
        write(600,*)"echo -e "" ",i-1,aREF*real(i-1,wr),"\n",i,aREF*real(i,wr), " \"
        write(600,*)" "" | psxy $geozone $geoproj -W5,gray,-- -O -K >>  $file"
        ! -------                                               --------     .
      enddo
      ! -------                                                 --------     .
      open(unit=30,file="OUTPUT/GMT/chat-dir"//trim(adjustl(char_nbseismes))//".txt",status='replace')
      do i=1,nDIR                                                             ! points réels des ondes directes
        write(30,*)XYDIR(i,1),XYDIR(i,2),XYDIR(i,3),sDIR(i,1),sDIR(i,2)
      enddo
      close(30)
      write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/chat-dir"//trim(adjustl(char_nbseismes))//".txt -O -K -St0.1i ", &
        "-Wthinnest -Exy -COUTPUT/GMT/colorpal3.cpt >>  $file"
      ! -------                                                 --------     .
      open(unit=31,file="OUTPUT/GMT/chat-ref"//trim(adjustl(char_nbseismes))//".txt",status='replace')
      do i=1,nREF                                                             ! points réels des ondes réfractées
        write(31,*)XYREF(i,1),XYREF(i,2),XYREF(i,3),sREF(i,1),sREF(i,2)
      enddo
      close(31)
      write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/chat-ref"//trim(adjustl(char_nbseismes))//".txt -O -K -Si0.1i ", &
        "-Wthinnest -Exy -COUTPUT/GMT/colorpal3.cpt >>  $file"
      ! -------                                                 --------     . POUR CE SEISME
      open(unit=32,file="OUTPUT/GMT/chat-all"//trim(adjustl(char_nbseismes))//".txt",status='replace')
      if (nbseismes.gt.1) then
```

```
        do i=1,nbtps(j)
          do k=1,nbtps(j)
            if ((D(j)%datatps(i)%andS=='S').and.(D(j)%datatps(k)%andS=='S') &
              .and.(D(j)%datatps(i)%typeonde==D(j)%datatps(k)%typeonde)) then
              one_tps_1%date = D(j)%datatps(i)%tpsR%date
              one_tps_1%sec = D(j)%datatps(i)%tpsR%secP
              one_tps_2%date = D(j)%datatps(k)%tpsR%date
              one_tps_2%sec = D(j)%datatps(k)%tpsR%secP
              call difftime(x1,one_tps_1,one_tps_2)
              one_tps_1%date = D(j)%datatps(i)%tpsR%date
              one_tps_1%sec = D(j)%datatps(i)%tpsR%secS
              one_tps_2%date = D(j)%datatps(k)%tpsR%date
              one_tps_2%sec = D(j)%datatps(k)%tpsR%secS
              call difftime(y1,one_tps_1,one_tps_2)
              write(32,*)abs(x1),abs(y1)
            endif
          enddo
        enddo
        write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/chat-all"//trim(adjustl(char_nbseismes))//".txt ", &
          " -O -K -Sc0.3i -Wthinnest >>  $file"
        close(32)
      endif
      ! ------------------------------------------------------------      .
      write(600,*)"pstext OUTPUT/GMT/chat-stationsDIR.d $geozone $geoproj -O -K -Gorange >> $file"
      write(600,*)"pstext OUTPUT/GMT/chat-stationsREF.d $geozone $geoproj -O -K -Gorange >> $file"
      ! ------------------------------------------------------------      . légende
      write(600,*)"###########################################"
      X = Xmaxi * 0.1_wr
      Y = Ymaxi * 0.8_wr
      if(XYREF(3,1).gt.0.0_wr) then
        ! si il existe des ondes réfractées :
        write(600,*)"echo",X,Y," | psxy $geozone $geoproj -O -K -St0.1i -Wthinnest -Gyellow >>  $file"
        write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5,gray,-.-. >>  $file"
        write(char1,'(f7.4)')aDIR
        write(char2,'(f7.4)')R2DIR
        write(600,*)"echo """,X+0.1_wr*X,Y,"15 0 4 LM ondes directes : V@-P@-/V@-S@- =",char1," \"
        write(600,*)" (@~\143@~@-2@- =",char2,")"," \"
        write(600,*)""" | pstext $geozone $geoproj -O -K >> $file"
        Y = Ymaxi * 0.75_wr
        write(char1,'(f7.4)')aREF
        write(char2,'(f7.4)')R2REF
        write(600,*)"echo",X,Y," | psxy $geozone $geoproj -O -K -Si0.1i -Wthinnest -Gyellow >>  $file"
        write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5,gray,-- >>  $file"
        write(600,*)"echo """,X+0.1_wr*X,Y,"15 0 4 LM ondes r\351fract\351es : V@-P@-/V@-S@- =",char1," \"
        write(600,*)" (@~\143@~@-2@- =",char2,")"" | pstext $geozone $geoproj -O -K >> $file"
        Y = Ymaxi * 0.70_wr
        write(char1,'(f7.4)')a
        write(char2,'(f7.4)')R2
        write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5 >>  $file"
        write(600,*)"echo """,X+0.1_wr*X,Y,"15 0 4 LM ensembles : V@-P@-/V@-S@- =","  \"
        write(600,*)char1," (@~\143@~@-2@- =",char2,")"" | pstext $geozone $geoproj -O -K >> $file"
      else
        Y = Ymaxi * 0.75_wr
        write(char1,'(f7.4)')a
        write(char2,'(f7.4)')R2
        write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5 >>  $file"
        write(600,*)"echo """,X,Y,"15 0 4 LM ondes directes : V@-P@-/V@-S@- =",char1," \"
        write(600,*)" (@~\143@~@-2@- =",char2,")"" | pstext $geozone $geoproj -O -K >> $file"
      endif
      ! -------                                                  --------    .
      write(600,*)"psscale -D1/-1/0.50E+01/0.25ch -B.25:""pond\351ration"": -S -I -COUTPUT/GMT/colorpal3.cpt -O -K >> $file"
      ! -------                                                  --------    .
      write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
      write(600,*)"###########################################"
      write(600,*)"ps2raster OUTPUT/GMT/chatelainplot"//trim(adjustl(char_nbseismes))//".ps -Tf -A"
```

```
356       write(600,'(2a)')"mv OUTPUT/GMT/chatelainplot"//trim(adjustl(char_nbseismes))//".pdf ", &
357       "OUTPUT/figures/chatelainplot"//trim(adjustl(char_nbseismes))//".pdf"
358       write(600,*)"##########################################"
359       write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
360       write(600,*)" echo $ELAPSED secondes"
361       call system_clock(Nnewtime,ratetime)
362       tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
363       write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
364       int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
365       ! ------------------------------------------------------------------ .
366     enddo
367     ! -------------------------------------------------------------------- .
368   end subroutine GMT_chatelain
369
370   ! ---------------------------------------------------------------------- .
371
372   subroutine distancePlot(a,x,y,d)
373     ! -------                                                  --------     .mh
374     ! 1) calcule de d, distance la plus courte entre un point (x,y)
375     ! et la droite de coef directeur a
376     ! ou
377     ! 2) calcul de d, distance horizontale ou verticale maximale
378     ! ou
379     ! 3) calcul de d, distance
380     ! -------                                                  --------     .
381     implicit none
382     ! -------                                                  --------     .
383     real (KIND=wr), intent(in) :: a                                        ! coeficient directeur
384     real (KIND=wr), intent(in) :: x,y                                      ! point de coordonnées (x,y)
385     real (KIND=wr), intent(inout) :: d
386     ! -------                                                  --------     .
387     real (KIND=wr) :: deltaX,deltaY
388     ! -------                                                  --------     .
389     deltaX=abs(y/a-x)
390     deltaY=abs(y-a*x)
391     ! -------                                                  --------     .
392     ! d=d+deltaY*sin(atan(deltaX/deltaY))                                  ! 1)
393     ! d=d+max(deltaX,deltaY)                                               ! 2)
394     !d=d+sqrt(deltaX*deltaX+deltaY*deltaY)                                 ! 3)
395     ! -------                                                  --------     .
396     d=d+max(deltaX,deltaY)
397     ! -------------------------------------------------------------------- .
398   end subroutine distancePlot
399
400 END MODULE figure_GMTchat
401
402
403
404 ! ********************************************************************** .
405 ! ********************************************************************** .
```

2.3 SRC/MOD/MOD_GMT/mkcoda.f90

```
1  ! permet la création des scripts GMT pour l'hodochrone et le calcul de la magnitude Md
2  ! mars 2014
3  ! ********************************************************************** .
4  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          -------- .
5  ! ********************************************************************** .
6  ! ---------------------------------------------------------------------- .
7
8  MODULE figure_GMTcoda
9
10    use modparam
11
12    implicit none
```

```
13
14
15      private
16
17      public  :: GMT_coda
18
19      ! ------------------------------------------------------------------      .
20      ! on défini _FILE_DIR_ en fonction du compilateur
21      ! variable permettant de tester l'existance d'un dossier
22
23  #ifdef __INTEL_COMPILER
24  #define _FILE_DIR_ DIRECTORY
25  #elif __GFORTRAN__
26  #define _FILE_DIR_ FILE
27  #endif
28
29      ! ------------------------------------------------------------------      .
30
31
32  CONTAINS
33
34  ! ------------------------------------------------------------------      .
35
36    subroutine GMT_coda(j,nbtps,datatps,param_best,xmaxcercle,lon,lat,acentroid)
37      ! -------                                                 ---------      .mh
38      ! Calcul les regressions sur les hodochrones
39      ! -------                                                 ---------      .
40      use typetemps
41      use time
42      use statistiques
43      use pb_direct
44      ! -------                                                 ---------      .
45      implicit none
46      integer(KIND=wi), intent (in) :: j
47      integer(KIND=wi), intent(in) :: nbtps
48      type(dataone), intent(in) :: datatps(nbtps)
49      type(parametre), intent(inout) :: param_best
50      real(KIND=wr), intent (in) :: xmaxcercle
51      real(KIND=wr), intent (in) :: lon,lat
52      type(amoho_centroid), intent (in) :: acentroid
53      ! -------                                                 ---------      .
54      type(date_sec) :: one_tps
55      type(stations) :: a_sta
56      integer (KIND=wi) :: i,k,l,ok
57      integer (KIND=wi) :: nPg, nPn ,nSg, nSn, Noldtime, Nnewtime, ratetime
58      real (KIND=wr), dimension(:,:), allocatable :: pt_Pg, pt_Pn ,pt_Sg, pt_Sn
59      real (KIND=wr) :: a_Pg,R2_Pg,a_Pn,b_Pn,R2_Pn,a_Sg,R2_Sg,a_Sn,b_Sn,R2_Sn
60      real (KIND=wr) :: min, max_0, max, discritiqueH
61      real (KIND=wr) :: tl, val,dx,dy,coefa1,coefb1,coefa2,coefb2
62      real (KIND=wr) :: duree,duree1,duree2,ml,depi,depi2
63      real (KIND=wr) :: Tpsmin,Tppmin,Tpsmax,Tppmax
64      real(KIND=wr) :: lon1,lon2,lat1,lat2,v1,v2
65      character (LEN=5) :: numberfile
66      character (LEN=4) :: nomstadoublets(nbtps+1),kstnm
67      logical :: deja,existe1
68      ! ------------------------------------------------------------------      .
69      nPg = 0                                                                   ! nombre de données
70      nPn = 0
71      nSg = 0
72      nSn = 0
73      do i=1,nbtps
74        if(datatps(i)%typeonde.eq.'N') then
75          nPn = nPn + 1
76        elseif(datatps(i)%typeonde.eq.'G') then
77          nPg = nPg + 1
```

```
78        else
79          write(*,*)'problème dans GMT_coda : onde P ni N ni G'
80          stop
81        endif
82        if(datatps(i)%andS.eq.'S') then
83          if(datatps(i)%typeonde.eq.'N') then
84            nSn = nSn + 1
85          elseif(datatps(i)%typeonde.eq.'G') then
86            nSg = nSg + 1
87          else
88            write(*,*)'problème dans GMT_coda : onde S ni N ni G'
89            stop
90          endif
91        endif
92      enddo
93      ! _______                                                  ________     . pour chaque vecteur
94      allocate (pt_Pg(nPg,4))
95      allocate (pt_Pn(nPn,4))
96      allocate (pt_Sg(nSg,4))
97      allocate (pt_Sn(nSn,4))
98      nPg = 0                                                                 ! nombre de données
99      nPn = 0
100     nSg = 0
101     nSn = 0
102     max = 0.0_wr
103     discritiqueH = 0.0_wr
104     do i=1,nbtps
105       one_tps%date = datatps(i)%tpsR%date
106       one_tps%sec = datatps(i)%tpsR%secP
107       if(datatps(i)%typeonde.eq.'N') then
108         nPn = nPn + 1
109         call difftime(pt_Pn(nPn,2),one_tps,param_best%Tzero)
110         pt_Pn(nPn,1) = datatps(i)%dhypo
111         pt_Pn(nPn,3) = datatps(i)%wp
112         if((pt_Pn(nPn,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max =pt_Pn(nPn,2)
113         discritiqueH = discritiqueH + datatps(i)%dcritiqueH
114         pt_Pn(nPn,4) = datatps(i)%sigP
115       elseif(datatps(i)%typeonde.eq.'G') then
116         nPg = nPg + 1
117         call difftime(pt_Pg(nPg,2),one_tps,param_best%Tzero)
118         pt_Pg(nPg,1) = datatps(i)%dhypo
119         pt_Pg(nPg,3) = datatps(i)%wp
120         if((pt_Pg(nPg,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max = pt_Pg(nPg,2)
121         pt_Pg(nPg,4) = datatps(i)%sigP
122       else
123         write(*,*)'problème dans GMT_coda : onde P ni N ni G'
124         stop
125       endif
126       if(datatps(i)%andS.eq.'S') then
127         if(datatps(i)%typeonde.eq.'N') then
128           nSn = nSn + 1
129           one_tps%date = datatps(i)%tpsR%date
130           one_tps%sec = datatps(i)%tpsR%secS
131           call difftime(pt_Sn(nSn,2),one_tps,param_best%Tzero)
132           pt_Sn(nSn,1) = datatps(i)%dhypo
133           pt_Sn(nSn,3) = datatps(i)%ws
134           if((pt_Sn(nSn,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max =pt_Sn(nSn,2)
135           discritiqueH = discritiqueH + datatps(i)%dcritiqueH
136           pt_Sn(nSn,4) = datatps(i)%sigS
137         elseif(datatps(i)%typeonde.eq.'G') then
138           nSg = nSg + 1
139           one_tps%date = datatps(i)%tpsR%date
140           one_tps%sec = datatps(i)%tpsR%secS
141           call difftime(pt_Sg(nSg,2),one_tps,param_best%Tzero)
142           pt_Sg(nSg,1) = datatps(i)%dhypo
```

```
          pt_Sg(nSg,3) = datatps(i)%ws
          pt_Sg(nSg,4) = datatps(i)%sigS
          if((pt_Sg(nSg,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max = pt_Sg(nSg,2)
        else
          write(*,*)'problème dans GMT_coda : onde S ni N ni G'
          stop
        endif
      endif
    enddo
    discritiqueH = discritiqueH / real(nSn+nPn,wr)
    ! ------                                                    --------     .
    call correlationaffpond(a_Pg,R2_Pg,nPg,pt_Pg)
    call correlationpond(a_Pn,b_Pn,R2_Pn,nPn,pt_Pn)
    call correlationaffpond(a_Sg,R2_Sg,nSg,pt_Sg)
    call correlationpond(a_Sn,b_Sn,R2_Sn,nSn,pt_Sn)
    ! ------                                                    --------     .
    max_0 = max
    ! ---------------------------------------------------------------     .
    write(*,*)"ecriture du script GMT Coda"
    write(600,*)"thegray1=240/240/240"
    write(600,*)"thegray2=230/230/230"
    write(600,*)"thegray3=220/220/220"
    write(600,*)"BEFORE=$SECONDS"
    call system_clock(Noldtime)
    write(600,*)"#########################################"
    write(600,*)"################## coda #################"
    write(600,*)"echo 'execution du script GMT coda Md'"
    write(numberfile(1:5),'(i5)')j
    write(600,*)"file=OUTPUT/GMT/coda"//"-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"geoproj=-JX13i/8i"                                 ! système de projection
    if (xmaxcercle/2.0_wr.lt.datatps(nbtps)%dhypo) then
      val=xmaxcercle/2.0_wr
    else
      val=datatps(nbtps)%dhypo
    endif
    write(600,'(a,E13.7,a,E13.7)')"geozone=-R0.0/",val*1.1_wr,"/-30.0/",max_0+500.0_wr
    if (max_0.gt.60.0_wr) then
      write(600,*)"psbasemap $geozone $geoproj -Ba50f25:""distance hypocentrale (km)"":",&
      "/a100f25:""temps d'arriv\351es des ondes (s)"":WenS -Xc -Yc -K >  $file"
    else
      write(600,*)"psbasemap $geozone $geoproj -Ba20f5:""distance hypocentrale (km)"":",&
      "/a100f25:""temps d'arriv\351es des ondes (s)"":WenS -Xc -Yc -K >  $file"
    endif
    ! ---------------------------------------------------------------     .
    ! ------                                                    --------     . ONDES DIRECTES
    min = 0.0_wr
    max = val*1.09_wr
    ! ------                                                    --------     . Pg par modèle
    call directe(param_best,min,Tpsmin,Tppmin)
    call directe(param_best,max,Tpsmax,Tppmax)
    write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$pp -O -K >>  $file"
    ! ------                                                    --------     . Sg par modèle
    write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$ss -O -K >>  $file"
    ! ------                                                    --------     . ONDES REFRACTEES
    min = discritiqueH
    if (IsNaN(discritiqueH)) min=max-5.0_wr ! arbitraire ...
    ! ------                                                    --------     . Pn par modèle
    if(FLAG_non_tabulaire) then                                     ! moho incliné
      a_sta%lon=param_best%lon
      a_sta%lat=param_best%lat
      a_sta%alti=0.0_wr
      call refracte_mohovar(acentroid,param_best,a_sta,min,Tpsmin,Tppmin)
      call refracte_mohovar(acentroid,param_best,a_sta,max,Tpsmax,Tppmax)
```

```
208     else
209       call refracte(param_best,min,Tpsmin,Tppmin)
210       call refracte(param_best,max,Tpsmax,Tppmax)
211     endif
212
213     write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
214     write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$pp,- -O -K >>  $file"
215     ! ------                                                  --------     . Sn par modèle
216     write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
217     write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$ss,- -O -K >>  $file"
218     ! ------------------------------------------------------------------   .
219     inquire (_FILE_DIR_="DATA/sac-"//trim(adjustl(numberfile)),exist=existe1) ! option différente selon compilo !
220     if ((existe1).and.(tracessac)) then
221       ! ------------------------------------------------------------------    . plot traces si existes
222       do i=1,nbtps
223         ! lecture un peu archaïque, mais permet un peu de souplesse dans le non des station et des fichiers sac
224         ! -------                                                 --------     . COMPOSANTE Z
225         if (datatps(i)%sta%staname(4:4)=='0') then                             ! nom station en trois caractere + "0"
226           write(600,'(5a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
227             "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
228             "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
229             "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
230             " 2>/dev/null | uniq | while read nom "
231         else
232           write(600,'(3a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
233             "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*Z.*SA* ", &
234             " 2>/dev/null | uniq | while read nom "
235         endif
236         write(600,*)" do "
237         write(600,*)" nombis=${nom/'DATA'/'OUTPUT'}"
238         write(600,*)" nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
239         write(600,*)" psxy $geozone $geoproj -W2,$thegray2 -O -K $nomter.txt -: >>  $file"
240         write(600,*)" done "
241         write(600,*)"rm -rf toto2.txt"
242         ! -------                                                 --------     . COMPOSANTE E
243         if (datatps(i)%sta%staname(4:4)=='0') then                             ! nom station en trois caractere + "0"
244         write(600,'(5a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*E.*SA* ", &
245           "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*E.*SA* ", &
246           "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*E.*SA* ", &
247           "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*E.*SA* ", &
248             " 2>/dev/null | uniq | while read nom "
249         else
250           write(600,'(3a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*E.*SA* ", &
251             "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*Z.*SA* ", &
252             " 2>/dev/null | uniq | while read nom "
253         endif
254         write(600,*)" do "
255         write(600,*)" nombis=${nom/'DATA'/'OUTPUT'}"
256         write(600,*)" nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
257         write(600,*)" psxy $geozone $geoproj -W2,$thegray2 -O -K $nomter.txt -: >>  $file"
258         write(600,*)" done "
259         write(600,*)"rm -rf toto2.txt"
260         ! -------                                                 --------     . COMPOSANTE N
261         if (datatps(i)%sta%staname(4:4)=='0') then                             ! nom station en trois caractere + "0"
262         write(600,'(5a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*N.*SA* ", &
263           "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*N.*SA* ", &
264           "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*N.*SA* ", &
265           "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*N.*SA* ", &
266             " 2>/dev/null | uniq | while read nom "
267         else
268           write(600,'(3a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
269             "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*N.*SA* ", &
270             " 2>/dev/null | uniq | while read nom "
271         endif
272         write(600,*)" do "
```

```
273           write(600,*)" nombis=${nom/'DATA'/'OUTPUT'}"
274           write(600,*)" nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
275           write(600,*)" psxy $geozone $geoproj -W2,$thegray3 -O -K $nomter.txt -: >>  $file"
276           write(600,*)" done "
277           write(600,*)"rm -rf toto2.txt"
278           ! -------                                                  --------     .
279         enddo
280         ! ------------------------------------------------------------------     .
281         ! calcul de la Magnitude Md                                              !
282         ! ------------------------------------------------------------------     .
283         ok=0
284         open(599-j, FILE = 'OUTPUT/GMT/scriptmag'//trim(adjustl(numberfile))//'.sh',status='replace',iostat = ok)
285         if(ok.ne.0) then
286           write(*,*)'problème dans GMT_coda : OUTPUT/GMT/scriptmag'//trim(adjustl(numberfile))//'.sh n''existe pas '
287           stop
288         endif
289         ! -------                                                  --------     . distance Pn > Pg
290         call directe(param_best,min,Tpsmin,Tppmin)                             ! coef dir. droite Pg
291         call directe(param_best,max,Tpsmax,Tppmax)
292         coefa1=(Tppmax-Tppmin)/(max-min)
293         coefb1=-coefa1*max+Tppmax
294         if(FLAG_non_tabulaire) then                                            ! moho incliné
295           a_sta%lon=param_best%lon
296           a_sta%lat=param_best%lat
297           a_sta%alti=0.0_wr
298           call refracte_mohovar(acentroid,param_best,a_sta,min,Tpsmin,Tppmin)
299           call refracte_mohovar(acentroid,param_best,a_sta,max,Tpsmax,Tppmax)
300         else
301           call refracte(param_best,min,Tpsmin,Tppmin)                          ! coef dir. droite Pg
302           call refracte(param_best,max,Tpsmax,Tppmax)
303         endif
304         coefa2=(Tppmax-Tppmin)/(max-min)
305         coefb2=-coefa2*max+Tppmax
306         call deuxdroites(coefa1,coefb1,coefa2,coefb2,dx,dy)                    ! intersection des droites Pg et Pn et dx
307         if (IsNaN(dx)) then
308           write(*,*)'problème dans GMT_coda : dx = NaN'
309           stop
310         endif
311         if (dx.lt.0.0_wr) then
312           write(*,*)'problème dans GMT_coda : dx < 0, bizarre !'
313           stop
314         endif
315         ! -------                                                  --------     .
316         do i=1,nbtps
317           if ((((datatps(i)%typeonde=='G').and.(datatps(i)%dhypo.le.dx)).or. &
318                ((datatps(i)%typeonde=='N').and.(datatps(i)%dhypo.ge.dx))).and. &
319                (datatps(i)%depi.gt.10.0_wr)) then ! premieres arrivée P (Pg ou Pn), distance épi > 10 km sinon saturation
320             ! lecture un peu archaïque, mais permet un peu de souplesse dans le non des station et des fichiers sac
321             ! -------                                                --------     .
322             if (datatps(i)%sta%staname(4:4)=='0') then                         ! nom station en trois caractères + "0"
323               write(599-j,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
324                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
325                 "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
326                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
327                 "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
328                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
329                 "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
330                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
331                 " 2>/dev/null | uniq | while read nom "
332               write(600,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
333                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
334                 "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
335                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
336                 "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
337                 "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
```

```
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
          " 2>/dev/null | uniq | while read nom "
      else
        write(599-j,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*sa* ", &
          " 2>/dev/null | uniq | while read nom "
        write(600,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*SA* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
          "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*Z.*sa* ", &
          " 2>/dev/null | uniq | while read nom "
      endif
      ! ------                                                ________    .
      write(599-j,'(a)')"do"
      write(600,'(a)')"do"
      write(599-j,'(a)')"deux=${nom/HZ/HE}"
      write(599-j,'(a)')"trois=${nom/HZ/HN}"
      ! ------                                                ________    . script sac
      write(599-j,'(a)')"sac << EOF >/dev/null"
      write(599-j,'(a)')"r $nom"
      write(599-j,'(a)')"mulf $nom"
      write(599-j,'(a)')"w 1.sac"
      write(599-j,'(a)')"r $deux"
      write(599-j,'(a)')"mulf $deux"
      write(599-j,'(a)')"w 2.sac"
      write(599-j,'(a)')"r $trois"
      write(599-j,'(a)')"mulf $trois"
      write(599-j,'(a)')"w 3.sac"
      write(599-j,'(a)')"r 1.sac"
      write(599-j,'(a)')"addf 2.sac"
      write(599-j,'(a)')"addf 3.sac"
      ! ------                                                ________    .
      ! new.sac = env(Z)**2 + env(N)**2 + env(E)**2 !
      ! ------                                                ________    .
      write(599-j,'(a)')"w new.SAC"
      write(599-j,'(a)')"quit"
      write(599-j,'(a)')"EOF"
      ! ------                                                ________    .
      write(599-j,'(a)')"nombis=${nom/'DATA'/'OUTPUT'}"
      write(599-j,'(a)')"nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
      write(600,'(a)')"nombis=${nom/'DATA'/'OUTPUT'}"
      write(600,'(a)')"nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
      write(599-j,*)" echo '",j,datatps(i)%tpsTh,param_best%Tzero,datatps(i)%depi, &
        datatps(i)%dhypo,val/15.0_wr,"'> toto1.txt"
      write(599-j,'(a)')" ./BIN/sac_coda.exe new.SAC $nomter-coda-"//trim(adjustl(numberfile))//".txt < toto1.txt "
      write(599-j,'(a)')"rm -rf toto1.txt "
      write(599-j,'(a)')"done "
      ! ------                                                ________    .
      write(600,*)"if test -f $nomter-coda-"//trim(adjustl(numberfile))//".txt ; then"
      write(600,*)" psxy $geozone $geoproj -W4,red -O -K $nomter-coda-"//trim(adjustl(numberfile))//".txt >>  $file"
      write(600,*)"fi"
      write(600,'(a)')"done "
      ! ------                                                ________    .
      write(599-j,'(a)')"rm -rf 1.sac 2.sac 3.sac new.SAC"
```

```
403           ! -------                                                  --------     .
404           else
405             write(599-j,'(a)')'### no mag :  '//datatps(i)%sta%staname//" - "//trim(adjustl(numberfile))
406           endif
407         enddo
408         Close(599-j)
409         ! -------                                                  --------     . pour gfortran :
410         !call execute_command_line ("chmod +x OUTPUT/GMT/scriptmag"//trim(adjustl(numberfile))//".sh", wait=.true.)
411         !call execute_command_line ("./OUTPUT/GMT/scriptmag"//trim(adjustl(numberfile))//".sh", wait=.true.)
412         ! -------                                                  --------     . pour gfortran & ifort :
413         call system ("chmod +x OUTPUT/GMT/scriptmag"//trim(adjustl(numberfile))//".sh")
414         call system ("./OUTPUT/GMT/scriptmag"//trim(adjustl(numberfile))//".sh")
415         ! -------                                                  --------     .
416         ! ---------------------------------------------------------------     .
417       endif
418       ! ---------------------------------------------------------------     .
419       ! ---------------------------------------------------------------     .
420       ! -------                                                  --------     . nom sta
421       l=1
422       do i=1,nbtps+1
423         nomstadoublets(i)='xxxx'
424       enddo
425       do i=1,nbtps
426         ! -------                                                  --------     . nom déja affiché ?
427         deja =.true.
428         do k=1,l
429           if (nomstadoublets(k).eq.datatps(i)%sta%staname) deja=.false.
430         enddo
431         ! -------                                                  --------     . affiche
432         if (deja) then
433           if (mod(i,3)==0) then
434             write(600,*)"echo '",datatps(i)%dhypo,max_0*0.9_wr, &
435             " 7 90 1 5 "//datatps(i)%sta%staname//"' | pstext $geoproj $geozone -O -K -C2 >> $file"
436             nomstadoublets(l)=datatps(i)%sta%staname
437             l=l+1
438           elseif (mod(i,2)==0) then
439             write(600,*)"echo '",datatps(i)%dhypo,max_0*0.8_wr, &
440             " 7 90 1 5 "//datatps(i)%sta%staname//"' | pstext $geoproj $geozone -O -K -C2 >> $file"
441             nomstadoublets(l)=datatps(i)%sta%staname
442             l=l+1
443           else
444             write(600,*)"echo '",datatps(i)%dhypo,max_0*0.7_wr, &
445             " 7 90 1 5 "//datatps(i)%sta%staname//"' | pstext $geoproj $geozone -O -K -C2 >> $file"
446             nomstadoublets(l)=datatps(i)%sta%staname
447             l=l+1
448           endif
449         endif
450       enddo
451
452
453       ! -------                                                  --------     . trace les magnitudes théoriques :
454       do i=1,5 ! Ml
455         ml=real(i,wr)
456         write(numberfile(1:5),'(i5)')i
457         depi=0.0_wr
458         do while (depi.lt.(val*1.1_wr))
459           duree1=10.0_wr**((ml+ 0.87_wr-0.0035_wr*depi)/2.0_wr)+depi/param_best%VC
460           depi2=depi+2._wr
461           duree2=10.0_wr**((ml+ 0.87_wr-0.0035_wr*depi2)/2.0_wr)+depi2/param_best%VC
462           write(600,*)"echo -e "" ",depi,duree1,"\n",depi2,duree2," \"
463           write(600,*)" "" | psxy $geozone $geoproj -W0.01i,blue -O -K >> $file"
464           depi=depi2
465         enddo
466         depi=10.0_wr
467         duree1=10.0_wr**((ml+ 0.87_wr-0.0035_wr*depi)/2.0_wr)!depi/param_best%VC
```

```
      write(600,*)"echo '",depi,duree1," 15 0 1 5 Md ="//trim(adjustl(numberfile))//"'",&
      " | pstext $geoproj $geozone -O -K -C2 >> $file"
    enddo
    write(numberfile(1:5),'(i5)')j
    ! -------                                                    --------     .
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
    write(600,*)"#########################################"
    write(600,*)"ps2raster OUTPUT/GMT/coda"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/coda"//"-"//trim(adjustl(numberfile))//".pdf ", &
    "OUTPUT/figures/coda"//"-"//trim(adjustl(numberfile))//".pdf "
    ! ------------------------------------------------------------     .
    deallocate (pt_Pg,pt_Pn,pt_Sg,pt_Sn)
    ! ------------------------------------------------------------     .
    ! plot carte avec des cercles à chaque station, focntion de la magnitude
    ! ------------------------------------------------------------     .
    write(600,*)"#########################################"
    write(*,*)"ecriture du script GMT_coda_map "
    write(600,*)"#########################################"
    write(numberfile(1:5),'(i5)')j
    write(600,*)"file=OUTPUT/GMT/coda_map-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"gmtset BASEMAP_TYPE plain"
    write(600,*)"labasemap1=-Bpa2g1.f.5/a1g1.f.25WeSn"
    ! -------                                                    --------     .
    v1 = 2.0_wr * pi * rT / 360.0_wr                                        ! km / degree en lon
    v2 = 2.0_wr * pi * rT * sin((90.0_wr-lat)/180.0_wr*pi) /360.0_wr        ! km / degree en lat
    ! -------                                                    --------     .
    lon1 = lon - (xmaxcercle / v2 * 1.125_wr) / 2.0_wr
    lon2 = lon + (xmaxcercle / v2 * 1.125_wr) / 2.0_wr
    lat1 = lat - (xmaxcercle / v1 * 1.125_wr) / 2.0_wr
    lat2 = lat + (xmaxcercle / v1 * 1.125_wr) / 2.0_wr
    ! -------                                                    --------     .
    write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",lon1,"/",lon2,"/",lat1,"/",lat2
    write(600,'(a11,E13.7,a1,E13.7,a3)')"geoproj=-JC",lon,"/",lat,"/7i"
    ! -------                                                    --------     .
    write(600,*)"bluef=""0/0/100"" "
    write(600,*)"makecpt -Cseis -I -T1.5/4.5/0.01 -Z > OUTPUT/GMT/neis.cpt"
    write(600,*)"pscoast $geozone $geoproj -Df+ -Ia/$bluef -S240/255/255 -G180/238/180 -W1 -K -Xc -X5.5i -Yc $labasemap1 >  $file"
    write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -m >> $file"
    ! -------                                                    --------     .
    inquire (FILE="OUTPUT/files/mag-"//trim(adjustl(numberfile))//".d",exist=existe1) ! option différente selon compilo !
    ok=0
    ! -------                                                    --------     .
    do while (.not.existe1)
      ok=ok+1
      call sleep(5)
      inquire (FILE="OUTPUT/files/mag-"//trim(adjustl(numberfile))//".d",exist=existe1) ! option différente selon compilo !
      if (ok.gt.5) then
        ! write(*,*)'problème dans GMT_coda : le fichier OUTPUT/files/mag-'//trim(adjustl(numberfile))//'.d n''existe pas, ok=',ok
        existe1=.true.
      endif
    enddo
    ! -------                                                    --------     .
    ok=0
    ! -------                                                    --------     .
    open(111, FILE ="OUTPUT/files/mag-"//trim(adjustl(numberfile))//".d",status='old',iostat = ok)
    if (ok .ne. 0) then
      write(600,*)'psxy $geozone $geoproj OUTPUT/GMT/ellipse-'//trim(adjustl(numberfile))//'.txt ', &
        '-Sa0.5 -W1,gray -Gblue -O >> $file'
    else
      do while(ok .eq. 0)
        ! -------                                                --------     .
        read(111,*,iostat = ok)kstnm,ml,duree,depi
        if (kstnm(4:4)==' ')kstnm(4:4)='0'
        if (ok .eq. 0) then
          do i=1,nbtps
```

```
533             if (datatps(i)%sta%staname==kstnm) then ! si bonne station
534                write(600,*)"echo ",datatps(i)%sta%lon,datatps(i)%sta%lat,ml,ml*0.08_wr-0.04_wr, &
535                  " | psxy $geozone $geoproj -Sci -Wthinnest -O -K -Ba0 -COUTPUT/GMT/neis.cpt >>  $file"
536             endif
537           enddo
538         endif
539         ! -------                                                    --------     .
540       end do
541       close(111)
542       ! ------------------------------------------------------------------     .
543       write(600,*)"echo '",lon,lat,"0 300 300' | psxy $geozone $geoproj -SE -W5,-- -O -K -N >> $file"
544       ! -------                                                    --------     .
545       write(600,*)'psxy $geozone $geoproj OUTPUT/GMT/ellipse-'//trim(adjustl(numberfile))//'.txt ', &
546         '-Sa0.5 -W1,gray -Gblue -O -K >> $file'
547       write(600,*)"echo '>' > OUTPUT/GMT/mag.legend"
548       write(600,*)"echo 'N4' >> OUTPUT/GMT/mag.legend"
549       write(600,*)"echo 'G0.3i' >> OUTPUT/GMT/mag.legend"
550       write(600,*)"echo 'S 0.28i c 0.08i  0/0/205 0.5p 0.525i  M@-d@- : 1.5' >> OUTPUT/GMT/mag.legend"
551       write(600,*)"echo 'S 0.28i c 0.12i  0/160/183 0.5p 0.525i  M@-d@- : 2.0' >> OUTPUT/GMT/mag.legend"
552       write(600,*)"echo 'S 0.28i c 0.16i  090/255/030 0.5p 0.525i  M@-d@- : 2.5' >> OUTPUT/GMT/mag.legend"
553       write(600,*)"echo 'S 0.28i c 0.20i  255/255/0 0.5p 0.525i  M@-d@- : 3.0' >> OUTPUT/GMT/mag.legend"
554       write(600,*)"echo 'G0.15i' >> OUTPUT/GMT/mag.legend"
555       write(600,*)"echo 'S 0.28i c 0.24i  255/170/0 0.5p 0.525i  M@-d@- : 3.5' >> OUTPUT/GMT/mag.legend"
556       write(600,*)"echo 'S 0.28i c 0.28i  255/042/0 0.5p 0.525i  M@-d@- : 4.0' >> OUTPUT/GMT/mag.legend"
557       write(600,*)"echo 'S 0.28i c 0.32i  173/0/0 0.5p 0.525i  M@-d@- : 4.5' >> OUTPUT/GMT/mag.legend"
558       write(600,*)"echo 'S 0.28i c 0.36i  0/0/0 0.5p .525i M@-d@- : 5.0' >> OUTPUT/GMT/mag.legend"
559       write(600,*)"pslegend -Dx4.5i/-0.4i/7i/1.i/TC $geozone $geoproj -O OUTPUT/GMT/mag.legend >> $file"
560     endif
561     ! -------                                                    --------     .
562     write(600,*)"ps2raster OUTPUT/GMT/coda_map-"//trim(adjustl(numberfile))//".ps -Tf -A"
563     write(600,'(2a)')"mv OUTPUT/GMT/coda_map-"//trim(adjustl(numberfile))//".pdf ", &
564       "OUTPUT/figures/coda_map-"//trim(adjustl(numberfile))//".pdf"
565     ! ------------------------------------------------------------------     .
566     write(600,*)"#############################################"
567     write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
568     write(600,*)" echo $ELAPSED secondes"
569     call system_clock(Nnewtime,ratetime)
570     tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
571     write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
572     int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
573     ! ------------------------------------------------------------------     .
574   end subroutine GMT_coda
575
576     ! ------------------------------------------------------------------     .
577
578   subroutine deuxdroites(a1,b1,a2,b2,x,y)
579     ! -------                                                    --------     .mh
580     ! point d'intesection (x,y) de deux droites 1 et 2,
581     ! de coef. dir. a et ordonnée à l'origine b
582     ! -------                                                    --------     .
583     implicit none
584     real(KIND=wr), intent (in) :: a1,b1,a2,b2
585     real(KIND=wr), intent (out) :: x,y
586     ! ------------------------------------------------------------------     .
587     if ((a1-a2).ne.0.0_wr) then
588       x=(b2-b1)/(a1-a2)
589       y=a1*x+b1
590     else
591       x=0.0_wr
592       y=a1*x+b1
593     endif
594     ! ------------------------------------------------------------------     .
595   end subroutine deuxdroites
596
597 END MODULE figure_GMTcoda
```

```
598
599
600
601 ! ********************************************************************     .
602 ! ********************************************************************     .
```

2.4 SRC/MOD/MOD_GMT/mkfcout.f90

```
1  ! permet la création des scripts GMT pour les figures sur le fonction coût
2  ! mars 2014
3  ! ********************************************************************     .
4  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
5  ! ********************************************************************     .
6  ! ------------------------------------------------------------------     .
7
8  MODULE figure_GMTfc
9
10     use modparam
11
12     implicit none
13
14     private
15
16     public  :: GMT_fc
17
18
19 CONTAINS
20
21 ! ------------------------------------------------------------------     .
22
23   subroutine GMT_fc(dp,nbChaineMV,nmod)
24     ! -------                                                 --------     .mh
25     use typetemps, only : densityplot
26     use figure_GMTpar, only : RVB
27     ! -------                                                 --------     .
28     implicit none
29     ! -------                                                 --------     .
30     type(densityplot), intent (in) :: dp                                    ! modèles retenus par McMC
31     integer(KIND=wi), intent (in) :: nbChaineMV
32     integer(KIND=wi), intent (in) :: nmod(nbChaineMV)
33     ! -------                                                 --------     .
34     integer(KIND=wi) :: i,j,k,maxiter,Noldtime, Nnewtime, ratetime
35     character(LEN=5) :: char
36     real(KIND=wr) :: tl
37     character(LEN=11) :: color
38     ! ------------------------------------------------------------------     . ecriture des misfit par chaîne
39     k=0
40     maxiter=0
41     do i=1,nbChaineMV
42        write(char,'(i5)')10000+i
43        open(unit=850+i,file="OUTPUT/GMT/themis"//char//".bin",STATUS="replace",access='direct',RECL=16)
44        do j=1,nmod(i)
45           k=k+1
46           write(850+i,rec=j)real(dp%mis%vec(k),8),real(j,8)
47           if(maxiter.lt.j)maxiter=j
48        enddo
49        close(850+i)
50     enddo
51     ! ------------------------------------------------------------------     . script GMT
52     write(*,*)"ecriture des script GMT_fc "
53     write(600,*)"BEFORE=$SECONDS"
54     call system_clock(Noldtime)
55     write(600,*)"#####################################"
56     write(600,*)"#####################################"
57     write(600,*)
```

```
    write(600,*)"echo 'execution du script GMT fonction coût'"
    write(600,*)"#########################################"
    write(600,*)"###########  fonction coût     ##########"
    write(600,*)"#########################################"
    ! ------------------------------------------------------------------ .
    write(600,'(a,E13.7,a1,E13.7)')"geozone=-R0/7.5/",dp%mis%themin-2.0_wr,"/",dp%mis%themax+2.0_wr
    write(600,*)"geoproj=-JX2i/4.5i"
    write(600,*)"file=OUTPUT/GMT/mishisto.ps"
    ! ------------------------------------------------------------------ . histogramme de la fonction cout
    write(600,*)"psbasemap $geozone $geoproj -Ba5f1:'effectif (%)':/a10Snew -K -X10i >  $file"
    write(600,'(a,E13.7,a1,E13.7,a)')"geozone=-R",dp%mis%themin-2.0_wr,"/",dp%mis%themax+2.0_wr,"/0/10"
    do i=1,nbChaineMV
      write(char,'(i5)')10000+i
      write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/themis"//char//".bin -bi1d -W0.2", &
      " -Ggray -Z1 -O -K -A >>  $file"
    enddo
    do i=1,nbChaineMV
      write(char,'(i5)')10000+i
      write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/themis"//char//".bin -bi2d", &
      " -W0.2 -S -L0/0 -Z1 -O -K -A >>  $file"
    enddo
    write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/lon_lat-1_tot.bin -bi3d -T2 -W0.2", &
    " -Gblue -Z1 -K -O -A  >>  $file"
    write(600,*)"pshistogram $geozone $geoproj OUTPUT/GMT/lon_lat-1_tot.bin -bi3d -T2 -W0.2 -S", &
    "  -L0/0 -Z1 -O -K -A >>  $file"
    ! ------------------------------------------------------------------ .
    write(600,'(a,i9.9,a,E13.7,a,E13.7)')"geozone=-R0/",maxiter,"/",dp%mis%themin-2.0_wr,"/",dp%mis%themax+2.0_wr
    write(600,*)"geoproj=-JX8i/4.5i"
    write(600,'(a31,i9.9,a)')"psbasemap $geozone $geoproj -Ba",(maxiter/1000)*100, &
    ":'mod\350les':/a10:'fonction co\373t':nSeW -K -O -X-8.5i >>  $file"

    do i=1,nbChaineMV
      write(char,'(i5)')10000+i
      call RVB(i,nbChaineMV,color)
      write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/themis"//char//".bin -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
    enddo
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
    ! ------------------------------------------------------------------ .
    write(600,*)"ps2raster OUTPUT/GMT/mishisto.ps -Tf -A"
    write(600,'(a)')"mv OUTPUT/GMT/mishisto.pdf OUTPUT/figures/mishisto.pdf"
    ! ------                                                  --------     . fin script GMT
    write(600,*)
    write(600,*)"#######################################"
    write(600,*)"#######################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
    int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------------ .
  end subroutine GMT_fc

END MODULE figure_GMTfc



! ********************************************************************** .
! ********************************************************************** .
```

2.5 SRC/MOD/MOD_GMT/mkhodo.f90

```
! permet la création des scripts GMT pour l'hodochrone
! mars 2014
! ********************************************************************** .
```

```
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
! ************************************************************************     .
! ------------------------------------------------------------------------     .

MODULE figure_GMThodo

    use modparam

    implicit none


    private

    public  :: GMT_Hodochrone


    ! ------------------------------------------------------------------     .
    ! on défini _FILE_DIR_ en fonction du compilateur
    ! variable permettant de tester l'existance d'un dossier

#ifdef __INTEL_COMPILER
#define _FILE_DIR_ DIRECTORY
#elif __GFORTRAN__
#define _FILE_DIR_ FILE
#endif

    ! ------------------------------------------------------------------     .


CONTAINS

! ------------------------------------------------------------------------     .

  subroutine GMT_Hodochrone(j,nbtps,datatps,param_best,xmaxcercle,acentroid)
    ! -------                                                 --------     .mh
    ! Calcul les regressions sur les hodochrones et affiche l'hodochrone
    ! -------                                                 --------     .
    use typetemps
    use time
    use statistiques
    use pb_direct
    ! -------                                                 --------     .
    implicit none
    integer(KIND=wi), intent (in) :: j
    integer(KIND=wi), intent(in) :: nbtps
    type(dataone), intent(in) :: datatps(nbtps)
    type(parametre), intent(inout) :: param_best
    real(KIND=wr), intent (in) :: xmaxcercle
    type(amoho_centroid), intent (in) :: acentroid
    ! ------                                                  --------     .
    type(date_sec) :: one_tps
    type(stations) :: a_sta
    integer (KIND=wi) :: i,k,l
    integer (KIND=wi) :: nPg, nPn ,nSg, nSn, Noldtime, Nnewtime, ratetime
    real (KIND=wr), dimension(:,:), allocatable :: pt_Pg, pt_Pn ,pt_Sg, pt_Sn
    real (KIND=wr) :: a_Pg,R2_Pg,a_Pn,b_Pn,R2_Pn,a_Sg,R2_Sg,a_Sn,b_Sn,R2_Sn
    real (KIND=wr) :: min, max_0, max, discritiqueH
    real (KIND=wr) :: X, Y, tl, val
    real (KIND=wr) :: Tpsmin,Tppmin,Tpsmax,Tppmax
    character (LEN=5) :: numberfile
    character (LEN=4) :: nomstadoublets(nbtps+1)
    logical :: deja,existe1
    ! ------------------------------------------------------------------     .
    nPg = 0                                                                  ! nombre de données
    nPn = 0
```

```
    nSg = 0
    nSn = 0
    do i=1,nbtps
      if(datatps(i)%typeonde.eq.'N') then
        nPn = nPn + 1
      elseif(datatps(i)%typeonde.eq.'G') then
        nPg = nPg + 1
      else
        write(*,*)'problème dans GMT_Hodochrone : onde P ni N ni G'
        stop
      endif
      if(datatps(i)%andS.eq.'S') then
        if(datatps(i)%typeonde.eq.'N') then
          nSn = nSn + 1
        elseif(datatps(i)%typeonde.eq.'G') then
          nSg = nSg + 1
        else
          write(*,*)'problème dans GMT_Hodochrone : onde S ni N ni G'
          stop
        endif
      endif
    enddo
    ! -------                                                  --------    . pour chaque vecteur
    allocate (pt_Pg(nPg,4))
    allocate (pt_Pn(nPn,4))
    allocate (pt_Sg(nSg,4))
    allocate (pt_Sn(nSn,4))
    nPg = 0                                                               ! nombre de données
    nPn = 0
    nSg = 0
    nSn = 0
    max = 0.0_wr
    discritiqueH = 0.0_wr
    do i=1,nbtps
      one_tps%date = datatps(i)%tpsR%date
      one_tps%sec = datatps(i)%tpsR%secP
      if(datatps(i)%typeonde.eq.'N') then
        nPn = nPn + 1
        call difftime(pt_Pn(nPn,2),one_tps,param_best%Tzero)
        pt_Pn(nPn,1) = datatps(i)%dhypo
        pt_Pn(nPn,3) = datatps(i)%wp
        if((pt_Pn(nPn,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max =pt_Pn(nPn,2)
        discritiqueH = discritiqueH + datatps(i)%dcritiqueH
        pt_Pn(nPn,4) = datatps(i)%sigP
      elseif(datatps(i)%typeonde.eq.'G') then
        nPg = nPg + 1
        call difftime(pt_Pg(nPg,2),one_tps,param_best%Tzero)
        pt_Pg(nPg,1) = datatps(i)%dhypo
        pt_Pg(nPg,3) = datatps(i)%wp
        if((pt_Pg(nPg,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max = pt_Pg(nPg,2)
        pt_Pg(nPg,4) = datatps(i)%sigP
      else
        write(*,*)'problème dans GMT_Hodochrone : onde P ni N ni G'
        stop
      endif
      if(datatps(i)%andS.eq.'S') then
        if(datatps(i)%typeonde.eq.'N') then
          nSn = nSn + 1
          one_tps%date = datatps(i)%tpsR%date
          one_tps%sec = datatps(i)%tpsR%secS
          call difftime(pt_Sn(nSn,2),one_tps,param_best%Tzero)
          pt_Sn(nSn,1) = datatps(i)%dhypo
          pt_Sn(nSn,3) = datatps(i)%ws
          if((pt_Sn(nSn,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max =pt_Sn(nSn,2)
          discritiqueH = discritiqueH + datatps(i)%dcritiqueH
```

```
134           pt_Sn(nSn,4) = datatps(i)%sigS
135         elseif(datatps(i)%typeonde.eq.'G') then
136           nSg = nSg + 1
137           one_tps%date = datatps(i)%tpsR%date
138           one_tps%sec = datatps(i)%tpsR%secS
139           call difftime(pt_Sg(nSg,2),one_tps,param_best%Tzero)
140           pt_Sg(nSg,1) = datatps(i)%dhypo
141           pt_Sg(nSg,3) = datatps(i)%ws
142           pt_Sg(nSg,4) = datatps(i)%sigS
143           if((pt_Sg(nSg,2).gt.max).and.(datatps(i)%depi.lt.xmaxcercle/2.0_wr)) max = pt_Sg(nSg,2)
144         else
145           write(*,*)'problème dans GMT_Hodochrone : onde S ni N ni G'
146           stop
147         endif
148       endif
149     enddo
150     discritiqueH = discritiqueH / real(nSn+nPn,wr)
151     ! -------                                                 --------     .
152     call correlationaffpond(a_Pg,R2_Pg,nPg,pt_Pg)
153     call correlationpond(a_Pn,b_Pn,R2_Pn,nPn,pt_Pn)
154     call correlationaffpond(a_Sg,R2_Sg,nSg,pt_Sg)
155     call correlationpond(a_Sn,b_Sn,R2_Sn,nSn,pt_Sn)
156     ! -------                                                 --------     .
157     max_0 = max
158     ! ------------------------------------------------------------------     .
159     write(*,*)"ecriture du script GMT Hodochrone"
160     write(600,*)"thegray1=200/200/200"
161     write(600,*)"thegray2=175/175/175"
162     write(600,*)"thegray3=150/150/150"
163     write(600,*)"BEFORE=$SECONDS"
164     call system_clock(Noldtime)
165     write(600,*)"#########################################"
166     write(600,*)"################ Hodochrone ##############"
167     write(600,*)"echo 'execution du script GMT Hodochrone'"
168     write(numberfile(1:5),'(i5)')j
169     write(600,*)"file=OUTPUT/GMT/hodochrone"//"-"//trim(adjustl(numberfile))//".ps"
170     write(600,*)"geoproj=-JX13i/8i"                                 ! système de projection
171     if (xmaxcercle/2.0_wr.lt.datatps(nbtps)%dhypo) then
172       val=xmaxcercle/2.0_wr
173     else
174       val=datatps(nbtps)%dhypo
175     endif
176     write(600,'(a12,E13.7,a3,E13.7)')"geozone=-R0/",val*1.2_wr,"/0/",max_0*1.1_wr
177     if (max_0.gt.60.0_wr) then
178       write(600,*)"psbasemap $geozone $geoproj -Ba50f25:""distance hypocentrale (km)"":",&
179       "/a15f5g60:""temps d'arriv\351es des ondes (s)"":WenS -Xc -Yc -K >  $file"
180     else
181       write(600,*)"psbasemap $geozone $geoproj -Ba20f5:""distance hypocentrale (km)"":",&
182       "/a5f1g60:""temps d'arriv\351es des ondes (s)"":WenS -Xc -Yc -K >  $file"
183     endif
184     ! -------                                                 --------     . plot distance critique
185     write(600,*)"echo -e "" ",discritiqueH,1,"\n",discritiqueH,max_0*1.09_wr," \"
186     write(600,*)" "" | psxy $geozone $geoproj -W0.01i,gray -O -K >>  $file"
187
188     ! ------------------------------------------------------------------     .
189     ! ------------------------------------------------------------------     .
190     inquire (_FILE_DIR_="DATA/sac-"//trim(adjustl(numberfile)),exist=existe1) ! option différente selon compilo !
191     if ((existe1).and.(tracessac)) then
192       ! ------------------------------------------------------------------     .
193       ! double si Pg et Pn (ou Sg et Sn), mais pas tres grave ....
194       ! ------------------------------------------------------------------     . plot traces si existes
195       do i=1,nbtps
196         ! lecture un peu archaïque, mais permet un peu de souplesse dans le non des station et des fichiers sac
197         ! -------                                                 --------     . COMPOSANTE Z
198         if (datatps(i)%sta%staname(4:4)=='0') then                              ! nom station en trois caractere + "0"
```

```
          write(600,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
            " 2>/dev/null | uniq | while read nom "
        else
          write(600,'(5a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*Z.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*Z.*sa* ", &
            " 2>/dev/null | uniq | while read nom "
        endif
        write(600,*)" do "
        write(600,*)" nombis=${nom/'DATA'/'OUTPUT'}"
        write(600,*)" nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
        write(600,*)" echo '",param_best%Tzero,datatps(i)%dhypo,val/15.0_wr,"' > toto2.txt"
        write(600,*)" ./BIN/sac_bin2txt.exe $nom $nomter.txt < toto2.txt "
        write(600,*)" psxy $geozone $geoproj -W2,$thegray1 -O -K $nomter.txt -: >>  $file"
        write(600,*)" done "
        write(600,*)"rm -rf toto2.txt"
        ! -------                                                 --------     . COMPOSANTE E
        if (datatps(i)%sta%staname(4:4)=='0') then                           ! nom station en trois caractere + "0"
          write(600,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*E.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*E.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*E.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*E.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
            " 2>/dev/null | uniq | while read nom "
        else
          write(600,'(5a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*E.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*E.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*Z.*sa* ", &
            " 2>/dev/null | uniq | while read nom "
        endif
        write(600,*)" do "
        write(600,*)" nombis=${nom/'DATA'/'OUTPUT'}"
        write(600,*)" nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
        write(600,*)" echo '",param_best%Tzero,datatps(i)%dhypo,val/15.0_wr,"' > toto2.txt"
        write(600,*)" ./BIN/sac_bin2txt.exe $nom $nomter.txt < toto2.txt "
        write(600,*)" psxy $geozone $geoproj -W2,$thegray2 -O -K $nomter.txt -: >>  $file"
        write(600,*)" done "
        write(600,*)"rm -rf toto2.txt"
        ! -------                                                 --------     . COMPOSANTE N
        if (datatps(i)%sta%staname(4:4)=='0') then                           ! nom station en trois caractere + "0"
          write(600,'(9a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*N.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*N.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*N.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*N.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:3)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname(1:4)//".*Z.*sa* ", &
            " 2>/dev/null | uniq | while read nom "
        else
          write(600,'(5a)')"ls DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*N.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//"*N.*SA* ", &
            "DATA/sac-"//trim(adjustl(numberfile))//"/"//datatps(i)%sta%staname//"*Z.*sa* ", &
```

```
264             "DATA/sac-"//trim(adjustl(numberfile))//"/*"//datatps(i)%sta%staname//".*Z.*sa* ", &
265             " 2>/dev/null | uniq | while read nom "
266         endif
267         write(600,*)" do "
268         write(600,*)" nombis=${nom/'DATA'/'OUTPUT'}"
269         write(600,*)" nomter=${nombis/sac-"//trim(adjustl(numberfile))//"/GMT}"
270         write(600,*)" echo '",param_best%Tzero,datatps(i)%dhypo,val/15.0_wr,"' > toto2.txt"
271         write(600,*)" ./BIN/sac_bin2txt.exe $nom $nomter.txt < toto2.txt "
272         write(600,*)" psxy $geozone $geoproj -W2,$thegray3 -O -K $nomter.txt -: >>  $file"
273         write(600,*)" done "
274         write(600,*)"rm -rf toto2.txt"
275       enddo
276       ! ------------------------------------------------------------------     .
277       ! ------------------------------------------------------------------     .
278     endif
279     ! ------------------------------------------------------------------     .
280     ! ------------------------------------------------------------------     .
281     ! ------------------------------------------------------------------     .
282
283
284     ! prendre en considération un modo incliné !  ... à revoir !
285
286
287     ! -------                                                 ---------     . traits réels (lois appliquées au jeu de paramètres)
288     ! -------                                                 ---------     . ONDES REFLECHIES puis REFRACTEES
289     max = val*1.09_wr
290     min = 0.0_wr
291     ! -------                                                 ---------     . pPn par modèle
292     !call pPn_sSn(param_best,max,Tpsmax,Tppmax)
293     !call pPn_sSn(param_best,min,Tpsmin,Tppmin)
294     !write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
295     !write(600,*)" "" | psxy $geozone $geoproj -W0.01i,LIGHTGREEN -O -K >>  $file"
296     ! -------                                                 ---------     . sSn par modèle
297     !write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
298     !write(600,*)" "" | psxy $geozone $geoproj -W0.01i,LIGHTGREEN -O -K >>  $file"
299     ! -------                                                 ---------     . ONDES REFLECHIES
300     !do i=int(param_best%Zhypo+2.0_wr),int(val*1.09_wr+0.5_wr),2
301       !min = real(i-1,wr)
302       !max = real(i+1,wr)
303       ! -------                                               ---------     . PmP par modèle
304       !call reflechie(param_best,min,Tpsmin,Tppmin)
305       !call reflechie(param_best,max,Tpsmax,Tppmax)
306       !write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
307       !write(600,*)" "" | psxy $geozone $geoproj -W0.01i,LIGHTORANGE -O -K >>  $file"
308       ! -------                                               ---------     . SmS par modèle
309       !write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
310       !write(600,*)" "" | psxy $geozone $geoproj -W0.01i,LIGHTORANGE -O -K >>  $file"
311     !enddo
312     ! -------                                                 ---------     . ONDES REFLECHIES 2
313     !do i=int(param_best%Zhypo+2.0_wr),int(val*1.09_wr+0.5_wr),2
314       !min = real(i-1,wr)
315       !max = real(i+1,wr)
316       ! -------                                               ---------     . 2PmP par modèle
317       !call reflechie2(param_best,min,Tpsmin,Tppmin)
318       !call reflechie2(param_best,max,Tpsmax,Tppmax)
319       !write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
320       !write(600,*)" "" | psxy $geozone $geoproj -W0.01i,LIGHTORANGE -O -K >>  $file"
321       ! -------                                               ---------     . 2SmS par modèle
322       !write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
323       !write(600,*)" "" | psxy $geozone $geoproj -W0.01i,LIGHTORANGE -O -K >>  $file"
324     !enddo
325     ! -------                                                 ---------     . ONDES DIRECTES
326     min = 0.0_wr
327     max = val*1.09_wr
328     ! -------                                                 ---------     . Pg par modèle
```

```
    call directe(param_best,min,Tpsmin,Tppmin)
    call directe(param_best,max,Tpsmax,Tppmax)
    write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$pp -O -K >>  $file"
    ! -------                                                  --------    . Sg par modèle
    write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$ss -O -K >>  $file"
    ! -------                                                  --------    . ONDES REFRACTEES
    min = discritiqueH
    ! -------                                                  --------    . Pn par modèle
    if(FLAG_non_tabulaire) then                                             ! moho incliné
      a_sta%lon=param_best%lon
      a_sta%lat=param_best%lat
      a_sta%alti=0.0_wr
      call refracte_mohovar(acentroid,param_best,a_sta,min,Tpsmin,Tppmin)
      call refracte_mohovar(acentroid,param_best,a_sta,max,Tpsmax,Tppmax)
    else
      call refracte(param_best,min,Tpsmin,Tppmin)
      call refracte(param_best,max,Tpsmax,Tppmax)
    endif
    write(600,*)"echo -e "" ",min,Tppmin,"\n",max,Tppmax," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$pp,- -O -K >>  $file"
    ! -------                                                  --------    . Sn par modèle
    write(600,*)"echo -e "" ",min,Tpsmin,"\n",max,Tpsmax," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$ss,- -O -K >>  $file"
    ! -------                                                  --------    . Pn par regression
    write(600,*)"echo -e "" ",min,a_Pn*min+b_Pn,"\n",max,a_Pn*max+b_Pn," \"
    write(600,*)" ""| psxy $geozone $geoproj -W0.001i -O -K >> $file"
    ! -------                                                  --------    . Sn par regression
    write(600,*)"echo -e "" ",min,a_Sn*min+b_Sn,"\n",max,a_Sn*max+b_Sn," \"
    write(600,*)" ""| psxy $geozone $geoproj -W0.001i -O -K >> $file"
    ! -------                                                  --------    . Pg par regression
    min = 0.0_wr
    write(600,*)"echo -e "" ",min,a_Pg*min,"\n",max,a_Pg*max," \"
    write(600,*)" ""| psxy $geozone $geoproj -W0.001i -O -K >>  $file"
    ! -------                                                  --------    . Sg par regression
    write(600,*)"echo -e "" ",min,a_Sg*min,"\n",max,a_Sg*max," \"
    write(600,*)" ""| psxy $geozone $geoproj -W0.001i -O -K >>  $file"
    ! -------                                                  --------    .
    do i=1,nPg                                                              ! points réels
      write(600,*)"echo",pt_Pg(i,1),pt_Pg(i,2),pt_Pg(i,3),pt_Pg(i,4)," \"
      write(600,*)" | psxy $geozone $geoproj -O -K -St0.1i -Wthinnest -Ey -COUTPUT/GMT/colorpal3.cpt >>  $file"
    enddo
    ! -------                                                  --------    .
    do i=1,nPn                                                              ! points réels
      write(600,*)"echo",pt_Pn(i,1),pt_Pn(i,2),pt_Pn(i,3),pt_Pn(i,4)," \"
      write(600,*)" | psxy $geozone $geoproj -O -K -Si0.1i -Wthinnest -Ey -COUTPUT/GMT/colorpal3.cpt >>  $file"
    enddo
    ! -------                                                  --------    .
    do i=1,nSg                                                              ! points réels
      write(600,*)"echo",pt_Sg(i,1),pt_Sg(i,2),pt_Sg(i,3),pt_Sg(i,4)," \"
      write(600,*)" | psxy $geozone $geoproj -O -K -Ss0.1i -Wthinnest -Ey -COUTPUT/GMT/colorpal3.cpt >>  $file"
    enddo
    ! -------                                                  --------    .
    do i=1,nSn                                                              ! points réels
      write(600,*)"echo",pt_Sn(i,1),pt_Sn(i,2),pt_Sn(i,3),pt_Sn(i,4)," \"
      write(600,*)" | psxy $geozone $geoproj -O -K -Sd0.1i -Wthinnest -Ey -COUTPUT/GMT/colorpal3.cpt >>  $file"
    enddo
    ! -------                                                  --------    . nom sta
    l=1
    do i=1,nbtps+1
      nomstadoublets(i)='xxxx'
    enddo
    do i=1,nbtps
      ! -------                                                --------    . nom déja affiché ?
```

```
      deja =.true.
      do k=1,l
        if (nomstadoublets(k).eq.datatps(i)%sta%staname) deja=.false.
      enddo
      ! _______                                          ________    . affiche
      if (deja) then
        if (mod(i,3)==0) then
          write(600,*)"echo '",datatps(i)%dhypo,max_0*0.9_wr, &
          " 7 90 1 5 "//datatps(i)%sta%staname//"' | pstext $geoproj $geozone -O -K -C2 >> $file"
          nomstadoublets(l)=datatps(i)%sta%staname
          l=l+1
        elseif (mod(i,2)==0) then
          write(600,*)"echo '",datatps(i)%dhypo,max_0*0.8_wr, &
          " 7 90 1 5 "//datatps(i)%sta%staname//"' | pstext $geoproj $geozone -O -K -C2 >> $file"
          nomstadoublets(l)=datatps(i)%sta%staname
          l=l+1
        else
          write(600,*)"echo '",datatps(i)%dhypo,max_0*0.7_wr, &
          " 7 90 1 5 "//datatps(i)%sta%staname//"' | pstext $geoproj $geozone -O -K -C2 >> $file"
          nomstadoublets(l)=datatps(i)%sta%staname
          l=l+1
        endif
      endif
    enddo
    ! _______                                          ________    .
    write(600,*)"#########################################"                ! Légende
    ! _______                                          ________    . figurés Pg
    X = val * 1.2_wr * 0.085_wr
    Y = max_0 * 1.1_wr  * 0.745_wr
    write(600,*)"echo",X,Y," \"
    write(600,*)" | psxy $geozone $geoproj -O -K -St0.1i -Wthinnest -Gyellow >>  $file"
    X = val * 1.2_wr  * 0.1_wr
    Y = max_0 * 1.1_wr  * 0.75_wr
    write(600,*)"echo """,X,Y," \"
    write(600,*)"15 0 4 LM ondes compressives directes"" | pstext $geozone $geoproj -O -K >> $file"
    ! _______                                          ________    . figurés Pn
    X = val * 1.2_wr * 0.085_wr
    Y = max_0 * 1.1_wr  * 0.695_wr
    write(600,*)"echo",X,Y," \"
    write(600,*)" | psxy $geozone $geoproj -O -K -Ss0.1i -Wthinnest -Gyellow >>  $file"
    X = val * 1.2_wr  * 0.1_wr
    Y = max_0 * 1.1_wr  * 0.70_wr
    write(600,*)"echo """,X,Y," \"
    write(600,*)"15 0 4 LM ondes cisaillantes directes"" | pstext $geozone $geoproj -O -K >> $file"
    ! _______                                          ________    . figurés Sg
    X = val * 1.2_wr * 0.085_wr
    Y = max_0 * 1.1_wr  * 0.645_wr
    write(600,*)"echo",X,Y," \"
    write(600,*)" | psxy $geozone $geoproj -O -K -Si0.1i -Wthinnest -Gyellow >>  $file"
    X = val * 1.2_wr  * 0.1_wr
    Y = max_0 * 1.1_wr  * 0.65_wr
    write(600,*)"echo """,X,Y," \"
    write(600,*)"15 0 4 LM ondes compressives r\351fract\351es"" | pstext $geozone $geoproj -O -K >> $file"
    ! _______                                          ________    . figurés Sn
    X = val * 1.2_wr * 0.085_wr
    Y = max_0 * 1.1_wr  * 0.595_wr
    write(600,*)"echo",X,Y," \"
    write(600,*)" | psxy $geozone $geoproj -O -K -Sd0.1i -Wthinnest -Gyellow >>  $file"
    X = val * 1.2_wr  * 0.1_wr
    Y = max_0 * 1.1_wr  * 0.60_wr
    write(600,*)"echo """,X,Y," \"
    write(600,*)"15 0 4 LM ondes cisaillantes r\351fract\351es"" | pstext $geozone $geoproj -O -K >> $file"
    ! _______                                          ________    . doites des modèles
    X = val * 1.2_wr  * 0.1_wr
    Y = max_0 * 1.1_wr  * 0.80_wr
```

```
    write(600,*)"echo """,X,Y," \"
    write(600,*)"15 0 4 LM selon le mod\350le de terre"" | pstext $geozone $geoproj -O -K >> $file"
    X = val * 1.2_wr * 0.085_wr
    write(600,*)"echo -e "" ",X-0.075_wr*X,Y,"\n",X-0.01_wr*X,Y," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$ss -O -K >>  $file"
    write(600,*)"echo -e "" ",X+0.01_wr*X,Y,"\n",X+0.075_wr*X,Y," \"
    write(600,*)" "" | psxy $geozone $geoproj -W0.05i,$pp -O -K >>  $file"
    ! -------                                                    --------    . doites de régressions
    X = val * 1.2_wr  * 0.1_wr
    Y = max_0 * 1.1_wr  * 0.85_wr
    write(600,*)"echo """,X,Y," \"
    write(600,*)"15 0 4 LM r\351gressions lin\351aires"" | pstext $geozone $geoproj -O -K >> $file"
    X = val * 1.2_wr * 0.085_wr
    write(600,*)"echo -e "" ",X-0.05_wr*X,Y,"\n",X+0.05_wr*X,Y," \"
    write(600,*)" ""| psxy $geozone $geoproj -W0.001i -O -K >> $file"
    ! -------                                                    --------    .
    write(600,*)"psscale -D1/-1/0.50E+01/0.25ch -B.25:""pond\351ration"": -S -I -COUTPUT/GMT/colorpal3.cpt -O -K >> $file"
    ! -------                                                    --------    .
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
    write(600,*)"################################################"
    write(600,*)"ps2raster OUTPUT/GMT/hodochrone"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/hodochrone"//"-"//trim(adjustl(numberfile))//".pdf ", &
    "OUTPUT/figures/hodochrone"//"-"//trim(adjustl(numberfile))//".pdf "
    write(600,*)"################################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
      int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------    .
    deallocate (pt_Pg,pt_Pn,pt_Sg,pt_Sn)
    ! ------------------------------------------------------------    .
  end subroutine GMT_Hodochrone

END MODULE figure_GMThodo



! ********************************************************************    .
! ********************************************************************    .
```

2.6 SRC/MOD/MOD_GMT/mkmap.f90

```
! permet la création des scripts GMT pour une carte de la région avec les stations
! mars 2014
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .

MODULE figure_GMTmap

    use modparam

    implicit none

    private

    public  :: GMT_map


CONTAINS

! ------------------------------------------------------------------    .
```

```
22
23  subroutine GMT_map(l,xmin,xmax,dp,nbtps,datatps)
24    ! ________                                                    ________     .mh
25    ! production du script GMT produisant une carte des stations
26    ! ________                                                    ________     .
27    use typetemps
28    ! ________                                                    ________     .
29    implicit none
30    integer(KIND=wi), intent (in) :: l
31    real(KIND=wr), intent(in) :: xmin,xmax
32    type(densityplot), intent(in) :: dp
33    integer(KIND=wi), intent(in) :: nbtps
34    type(dataone), intent(in) :: datatps(nbtps)
35    ! ________                                                    ________     .
36    type(stations) :: datasta
37    real(KIND=wr) :: v1, v2
38    real(KIND=wr) :: X, Y, tl
39    real(KIND=wr) :: lon1,lon2,lat1,lat2, azim(36), max
40    integer(KIND=wi) :: i,j,k,ok,Noldtime, Nnewtime, ratetime
41    character (LEN=5) :: numberfile
42    ! ____________________________________________________________     .
43    write(*,*)"ecriture du script GMT_map "
44    write(600,*)"BEFORE=$SECONDS"
45    call system_clock(Noldtime)
46    write(600,*)"#######################################"
47    write(600,*)"#######################################"
48    write(600,*)
49    write(600,*)"echo 'execution du script GMT map'"
50    write(600,*)"###########################################"
51    write(600,*)"############     carte GMT       ##########"
52    write(600,*)"###########################################"
53    write(numberfile(1:5),'(i5)')l
54    write(600,*)"file=OUTPUT/GMT/map-"//trim(adjustl(numberfile))//".ps"
55    write(600,*)"gmtset BASEMAP_TYPE plain"
56    write(600,*)"labasemap1=-Bpa2g1.f.5/a1g1.f.25WeSn"
57    write(600,*)"grdfile=SRC/FILES/bath1.bin"
58    ! ________                                                    ________     .
59    v1 = 2.0_wr * pi * rT / 360.0_wr                                         ! km / degree en lon
60    v2 = 2.0_wr * pi * rT * sin((90.0_wr-dp%lat(l)%vec10000(1,1))/180.0_wr*pi) /360.0_wr ! km / degree en lat
61    ! ________                                                    ________     .
62    lon1 = dp%lon(l)%vec10000(1,1) - (xmax / v2 * 1.125_wr) / 2.0_wr
63    lon2 = dp%lon(l)%vec10000(1,1) + (xmax / v2 * 1.125_wr) / 2.0_wr
64    lat1 = dp%lat(l)%vec10000(1,1) - (xmax / v1 * 1.125_wr) / 2.0_wr
65    lat2 = dp%lat(l)%vec10000(1,1) + (xmax / v1 * 1.125_wr) / 2.0_wr
66    ! ________                                                    ________     .
67    write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",lon1,"/",lon2,"/",lat1,"/",lat2
68    write(600,'(a11,E13.7,a1,E13.7,a3)')"geoproj=-JC",dp%lon(l)%vec10000(1,1),"/",dp%lat(l)%vec10000(1,1),"/7i"
69    ! ________                                                    ________     .
70    write(600,*)"bluef=""0/0/100"" "
71    write(600,*)"grdgradient $grdfile -A0/270 -GOUTPUT/GMT/grd.gradients -Ne0.6 "
72    write(600,*)"grdimage $geozone $geoproj $grdfile -CSRC/FILES/mytopo.cpt -Sc/1 $labasemap1 \"
73    write(600,*)"OUTPUT/GMT/grd.gradients -Sn -Xc -X5.5i -Yc -K > $file"
74    write(600,*)"pscoast $geozone $geoproj -Df+ -Ia/$bluef -W1 -O -K >>  $file"
75    write(600,*)"grdcontour $grdfile $geozone $geoproj -Ba0 -C1000 -L-10000/-10 -W2 -O -K >> $file"
76    write(600,*)"grdcontour $grdfile $geozone $geoproj -Ba0 -C1000 -L10/1000 -W2 -O -K >> $file"
77    ! ________                                                    ________     .
78    write(600,*)"######### cercles de pondération ########"
79    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"0",xmin,xmin, &
80      " | psxy $geozone $geoproj -SE -W13,white -O -K >> $file"
81    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"0",xmin,xmin, &
82      " | psxy $geozone $geoproj -SE -W10 -O -K >> $file"
83    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"0",xmax,xmax, &
84      " | psxy $geozone $geoproj -SE -W13,white -O -K >> $file"
85    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"0",xmax,xmax, &
86      " | psxy $geozone $geoproj -SE -W10  -O -K >> $file"
```

```
87    ! -------                                                  --------    .
88    write(600,*)"######### affiche les stations ##########"
89    ok = 0                                                               ! toutes les stations
90    open(800, FILE = 'DATA/sta.d',status='old',iostat = ok)
91    if (ok .ne. 0) then
92      write(*,*)'problème dans GMT_map : le fichier data/sta.d n''existe pas '
93      stop
94    endif
95    do while(ok .eq. 0)
96      read(800,*,iostat = ok)datasta
97      if ((datasta%lon.gt.lon1).and.(datasta%lon.lt.lon2).and.(datasta%lat.gt.lat1).and.(datasta%lat.lt.lat2)) then
98        write(600,*)" echo ",datasta%lon,datasta%lat," | psxy $geoproj $geozone -St0.05i -Ggrey -Lk -Wthinnest -O -K >> $file"
99      endif
100   enddo
101   close(800)
102   ! -------  relie les stations utilisées                      --------    .
103   do i=1,nbtps
104     k=0                                                                ! nb de pointé pur cette station
105     do j=1,nbtps
106       if (datatps(i)%sta%staname.eq.datatps(j)%sta%staname)   then
107         k =  k + 1
108         if(datatps(i)%tpsR%secP.eq.datatps(j)%tpsR%secP) then
109           if (datatps(j)%andS.eq."S") k =  k + 1
110         endif
111         if (datatps(i)%tpsR%secP.ne.datatps(j)%tpsR%secP) then
112           if (datatps(j)%andS.eq."S") k =  k + 1
113         endif
114       endif
115     enddo
116     if ((k.lt.0).or.(k.gt.4)) write(*,*)'problème dans GMT_map : nombre de données par station incorrecte !'
117     if (k .eq. 1) then
118       write(600,*)" echo -e "" ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"\n", &
119       datatps(i)%sta%lon,datatps(i)%sta%lat," "" | psxy $geozone $geoproj -L -K -O -W2,green >> $file"
120     endif
121     if (k .eq. 2) then
122       write(600,*)" echo -e "" ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"\n", &
123       datatps(i)%sta%lon,datatps(i)%sta%lat," "" | psxy $geozone $geoproj -L -K -O -W4,yellow >> $file"
124     endif
125     if (k .eq. 3) then
126       write(600,*)" echo -e "" ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"\n", &
127       datatps(i)%sta%lon,datatps(i)%sta%lat," "" | psxy $geozone $geoproj -L -K -O -W6,orange >> $file"
128     endif
129     if (k .eq. 4) then
130       write(600,*)" echo -e "" ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1),"\n", &
131       datatps(i)%sta%lon,datatps(i)%sta%lat," "" | psxy $geozone $geoproj -L -K -O -W8,red >> $file"
132     endif
133   enddo
134   ! -------                                                  --------    .
135   do i=1,nbtps                                                         ! affiche les stations utilisées
136     write(600,*)" echo ",datatps(i)%sta%lon,datatps(i)%sta%lat, &
137     " | psxy $geoproj $geozone -St0.1i -Gblue -Lk -Wthinnest -O -K >> $file"
138   enddo
139   do i=1,nbtps                                                         ! affiche les noms des stations utilisées
140     write(600,*)" echo ",datatps(i)%sta%lon+0.4_wr,datatps(i)%sta%lat,"8 0 5 6 "//datatps(i)%sta%staname, &
141     " | pstext $geozone $geoproj -O -K >> $file"
142   enddo
143   ! -------                                                  --------    .
144   write(600,*)"###### Limites du massif Armoricain #####"
145   write(600,*)"psxy $geozone $geoproj -A -W4,gray -O SRC/FILES/limitesMA -K -M >> $file"
146   ! -------                                                  --------    .
147   write(600,*)"################ Epicentre ################"
148   write(600,*)"echo",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1)," \"
149   write(600,*)" | psxy $geozone $geoproj -L -K -O -Wthinnest -Ggreen -Sa0.20i >> $file"
150   ! -------                                                  --------    .
151   write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
```

```
    ! ------- légende :                                                ---------     .
    write(600,*)"################ légende ################"
    write(600,*)"geozone=-R0/1/0/1.25"
    write(600,*)"geoproj=-JX2i/1i"
    X=.1_wr
    Y=.25_wr
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -X-2.5i -O -K >>  $file"
    write(600,*)"echo ",X+.5_wr,Y," 15 0 5 6 ""1 donn\351es""  | pstext $geozone $geoproj -O -K >> $file"
    write(600,*)" echo -e "" ",X,Y,"\n",X+.25_wr,Y," "" | psxy $geozone $geoproj -L -K -O -W2,green >> $file"
    Y=.5_wr
    write(600,*)"echo ",X+.5_wr,Y," 15 0 5 6 ""2 donn\351es"" | pstext $geozone $geoproj -O -K >> $file"
    write(600,*)" echo -e "" ",X,Y,"\n",X+.25_wr,Y," "" | psxy $geozone $geoproj -L -K -O -W4,yellow >> $file"
    Y=.75_wr
    write(600,*)"echo ",X+.5_wr,Y," 15 0 5 6 ""3 donn\351es"" | pstext $geozone $geoproj -O -K >> $file"
    write(600,*)" echo -e "" ",X,Y,"\n",X+.25_wr,Y," "" | psxy $geozone $geoproj -L -K -O -W6,orange >> $file"
    Y=1._wr
    write(600,*)"echo ",X+.5_wr,Y," 15 0 5 6 ""4 donn\351es"" | pstext $geozone $geoproj -O -K >> $file"
    write(600,*)" echo -e "" ",X,Y,"\n",X+.25_wr,Y," "" \ | psxy $geozone $geoproj -L -K -O -W8,red >> $file"
    ! -------                                                  ---------     .
    write(600,*)"################## rose ################"
    write(600,*)"geozone=-R0/1/0/1.25"
    write(600,*)"geoproj=-JX2i/2i"
    ! -------                                                  ---------     . initialisation compteur azimutal (10 degrés)
    do j=1,36
      azim(j) = 0.0_wr
    enddo
    ! -------                                                  ---------     . compteur azimutal (10 degrés) pour toutes données
    open(801, FILE = 'OUTPUT/GMT/baz'//"-"//trim(adjustl(numberfile))//'.d',status='replace',iostat = ok)
    do i=1,nbtps
      write(801,*)datatps(i)%baz, datatps(i)%wp
      do j=1,36
        if((datatps(i)%baz.ge.((real(j,wr)*10.0_wr)-10.00_wr)).and.(datatps(i)%baz.lt.(real(j,wr)*10.0_wr))) then
          azim(j) = azim(j) + datatps(i)%wp
        endif
      enddo
      if (datatps(i)%andS .eq. "S") then
        write(801,*)datatps(i)%baz, datatps(i)%ws
        do j=1,36
          if((datatps(i)%baz.ge.((real(j,wr)*10.0_wr)-10.00_wr)).and.(datatps(i)%baz.lt.(real(j,wr)*10.0_wr))) then
            azim(j) = azim(j) + datatps(i)%ws
          endif
        enddo
      endif
    enddo
    close(801)
    ! -------                                                  ---------     . mode du compteur
    max=-1.0_wr
    do j=1,36
      if(max.lt.azim(j)) max = azim(j)
    enddo
    ! -------                                                  ---------     . pour données directes
    open(802, FILE = 'OUTPUT/GMT/baz1_'//"-"//trim(adjustl(numberfile))//'.d',status='replace',iostat = ok)
    do i=1,nbtps
      if(datatps(i)%typeonde .eq. "G") then
        write(802,*)datatps(i)%baz, datatps(i)%wp/max
        if (datatps(i)%andS .eq. "S") write(802,*)datatps(i)%baz, datatps(i)%ws/max
      endif
    enddo
    close(802)
    ! -------                                                  ---------     . pour données réfractées
    open(803, FILE = 'OUTPUT/GMT/baz2_'//"-"//trim(adjustl(numberfile))//'.d',status='replace',iostat = ok)
    do i=1,nbtps
      if(datatps(i)%typeonde .eq. "N") then
        write(803,*)datatps(i)%baz, datatps(i)%wp/max
        if (datatps(i)%andS .eq. "S") write(803,*)datatps(i)%baz, datatps(i)%ws/max
```

```
      endif
    enddo
    close(803)
    ! -------                                                    --------     . légende 1
    write(600,*)"echo ""0 1 15 0 4 LM couverture azimutale "" |", &
      "pstext $geozone $geoproj -Y5i -O -K >> $file"
    write(600,*)"echo ""0 0.85 15 0 4 LM pond\351r\351e : "" |", &
      "pstext $geozone $geoproj -O -K >> $file"
    ! -------                                                    --------     . légende 2
    write(600,*)"echo ""0 0.6 15 0 4 LM - ondes @;blue;directes"" |", &
      "pstext $geozone $geoproj -O -K >> $file"
    write(600,*)"echo ""0 0.45 15 0 4 LM - ondes @;olivedrab4;r\351fract\351es"" |", &
      "pstext $geozone $geoproj -O -K >> $file"
    ! -------                                                    --------     . rose données directes
    write(600,*)"psrose ./OUTPUT/GMT/baz"//"-"//trim(adjustl(numberfile))//".d -: -A10 -S.75in", &
      " -Ggray -R0/1/0/360 -F -L'@'/'@'/'@'/'Nord' -Y-1.5i -B.25g0.25/30g30 -O -K >> $file"
    write(600,*)"psrose ./OUTPUT/GMT/baz1-"//"-"//trim(adjustl(numberfile))//".d -: -A10 -S.75i", &
      " -Gblue -R0/1/0/360 -W2 -F -L'@'/'@'/'@'/'Nord' -O -K >> $file "
    ! -------                                                    --------     . rose données réfractées
    write(600,*)"psrose ./OUTPUT/GMT/baz"//"-"//trim(adjustl(numberfile))//".d -: -A10 -S.75in", &
      " -Ggray -R0/1/0/360 -F -L'@'/'@'/'@'/'Nord' -Y-2i -B.25g0.25/30g30 -O -K >> $file "
    write(600,*)"psrose ./OUTPUT/GMT/baz2-"//"-"//trim(adjustl(numberfile))//".d -: -A10 -S.75i", &
      " -Golivedrab4 -R0/1/0/360 -W2 -F -L'@'/'@'/'@'/'Nord' -O >> $file "
    ! -------                                                    --------     . fin
    write(600,*)"ps2raster OUTPUT/GMT/map-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/map-"//trim(adjustl(numberfile))//".pdf ", &
      "OUTPUT/figures/map-"//trim(adjustl(numberfile))//".pdf"
    ! -------                                                    --------     .
    write(600,*)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
      int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------------     .
  end subroutine GMT_map

END MODULE figure_GMTmap



! ********************************************************************     .
! ********************************************************************     .
```

2.7 SRC/MOD/MOD_GMT/mkparamiter.f90

```
! permet la création des scripts GMT pour les figures sur chaque parametre vs itération et sur les fonctions d'autocorrélation
! mars-octobre 2014
! ********************************************************************     .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
! ********************************************************************     .
! ------------------------------------------------------------------     .

MODULE figure_GMTpar

    use modparam

    implicit none

    private

    public  :: GMT_param
```

```
    public  :: RVB


CONTAINS

! ------------------------------------------------------------------      .

subroutine GMT_param(dp,nbChaineMV,nmod)
    ! -------                                                 ---------     .mh
    ! figures des chaines après échantillonage
    ! ------------------------------------------------------------------      .
    use typetemps
    ! -------                                                 ---------     .
    implicit none
    ! -------                                                 ---------     .
    type(densityplot), intent (in) :: dp                                    ! modèles retenus par McMC
    integer(KIND=wi), intent (in) :: nbChaineMV
    integer(KIND=wi), intent (in) :: nmod(nbChaineMV)
    ! -------                                                 ---------     .
    integer(KIND=wi) :: i,j,k,l,m,maxiter,val,Noldtime, Nnewtime, ratetime
    real(KIND=wr) :: tl
    character(LEN=5) :: char
    character(LEN=11) :: color, char1, char2
    character (LEN=5) :: numberfile
    ! ------------------------------------------------------------------      . ecriture des paramètre de terre, par chaîne, avec rééchantillonnage
    k=0
    maxiter=0
    do i=1,nbChaineMV
      write(char,'(i5)')10000+i
      open(unit=1200+8*i,file="OUTPUT/GMT/theVC"//char//".bin",STATUS="replace",access='direct',RECL=16)
      open(unit=1201+8*i,file="OUTPUT/GMT/theVM"//char//".bin",STATUS="replace",access='direct',RECL=16)
      open(unit=1202+8*i,file="OUTPUT/GMT/theVpVs"//char//".bin",STATUS="replace",access='direct',RECL=16)
      open(unit=1203+8*i,file="OUTPUT/GMT/theZmoho"//char//".bin",STATUS="replace",access='direct',RECL=16)
      l=0
      do j=1,nmod(i)
        k=k+1
        val = max(100,(nmod(i)/2000))
        if (mod(k,val)==0) then
          l=l+1
          write(1200+8*i,rec=l)real(dp%VC%vec(k),8),real(j,8)
          write(1201+8*i,rec=l)real(dp%VM%vec(k),8),real(j,8)
          write(1202+8*i,rec=l)real(dp%VpVs%vec(k),8),real(j,8)
          write(1203+8*i,rec=l)real(dp%Zmoho%vec(k),8),real(j,8)
        endif
      if(maxiter.lt.j)maxiter=j
    enddo
    close(1200+8*i)
    close(1201+8*i)
    close(1202+8*i)
    close(1203+8*i)
    enddo
    ! ------------------------------------------------------------------      . ecriture des paramètre hypocentraux, par chaîne, avec rééchantillonnage
    do m=1,nbseismes
      write(numberfile(1:5),'(i5)')m
      k=0
      maxiter=0
      do i=1,nbChaineMV
        write(char,'(i5)')10000+i
        open(unit=1204+8*i,file="OUTPUT/GMT/theLon"//char//"-"//trim(adjustl(numberfile))//".bin", &
        STATUS="replace",access='direct',RECL=16)
        open(unit=1205+8*i,file="OUTPUT/GMT/theLat"//char//"-"//trim(adjustl(numberfile))//".bin", &
        STATUS="replace",access='direct',RECL=16)
        open(unit=1206+8*i,file="OUTPUT/GMT/theZhypo"//char//"-"//trim(adjustl(numberfile))//".bin", &
        STATUS="replace",access='direct',RECL=16)
        open(unit=1207+8*i,file="OUTPUT/GMT/theTzero"//char//"-"//trim(adjustl(numberfile))//".bin", &
```

```
        STATUS="replace",access='direct',RECL=16)
        l=0
        do j=1,nmod(i)
           k=k+1
           val = max(100,(nmod(i)/2000))
           if (mod(k,val)==0) then
             l=l+1
             write(1204+8*i,rec=l)real(dp%Lon(m)%vec(k),8),real(j,8)
             write(1205+8*i,rec=l)real(dp%Lat(m)%vec(k),8),real(j,8)
             write(1206+8*i,rec=l)real(dp%Zhypo(m)%vec(k),8),real(j,8)
             write(1207+8*i,rec=l)real(dp%Tzero(m)%vec(k),8),real(j,8)
           endif
           if(maxiter.lt.j)maxiter=j
        enddo
      close(1204+8*i)
      close(1205+8*i)
      close(1206+8*i)
      close(1207+8*i)
      enddo
    enddo
    m=1
    ! ------------------------------------------------------------------      .
    ! plot les différentes réalisations pour chaque parametre
    ! ------------------------------------------------------------------      .
    write(*,*)"ecriture des script GMT_param "
    write(600,*)"BEFORE=$SECONDS"
    call system_clock(Noldtime)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)
    write(600,*)"echo 'execution du script GMT param vs iter'"
    write(600,*)"#########################################"
    write(600,*)"############  param vs iter     ##########"
    write(600,*)"#########################################"
    ! ------------------------------------------------------------------      .
    write(600,*)"geoproj=-JX13.5i/4.5i"
    write(600,*)"file=OUTPUT/GMT/VCVM_histo.ps"
    write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%VC%themin,"/",dp%VC%themax
    ! ------------------------------------------------------------------      .
    write(char1(1:11),'(i11)')(maxiter/1000)*250
    write(char2(1:11),'(i11)')val
    ! ------------------------------------------------------------------      .
    write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
      "-Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
      "51':/a",real(int(((dp%VC%themax-dp%VC%themin)/5.0_wr)*10.0_wr,wi),wr)/10._wr,":'"//dp%VC%char//"':nSeW", &
      " -K -Y6.3i -Xc >  $file"
    do i=1,nbChaineMV
      write(char,'(i5)')10000+i
      call RVB(i,nbChaineMV,color)
      write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theVC"//char//".bin -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
    enddo
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
    write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%VM%themin,"/",dp%VM%themax
    write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
      " -Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
      "51':/a",real(int(((dp%VM%themax-dp%VM%themin)/5.0_wr)*10.0_wr),wr)/10.0_wr,":'"//dp%VM%char//"':nSeW", &
      " -K -O -Y-5.5i >>  $file"
    do i=1,nbChaineMV
      write(char,'(i5)')10000+i
      call RVB(i,nbChaineMV,color)
      write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theVM"//char//".bin -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
    enddo
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
    write(600,*)"ps2raster OUTPUT/GMT/VCVM_histo.ps -Tf -A"
    write(600,'(a)')"mv OUTPUT/GMT/VCVM_histo.pdf OUTPUT/figures/VCVM_histo.pdf"
```

```
147     ! ------------------------------------------------------------------      .
148     ! ------------------------------------------------------------------      .
149     write(600,*)"file=OUTPUT/GMT/VpVsZmoho_histo.ps"
150     write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%VpVs%themin,"/",dp%VpVs%themax
151     write(600,'(2a)')"psbasemap $geozone $geoproj -Ba"//trim(adjustl(char1))//":'mod\350les ", &
152       "\050\351chantillonnage 1/"//trim(adjustl(char2))//"\051':/a0.05:'"//dp%VpVs%char//"':nSeW -K -Y6.3i -Xc >  $file"
153     do i=1,nbChaineMV
154       write(char,'(i5)')10000+i
155       call RVB(i,nbChaineMV,color)
156       write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theVpVs"//char//".bin -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
157     enddo
158     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
159     write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%Zmoho%themin,"/",dp%Zmoho%themax
160     write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
161       " -Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
162       "51':/a",real(int(((dp%Zmoho%themax-dp%Zmoho%themin)/5.0_wr)*5.0_wr),wr)/10._wr,":'"//dp%Zmoho%char//"':nSeW", &
163       " -K -O -Y-5.5i >>  $file"
164     do i=1,nbChaineMV
165       write(char,'(i5)')10000+i
166       call RVB(i,nbChaineMV,color)
167       write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theZmoho"//char//".bin -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
168     enddo
169     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
170     write(600,*)"ps2raster OUTPUT/GMT/VpVsZmoho_histo.ps -Tf -A"
171     write(600,'(a)')"mv OUTPUT/GMT/VpVsZmoho_histo.pdf OUTPUT/figures/VpVsZmoho_histo.pdf"
172     ! ------------------------------------------------------------------      .
173     ! ------------------------------------------------------------------      .
174     do m=1,nbseismes
175       write(numberfile(1:5),'(i5)')m
176       write(600,*)"file=OUTPUT/GMT/LatLon_histo"//"-"//trim(adjustl(numberfile))//".ps"
177       write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%Lon(m)%themin,"/",dp%Lon(m)%themax
178       write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
179         " -Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
180         "51':/a",real(int(((dp%Lon(m)%themax-dp%Lon(m)%themin)/5.0_wr)*200.0_wr),wr)/200.0_wr,":'"//dp%Lon(m)%char//"':nSeW", &
181         " -K -Y6.3i -Xc >  $file"
182       do i=1,nbChaineMV
183         write(char,'(i5)')10000+i
184         call RVB(i,nbChaineMV,color)
185         write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theLon"//char//"-"//trim(adjustl(numberfile))//".bin", &
186           " -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
187       enddo
188       write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
189       write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%Lat(m)%themin,"/",dp%Lat(m)%themax
190       write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
191         " -Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
192         "51':/a",real(int(((dp%Lat(m)%themax-dp%Lat(m)%themin)/5.0_wr)*200.0_wr),wr)/200.0_wr,":'"//dp%Lat(m)%char//"':nSeW", &
193         " -K -O -Y-5.5i >>  $file"
194       do i=1,nbChaineMV
195         write(char,'(i5)')10000+i
196         call RVB(i,nbChaineMV,color)
197         write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theLat"//char//"-"//trim(adjustl(numberfile))//".bin", &
198           " -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
199       enddo
200       write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
201       write(600,*)"ps2raster OUTPUT/GMT/LatLon_histo"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
202       write(600,'(2a)')"mv OUTPUT/GMT/LatLon_histo"//"-"//trim(adjustl(numberfile))//".pdf ", &
203         " OUTPUT/figures/LatLon_histo"//"-"//trim(adjustl(numberfile))//".pdf"
204     enddo
205     ! ------------------------------------------------------------------      .
206     ! ------------------------------------------------------------------      .
207     do m=1,nbseismes
208       write(numberfile(1:5),'(i5)')m
209       write(600,*)"file=OUTPUT/GMT/ZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".ps"
210       write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%Tzero(m)%themin,"/",dp%Tzero(m)%themax
211       write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
```

```
        " -Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
        "51':/a",real(int(((dp%Tzero(m)%themax-dp%Tzero(m)%themin)/5.0_wr)*10.0_wr),wr)/10.0_wr,":'"//dp%Tzero(m)%char//"':nSeW", &
        " -K -Y6.3i -Xc >  $file"
      do i=1,nbChaineMV
        write(char,'(i5)')10000+i
        call RVB(i,nbChaineMV,color)
        write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theTzero"//char//"-"//trim(adjustl(numberfile))//".bin", &
          " -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
      enddo
      write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
      write(600,'(a12,i9.9,a1,E13.7,a1,E13.7)')"geozone=-R0/",maxiter,"/",dp%Zhypo(m)%themin,"/",dp%Zhypo(m)%themax
      write(600,'(3a,E13.7,2a)')"psbasemap $geozone $geoproj ", &
        " -Ba"//trim(adjustl(char1))//":'mod\350les \050\351chantillonnage 1/"//trim(adjustl(char2))//"\0", &
        "51':/a",real(int(((dp%Zhypo(m)%themax-dp%Zhypo(m)%themin)/5.0_wr)*10.0_wr),wr)/10.0_wr,":'"//dp%Zhypo(m)%char//"':nSeW", &
        " -K -O -Y-5.5i >>  $file"
      do i=1,nbChaineMV
        write(char,'(i5)')10000+i
        call RVB(i,nbChaineMV,color)
        write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/theZhypo"//char//"-"//trim(adjustl(numberfile))//".bin", &
          " -bi2d -Wthinnest,"//color//" -O -K -: >>  $file"
      enddo
      write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
      write(600,*)"ps2raster OUTPUT/GMT/ZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
      write(600,'(2a)')"mv OUTPUT/GMT/ZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".pdf ", &
        "OUTPUT/figures/ZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".pdf"
    enddo
    ! -----------------------------------------------------------------    .
    write(600,*)
    write(600,*)"#*************************************#"
    write(600,*)"#*************************************#"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
      int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! -----------------------------------------------------------------    .


    ! -----------------------------------------------------------------    .
    ! plot figures fonctions d'autocorrélation
    ! -----------------------------------------------------------------    .
    write(*,*)"ecriture des script GMT_autocorr "
    write(600,*)"BEFORE=$SECONDS"
    call system_clock(Noldtime)
    write(600,*)"#*************************************#"
    write(600,*)"#*************************************#"
    write(600,*)
    write(600,*)"echo 'execution du script GMT autocorr'"
    write(600,*)"#######################################"
    write(600,*)"###########     autocorr      ##########"
    write(600,*)"#######################################"
    ! -----------------------------------------------------------------    .
    write(600,*)"geoproj=-JX13.5i/4.5i"
    ! -----------------------------------------------------------------    .
    write(600,'(a,i9.9,a)')"geozone=-R-100/",autocorr,"/-1/1"
    write(600,*)"file=OUTPUT/GMT/autoVCVM_histo.ps"
    ! -----------------------------------------------------------------    .
    write(600,'(a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur V@-C @-':nSeW -K -Y6.3i -Xc >  $file"
    do i=1,nbChaineMV
      write(char,'(i5)')i
      call RVB(i,nbChaineMV,color)
      write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/autovar_VC"//trim(adjustl(char))//".txt ", &
        "-Wthinnest,"//color//" -O -K >>  $file"
    enddo
```

```
277     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
278     write(600,'(a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur V@-M @-':nSeW -K -O -Y-5.5i >>  $file"
279     do i=1,nbChaineMV
280       write(char,'(i5)')i
281       call RVB(i,nbChaineMV,color)
282       write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/autovar_VM"//trim(adjustl(char))//".txt ", &
283         "-Wthinnest,"//color//" -O -K >>  $file"
284     enddo
285     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
286     write(600,*)"ps2raster OUTPUT/GMT/autoVCVM_histo.ps -Tf -A"
287     write(600,'(a)')"mv OUTPUT/GMT/autoVCVM_histo.pdf OUTPUT/figures/autoVCVM_histo.pdf"
288     ! ------------------------------------------------------------------    .
289     ! ------------------------------------------------------------------    .
290     write(600,*)"file=OUTPUT/GMT/autoVpVsZmoho_histo.ps"
291     write(600,'(2a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur V@-P @- / V@-S @- ':nSeW ", &
292       " -K -Y6.3i -Xc >  $file"
293     do i=1,nbChaineMV
294       write(char,'(i5)')i
295       call RVB(i,nbChaineMV,color)
296       write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/autovar_VpVs"//trim(adjustl(char))//".txt ", &
297         "-Wthinnest,"//color//" -O -K >>  $file"
298     enddo
299     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
300     write(600,'(2a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur la profondeur du moho ':nSeW ", &
301       "-K -O -Y-5.5i >>  $file"
302     do i=1,nbChaineMV
303       write(char,'(i5)')i
304       call RVB(i,nbChaineMV,color)
305       write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/autovar_Zmoho"//trim(adjustl(char))//".txt ", &
306         "-Wthinnest,"//color//" -O -K >>  $file"
307     enddo
308     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
309     write(600,*)"ps2raster OUTPUT/GMT/autoVpVsZmoho_histo.ps -Tf -A"
310     write(600,'(a)')"mv OUTPUT/GMT/autoVpVsZmoho_histo.pdf OUTPUT/figures/autoVpVsZmoho_histo.pdf"
311     ! ------------------------------------------------------------------    .
312     ! ------------------------------------------------------------------    .
313     do m=1,nbseismes
314       write(numberfile(1:5),'(i5)')m
315       write(600,*)"file=OUTPUT/GMT/autoLatLon_histo"//"-"//trim(adjustl(numberfile))//".ps"
316       write(600,'(2a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur la longitude':nSeW", &
317         " -K -Y6.3i -Xc >  $file"
318       do i=1,nbChaineMV
319         write(char,'(i5)')i
320         call RVB(i,nbChaineMV,color)
321         write(600,*)"psxy $geozone $geoproj ", &
322           "OUTPUT/GMT/autovar_lon_"//trim(adjustl(numberfile))//"_"//trim(adjustl(char))//".txt ", &
323           "-Wthinnest,"//color//" -O -K >>  $file"
324       enddo
325       write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
326       write(600,'(2a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur la latitude':nSeW ", &
327         "-K -O -Y-5.5i >>  $file"
328       do i=1,nbChaineMV
329         write(char,'(i5)')i
330         call RVB(i,nbChaineMV,color)
331         write(600,*)"psxy $geozone $geoproj ", &
332           "OUTPUT/GMT/autovar_lat_"//trim(adjustl(numberfile))//"_"//trim(adjustl(char))//".txt ", &
333           "-Wthinnest,"//color//" -O -K >>  $file"
334       enddo
335       write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
336       write(600,*)"ps2raster OUTPUT/GMT/autoLatLon_histo"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
337       write(600,'(2a)')"mv OUTPUT/GMT/autoLatLon_histo"//"-"//trim(adjustl(numberfile))//".pdf ", &
338         "OUTPUT/figures/autoLatLon_histo"//"-"//trim(adjustl(numberfile))//".pdf"
339     enddo
340     ! ------------------------------------------------------------------    .
341     ! ------------------------------------------------------------------    .
```

```
342     do m=1,nbseismes
343       write(numberfile(1:5),'(i5)')m
344       write(600,*)"file=OUTPUT/GMT/autoZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".ps"
345
346       write(600,'(2a)')"psbasemap $geozone $geoproj -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur le temps initial ':nSeW ", &
347         "-K -Y6.3i -Xc >  $file"
348       do i=1,nbChaineMV
349         write(char,'(i5)')i
350         call RVB(i,nbChaineMV,color)
351         write(600,*)"psxy $geozone $geoproj ", &
352           "OUTPUT/GMT/autovar_Tzero_"//trim(adjustl(numberfile))//"_"//trim(adjustl(char))//".txt ", &
353           " -Wthinnest,"//color//" -O -K >>  $file"
354       enddo
355       write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
356       write(600,'(2a)')"psbasemap $geozone $geoproj ", &
357         " -Ba1000f500g10000:k:/a0.5g5:'r@-k@- sur la profondeur de l\234hypocentre':nSeW -K -O -Y-5.5i >>  $file"
358       do i=1,nbChaineMV
359         write(char,'(i5)')i
360         call RVB(i,nbChaineMV,color)
361         write(600,*)"psxy $geozone $geoproj ", &
362           "OUTPUT/GMT/autovar_Zhypo_"//trim(adjustl(numberfile))//"_"//trim(adjustl(char))//".txt ", &
363           "-Wthinnest,"//color//" -O -K >>  $file"
364       enddo
365
366       write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
367       write(600,*)"ps2raster OUTPUT/GMT/autoZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
368       write(600,'(2a)')"mv OUTPUT/GMT/autoZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".pdf ",&
369         "OUTPUT/figures/autoZhypoTzero_histo"//"-"//trim(adjustl(numberfile))//".pdf"
370     enddo
371     ! ------------------------------------------------------------------     .
372     write(600,*)
373     write(600,*)"#####################################"
374     write(600,*)"#####################################"
375     write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
376     write(600,*)" echo $ELAPSED secondes"
377     call system_clock(Nnewtime,ratetime)
378     tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
379     write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
380     int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
381     ! ------------------------------------------------------------------     .
382
383   end subroutine GMT_param
384
385     ! ------------------------------------------------------------------     .
386
387   subroutine RVB(i,n,color)
388     ! -------                                                 ---------     .mh
389     ! color en RVB sur vecteur de i sur n
390     ! -------                                                 ---------     .
391     implicit none
392     ! -------                                                 ---------     .
393     integer(KIND=wi), intent (in) :: i,n
394     character(LEN=11), intent (out) :: color
395     ! -------                                                 ---------     .
396     real(KIND=wr) :: val,r,v,b
397     ! ------------------------------------------------------------------     .
398     val=15.0_wr+real(int(real(n-i,wr)/real(n,wr)*250.0_wr,wi),wr)*3.0_wr    ! décale pour le fun (+15)
399     ! -------                                                 ---------     . rouge
400     if (val.gt.255.0_wr) then
401       r=0.0_wr
402     endif
403     if (val.lt.(255.0_wr/2.0_wr)) then
404       r=real(int(val),wr)*2.0_wr
405     endif
406     if ((val.gt.(255.0_wr/2.0_wr)).and.(val.lt.255.0_wr)) then
```

```
      r=255.0_wr*2.0_wr-real(int(val),wr)*2.0_wr
    endif
    ! -------                                                  ---------     . vert
    if (val.gt.(255.0_wr*2.0_wr)) then
      v=0.0_wr
    endif
    if (val.lt.255.0_wr) then
      v=0.0_wr
    endif
    if ((val.gt.255.0_wr).and.(val.lt.(255.0_wr+255.0_wr/2.0_wr))) then
      v=real(int(val),wr)*2.0_wr-2.0_wr*255.0_wr
    endif
    if ((val.gt.(255.0_wr+255.0_wr/2.0_wr)).and.(val.lt.(255.0_wr*2.0_wr))) then
      v=255.0_wr*4.0_wr-real(int(val),wr)*2.0_wr
    endif
    ! -------                                                  ---------     . bleu
    if (val.lt.(255.0_wr*2.0_wr)) then
      b=0.0_wr
    endif
    if ((val.lt.(255.0_wr*2.0_wr+255.0_wr/2.0_wr)).and.(val.gt.(255.0_wr*2.0_wr))) then
      b=real(int(val),wr)*2.0_wr-4.0_wr*255.0_wr
    endif
    if (val.gt.(255.0_wr*2.0_wr+255.0_wr/2.0_wr)) then
      b=255.0_wr*6.0_wr-real(int(val),wr)*2.0_wr
    endif
    write(color,'(i3.3,a1,i3.3,a1,i3.3)')int(r,wi),"/",int(v,wi),"/",int(b,wi)
    ! ------------------------------------------------------------------     .
  end subroutine RVB

END MODULE figure_GMTpar



! ********************************************************************     .
! ********************************************************************     .
```

2.8 SRC/MOD/MOD_GMT/mkmatricecorrel.f90

```
! permet la création des scripts GMT pour les figures
! octobre 2014
! ********************************************************************     .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------     .
! ********************************************************************     .
! ------------------------------------------------------------------     .

MODULE figure_GMTmCorr

    use modparam

    implicit none

    private

    public  :: GMT_mCorr


CONTAINS

! ------------------------------------------------------------------     .

subroutine GMT_mCorr(dp)
    ! -------                                                  ---------     .mh
    ! figures de la matrice de corrélation
    ! ------------------------------------------------------------------     .
    use typetemps
```

```
    use statistiques
    ! ------                                                  --------    .
    implicit none
    ! ------                                                  --------    .
    type(densityplot), intent (in) :: dp                                  ! modèles retenus par McMC
    ! ------                                                  --------    .
    integer(KIND=wi) :: i,j,nbpar,Noldtime, Nnewtime, ratetime
    real(KIND=wr) :: rp, tl
    real(KIND=wr), dimension(:,:), allocatable :: tabCorrel
    character(LEN=15), dimension(:), allocatable :: namepar
    character (LEN=5) :: numberchaine
    ! ------------------------------------------------------------    .
    nbpar=4+nbseismes*4
    ! ------                                                  --------    .
    allocate(tabCorrel(nbpar,dp%nbparam))
    allocate(namepar(nbpar))
    tabCorrel(1,:)=dp%VC%vec ; namepar(1)='VC'
    tabCorrel(2,:)=dp%VM%vec ; namepar(2)='VM'
    tabCorrel(3,:)=dp%Zmoho%vec ; namepar(3)='Zmoho'
    tabCorrel(4,:)=dp%VpVs%vec ; namepar(4)='VpVs'
    j=1
    do i=5,nbpar,4
      write(numberchaine(1:5),'(i5)')j
      tabCorrel(i,:)=dp%Lat(j)%vec ; namepar(i)='Lat_('//trim(adjustl(numberchaine))//")"
      tabCorrel(i+1,:)=dp%Lon(j)%vec ; namepar(i+1)='Lon_('//trim(adjustl(numberchaine))//")"
      tabCorrel(i+2,:)=dp%Zhypo(j)%vec ; namepar(i+2)='Zhypo_('//trim(adjustl(numberchaine))//")"
      tabCorrel(i+3,:)=dp%Tzero(j)%vec ; namepar(i+3)='Tzero_('//trim(adjustl(numberchaine))//")"
      j=j+1
    enddo
    ! ------                                                  --------    .
    open(unit=100,file="OUTPUT/GMT/matrice.corr",STATUS="replace")
    do i=1,nbpar
      do j=1,nbpar
        call Rpcalc(tabCorrel(i,:),tabCorrel(j,:),dp%nbparam,Rp)
        write(100,'(f6.1,1x,f6.1,1x,f9.6,1x,a,1x,a)') real(i,wr)-0.5_wr, real(j,wr)-0.5_wr ,Rp, &
          trim(adjustl(namepar(i))), trim(adjustl(namepar(j)))
      enddo
    enddo
    deallocate(tabCorrel)
    close(100)
    ! ------------------------------------------------------------    .
    write(*,*)"ecriture des script GMT_matCorr "
    write(600,*)"BEFORE=$SECONDS"
    call system_clock(Noldtime)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)
    write(600,*)"echo 'execution du script GMT matCorr'"
    write(600,*)"#########################################"
    write(600,*)"##########     autocorr      ##########"
    write(600,*)"#########################################"
    ! ------------------------------------------------------------    .
    write(600,*)"geoproj=-JX5i"
    write(600,'(a,i9.9,a,i9.9)')" geozone=-R-0/",nbpar,"/0/",nbpar
    write(600,*)"file=OUTPUT/GMT/matCorr.ps"
    ! ------                                                  --------    .
    write(600,'(a)')"echo '-1.00 000 000 255 -0.25 255 255 255' > OUTPUT/GMT/colorpal6.cpt"
    write(600,'(a)')"echo '-0.25 255 255 255 0.250 255 255 255' >> OUTPUT/GMT/colorpal6.cpt"
    write(600,'(a)')"echo '0.250 255 255 255 1.000 255 000 000' >> OUTPUT/GMT/colorpal6.cpt"
    write(600,*)"cat OUTPUT/GMT/matrice.corr | awk '{print $1, $2, $3}' > OUTPUT/GMT/toto.d"
    write(600,'(a,i9.9,2a)')" head -",nbpar," OUTPUT/GMT/matrice.corr | awk ", &
        "'{print -1.0, $2, 4, 0, 4, ""RM"", $5}' > OUTPUT/GMT/toto2.d"
    write(600,'(a,i9.9,2a)')" head -",nbpar," OUTPUT/GMT/matrice.corr | awk ", &
        "'{print $2, -1.0, 4, -90, 4, ""LM"", $5}' > OUTPUT/GMT/toto3.d"
    ! ------                                                  --------    .
```

```
    write(600,*)"xyz2grd -I.5 $geozone -GOUTPUT/GMT/toto1.grd OUTPUT/GMT/toto.d"
    write(600,*)"grdsample OUTPUT/GMT/toto1.grd -I1 -GOUTPUT/GMT/toto.grd -F"
    write(600,*)"grdimage OUTPUT/GMT/toto.grd $geozone $geoproj -COUTPUT/GMT/colorpal6.cpt -K -Sn -Xc -Yc > $file"
    write(600,*)"pstext OUTPUT/GMT/toto2.d $geozone $geoproj -O -K -N -Ba0 >> $file"
    write(600,*)"pstext OUTPUT/GMT/toto3.d $geozone $geoproj -O -K -N -Ba0 >> $file"
    do i=0,nbpar,4
      write(600,*)"echo -e '",i,0,"\n",i,nbpar,"' | psxy $geozone $geoproj -O -K -W3 >> $file"
      write(600,*)"echo -e '",0,i,"\n",nbpar,i,"' | psxy $geozone $geoproj -O -K -W3 >> $file"
    enddo
    write(600,*)"psscale -D5.5i/2.5i/5i/0.2i -O -K -COUTPUT/GMT/colorpal6.cpt -B0.2:'corr\351lation': >> $file"
    ! ------                                                    --------      .
    write(600,*)"psbasemap $geozone $geoproj -Ba0g1 -O >>  $file"
    write(600,*)"ps2raster OUTPUT/GMT/matCorr.ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/matCorr.pdf OUTPUT/figures/matCorr.pdf"
    ! ------------------------------------------------------------------      .
    write(600,*)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
    int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------------      .

  end subroutine GMT_mCorr

    ! ------------------------------------------------------------------      .

END MODULE figure_GMTmCorr



! ********************************************************************      .
! ********************************************************************      .
```

2.9 SRC/MOD/MOD_GMT/mkres.f90

```
! permet la création des scripts GMT pour une carte des résidus par stations
! mars 2014
! ********************************************************************      .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------      .
! ********************************************************************      .
! ------------------------------------------------------------------      .

MODULE figure_GMTres

    use modparam

    implicit none

    private

    public  :: GMT_res, GMT_resSTA


    ! ------------------------------------------------------------------      .

    TYPE sta_tps
        character(LEN=4) :: staname                                     ! nom de la station
        real(KIND=wr) :: lonSTA, latSTA                                 ! coordonnées de la station et résidus
        real(KIND=wr) :: Ttot, TPn, TPg, TSn, TSg                       ! résidus (somme absolue, puis par type d'onde)
        real(KIND=wr) :: TpsPn, TpsPg, TpsSn, TpsSg                     ! temps (absolue)
        real(KIND=wr) :: pdsPn, pdsPg, pdsSn, pdsSg                     ! pondérations par type d'onde
```

```
27          real(KIND=wr) :: disthypo
28      END TYPE sta_tps
29
30  CONTAINS
31
32  ! ------------------------------------------------------------------     .
33
34    subroutine GMT_resSTA(nbsta,nbtps,D,nomsta)
35      ! -------                                                 --------     .mh
36      ! plot residus des tous les séismes à chaque station pour un modèle (ensemble des 1000 meilleurs)
37      ! -------                                                 --------     .
38      use typetemps
39      use time
40      use sub_param
41      use statistiques
42      implicit none
43      ! -------                                                 --------     .
44      integer(KIND=wi), intent(in) :: nbtps(nbseismes),nbsta                 ! nombre de données de temps et de station possible
45      type(dataall), intent(in) :: D(nbseismes)                              ! données
46      character(LEN=4), dimension(:), allocatable, intent(out)  :: nomsta
47      ! -------                                                 --------     .
48      ! pour "biner" les résidus et definir si la distribution est gaussienne
49      ! delta = 0.05 sec ;  borne inf = numero * delta - delta/2
50      real(KIND=wr) :: binage(-5000:5000), binageR(-5000:5000)
51      real(KIND=wr) :: a,b,R2,XY(10001,3),Rp
52      real(KIND=wr), dimension(:), allocatable :: XYbis(:,:)
53      ! -------                                                 --------     .
54      real(KIND=wr) :: val,X,Y,ymay
55      real(KIND=wr) :: lon,lat,alti
56      real(KIND=wr) :: moy(4),ec(4),med(4),gauss(4)
57      real(KIND=wr), dimension(:), allocatable  :: vec
58      integer(KIND=wi) :: i,j,k,ok,sta,noctet
59      integer(KIND=wi) :: iPg,iPn,iSg,iSn, nb(4)
60      logical :: test, existe(4)
61      character(LEN=4) :: aname(nbsta+1)
62      integer(KIND=wi), dimension(:,:), allocatable :: anbname
63      ! ------------------------------------------------------------------     . nb de stations différentes
64      ! -------                                                 --------     . initialistaion
65      aname(1)=D(1)%datatps(1)%sta%staname
66      do i=2,nbsta+1
67        aname(i)='123_'
68      enddo
69      sta=0
70      ! -------                                                 --------     .
71      do i=1,nbseismes
72        do j=1,nbtps(i)
73          test=.true.
74          k=1
75          do while (aname(k).ne.'123_')
76            if (aname(k)==D(i)%datatps(j)%sta%staname) test=.false.
77            k=k+1
78          enddo
79          if(test) then
80            sta=sta+1
81            aname(sta)=D(i)%datatps(j)%sta%staname
82          endif
83        enddo
84      enddo
85      allocate(nomsta(sta))
86      ! ------------------------------------------------------------------     . un fichier de résidus par station
87      do k=1,sta
88        open(950, FILE ='OUTPUT/GMT/res-PG-'//aname(k)//'.d',status='replace')
89        open(951, FILE ='OUTPUT/GMT/res-PN-'//aname(k)//'.d',status='replace')
90        open(952, FILE ='OUTPUT/GMT/res-SG-'//aname(k)//'.d',status='replace')
91        open(953, FILE ='OUTPUT/GMT/res-SN-'//aname(k)//'.d',status='replace')
```

```
      nomsta(k)=aname(k)
      iPg=0
      iPn=0
      iSg=0
      iSn=0
      val=0.0_wr
      do i=1,nbseismes
        do j=1,nbtps(i)
          if (aname(k)==D(i)%datatps(j)%sta%staname) then
            if (D(i)%datatps(j)%typeonde=='G') then
              write(950,*)D(i)%datatps(j)%dTP
              iPg=iPg+1
              if(D(i)%datatps(j)%andS=='S') then
                write(952,*)D(i)%datatps(j)%dTS
                iSg=iSg+1
              endif
            else if (D(i)%datatps(j)%typeonde=='N') then
              write(951,*)D(i)%datatps(j)%dTP
              iPn=iPn+1
              if(D(i)%datatps(j)%andS=='S') then
                write(953,*)D(i)%datatps(j)%dTS
                iSn=iSn+1
              endif
            else
              write(*,*)'problème dans GMT_resSTA : onde ni directe ni réfractée'
            endif
            if (abs(D(i)%datatps(j)%dTP).gt.val) val=abs(D(i)%datatps(j)%dTP)
            if (abs(D(i)%datatps(j)%dTS).gt.val) val=abs(D(i)%datatps(j)%dTS)
          endif
        enddo
      enddo
      val= max((real(int(val+0.55_wr,wi),wr)*10.0_wr)/10._wr,1.1_wr)        ! bornes du graph
      close(950)
      close(951)
      close(952)
      close(953)
      X = -val + 0.3_wr*(2.0_wr*val)
      Y = 95.0_wr
      ! ------------------------------------------------------------------     . script GMT
      write(*,*)"ecriture des script GMT_res STA "//aname(k)
      write(600,*)"#######################################"
      write(600,*)"#######################################"
      write(600,*)
      write(600,*)"echo 'execution du script GMT res STA'"//aname(k)
      write(600,*)"###########################################"
      write(600,*)"###########      histo      ###########"
      write(600,*)"###########################################"
      ! ------------------------------------------------------------------     .
      write(600,'(a10,E13.7,a1,E13.7,a6)')"geozone=-R",-val,"/",val,"/0/100"
      write(600,*)"geoproj=-JX4.25i"
      write(600,*)"file=OUTPUT/GMT/resSTA"//"-"//aname(k)//".ps"
      ! ---------                                                 ---------     .
      if (iPg.gt.0) then
        write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:", &
        "'r\351sidus ondes compressives directes':/a10:'effectif (%)':nSeW ", &
        " -X2.5i -K > $file "
        write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Z1 OUTPUT/GMT/res-PG-"//aname(k)//".d -O -K >> $file"
        write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iPg,""" | pstext $geozone $geoproj -O -K >> $file"
        write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$pp -Z1 OUTPUT/GMT/res-PG-"//aname(k)//".d -L0/0  -O -K >> $file"
        write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iPg,""" | pstext $geozone $geoproj -O -K >> $file"
        write(600,*)"psbasemap $geozone $geoproj -Bg1/a0 -O -K >> $file "

      else
        write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:", &
        "'r\351sidus ondes compressives directes':/a10:'effectif (%)':nSeW ", &
```

```
157         " -X2.5i -K > $file "
158         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : 0"" | pstext $geozone $geoproj -O -K >> $file"
159       endif
160       ! ------                                                  --------    .
161       if (iPn.gt.0) then
162         write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:'r\351sidus ondes compressives r\351fract\351es'", &
163         ":/a10:'effectif (%)':nSew -O -K -X5.5i >>  $file"
164         write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0 ", &
165         " -Z1 -K OUTPUT/GMT/res-PN-"//aname(k)//".d -O -K >> $file "
166         write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$pp -L0/0 -Ba0 ", &
167         " -Z1 -K OUTPUT/GMT/res-PN-"//aname(k)//".d -O -K >> $file "
168         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iPn,""" | pstext $geozone $geoproj -O -K >> $file"
169         write(600,*)"psbasemap $geozone $geoproj -Bg1/a0 -O -K >> $file "
170       else
171         write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:'r\351sidus ondes compressives r\351fract\351es'", &
172         ":/a10:'effectif (%)':nSew -O -K -X5.5i >>  $file"
173         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : 0"" | pstext $geozone $geoproj -O -K >> $file"
174       endif
175       ! ------                                                  --------    .
176       if (iSg.gt.0) then
177         write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:", &
178         "'r\351sidus ondes cisaillantes directes':/a10:'effectif (%)':nSew -O  -K -Y5.5i >>  $file"
179         write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0", &
180         " -Z1 -K OUTPUT/GMT/res-SG-"//aname(k)//".d -O -K >> $file "
181         write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$ss -L0/0 -Ba0", &
182         " -Z1 -K OUTPUT/GMT/res-SG-"//aname(k)//".d -O -K >> $file "
183         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iSg,""" | pstext $geozone $geoproj -O -K >> $file"
184         write(600,*)"psbasemap $geozone $geoproj -Bg1/a0 -O -K >> $file "
185       else
186         write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:", &
187         "'r\351sidus ondes cisaillantes directes':/a10:'effectif (%)':nSew -O  -K -Y5.5i >>  $file"
188         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : 0"" | pstext $geozone $geoproj -O -K >> $file"
189       endif
190       ! ------                                                  --------    .
191       if (iSn.gt.0) then
192         write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:'r\351sidus ondes cisaillantes r\351fract\351es'", &
193         ":/a10:'effectif (%)':nSeW -O -K -X-5.5i >>  $file"
194         write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0 ", &
195         " -Z1 OUTPUT/GMT/res-SN-"//aname(k)//".d -O -K >> $file "
196         write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$ss -L0/0 -Ba0 ", &
197         " -Z1 OUTPUT/GMT/res-SN-"//aname(k)//".d -O -K >> $file "
198         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iSn,""" | pstext $geozone $geoproj -O -K >> $file"
199         write(600,*)"psbasemap $geozone $geoproj -Bg1/a0 -O >> $file "
200       else
201         write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:'r\351sidus ondes cisaillantes r\351fract\351es'", &
202         ":/a10:'effectif (%)':nSeW -O -K  -X-5.5i >>  $file"
203         write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : 0"" | pstext $geozone $geoproj -O >> $file"
204       endif
205       ! ------                                                  --------    .
206       write(600,*)"ps2raster OUTPUT/GMT/resSTA-"//aname(k)//".ps -Tf -A"
207       write(600,'(a)')"mv OUTPUT/GMT/resSTA-"//aname(k)//".pdf OUTPUT/figures/resSTA-"//aname(k)//".pdf"
208       write(600,*)
209     ! ------------------------------------------------------------    .
210     enddo
211     ! ------------------------------------------------------------    .
212
213     write(600,*)"#########################################"
214     write(600,*)"###########     all histo       ##########"
215     write(600,*)"#########################################"
216
217     write(600,*)"geoproj=-JX5i"
218     write(600,*)"file=OUTPUT/GMT/Allres.ps"
219     write(600,*)"cat OUTPUT/GMT/residus*pg OUTPUT/GMT/residus*pn OUTPUT/GMT/residus*sg ", &
220       "OUTPUT/GMT/residus*sn > OUTPUT/GMT/allres.sn"
221     write(600,*)"cat OUTPUT/GMT/residus*pg OUTPUT/GMT/residus*pn OUTPUT/GMT/residus*sg > OUTPUT/GMT/allres.sg"
```

```
    write(600,*)"cat OUTPUT/GMT/residus*pg OUTPUT/GMT/residus*pn > OUTPUT/GMT/allres.pn"
    write(600,*)"cat OUTPUT/GMT/residus*pg > OUTPUT/GMT/allres.pg"

    write(600,*)"pshistogram -W0.02  -Gred  OUTPUT/GMT/allres.sn -IO > OUTPUT/GMT/allres.all 2>/dev/null "
    write(600,*)"minmax -I1.5 OUTPUT/GMT/allres.all > OUTPUT/GMT/geozonehistoall 2>/dev/null "
    write(600,*)"read geozone < OUTPUT/GMT/geozonehistoall"

    write(600,*)"pshistogram  $geozone $geoproj -W0.02  -Gred  OUTPUT/GMT/allres.sn -K ", &
      "-Ba.5f.25g10:'r\351sidus (s)':/a5f1:'effectif':nSeW > $file"

    write(600,*)"pshistogram  $geozone $geoproj -W0.02  -G$ss  OUTPUT/GMT/allres.sg -K -O >> $file"
    write(600,*)"pshistogram  $geozone $geoproj -W0.02  -Gblue OUTPUT/GMT/allres.pn -K -O >> $file"
    write(600,*)"pshistogram  $geozone $geoproj -W0.02  -G$pp  OUTPUT/GMT/allres.pg -O >> $file"

    write(600,*)"ps2raster $file -Tf -A"
    write(600,'(a)')"mv OUTPUT/GMT/Allres.pdf OUTPUT/figures/Allres.pdf "
    write(600,*)
    ! ------------------------------------------------------------------    .
    val=0.0_wr
    if (FLAGresSTA) then
      nb(:)=-1
      ! -------                                                 --------    .
      ! plot residus des tous les séismes à chaque station pour tous les modèles sélectionnés
      ! -------                                                 --------    .
      inquire (iolength = noctet)val
      ! ------------------------------------------------------------------    .
      open(956, FILE ='OUTPUT/input/sta_new.d',status='replace')
      open(957, FILE ='OUTPUT/GMT/sta_RES_TOT2latex.txt',status='replace')
      open(958, FILE ='OUTPUT/GMT/sta_RES_TOT.txt',status='replace')
      ! ------------------------------------------------------------------    .
      ! -------  relecture des fichier et calcul de la moyenne et médiane  .
      do k=1,sta
        moy=0.0_wr
        med=0.0_wr
        ec=0.0_wr
        ! -------                                                 --------    . PG
        inquire(file='OUTPUT/files/STA/'//aname(k)//'-PG.bin',exist=existe(1)) ! on teste l'existence du fichier
        ok=0
        if (existe(1)) then
          open(111, FILE ='OUTPUT/files/STA/'//aname(k)//'-PG.bin',status='old',access='direct',RECL=(noctet),iostat = ok)
          if (ok .ne. 0) then
            write(*,*)'problème dans GMT_resSTA : le fichier OUTPUT/files/STA/'//aname(k)//'-PG.bin n''existe pas'
            stop
          endif
          nb(1)=0
          do while(ok.eq.0)                                             ! boucle pour compter le nombre de lignes du fichier
            nb(1)=nb(1)+1
            read(111,rec=nb(1),iostat = ok)
          end do
          nb(1)=nb(1)-1
          allocate(vec(nb(1)))
          do j=1,nb(1)
            read(111,rec=j)vec(j)
            if (abs(vec(j)).gt.val)val=abs(vec(j)) ! garde le max pour affichage
          enddo
          close(111)
          call moy_ec(vec,nb(1),nb(1),moy(1),ec(1))
          call mediane(2.0_wr,vec,nb(1),med(1))
          ! ------------------------------------------------------------    .
          ! calcul de la droite de Henry
          ! méthode graphique pour verifier si une distribution est gaussienne
          ! ------------------------------------------------------------    .
          binage(:)=0.0_wr                                              ! initialisation
          XY(:,:)=0.0_wr
          do j=1,nb(1)
```

```
287             i=int((vec(j)+0.0005_wr)/0.001_wr,wi)
288             if (i.gt.5000) i=5000
289             if (i.lt.-5000) i=-5000
290             binage(i)=binage(i)+1.0_wr                                  ! binage
291           enddo
292           binage(-5000)=binage(-5000)/real(nb(1),wr)
293           do j=-4999,5000
294             binage(j)=binage(j)/real(nb(1),wr)+binage(j-1)              ! normalise et transforme en distribution cumulée
295           enddo
296           i=0
297           do j=-5000,5000
298             if ((binage(j).gt.0.15_wr).and.(binage(j).lt.0.85_wr)) then
299               call inv_normal_cumulative_distrib_func(binage(j),binageR(j))! computing the inverse normal cumulative distribution function
300               i=i+1
301               XY(i,1)=real(j,wr)*0.001_wr-0.0005_wr
302               XY(i,2)=binageR(j)
303               XY(i,3)=1.0_wr                                            ! pondération constante
304             endif
305           enddo
306           allocate (XYbis(i,3))
307           do j=1,i
308             XYbis(j,1)=XY(j,1)
309             XYbis(j,2)=XY(j,2)
310             XYbis(j,3)=XY(j,3)
311           enddo
312           call correlationpond(a,b,R2,i,XYbis)                          ! regession linéaire sur binageR
313           do j=1,i
314             XYbis(j,1)=a*XYbis(j,1)+b
315           enddo
316           if (i.gt.2) then
317             Call Rpcalc(XYbis,i,i,Rp)                                   ! coeficient de corrélation linéaire de Bravais-Pearson
318           else
319             Rp=0.0_wr
320           endif
321           deallocate (XYbis)
322           gauss(1)=Rp
323           ! ------------------------------------------------------------     .
324           deallocate(vec)
325         else
326           nb(1)=0
327           moy(1)=0.0_wr
328           ec(1)=0.0_wr
329           gauss(1)=0.0_wr
330         endif
331         ! -------                                          --------     . PN
332         inquire(file='OUTPUT/files/STA/'//aname(k)//'-PN.bin',exist=existe(2)) ! on teste l'existence du fichier
333         ok=0
334         if (existe(2)) then
335           open(111, FILE ='OUTPUT/files/STA/'//aname(k)//'-PN.bin',status='old',access='direct',RECL=(noctet),iostat = ok)
336           if (ok .ne. 0) then
337             write(*,*)'problème dans GMT_resSTA : le fichier OUTPUT/files/STA/'//aname(k)//'-PN.bin n''existe pas'
338             stop
339           endif
340           nb(2)=0
341           do while(ok.eq.0)                                             ! boucle pour compter le nombre de lignes du fichier
342             nb(2)=nb(2)+1
343             read(111,rec=nb(2),iostat = ok)
344           end do
345           nb(2)=nb(2)-1
346           allocate(vec(nb(2)))
347           do j=1,nb(2)
348             read(111,rec=j)vec(j)
349             if (abs(vec(j)).gt.val)val=abs(vec(j)) ! garde le max pour affichage
350             enddo
351           close(111)
```

```
352           call moy_ec(vec,nb(2),nb(2),moy(2),ec(2))
353           call mediane(2.0_wr,vec,nb(2),med(2))
354           ! ------------------------------------------------------------     .
355           ! calcul de la droite de Henry
356           ! méthode graphique pour verifier si une distribution est gaussienne
357           ! ------------------------------------------------------------     .
358           binage(:)=0.0_wr                                                   ! initialisation
359           XY(:,:)=0.0_wr
360           do j=1,nb(2)
361             i=int((vec(j)+0.0005_wr)/0.001_wr,wi)
362             if (i.gt.5000) i=5000
363             if (i.lt.-5000) i=-5000
364             binage(i)=binage(i)+1.0_wr                                       ! binage
365           enddo
366           binage(-5000)=binage(-5000)/real(nb(2),wr)
367           do j=-4999,5000
368             binage(j)=binage(j)/real(nb(2),wr)+binage(j-1)                   ! normalise et transforme en distribution cumulée
369           enddo
370           i=0
371           do j=-5000,5000
372             if ((binage(j).gt.0.15_wr).and.(binage(j).lt.0.85_wr)) then
373               call inv_normal_cumulative_distrib_func(binage(j),binageR(j))! computing the inverse normal cumulative distribution function
374               i=i+1
375               XY(i,1)=real(j,wr)*0.001_wr-0.0005_wr
376               XY(i,2)=binageR(j)
377               XY(i,3)=1.0_wr                                               ! pondération constante
378             endif
379           enddo
380           allocate (XYbis(i,3))
381           do j=1,i
382             XYbis(j,1)=XY(j,1)
383             XYbis(j,2)=XY(j,2)
384             XYbis(j,3)=XY(j,3)
385           enddo
386           call correlationpond(a,b,R2,i,XYbis)                             ! regession linéaire sur binageR
387           do j=1,i
388             XYbis(j,1)=a*XYbis(j,1)+b
389           enddo
390           if (i.gt.2) then
391             Call Rpcalc(XYbis,i,i,Rp)                                      ! coeficient de corrélation linéaire de Bravais-Pearson
392           else
393             Rp=0.0_wr
394           endif
395           deallocate (XYbis)
396           gauss(2)=Rp
397           ! ------------------------------------------------------------     .
398           deallocate(vec)
399         else
400           nb(2)=0
401           moy(2)=0.0_wr
402           ec(2)=0.0_wr
403           gauss(2)=0.0_wr
404         endif
405         ! -------                                          ---------     . SG
406         inquire(file='OUTPUT/files/STA/'//aname(k)//'-SG.bin',exist=existe(3)) ! on teste l'existence du fichier
407         ok=0
408         if (existe(3)) then
409           open(111, FILE ='OUTPUT/files/STA/'//aname(k)//'-SG.bin',status='old',access='direct',RECL=(noctet),iostat = ok)
410           if (ok .ne. 0) then
411             write(*,*)'problème dans GMT_resSTA : le fichier OUTPUT/files/STA/'//aname(k)//'-SG.bin n''existe pas'
412             stop
413           endif
414           nb(3)=0
415           do while(ok.eq.0)                                                ! boucle pour compter le nombre de lignes du fichier
416             nb(3)=nb(3)+1
```

```
            read(111,rec=nb(3),iostat = ok)
          end do
          nb(3)=nb(3)-1
          allocate(vec(nb(3)))
          do j=1,nb(3)
            read(111,rec=j)vec(j)
            if (abs(vec(j)).gt.val)val=abs(vec(j)) ! garde le max pour affichage
          enddo
          close(111)
          call moy_ec(vec,nb(3),nb(3),moy(3),ec(3))
          call mediane(2.0_wr,vec,nb(3),med(3))
          ! ------------------------------------------------------------     .
          ! calcul de la droite de Henry
          ! méthode graphique pour verifier si une distribution est gaussienne
          ! ------------------------------------------------------------     .
          binage(:)=0.0_wr                                                   ! initialisation
          XY(:,:)=0.0_wr
          do j=1,nb(3)
            i=int((vec(j)+0.0005_wr)/0.001_wr,wi)
            if (i.gt.5000) i=5000
            if (i.lt.-5000) i=-5000
            binage(i)=binage(i)+1.0_wr                                       ! binage
          enddo
            binage(-5000)=binage(-5000)/real(nb(3),wr)
          do j=-4999,5000
            binage(j)=binage(j)/real(nb(3),wr)+binage(j-1)                   ! normalise et transforme en distribution cumulée
          enddo
          i=0
          do j=-5000,5000
            if ((binage(j).gt.0.15_wr).and.(binage(j).lt.0.85_wr)) then
              call inv_normal_cumulative_distrib_func(binage(j),binageR(j))! computing the inverse normal cumulative distribution function
              i=i+1
              XY(i,1)=real(j,wr)*0.001_wr-0.0005_wr
              XY(i,2)=binageR(j)
              XY(i,3)=1.0_wr                                                 ! pondération constante
            endif
          enddo
          allocate (XYbis(i,3))
          do j=1,i
            XYbis(j,1)=XY(j,1)
            XYbis(j,2)=XY(j,2)
            XYbis(j,3)=XY(j,3)
          enddo
          call correlationpond(a,b,R2,i,XYbis)                               ! regession linéaire sur binageR
          do j=1,i
            XYbis(j,1)=a*XYbis(j,1)+b
          enddo
          if (i.gt.2) then
            Call Rpcalc(XYbis,i,i,Rp)                                        ! coeficient de corrélation linéaire de Bravais-Pearson
          else
            Rp=0.0_wr
          endif
          deallocate (XYbis)
          gauss(3)=Rp
          ! ------------------------------------------------------------     .
          deallocate(vec)
        else
          nb(3)=0
          moy(3)=0.0_wr
          ec(3)=0.0_wr
          gauss(3)=0.0_wr
        endif
        ! -------                                          --------     . SN
        inquire(file='OUTPUT/files/STA/'//aname(k)//'-SN.bin',exist=existe(4)) ! on teste l'existence du fichier
        ok=0
```

```
482         if (existe(4)) then
483           open(111, FILE ='OUTPUT/files/STA/'//aname(k)//'-SN.bin',status='old',access='direct',RECL=(noctet),iostat = ok)
484           if (ok .ne. 0) then
485             write(*,*)'problème dans GMT_resSTA : le fichier OUTPUT/files/STA/'//aname(k)//'-SN.bin n''existe pas'
486             stop
487           endif
488           nb(4)=0
489           do while(ok.eq.0)                                          ! boucle pour compter le nombre de lignes du fichier
490             nb(4)=nb(4)+1
491             read(111,rec=nb(4),iostat = ok)
492           end do
493           nb(4)=nb(4)-1
494           allocate(vec(nb(4)))
495           do j=1,nb(4)
496             read(111,rec=j)vec(j)
497             if (abs(vec(j)).gt.val)val=abs(vec(j)) ! garde le max pour affichage
498           enddo
499           close(111)
500           call moy_ec(vec,nb(4),nb(4),moy(4),ec(4))
501           call mediane(2.0_wr,vec,nb(4),med(4))
502           ! ---------------------------------------------------------    .
503           ! calcul de la droite de Henry
504           ! méthode graphique pour verifier si une distribution est gaussienne
505           ! ---------------------------------------------------------    .
506           binage(:)=0.0_wr                                           ! initialisation
507           XY(:,:)=0.0_wr
508           do j=1,nb(4)
509             i=int((vec(j)+0.0005_wr)/0.001_wr,wi)
510             if (i.gt.5000) i=5000
511             if (i.lt.-5000) i=-5000
512             binage(i)=binage(i)+1.0_wr                               ! binage
513           enddo
514           binage(-5000)=binage(-5000)/real(nb(4),wr)
515           do j=-4999,5000
516             binage(j)=binage(j)/real(nb(4),wr)+binage(j-1)           ! normalise et transforme en distribution cumulée
517           enddo
518           i=0
519           do j=-5000,5000
520             if ((binage(j).gt.0.15_wr).and.(binage(j).lt.0.85_wr)) then
521               call inv_normal_cumulative_distrib_func(binage(j),binageR(j))! computing the inverse normal cumulative distribution function
522               i=i+1
523               XY(i,1)=real(j,wr)*0.001_wr-0.0005_wr
524               XY(i,2)=binageR(j)
525               XY(i,3)=1.0_wr                                         ! pondération constante
526             endif
527           enddo
528           allocate (XYbis(i,3))
529           do j=1,i
530             XYbis(j,1)=XY(j,1)
531             XYbis(j,2)=XY(j,2)
532             XYbis(j,3)=XY(j,3)
533           enddo
534           call correlationpond(a,b,R2,i,XYbis)                       ! regession linéaire sur binageR
535           do j=1,i
536             XYbis(j,1)=a*XYbis(j,1)+b
537           enddo
538           if (i.gt.2) then
539             Call Rpcalc(XYbis,i,i,Rp)                                ! coeficient de corrélation linéaire de Bravais-Pearson
540           else
541             Rp=0.0_wr
542           endif
543           deallocate (XYbis)
544           gauss(4)=Rp
545           ! ---------------------------------------------------------    .
546           deallocate(vec)
```

```
      else
        nb(4)=0
        moy(4)=0.0_wr
        ec(4)=0.0_wr
        gauss(4)=0.0_wr
      endif
      ymay = 1.0_wr
      val = min(val,2.0_wr)
      X = -val + 0.1_wr*val
      Y = 0.9_wr*ymay
      ! ------------------------------------------------------------     . print figure
      write(*,*)"ecriture des script GMT_res STA TOT "//aname(k)
      write(600,*)"#######################################"
      write(600,*)"#######################################"
      write(600,*)
      write(600,*)"echo 'execution du script GMT res STA TOT ",aname(k),"'"
      write(600,*)"###########################################"
      write(600,*)"###########      histo        ###########"
      write(600,*)"###########################################"
      ! ------------------------------------------------------------     .
      write(600,'(a,E13.7,a,E13.7,a,E13.7)')"geozone=-R",-val,"/",val,"/0/",ymay
      write(600,*)"geoproj=-JX4.25i"
      write(600,*)"file=OUTPUT/GMT/resSTA_TOT"//"-"//aname(k)//".ps"
      ! -------                                                 --------     . PG
      if (nb(1).gt.0) then
        write(600,*)"psbasemap $geozone $geoproj -Bpa0.5g1/a0 -Bsf0.1:", &
        "'r\351sidus ondes compressives directes':/a.15:'effectif (%)':nSeW ", &
        " -X2.5i -K > $file "
        write(600,*)"echo -e ' ",moy(1)," 0.01 \n ",moy(1),ymay*.99_wr,"'| psxy $geozone $geoproj -W2,green -O -K >> $file"
        write(600,*)"echo -e ' ",moy(1)+ec(1)," 0.01 \n ",moy(1)+ec(1),ymay*.99_wr, &
        "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
        write(600,*)"echo -e ' ",moy(1)-ec(1)," 0.01 \n ",moy(1)-ec(1),ymay*.99_wr, &
        "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
        write(600,*)"echo -e ' ",med(1)," 0.01 \n ",med(1),ymay*.99_wr, &
        "'| psxy $geozone $geoproj -W2,blue -O -K >> $file"
        write(600,*)"pshistogram -F $geozone $geoproj -W0.001 -G$pp -Z1 OUTPUT/files/STA/"//aname(k)//"-PG.bin ", &
        "-L0/0 -bi1d -O -K >> $file"
        write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : ",nb(1),""" | pstext $geozone $geoproj -O -K >> $file"
        Y = 0.8_wr*ymay
        write(600,'(a,f13.5,1x,f13.5,a,f13.5,a)')"echo """,X,Y," 15 0 5 LT Rp : ",gauss(1), &
          """ | pstext $geozone $geoproj -O -K >> $file"
        Y = 0.9_wr*ymay
      else
        write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:", &
        "'r\351sidus ondes compressives directes':/a.15:'effectif (%)':nSeW ", &
        " -X2.5i -K > $file "
        write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : 0"" | pstext $geozone $geoproj -O -K >> $file"
      endif
      ! -------                                                 --------     . PN
      if (nb(2).gt.0) then
        write(600,*)"psbasemap $geozone $geoproj -Bpa0.5g1/a0 -Bsf0.1:'r\351sidus ondes compressives r\351fract\351es'", &
        ":/a.15nSew -O -K -X5.5i >>  $file"
        write(600,*)"echo -e ' ",moy(2)," 0.01 \n ",moy(2),ymay*.99_wr,"'| psxy $geozone $geoproj -W2,green -O -K >> $file"
        write(600,*)"echo -e ' ",moy(2)+ec(2)," 0.01 \n ",moy(2)+ec(2),ymay*.99_wr, &
        "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
        write(600,*)"echo -e ' ",moy(2)-ec(2)," 0.01 \n ",moy(2)-ec(2),ymay*.99_wr, &
        "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
        write(600,*)"echo -e ' ",med(2)," 0.01 \n ",med(2),ymay*.99_wr, &
        "'| psxy $geozone $geoproj -W2,blue -O -K >> $file"
        write(600,*)"pshistogram -F $geozone $geoproj -W0.001 -G$pp -L0/0 -Ba0 ", &
        " -Z1 -K OUTPUT/files/STA/"//aname(k)//"-PN.bin -bi1d -O >> $file "
        write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : ",nb(2),""" | pstext $geozone $geoproj -O -K >> $file"
        Y = 0.8_wr*ymay
        write(600,'(a,f13.5,1x,f13.5,a,f13.5,a)')"echo """,X,Y," 15 0 5 LT Rp : ",gauss(2), &
          """ | pstext $geozone $geoproj -O -K >> $file"
```

```
612           Y = 0.9_wr*ymay
613         else
614           write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:'r\351sidus ondes compressives r\351fract\351es'", &
615           ":/a.15:'effectif (%)':nSew -O -K -X5.5i >>  $file"
616           write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : 0"" | pstext $geozone $geoproj -O -K >> $file"
617         endif
618         ! -------                                                    --------     . SG
619         if (nb(3).gt.0) then
620           write(600,*)"psbasemap $geozone $geoproj -Bpa0.5g1/a0 -Bsf0.1:", &
621           "'r\351sidus ondes cisaillantes directes':/a.15:'effectif (%)':nSew -O  -K -Y5.5i >>  $file"
622           write(600,*)"echo -e ' ",moy(3)," 0.01 \n ",moy(3),ymay*.99_wr,"'| psxy $geozone $geoproj -W2,green -O -K >> $file"
623           write(600,*)"echo -e ' ",moy(3)+ec(3)," 0.01 \n ",moy(3)+ec(3),ymay*.99_wr, &
624           "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
625           write(600,*)"echo -e ' ",moy(3)-ec(3)," 0.01 \n ",moy(3)-ec(3),ymay*.99_wr, &
626           "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
627           write(600,*)"echo -e ' ",med(3)," 0.01 \n ",med(3),ymay*.99_wr, &
628           "'| psxy $geozone $geoproj -W2,blue -O -K >> $file"
629           write(600,*)"pshistogram -F $geozone $geoproj -W0.001 -G$ss -L0/0 -Ba0 -bi1d ", &
630           " -Z1 -K OUTPUT/files/STA/"//aname(k)//"-SG.bin -O >> $file "
631           write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : ",nb(3),""" | pstext $geozone $geoproj -O -K >> $file"
632           Y = 0.8_wr*ymay
633           write(600,'(a,f13.5,1x,f13.5,a,f13.5,a)')"echo """,X,Y," 15 0 5 LT Rp : ",gauss(3), &
634             """ | pstext $geozone $geoproj -O -K >> $file"
635           Y = 0.9_wr*ymay
636         else
637           write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:", &
638           "'r\351sidus ondes cisaillantes directes':/a.15:'effectif (%)':nSew -O  -K -Y5.5i >>  $file"
639           write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : 0"" | pstext $geozone $geoproj -O -K >> $file"
640         endif
641         ! -------                                                    --------     . SN
642         if (nb(4).gt.0) then
643           write(600,*)"psbasemap $geozone $geoproj -Bpa0.5g1/a0 -Bsf0.1:'r\351sidus ondes cisaillantes r\351fract\351es'", &
644           ":/a.15:'effectif (%)':nSeW -O -K -X-5.5i >>  $file"
645           write(600,*)"echo -e ' ",moy(4)," 0.01 \n ",moy(4),ymay*.99_wr,"'| psxy $geozone $geoproj -W2,green -O -K >> $file"
646           write(600,*)"echo -e ' ",moy(4)+ec(4)," 0.01 \n ",moy(4)+ec(4),ymay*.99_wr, &
647           "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
648           write(600,*)"echo -e ' ",moy(4)-ec(4)," 0.01 \n ",moy(4)-ec(4),ymay*.99_wr, &
649           "'| psxy $geozone $geoproj -W2,green,-- -O -K >> $file"
650           write(600,*)"echo -e ' ",med(4)," 0.01 \n ",med(4),ymay*.99_wr, &
651           "'| psxy $geozone $geoproj -W2,blue -O -K >> $file"
652           write(600,*)"pshistogram -F $geozone $geoproj -W0.001 -G$ss -L0/0 -Ba0 ", &
653           " -Z1 OUTPUT/files/STA/"//aname(k)//"-SN.bin -O -K -bi1d >> $file "
654           write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : ",nb(4),""" | pstext $geozone $geoproj -O -K >> $file"
655           Y = 0.8_wr*ymay
656           write(600,'(a,f13.5,1x,f13.5,a,f13.5,a)')"echo """,X,Y," 15 0 5 LT Rp : ",gauss(4), &
657             """ | pstext $geozone $geoproj -O >> $file"
658           Y = 0.9_wr*ymay
659         else
660           write(600,*)"psbasemap $geozone $geoproj -Bpa0.5/a0 -Bsf0.1:'r\351sidus ondes cisaillantes r\351fract\351es'", &
661           ":/a.15:'effectif (%)':nSeW -O -K  -X-5.5i >>  $file"
662           write(600,*)"echo """,X,Y," 15 0 5 LT donn\351es : 0"" | pstext $geozone $geoproj -O >> $file"
663         endif
664         ! -------                                                    --------     .
665         write(600,*)"ps2raster OUTPUT/GMT/resSTA_TOT-"//aname(k)//".ps -Tf -A"
666         write(600,'(a)')"mv OUTPUT/GMT/resSTA_TOT-"//aname(k)//".pdf OUTPUT/figures/resSTA_TOT-"//aname(k)//".pdf"
667         write(600,*)
668         ! ------------------------------------------------------------------     .
669         allocate(anbname(sta,4))
670         anbname=0
671
672         do i=1,nbseismes
673           do j=1,nbtps(i)
674             if (aname(k)==D(i)%datatps(j)%sta%staname) then
675               lon=D(i)%datatps(j)%sta%lon
676               lat=D(i)%datatps(j)%sta%lat
```

```
677               alti=D(i)%datatps(j)%sta%alti
678
679               if (D(i)%datatps(j)%typeonde=='G') then
680                 anbname(k,1)=anbname(k,1)+1
681                 if (D(i)%datatps(j)%andS=='S') then
682                   anbname(k,3)=anbname(k,3)+1
683                 endif
684               elseif(D(i)%datatps(j)%typeonde=='N') then
685                 anbname(k,2)=anbname(k,2)+1
686                 if (D(i)%datatps(j)%andS=='S') then
687                   anbname(k,4)=anbname(k,4)+1
688                 endif
689               else
690                 write(*,*)'problème dans GMT_resSTA : onde ni G ni N'
691                 stop
692               endif
693
694
695             endif
696           enddo
697         enddo
698
699         ! ------------------------------------------------------------------      .
700
701         write(957,1122)aname(k)," & \np{",moy(1),"} {\small ($\pm$ \np{",2.0_wr*ec(1),"}}) & \np{",med(1),"} \\", &
702           " & \np{",moy(2),"} {\small ($\pm$ \np{",2.0_wr*ec(2),"}}) & \np{",med(2),"} \\", &
703           " & \np{",moy(3),"} {\small ($\pm$ \np{",2.0_wr*ec(3),"}}) & \np{",med(3),"} \\", &
704           " & \np{",moy(4),"} {\small ($\pm$ \np{",2.0_wr*ec(4),"}}) & \np{",med(4),"} \\"
705
706         ! ------------------------------------------------------------------      .
707         !  au moins trois stations ...
708         ! ------------------------------------------------------------------      .
709         if (anbname(k,1).lt.2) moy(1)=0.0_wr ! PG
710         if (anbname(k,2).lt.2) moy(2)=0.0_wr ! PN
711         if (anbname(k,3).lt.2) moy(3)=0.0_wr ! SG
712         if (anbname(k,4).lt.2) moy(4)=0.0_wr ! SN
713
714         if (anbname(k,1).lt.2) med(1)=0.0_wr ! PG
715         if (anbname(k,2).lt.2) med(2)=0.0_wr ! PN
716         if (anbname(k,3).lt.2) med(3)=0.0_wr ! SG
717         if (anbname(k,4).lt.2) med(4)=0.0_wr ! SN
718
719         ! ------------------------------------------------------------------      .
720         write(958,*)aname(k),lat,lon,alti,moy(1),2.0_wr*ec(1),med(1),(gauss(1)*1000.0_wr-990.0_wr)/40.0_wr, &
721                                           moy(2),2.0_wr*ec(2),med(2),(gauss(2)*1000.0_wr-990.0_wr)/40.0_wr, &
722                                           moy(3),2.0_wr*ec(3),med(3),(gauss(3)*1000.0_wr-990.0_wr)/40.0_wr, &
723                                           moy(4),2.0_wr*ec(4),med(4),(gauss(4)*1000.0_wr-990.0_wr)/40.0_wr
724
725         ! ------------------------------------------------------------------      .
726         write(956,*)aname(k),lat,lon,alti,med(1),med(2),med(3),med(4)
727
728         deallocate(anbname)
729       enddo
730       close(956)
731       close(957)
732       close(958)
733       ! ------------------------------------------------------------------      .
734     endif
735     ! ------------------------------------------------------------------      .
736     1122 format (2a,f10.3,a,f10.2,a,f10.3,2a,f10.3,a,f10.2,a,f10.3,2a,f10.3,a,f10.2,a,f10.3,2a,f10.3,a,f10.2,a,f10.3,a)
737
738     ! ------------------------------------------------------------------      .
739   end subroutine GMT_resSTA
740
741   ! ------------------------------------------------------------------      .
```

```
742
743
744   subroutine GMT_res(l,xmax,dp,nbtps,datatps)
745     ! _______                                                  ________     .mh
746     ! production du script GMT produisant une carte des stations
747     ! _______                                                  ________     .
748     use typetemps
749     use time
750     implicit none
751     ! _______                                                  ________     .
752     real(KIND=wr), intent(in) :: xmax
753     type(densityplot), intent(in) :: dp
754     integer(KIND=wi), intent(in) :: nbtps, l
755     type(dataone), intent(in) :: datatps(nbtps)
756     ! ______                                                   ________     .
757     type(sta_tps), dimension(:), allocatable :: statps
758     integer(KIND=wi) :: iPn, iPg, iSn, iSg
759     type(stations) :: datasta
760     real(KIND=wr) :: v1, v2, amax, val, size, tl, X, Y
761     real(KIND=wr) :: lat,lon,alti,moy(4),ec(4),med(4),gauss(4)
762     type(date_sec) :: tps_ref, a_time
763     real(KIND=wr) :: lon1,lon2,lat1,lat2
764     integer(KIND=wi) :: i,j,k,n,n2,ok,Noldtime, Nnewtime, ratetime, nbsta
765     character (LEN=13) :: sizename
766     character (LEN=5) :: numberfile
767     character (LEN=4) :: aname
768     ! ____________________________________________________________     .
769     write(numberfile(1:5),'(i5)')l
770     ! ____________________________________________________________     . nombre de stations
771     n=0
772     n2=0
773     do i=1,nbtps
774       k=0
775       do j=1,nbtps
776         if (datatps(i)%sta%staname.eq.datatps(j)%sta%staname) then
777           k=k+1
778         endif
779       enddo
780       if (k.eq.1) n=n+1
781       if (k.eq.2) n2=n2+1
782       if (k.eq.3) then
783         write(*,*)'problème dans GMT_res 1 : station présentes 3 fois'
784         stop
785       endif
786     enddo
787     n=n+n2/2
788     allocate(statps(n))
789     ! ____________________________________________________________     . initialisation
790     do i=1,n
791       statps(i)%staname = "0000"
792       statps(i)%lonSTA = 0.0_wr
793       statps(i)%latSTA = 0.0_wr
794       statps(i)%TPn = 0.0_wr
795       statps(i)%TSn = 0.0_wr
796       statps(i)%TPg = 0.0_wr
797       statps(i)%TSg = 0.0_wr
798       statps(i)%Ttot = 0.0_wr
799       statps(i)%pdsPn = 0.0_wr
800       statps(i)%pdsSn = 0.0_wr
801       statps(i)%pdsPg = 0.0_wr
802       statps(i)%pdsSg = 0.0_wr
803       statps(i)%TpsPn = 0.0_wr
804       statps(i)%TpsPg = 0.0_wr
805       statps(i)%TpsSn = 0.0_wr
806       statps(i)%TpsSg = 0.0_wr
```

```
807       statps(i)%disthypo = 0.0_wr
808     enddo
809     iPn=0
810     iPg=0
811     iSn=0
812     iSg=0
813     ! ------------------------------------------------------------------     .
814     i=0
815     tps_ref=dp%temps_ref(l)
816     tps_ref%sec=dp%Tzero(l)%vec(1)
817     call basetime(tps_ref)
818     donnees : do j=1,nbtps
819       n2 = 1
820       do k=1,n
821         if (datatps(j)%sta%staname.eq.statps(k)%staname) n2 = n2 +1        ! station déjà présente ?
822       enddo
823       if (n2.eq.1) then                                                    ! non
824         i = i +1
825         statps(i)%staname = datatps(j)%sta%staname
826         statps(i)%disthypo = datatps(j)%dhypo
827         statps(i)%lonSTA = datatps(j)%sta%lon
828         statps(i)%latSTA = datatps(j)%sta%lat
829         if (datatps(j)%typeonde.eq."N") then
830           iPn = iPn +1
831           statps(i)%TPn = datatps(j)%dTP
832           statps(i)%pdsPn = datatps(j)%wp
833           a_time%date=datatps(j)%tpsR%date
834           a_time%sec=datatps(j)%tpsR%secP
835           call basetime(a_time)
836           call difftime(statps(i)%TpsPn,a_time,tps_ref)
837           if (datatps(j)%andS.eq."S") then
838             iSn = iSn +1
839             statps(i)%TSn = datatps(j)%dTS
840             statps(i)%pdsSn = datatps(j)%ws
841             a_time%date=datatps(j)%tpsR%date
842             a_time%sec=datatps(j)%tpsR%secS
843             call basetime(a_time)
844             call difftime(statps(i)%TpsSn,a_time,tps_ref)
845           endif
846         elseif (datatps(j)%typeonde.eq."G") then
847           iPg = iPg +1
848           statps(i)%TPg = datatps(j)%dTP
849           statps(i)%pdsPg = datatps(j)%wp
850           a_time%date=datatps(j)%tpsR%date
851           a_time%sec=datatps(j)%tpsR%secP
852           call basetime(a_time)
853           call difftime(statps(i)%TpsPg,a_time,tps_ref)
854           if (datatps(j)%andS.eq."S") then
855             iSg = iSg +1
856             statps(i)%TSg = datatps(j)%dTS
857             statps(i)%pdsSg = datatps(j)%ws
858             a_time%date=datatps(j)%tpsR%date
859             a_time%sec=datatps(j)%tpsR%secS
860             call basetime(a_time)
861             call difftime(statps(i)%TpsSg,a_time,tps_ref)
862            endif
863         else
864           write(*,*)'problème dans GMT_res 1 : onde ni réfractée ni directe'
865           stop
866         endif
867     ! ------------------------------------------------------------------     .
868       elseif (n2.eq.2) then                                                ! oui
869         do k=1,n
870           if (datatps(j)%sta%staname.eq.statps(k)%staname) n2 = k
871         enddo
```

```
872         if (datatps(j)%typeonde.eq."N") then
873           iPn = iPn +1
874           statps(n2)%TPn = datatps(j)%dTP
875           statps(n2)%pdsPn = datatps(j)%wp
876           a_time%date=datatps(j)%tpsR%date
877           a_time%sec=datatps(j)%tpsR%secP
878           call basetime(a_time)
879           call difftime(statps(n2)%TpsPn,a_time,tps_ref)
880           if (datatps(j)%andS.eq."S") then
881             iSn = iSn +1
882             statps(n2)%TSn = datatps(j)%dTS
883             statps(n2)%pdsSn = datatps(j)%ws
884             a_time%date=datatps(j)%tpsR%date
885             a_time%sec=datatps(j)%tpsR%secS
886             call basetime(a_time)
887             call difftime(statps(n2)%TpsSn,a_time,tps_ref)
888           endif
889         elseif (datatps(j)%typeonde.eq."G") then
890           iPg = iPg +1
891           statps(n2)%TPg = datatps(j)%dTP
892           statps(n2)%pdsPg = datatps(j)%wp
893           a_time%date=datatps(j)%tpsR%date
894           a_time%sec=datatps(j)%tpsR%secP
895           call basetime(a_time)
896           call difftime(statps(n2)%TpsPg,a_time,tps_ref)
897           if (datatps(j)%andS.eq."S") then
898             iSg = iSg +1
899             statps(n2)%TSg = datatps(j)%dTS
900             statps(n2)%pdsSg = datatps(j)%ws
901             a_time%date=datatps(j)%tpsR%date
902             a_time%sec=datatps(j)%tpsR%secS
903             call basetime(a_time)
904             call difftime(statps(n2)%TpsSg,a_time,tps_ref)
905           endif
906         else
907           write(*,*)'problème dans GMT_res 2 : onde ni réfractée ni directe'
908           stop
909         endif
910       else                                                              ! problème
911         write(*,*)'problème dans GMT_res 2 : station présentes 3 fois'
912         stop
913       endif
914     enddo donnees
915     ! ------------------------------------------------------------      .
916     amax=-1.0_wr
917     do i=1,n
918       statps(i)%Ttot = abs(statps(i)%TPg) + abs(statps(i)%TPn) + &
919       abs(statps(i)%TSg) + abs(statps(i)%TSn)
920       if (amax.lt.statps(i)%TPg) amax = statps(i)%TPg
921       if (amax.lt.statps(i)%TSg) amax = statps(i)%TSg
922       if (amax.lt.statps(i)%TPn) amax = statps(i)%TPn
923       if (amax.lt.statps(i)%TSn) amax = statps(i)%TSn
924     enddo
925     val= max(real(int(real(int(amax*10._wr),wr)/10._wr+1._wr),wr),1.1_wr)  ! bornes du graph
926     ! ------------------------------------------------------------      .
927     open(950, FILE ='OUTPUT/residus'//"-"//trim(adjustl(numberfile))//'.d',status='replace')
928     do i=1,n
929
930         if(statps(i)%TPg.ne.0.0_wr) write(950,1001)"PG ",statps(i)%staname,statps(i)%lonSTA, &
931           statps(i)%latSTA,statps(i)%TPg,statps(i)%pdsPg,abs(statps(i)%TPg*100._wr/statps(i)%TpsPg), &
932           statps(i)%disthypo
933         if(statps(i)%TPn.ne.0.0_wr) write(950,1001)"PN ",statps(i)%staname,statps(i)%lonSTA, &
934           statps(i)%latSTA,statps(i)%TPn,statps(i)%pdsPn,abs(statps(i)%TPn*100._wr/statps(i)%TpsPn), &
935           statps(i)%disthypo
936         if(statps(i)%TSg.ne.0.0_wr) write(950,1001)"SG ",statps(i)%staname,statps(i)%lonSTA, &
```

```
          statps(i)%latSTA,statps(i)%TSg,statps(i)%pdsSg,abs(statps(i)%TSg*100._wr/statps(i)%TpsSg), &
          statps(i)%disthypo
        if(statps(i)%TSn.ne.0.0_wr) write(950,1001)"SN ",statps(i)%staname,statps(i)%lonSTA, &
          statps(i)%latSTA,statps(i)%TSn,statps(i)%pdsSn,abs(statps(i)%TSn*100._wr/statps(i)%TpsSn), &
          statps(i)%disthypo
      enddo
      1001 format (a,a,f10.4,f10.4,f10.4,f10.4,1x,f15.10,1x,f15.10)
      close(950)
      ! ------------------------------------------------------------------      .
      ! ------------------------------------------------------------------      .
      write(*,*)"ecriture des script GMT_res "
      write(600,*)"BEFORE=$SECONDS"
      call system_clock(Noldtime)
      write(600,*)"#######################################"
      write(600,*)"#######################################"
      write(600,*)
      write(600,*)"echo 'execution du script GMT res'"
      write(600,*)"###########################################"
      write(600,*)"###########      histo        ##########"
      write(600,*)"###########################################"
      ! ------------------------------------------------------------------      .
      write(600,'(a10,E13.7,a1,E13.7,a6)')"geozone=-R",-val,"/",val,"/0/100"
      write(600,*)"geoproj=-JX4.25i"
      write(600,*)"file=OUTPUT/GMT/reshisto"//"-"//trim(adjustl(numberfile))//".ps"
      ok=0
      ! -------                                                  --------      .
      open(955, FILE = "OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot",status='replace',iostat = ok)
      do i=1,n
        if(statps(i)%TPg.ne.0.0_wr) write(955,*)statps(i)%TPg
        if(statps(i)%TPn.ne.0.0_wr) write(955,*)statps(i)%TPn
        if(statps(i)%TSg.ne.0.0_wr) write(955,*)statps(i)%TSg
        if(statps(i)%TSn.ne.0.0_wr) write(955,*)statps(i)%TSn
      enddo
      close(955)
      ! -------                                                  --------      .
      open(951, FILE = "OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".pg",status='replace',iostat = ok)
      j=0
      do i=1,n
        if(statps(i)%TPg.ne.0.0_wr) then
          write(951,*)statps(i)%TPg
          j=j+1
        endif
      enddo
      close(951)

      if(j.gt.0) then
        write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0.0 -Z1 -K OUTPUT/GMT/", &
        "residus"//"-"//trim(adjustl(numberfile))//".pg -X2.5i > $file "
        write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
        "'r\351sidus ondes compressives directes':/a10:'effectif (%)':nSeW ", &
        " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O >> $file "
      else
        write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
        "'r\351sidus ondes compressives directes':/a10:'effectif (%)':nSeW ", &
        " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -X2.5i > $file "
      endif
      X = -val + 0.3_wr*(2.0_wr*val)
      Y = 95.0_wr
      if(j.gt.0) write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$pp -L0/0 -Ba0 -Z1 -O -K", &
      " OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".pg >> $file "
      write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iPg,""" | pstext $geozone $geoproj -O -K >> $file"
      ! -------                                                  --------      .
      open(952, FILE = "OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".pn",status='replace',iostat = ok)
      j=0
      do i=1,n
```

```
      if(statps(i)%TPn.ne.0.0_wr) then
        write(952,*)statps(i)%TPn
        j=j+1
      endif
    enddo
    close(952)
    if(j.gt.0) then
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0.0 -Z1 -K OUTPUT/GMT/", &
      "residus"//"-"//trim(adjustl(numberfile))//".pn -O -X5.5i >> $file "
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
      "'r\351sidus ondes compressives r\351fract\351es':/a10:'effectif (%)':nSew ", &
      " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O >> $file "
    else
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
      "'r\351sidus ondes compressives r\351fract\351es':/a10:'effectif (%)':nSew ", &
      " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O -X5.5i >> $file "
    endif
    if(j.gt.0) write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$pp -L0/0 -Ba0 -Z1 -O -K", &
    " OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".pn >> $file "
    write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iPn,""" | pstext $geozone $geoproj -O -K >> $file"
    ! ------                                                    -------                .
    open(953, FILE = "OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".sg",status='replace',iostat = ok)
    j=0
    do i=1,n
      if(statps(i)%TSg.ne.0.0_wr) then
        write(953,*)statps(i)%TSg
        j=j+1
      endif
    enddo
    close(953)

    if(j.gt.0) then
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0.0 -Z1 -K OUTPUT/GMT/", &
      "residus"//"-"//trim(adjustl(numberfile))//".sg -O -Y5.5i >> $file "
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
      "'r\351sidus ondes cisaillantes directes':/a10:'effectif (%)':nSew", &
      " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O >> $file "
    else
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
      "'r\351sidus ondes cisaillantes directes':/a10:'effectif (%)':nSew", &
      " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O -Y5.5i >> $file "
    endif
    if(j.gt.0) write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$ss -L0/0 -Ba0 -Z1 -O -K", &
    " OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".sg >> $file "
    write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iSg,""" | pstext $geozone $geoproj -O -K >> $file"
    ! ------                                                    -------                .
    open(954, FILE = "OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".sn",status='replace',iostat = ok)
    j=0
      do i=1,n
    if(statps(i)%TSn.ne.0.0_wr) then
      write(954,*)statps(i)%TSn
      j=j+1
    endif
    enddo
    close(954)
    if(j.gt.0) then
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G225 -Q -Ba0.0 -Z1 -K OUTPUT/GMT/", &
      "residus"//"-"//trim(adjustl(numberfile))//".sn -O -X-5.5i  >> $file "
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
      "'r\351sidus ondes cisaillantes r\351fract\351es':/a10:'effectif (%)':nSeW ", &
      " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O >> $file "
    else
      write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -Ggray -L0/0 -Bpa1.0g10/a0 -Bsf0.1:", &
      "'r\351sidus ondes cisaillantes r\351fract\351es':/a10:'effectif (%)':nSeW ", &
      " -Z1 -K OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".tot -O -X-5.5i >> $file "
```

```
    endif
    if(j.gt.0) write(600,*)"pshistogram -F $geozone $geoproj -W0.05 -G$ss -L0/0 -Ba0 -Z1 -O -K", &
    " OUTPUT/GMT/residus"//"-"//trim(adjustl(numberfile))//".sn  >> $file "
    write(600,*)"echo """,X,Y," 15 0 5 6 donn\351es : ",iSn,""" | pstext $geozone $geoproj -O >> $file"
    ! -------                                                    --------    .
    write(600,*)"ps2raster OUTPUT/GMT/reshisto"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/reshisto"//"-"//trim(adjustl(numberfile))//".pdf ", &
    " OUTPUT/figures/reshisto"//"-"//trim(adjustl(numberfile))//".pdf"
    ! ------------------------------------------------------------    .
    ! ------------------------------------------------------------    .
    write(600,*)"gmtset BASEMAP_TYPE plain"
    write(600,*)"grdfile=SRC/FILES/bath1.bin"
    write(600,*)"bluef=""0/0/100"" "
    ! -------                                                   --------    .
    v1 = 2.0_wr * pi * rT / 360.0_wr                                        ! km / degree en lon
    v2 = 2.0_wr * pi * rT * sin((90.0_wr-dp%lat(l)%vec10000(1,1))/180.0_wr*pi) /360.0_wr ! km / degree en lat
    ! -------                                                   --------    .
    lon1 = dp%lon(l)%vec10000(1,1) - (xmax / v2 * .99_wr) / 2.0_wr
    lon2 = dp%lon(l)%vec10000(1,1) + (xmax / v2 * .99_wr) / 2.0_wr
    lat1 = dp%lat(l)%vec10000(1,1) - (xmax / v1 * .99_wr) / 2.0_wr
    lat2 = dp%lat(l)%vec10000(1,1) + (xmax / v1 * .99_wr) / 2.0_wr
    if (lon1.lt.-6.0_wr) lon1=-6.0_wr
    if (lat2.gt.52.0_wr) lat2=52.0_wr
    ! -------                                                   --------    .
    write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",lon1,"/",lon2,"/",lat1,"/",lat2
    write(600,'(a12,E13.7,a1,E13.7,a1,E13.7,a1,E13.7,a3,E13.7)')"geozone3d=-R",lon1,"/",lon2,"/",lat1,"/",lat2,"/0/",val
    write(600,'(a11,E13.7,a1,E13.7,a3)')"geoproj=-JC",dp%lon(l)%vec10000(1,1),"/",dp%lat(l)%vec10000(1,1),"/20.i"
    ! ------------------------------------------------------------    .
    ! pour Pg
    ! ------------------------------------------------------------    .
    write(600,*)"file=OUTPUT/GMT/resPg"//"-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
    " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
    ! -------                                                   --------    .
    write(600,*)"######### affiche les stations ##########"
    ok = 0                                                                  ! toutes les stations
    open(960, FILE ='DATA/sta.d',status='old',iostat = ok)
    if (ok .ne. 0) then
      write(*,*)'problème dans GMT_map : le fichier data/sta.d n''existe pas'
      stop
    endif
    do while(ok .eq. 0)
      read(960,*,iostat = ok)datasta
      if ((datasta%lon.gt.lon1).and.(datasta%lon.lt.lon2).and.(datasta%lat.gt.lat1).and.(datasta%lat.lt.lat2)) then
        write(600,*)" echo ",datasta%lon,datasta%lat, &
        " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Ggrey -Lk -Wthinnest -O -K -E200/50 >> $file"
      endif
    end do
    close(960)
    ! -------                                                   --------    .
    do i=1,nbtps                                                            ! affiche les stations utilisées
      write(600,*)" echo ",datatps(i)%sta%lon,datatps(i)%sta%lat, &
      " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Gblue -Lk -Wthinnest -O -K -E200/50 >> $file"
    enddo
    ! -------                                                   --------    .
    write(600,*)"###### Limites du massif Armoricain #####"
    write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
    ! -------                                                   --------    .
    write(600,*)"################ Epicentre ################"
    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1)," \"
    write(600,*)" 0.0 | psxyz $geozone3d $geoproj -JZ2.5i -L -K -O -Wthinnest -Ggreen -Sa0.5i -E200/50 >> $file"
    ! -------                                                   --------    .
    write(600,*)"################  barres  ################"
    do i=1,n
      size = 0.05_wr + 0.15_wr*statps(i)%pdsPg
```

```
      val = abs(statps(i)%TPg)
      write(sizename,'(E13.7)')size
      if (statps(i)%TPg.gt.0.0_wr) then
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -G$pp -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      else
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -Gblue -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      endif
    enddo
    ! -------                                                    --------    .
    write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
    ! -------- légende :                                         --------    .
    write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
    " 20 0 4 LM ondes compressives directes | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
    ! -------                                                    --------    . fin
    write(600,*)"ps2raster OUTPUT/GMT/resPg"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/resPg"//"-"//trim(adjustl(numberfile))//".pdf ", &
    "OUTPUT/figures/resPg"//"-"//trim(adjustl(numberfile))//".pdf"
    ! ------------------------------------------------------------    .
    ! pour Pn :
    ! ------------------------------------------------------------    .
    write(600,*)"file=OUTPUT/GMT/resPn"//"-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
    " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
    ! -------                                                    --------    .
    write(600,*)"######### affiche les stations ##########"
    ok = 0                                                                 ! toutes les stations
    open(961, FILE ='DATA/sta.d',status='old',iostat = ok)
    if (ok .ne. 0) then
      write(*,*)'problème dans GMT_map : le fichier data/sta.d n''existe pas'
      stop
    endif
    do while(ok .eq. 0)
      read(961,*,iostat = ok)datasta
      if ((datasta%lon.gt.lon1).and.(datasta%lon.lt.lon2).and.(datasta%lat.gt.lat1).and.(datasta%lat.lt.lat2)) then
        write(600,*)" echo ",datasta%lon,datasta%lat, &
        " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Ggrey -Lk -Wthinnest -O -K -E200/50 >> $file"
      endif
    end do
    close(961)
    ! -------                                                    --------    .
    do i=1,nbtps                                                           ! affiche les stations utilisées
      write(600,*)" echo ",datatps(i)%sta%lon,datatps(i)%sta%lat, &
      " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Gblue -Lk -Wthinnest -O -K -E200/50 >> $file"
    enddo
    ! -------                                                    --------    .
    write(600,*)"###### Limites du massif Armoricain #####"
    write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
    ! -------                                                    --------    .
    write(600,*)"################ Epicentre ################"
    write(600,*)"echo ",dp%lon(1)%vec10000(1,1),dp%lat(1)%vec10000(1,1)," \"
    write(600,*)" 0.0 | psxyz $geozone3d $geoproj -JZ2.5i -L -K -O -Wthinnest -Ggreen -Sa0.5i -E200/50 >> $file"
    ! -------                                                    --------    .
    write(600,*)"################  barres  ################"
    do i=1,n
      size = 0.05_wr + 0.15_wr*statps(i)%pdsPn
      val = abs(statps(i)%TPn)
      write(sizename,'(E13.7)')size
      if (statps(i)%TPn.gt.0.0_wr) then
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -G$pp -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      else
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -Gblue -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
```

```
      endif
    enddo
    ! -------                                                    --------    .
    write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
    ! -------- légende :                                        --------    .
    write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
    " 20 0 4 LM 'ondes compressives r\351fract\351es' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
    ! -------                                                    --------    . fin
    write(600,*)"ps2raster OUTPUT/GMT/resPn"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/resPn"//"-"//trim(adjustl(numberfile))//".pdf ", &
    "OUTPUT/figures/resPn"//"-"//trim(adjustl(numberfile))//".pdf"
    ! ------------------------------------------------------------    .
    ! pour Sg
    ! ------------------------------------------------------------    .
    write(600,*)"file=OUTPUT/GMT/resSg"//"-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
    " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
    ! -------                                                    --------    .
    write(600,*)"######### affiche les stations ##########"
    ok = 0                                                                 ! toutes les stations
    open(962, FILE ='DATA/sta.d',status='old',iostat = ok)
    if (ok .ne. 0) then
      write(*,*)'problème dans GMT_map : le fichier data/sta.d n''existe pas'
      stop
    endif
    do while(ok .eq. 0)
      read(962,*,iostat = ok)datasta
      if ((datasta%lon.gt.lon1).and.(datasta%lon.lt.lon2).and.(datasta%lat.gt.lat1).and.(datasta%lat.lt.lat2)) then
        write(600,*)" echo ",datasta%lon,datasta%lat, &
        " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Ggrey -Lk -Wthinnest -O -K -E200/50 >> $file"
      endif
    end do
    close(962)
    ! -------                                                    --------    .
    do i=1,nbtps                                                           ! affiche les stations utilisées
      write(600,*)" echo ",datatps(i)%sta%lon,datatps(i)%sta%lat, &
      " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Gblue -Lk -Wthinnest -O -K -E200/50 >> $file"
    enddo
    ! -------                                                    --------    .
    write(600,*)"###### Limites du massif Armoricain #####"
    write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
    ! -------                                                    --------    .
    write(600,*)"################ Epicentre ################"
    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1)," \"
    write(600,*)" 0.0 | psxyz $geozone3d $geoproj -JZ2.5i -L -K -O -Wthinnest -Ggreen -Sa0.5i -E200/50 >> $file"
    ! -------                                                    --------    .
    write(600,*)"################  barres  ################"
    do i=1,n
      size = 0.05_wr + 0.15_wr*statps(i)%pdsSg
      val = abs(statps(i)%TSg)
      write(sizename,'(E13.7)')size
      if (statps(i)%TSg.gt.0.0_wr) then
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -G$ss -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      else
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -Gred -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      endif
    enddo
    ! -------                                                    --------    .
    write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
    ! -------- légende :                                        --------    .
    write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
    " 20 0 4 LM ondes cisaillantes directes | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
    ! -------                                                    --------    . fin
```

```
    write(600,*)"ps2raster OUTPUT/GMT/resSg"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/resSg"//"-"//trim(adjustl(numberfile))//".pdf ", &
    " OUTPUT/figures/resSg"//"-"//trim(adjustl(numberfile))//".pdf"
    ! ------------------------------------------------------------------ .
    ! pour Sn :
    ! ------------------------------------------------------------------ .
    write(600,*)"file=OUTPUT/GMT/resSn"//"-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
    " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
    ! -------                                                 -------- .
    write(600,*)"######### affiche les stations ##########"
    ok = 0                                                            ! toutes les stations
    open(963, FILE ='DATA/sta.d',status='old',iostat = ok)
    if (ok .ne. 0) then
      write(*,*)'problème dans GMT_map : le fichier data/sta.d n''existe pas'
      stop
    endif
    do while(ok .eq. 0)
      read(963,*,iostat = ok)datasta
      if ((datasta%lon.gt.lon1).and.(datasta%lon.lt.lon2).and.(datasta%lat.gt.lat1).and.(datasta%lat.lt.lat2)) then
        write(600,*)" echo ",datasta%lon,datasta%lat, &
        " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Ggrey -Lk -Wthinnest -O -K -E200/50 >> $file"
      endif
    end do
    close(963)
    ! -------                                                 -------- .
    do i=1,nbtps                                                      ! affiche les stations utilisées
      write(600,*)" echo ",datatps(i)%sta%lon,datatps(i)%sta%lat, &
      " 0.0 | psxyz $geoproj $geozone3d -JZ2.5i -St0.1i -Gblue -Lk -Wthinnest -O -K -E200/50 >> $file"
    enddo
    ! -------                                                 -------- .
    write(600,*)"###### Limites du massif Armoricain #####"
    write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
    ! -------                                                 -------- .
    write(600,*)"################ Epicentre ################"
    write(600,*)"echo ",dp%lon(l)%vec10000(1,1),dp%lat(l)%vec10000(1,1)," \"
    write(600,*)" 0.0 | psxyz $geozone3d $geoproj -JZ2.5i -L -K -O -Wthinnest -Ggreen -Sa0.5i -E200/50 >> $file"
    ! -------                                                 -------- .
    write(600,*)"################  barres  ################"
    do i=1,n
      size = 0.05_wr + 0.15_wr*statps(i)%pdsSn
      val = abs(statps(i)%TSn)
      write(sizename,'(E13.7)')size
      if (statps(i)%TSn.gt.0.0_wr) then
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -G$ss -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      else
        if (val.ne.0.0_wr) write(600,*)"echo ",statps(i)%lonSTA,statps(i)%latSTA," ",val, &
        " | psxyz $geoproj $geozone3d -JZ2.5i -So"//sizename//"ib0.0 -Gred -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
      endif
    enddo
    ! -------                                                 -------- .
    write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
    ! ------- légende :                                      -------- .
    write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
    " 20 0 4 LM 'ondes cisaillantes r\351fract\351es' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
    ! -------                                                 -------- . fin
    write(600,*)"ps2raster OUTPUT/GMT/resSn"//"-"//trim(adjustl(numberfile))//".ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/resSn"//"-"//trim(adjustl(numberfile))//".pdf ", &
    "OUTPUT/figures/resSn"//"-"//trim(adjustl(numberfile))//".pdf"
    ! ------------------------------------------------------------------ .
    deallocate(statps)
    ! ------------------------------------------------------------------ .
    ! ------------------------------------------------------------------ .
    if ((FLAGresSTA).and.(l==nbseismes)) then
```

```
      write(600,'(a12,E13.7,a1,E13.7,a1,E13.7,a1,E13.7,a)')"geozone3d=-R",lon1,"/",lon2,"/",lat1,"/",lat2,"/0/3.0"
      nbsta=0
      ok=0
      open(959, FILE ='OUTPUT/GMT/sta_RES_TOT.txt',status='old',iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans GMT_res : le fichier OUTPUT/GMT/sta_RES_TOT.txt n''existe pas'
        stop
      endif
      do while(ok .eq. 0)                                                ! boucle pour compter le nombre de lignes du fichier
        read(959,*,iostat = ok)
        if (ok .eq. 0) nbsta = nbsta + 1
      end do
      rewind(959)
      ok=0
      ! ------------------------------------------------------------     .
      ! Pg
      ! ------------------------------------------------------------     . MOY
      write(600,*)"echo 'execution du script GMT res Pg - TOT'"
      write(600,*)"file=OUTPUT/GMT/resTOTPgmoy.ps"
      write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
        " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
      write(600,*)"###### Limites du massif Armoricain #####"
      write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
      ! -------                                                 --------     .
      write(600,*)"#################  barres  ################"
      do i=1,nbsta
        read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
                                      moy(2),ec(2),med(2),gauss(2), &
                                      moy(3),ec(3),med(3),gauss(3), &
                                      moy(4),ec(4),med(4),gauss(4)

        val = abs(moy(1))
        if (gauss(1).ne.0.0_wr) then
          if (moy(1).gt.0.0_wr) then
            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
              " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(1), &
              "ib0.0 -G$pp -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
            if ((moy(1)-ec(1)).gt.0.0_wr) then

              write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val-ec(1), &
                " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(1), &
                "ib0.0 -G$pp -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
            endif
            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val+ec(1), &
              " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(1), &
              "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
          else
            if (moy(1).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
              lon,lat," ",val," | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(1), &
              "ib0.0 -Gblue -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
            if ((moy(1)-ec(1)).gt.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
              lon,lat," ",val-ec(1)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(1), &
              "ib0.0 -Gblue -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
            if (moy(1).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
              lon,lat," ",val+ec(1)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(1), &
              "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
          endif
        endif
      enddo
      ! -------                                                 --------     .
      write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
      ! -------- légende :                                     --------     .
      write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
      " 20 0 4 LM 'ondes compressives directes' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
      ! -------                                                 --------     . fin
```

```
1392        write(600,*)"ps2raster OUTPUT/GMT/resTOTPgmoy.ps -Tf -A"
1393        write(600,'(2a)')"mv OUTPUT/GMT/resTOTPgmoy.pdf OUTPUT/figures/resTOTPgmoy.pdf"
1394        ! -------                                                    ---------     .
1395        rewind(959)
1396        ! -------                                                    ---------     . MODE
1397        write(600,*)"file=OUTPUT/GMT/resTOTPgmed.ps"
1398        write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
1399          " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
1400        write(600,*)"###### Limites du massif Armoricain #####"
1401        write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
1402        ! -------                                                    ---------     .
1403        write(600,*)"################  barres  ###############"
1404        do i=1,nbsta
1405          read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
1406                                        moy(2),ec(2),med(2),gauss(2), &
1407                                        moy(3),ec(3),med(3),gauss(3), &
1408                                        moy(4),ec(4),med(4),gauss(4)
1409          val = abs(med(1))
1410          if (gauss(1).ne.0.0_wr) then
1411            if (med(1).gt.0.0_wr) then
1412              write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
1413                " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(1), &
1414                "ib0.0 -G$pp -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
1415            else
1416              if (med(1).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
1417              " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(1), &
1418              "ib0.0 -Gblue -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
1419            endif
1420          endif
1421        enddo
1422        ! -------                                                    ---------     .
1423        write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
1424        ! -------- légende :                                        ---------     .
1425        write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
1426          " 20 0 4 LM 'ondes compressives directes' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
1427        ! -------                                                    ---------     . fin
1428        write(600,*)"ps2raster OUTPUT/GMT/resTOTPgmed.ps -Tf -A"
1429        write(600,'(2a)')"mv OUTPUT/GMT/resTOTPgmed.pdf OUTPUT/figures/resTOTPgmed.pdf"
1430        ! -------                                                    ---------     .
1431        rewind(959)
1432        ! ---------------------------------------------------------------     .
1433        ! Sg
1434        ! ---------------------------------------------------------------     . MOY
1435        write(600,*)"echo 'execution du script GMT res Sg - TOT'"
1436        write(600,*)"file=OUTPUT/GMT/resTOTSgmoy.ps"
1437        write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
1438          " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
1439        write(600,*)"###### Limites du massif Armoricain #####"
1440        write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
1441        ! -------                                                    ---------     .
1442        write(600,*)"################  barres  ###############"
1443        do i=1,nbsta
1444          read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
1445                                        moy(2),ec(2),med(2),gauss(2), &
1446                                        moy(3),ec(3),med(3),gauss(3), &
1447                                        moy(4),ec(4),med(4),gauss(4)
1448          val = abs(moy(3))
1449          if (gauss(3).ne.0.0_wr) then
1450            if (moy(3).gt.0.0_wr) then
1451              write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
1452                " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(3), &
1453                "ib0.0 -G$ss -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
1454              if ((moy(3)-ec(3)).gt.0.0_wr) then
1455                write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val-ec(3), &
1456                  " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(3), &
```

```
1457                "ib0.0 -G$ss -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
1458            endif
1459            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val+ec(3), &
1460              " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(3), &
1461              "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
1462          else
1463            if (moy(3).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
1464              lon,lat," ",val," | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(3), &
1465              "ib0.0 -Gred -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
1466            if ((moy(3)-ec(3)).gt.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
1467              lon,lat," ",val-ec(3)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(3), &
1468              "ib0.0 -Gred -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
1469            if (moy(3).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
1470              lon,lat," ",val+ec(3)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(3), &
1471              "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
1472            endif
1473          endif
1474        enddo
1475        ! ------                                                  --------    .
1476        write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
1477        ! ------- légende :                                      --------    .
1478        write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
1479          " 20 0 4 LM 'ondes cisaillantes directes' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
1480        ! ------                                                  --------    . fin
1481        write(600,*)"ps2raster OUTPUT/GMT/resTOTSgmoy.ps -Tf -A"
1482        write(600,'(2a)')"mv OUTPUT/GMT/resTOTSgmoy.pdf OUTPUT/figures/resTOTSgmoy.pdf"
1483        ! ------                                                  --------    .
1484        rewind(959)
1485        ! ------                                                  --------    . MODE
1486        write(600,*)"file=OUTPUT/GMT/resTOTSgmed.ps"
1487        write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
1488        " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
1489        write(600,*)"###### Limites du massif Armoricain #####"
1490        write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
1491        ! ------                                                  --------    .
1492        write(600,*)"#################  barres  ################"
1493        do i=1,nbsta
1494          read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
1495                                        moy(2),ec(2),med(2),gauss(2), &
1496                                        moy(3),ec(3),med(3),gauss(3), &
1497                                        moy(4),ec(4),med(4),gauss(4)
1498          val = abs(med(3))
1499          if (gauss(3).ne.0.0_wr) then
1500            if (med(3).gt.0.0_wr) then
1501              write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
1502                " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(3), &
1503                "ib0.0 -G$ss -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
1504            else
1505              if (med(3).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
1506                " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(3), &
1507                "ib0.0 -Gred -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
1508            endif
1509          endif
1510        enddo
1511        ! ------                                                  --------    .
1512        write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
1513        ! ------- légende :                                      --------    .
1514        write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
1515        " 20 0 4 LM 'ondes cisaillantes directes' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
1516        ! ------                                                  --------    . fin
1517        write(600,*)"ps2raster OUTPUT/GMT/resTOTSgmed.ps -Tf -A"
1518        write(600,'(2a)')"mv OUTPUT/GMT/resTOTSgmed.pdf OUTPUT/figures/resTOTSgmed.pdf"
1519        ! ------                                                  --------    .
1520        rewind(959)
1521        ! ------------------------------------------------------------    .
```

```
      ! Pn
      ! ------------------------------------------------------------------    . MOY
      write(600,*)"echo 'execution du script GMT res Pn - TOT'"
      write(600,*)"file=OUTPUT/GMT/resTOTPnmoy.ps"
      write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
        " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
      write(600,*)"###### Limites du massif Armoricain #####"
      write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
      ! -------                                                  --------    .
      write(600,*)"################  barres  ################"
      do i=1,nbsta
        read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
                                      moy(2),ec(2),med(2),gauss(2), &
                                      moy(3),ec(3),med(3),gauss(3), &
                                      moy(4),ec(4),med(4),gauss(4)
        val = abs(moy(2))
        if (gauss(2).ne.0.0_wr) then
          if (moy(2).gt.0.0_wr) then
            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
              " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(2), &
              "ib0.0 -G$pp -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
            if ((moy(2)-ec(2)).gt.0.0_wr) then
              write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val-ec(2), &
                " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(2), &
                "ib0.0 -G$pp -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
            endif
            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val+ec(2), &
              " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(2), &
              "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
          else
            if (moy(2).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
              lon,lat," ",val," | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(2), &
              "ib0.0 -Gblue -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
            if ((moy(2)-ec(2)).gt.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
              lon,lat," ",val-ec(2)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(2), &
              "ib0.0 -Gblue -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
            if (moy(2).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
              lon,lat," ",val+ec(2)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(2), &
              "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
          endif
        endif
      enddo
      ! -------                                                  --------    .
      write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
      ! ------- légende :                                       --------    .
      write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
        " 20 0 4 LM 'ondes compressives r\351fract\351es' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
      ! -------                                                  --------    . fin
      write(600,*)"ps2raster OUTPUT/GMT/resTOTPnmoy.ps -Tf -A"
      write(600,'(2a)')"mv OUTPUT/GMT/resTOTPnmoy.pdf OUTPUT/figures/resTOTPnmoy.pdf"
      ! -------                                                  --------    .
      rewind(959)
      ! -------                                                  --------    . MODE
      write(600,*)"file=OUTPUT/GMT/resTOTPnmed.ps"
      write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
        " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
      write(600,*)"###### Limites du massif Armoricain #####"
      write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
      ! -------                                                  --------    .
      write(600,*)"################  barres  ################"
      do i=1,nbsta
        read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
                                      moy(2),ec(2),med(2),gauss(2), &
                                      moy(3),ec(3),med(3),gauss(3), &
                                      moy(4),ec(4),med(4),gauss(4)
```

```
      val = abs(med(2))
      if (gauss(2).ne.0.0_wr) then
        if (med(2).gt.0.0_wr) then
          write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
            " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(2), &
            "ib0.0 -G$pp -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
        else
          if (med(2).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
            " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(2), &
            "ib0.0 -Gblue -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
        endif
      endif
    enddo
    ! -------                                                    --------     .
    write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
    ! -------- légende :                                        --------     .
    write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
      " 20 0 4 LM 'ondes compressives r\351fract\351es' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
    ! -------                                                    --------     . fin
    write(600,*)"ps2raster OUTPUT/GMT/resTOTPnmed.ps -Tf -A"
    write(600,'(2a)')"mv OUTPUT/GMT/resTOTPnmed.pdf OUTPUT/figures/resTOTPnmed.pdf"
    ! -------                                                    --------     .
    rewind(959)
    ! ------------------------------------------------------------------     .
    ! Sn
    ! ------------------------------------------------------------------     . MOY
    write(600,*)"echo 'execution du script GMT res Sn - TOT'"
    write(600,*)"file=OUTPUT/GMT/resTOTSnmoy.ps"
    write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
      " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
    write(600,*)"###### Limites du massif Armoricain #####"
    write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
    ! -------                                                    --------     .
    write(600,*)"#################  barres  ################"
    do i=1,nbsta
      read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
                                    moy(2),ec(2),med(2),gauss(2), &
                                    moy(3),ec(3),med(3),gauss(3), &
                                    moy(4),ec(4),med(4),gauss(4)
      val = abs(moy(4))
      if (gauss(4).ne.0.0_wr) then
        if (moy(4).gt.0.0_wr) then
          write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
            " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(4), &
            "ib0.0 -G$ss -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
          if ((moy(4)-ec(4)).gt.0.0_wr) then
            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val-ec(4), &
              " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(4), &
              "ib0.0 -G$ss -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
          endif
          write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val+ec(4), &
            " | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(4), &
            "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
        else
          if (moy(4).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
            lon,lat," ",val," | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(4), &
            "ib0.0 -Gred -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
          if ((moy(4)-ec(4)).gt.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
            lon,lat," ",val-ec(4)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(4), &
            "ib0.0 -Gred -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
          if (moy(4).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ", &
            lon,lat," ",val+ec(4)," | psxyz $geoproj $geozone3d -JZ2.5i -SO",gauss(4), &
            "ib0.0 -Wthinner,black -O -K -Ba0 -E200/50 -N >> $file"
        endif
      endif
```

```
      enddo
      ! -------                                                    --------     .
      write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
      ! -------- légende :                                        --------     .
      write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
        " 20 0 4 LM 'ondes cisaillantes r\351fract\351es' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
      ! -------                                                    --------     . fin
      write(600,*)"ps2raster OUTPUT/GMT/resTOTSnmoy.ps -Tf -A"
      write(600,'(2a)')"mv OUTPUT/GMT/resTOTSnmoy.pdf OUTPUT/figures/resTOTSnmoy.pdf"
      ! -------                                                    --------     .
      rewind(959)
      ! -------                                                    --------     . MODE
      write(600,*)"file=OUTPUT/GMT/resTOTSnmed.ps"
      write(600,*)"pscoast $geozone3d $geoproj -JZ2.5i -S240/255/255 -G180/238/180 -N1 -Df+ -Ia/blue -W1 -K ", &
        " -E200/50 -Ba2f1g1/a2f1g1/a1:secondes:WSneZ -Xc -Yc >  $file"
      write(600,*)"###### Limites du massif Armoricain #####"
      write(600,*)"psxyz $geozone $geoproj -W4,gray -O SRC/FILES/limitesMA -K -E200/50 -M >> $file"
      ! -------                                                    --------     .
      write(600,*)"################  barres  ################"
      do i=1,nbsta
        read(959,*)aname,lat,lon,alti,moy(1),ec(1),med(1),gauss(1), &
                                      moy(2),ec(2),med(2),gauss(2), &
                                      moy(3),ec(3),med(3),gauss(3), &
                                      moy(4),ec(4),med(4),gauss(4)
        val = abs(med(4))
        if (gauss(4).ne.0.0_wr) then
          if (med(4).gt.0.0_wr) then
            write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
              " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(4), &
              "ib0.0 -G$ss -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
          else
            if (med(4).ne.0.0_wr) write(600,'(a,f13.5,1x,f13.5,a,f13.5,a,E13.7,a)')"echo ",lon,lat," ",val, &
              " | psxyz $geoproj $geozone3d -JZ2.5i -So",gauss(4), &
              "ib0.0 -Gred -Wthinner -O -K -Ba0 -E200/50 -N >> $file"
          endif
        endif
      enddo
      ! -------                                                    --------     .
      write(600,*)"psbasemap $geozone3d $geoproj -JZ2.5i -Ba0 -O -K -E200/50 >>  $file"
      ! -------- légende :                                        --------     .
      write(600,*)"echo ",lon1+(lon2-lon1)*0.05_wr,lat1+(lat2-lat1)*1.05_wr, &
        " 20 0 4 LM 'ondes cisaillantes r\351fract\351es' | pstext $geozone3d -JZ2.5i $geoproj -O -N -E200/50 >> $file"
      ! -------                                                    --------     . fin
      write(600,*)"ps2raster OUTPUT/GMT/resTOTSnmed.ps -Tf -A"
      write(600,'(2a)')"mv OUTPUT/GMT/resTOTSnmed.pdf OUTPUT/figures/resTOTSnmed.pdf"
      close(959)
    endif
    ! -------                                                    --------     .
    write(600,*)
    write(600,*)"#######################################"
    write(600,*)"#######################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
    int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------------     .
  end subroutine GMT_res

END MODULE figure_GMTres



! ********************************************************************     .
```

```
1717 ! ********************************************************************     .
```

2.10 SRC/MOD/MOD_GMT/mkcarriere.f90

```
1  ! permet la création des scripts GMT pour une carte de la région avec les carrières
2  ! octobre 2015
3  ! ********************************************************************     .
4  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
5  ! ********************************************************************     .
6  ! ------------------------------------------------------------------     .
7  ! merci à Pascal Guterman pascal.guterman@cnrs.fr pour les données          !
8  ! de carrieres (Travail de Frechet & Thouvenot 2012)                        !
9  ! adaptation en ForTran du module python/seiscomp de Pascal Guterman        !
10 ! ------------------------------------------------------------------     .
11
12 MODULE figure_GMTcarriere
13
14    use modparam
15    use typetemps, only : date_sec
16    use time
17    use distance_epi
18
19    implicit none
20
21    private
22
23    public  :: GMT_carriere
24
25  ! -------                                                  --------     .
26    TYPE catalogueSiHex
27         real(KIND=wr) :: sec,lat,lon,pfd,mag,annee,distcarriere,heure360
28         integer(KIND=wi) :: number,mm,jj,aaaa,hh,min,jday
29         character(LEN=8) :: orga
30         character(LEN=2) :: type
31    END TYPE catalogueSiHex
32  ! -------                                                  --------     .
33
34 CONTAINS
35
36 ! ------------------------------------------------------------------     .
37
38   subroutine GMT_carriere(l,seislon,seislat,seistps)
39     ! -------                                                --------     .mh
40     ! production du script GMT produisant une carte
41     ! et des diagrammes polaires en vue de discriminer les tirs de carrière
42     ! -------                                                --------     .
43     implicit none
44     integer(KIND=wi), intent (in) :: l
45     real(KIND=wr), intent (in) :: seislon,seislat                         ! lon lat du seisme
46     type(date_sec), intent (inout) :: seistps
47     ! -------                                                --------     .
48     type(catalogueSiHex) :: ev,refold,refnew
49     ! -------                                                --------     .
50     real(KIND=wr) :: v1, v2
51     real(KIND=wr) :: clon,clat,cdist
52     real(KIND=wr) :: tl
53     real(KIND=wr) :: lon1,lon2,lat1,lat2
54     integer(KIND=wi) :: i,ok,ok2,Noldtime, Nnewtime, ratetime
55     character (LEN=5) :: numberfile
56     logical :: existe1
57     ! ----------------------------------------------------------------     .
58     real(KIND=wr), parameter :: xmax=25.0_wr                              ! distance à l'épicentre
59     ! ----------------------------------------------------------------     .
60     write(*,*)"ecriture du script GMT_carrieres "
61     write(600,*)"BEFORE=$SECONDS"
```

```
62      call system_clock(Noldtime)
63      write(600,*)"#####################################"
64      write(600,*)"#####################################"
65      write(600,*)
66      write(600,*)"echo 'execution du script GMT carrieres'"
67      write(600,*)"#############################################"
68      write(600,*)"###########     carte GMT       ##########"
69      write(600,*)"#############################################"
70      write(numberfile(1:5),'(i5)')1
71      ! ------------------------------------------------------------------    .
72
73
74      inquire (FILE="DATA/files/catalogue_SiHEX_ke_FR.d",exist=existe1)
75      if (existe1) then
76        ! ------                                                 --------    . fichiers
77        ok = 0
78        open(95, FILE = 'DATA/files/carriere.d',status='old',iostat = ok)
79        if (ok .ne. 0) then
80          write(*,*)'problème de fichier dans GMT_carriere 1'
81          stop
82        endif
83        open(96, FILE = 'DATA/files/catalogue.d',status='old',iostat = ok)
84        if (ok .ne. 0) then
85          write(*,*)'problème de fichier dans GMT_carriere 1'
86          stop
87        endif
88        open(97, FILE = 'DATA/files/catalogue_non_tecto.d',status='old',iostat = ok)
89        if (ok .ne. 0) then
90          write(*,*)'problème de fichier dans GMT_carriere 2'
91        stop
92        endif
93        open(98, FILE = 'DATA/files/catalogue_SiHEX_all_Ma.d',status='old',iostat = ok)
94        if (ok .ne. 0) then
95          write(*,*)'problème de fichier dans GMT_carriere 1'
96          stop
97        endif
98        open(99, FILE = 'DATA/files/catalogue_SiHEX_ke_FR.d',status='old',iostat = ok)
99        if (ok .ne. 0) then
100         write(*,*)'problème de fichier dans GMT_carriere 2'
101         stop
102       endif
103       ! ------                                                 --------    .
104       open(101, FILE = "OUTPUT/GMT/cata_all-"//trim(adjustl(numberfile))//".d",status='replace',iostat = ok)
105       open(103, FILE = "OUTPUT/GMT/SiHEXP1-"//trim(adjustl(numberfile))//".d",status='replace',iostat = ok)
106       open(104, FILE = "OUTPUT/GMT/SiHEXP2-"//trim(adjustl(numberfile))//".d",status='replace',iostat = ok)
107       if (ok .ne. 0) then
108         write(*,*)'problème de fichier dans GMT_carriere'
109         stop
110       endif
111       ! ------                                                 --------    .
112       refold%aaaa=1962
113       refold%mm=1
114       refold%jj=1
115       call JDATE (refold%jday, refold%aaaa, refold%mm, refold%jj)
116
117       refnew%aaaa=2015
118       refnew%mm=1
119       refnew%jj=1
120       call JDATE (refnew%jday, refnew%aaaa, refnew%mm, refnew%jj)
121       ! ------------------------------------------------------------------    .
122       ok2=0
123       refold%distcarriere=9999.9_wr
124       do while(ok2.eq.0)
125         read(95,*,iostat=ok2)clon, clat
126         call dellipsgc(clat, clon, seislat, seislon, cdist)
```

```
      if (cdist.le.refold%distcarriere) refold%distcarriere=cdist
    end do
    ! ------------------------------------------------------------------     . catalogue_SiHEX_ke_FR.d
    ok=0
    do while(ok.eq.0)
      read(99,12345,iostat=ok)ev%number,ev%jj,ev%mm,ev%aaaa,ev%hh,ev%min,ev%sec,ev%lat,ev%lon,ev%pfd,ev%orga,ev%type,ev%mag
      ! -------                                                   ---------     .
      call JDATE (ev%jday,ev%aaaa,ev%mm,ev%jj)
      ev%heure360=real(ev%hh,wr)+real(ev%min,wr)/60._wr+ev%sec/3600._wr
      call polar2time(ev%heure360)
      ! -------                                                   ---------     .
      ok2=0
      ev%distcarriere=9999.9_wr
      clat=seislat
      clon=seislon
      call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
      if(cdist.lt.xmax) then
        rewind(95)
        do while(ok2.eq.0)
          read(95,*,iostat=ok2)clon,clat
          call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
          if (cdist.le.ev%distcarriere) ev%distcarriere=cdist
        end do
        ! -------                                                 ---------     .
        if ((ok.eq.0).and.(ev%distcarriere.le.xmax)) then
          write(101,*)ev%lon,ev%lat,1.0,'catalogue_SiHEX_ke_FR.d'         ! cata_all
          call JDATE (i,ev%aaaa,1,1)
          lat2=real(ev%aaaa,wr)+real(ev%jday-i,wr)/365.25_wr-real(refold%aaaa,wr)
          write(103,*)ev%heure360,lat2,1.0,'catalogue_SiHEX_ke_FR.d'      ! Plot polaire 1, SiHEXP1
          write(104,*)ev%heure360,ev%distcarriere,1.0,'catalogue_SiHEX_ke_FR.d' ! Plot polaire 2, SiHEXP2
        endif
      endif
      ! -------                                                   ---------     .
    end do
    ! ------------------------------------------------------------------     . catalogue_SiHEX_all_Ma.d
    ok=0
    do while(ok.eq.0)
      read(98,12346,iostat=ok)ev%number,ev%aaaa,ev%mm,ev%jj,ev%hh,ev%min,ev%sec,ev%lat,ev%lon,ev%pfd,ev%orga,ev%type,ev%mag
      ! -------                                                   ---------     .
      call JDATE (ev%jday,ev%aaaa,ev%mm,ev%jj)
      ev%heure360=real(ev%hh,wr)+real(ev%min,wr)/60._wr+ev%sec/3600._wr
      call polar2time(ev%heure360)
      ! -------                                                   ---------     .
      ok2=0
      ev%distcarriere=9999.9_wr
      clat=seislat
      clon=seislon
      call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
      if(cdist.lt.xmax) then
        rewind(95)
        do while(ok2.eq.0)
          read(95,*,iostat=ok2)clon,clat
          call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
          if (cdist.le.ev%distcarriere) ev%distcarriere=cdist
        end do
        ! -------                                                 ---------     .
        if ((ok.eq.0).and.(ev%distcarriere.le.xmax)) then
          write(101,*)ev%lon,ev%lat,-1.0,'catalogue_SiHEX_all_Ma.d'       ! cata_all
          call JDATE (i,ev%aaaa,1,1)
          lat2=real(ev%aaaa,wr)+real(ev%jday-i,wr)/365.25_wr-real(refold%aaaa,wr)
          write(103,*)ev%heure360,lat2,-1.0,'catalogue_SiHEX_all_Ma.d'    ! Plot polaire 1, SiHEXP1
          write(104,*)ev%heure360,ev%distcarriere,-1.0,'catalogue_SiHEX_all_Ma.d' ! Plot polaire 2, SiHEXP2
        endif
      endif
      ! -------                                                   ---------     .
```

```
192       end do
193       ! ------------------------------------------------------------------      . catalogue.d
194       ok=0
195       do while(ok.eq.0)
196         read(96,*,iostat=ok)ev%aaaa,ev%mm,ev%jj,ev%hh,ev%min,ev%sec,ev%lat,ev%lon,ev%mag,ev%pfd,ev%orga
197         ev%number=0
198         ev%type='ke'
199         ! --------                                                 ---------     .
200         call JDATE (ev%jday,ev%aaaa,ev%mm,ev%jj)
201         ev%heure360=real(ev%hh,wr)+real(ev%min,wr)/60._wr+ev%sec/3600._wr
202         call polar2time(ev%heure360)
203         ! --------                                                 ---------     .
204         ok2=0
205         ev%distcarriere=9999.9_wr
206         clat=seislat
207         clon=seislon
208         call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
209         if(cdist.lt.xmax) then
210           rewind(95)
211           do while(ok2.eq.0)
212             read(95,*,iostat=ok2)clon,clat
213             call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
214             if (cdist.le.ev%distcarriere) ev%distcarriere=cdist
215           end do
216           ! -------                                                 ---------     .
217           if ((ok.eq.0).and.(ev%distcarriere.le.xmax)) then
218             write(101,*)ev%lon,ev%lat,2.0,'catalogue.d'                          ! cata_all
219             call JDATE (i,ev%aaaa,1,1)
220             lat2=real(ev%aaaa,wr)+real(ev%jday-i,wr)/365.25_wr-real(refold%aaaa,wr)
221             write(103,*)ev%heure360,lat2,2.0,'catalogue.d'                       ! Plot polaire 1, SiHEXP1
222             write(104,*)ev%heure360,ev%distcarriere,2.0,'catalogue.d'            ! Plot polaire 2, SiHEXP2
223           endif
224         endif
225         ! --------                                                 ---------     .
226       end do
227       ! ------------------------------------------------------------------      . catalogue_non_tecto.d
228       ok=0
229       do while(ok.eq.0)
230         read(97,12347,iostat=ok)ev%jj,ev%mm,ev%aaaa,ev%hh,ev%min,i,ev%lat,ev%lon,ev%mag,ev%type
231         ev%sec=real(i,wr)
232         ev%pfd=0.0_wr
233         ev%orga='tir'
234         ev%number=0
235         ! --------                                                 ---------     .
236         call JDATE (ev%jday,ev%aaaa,ev%mm,ev%jj)
237         ev%heure360=real(ev%hh,wr)+real(ev%min,wr)/60._wr+ev%sec/3600._wr
238         call polar2time(ev%heure360)
239         ! --------                                                 ---------     .
240         ok2=0
241         ev%distcarriere=9999.9_wr
242         clat=seislat
243         clon=seislon
244         call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
245         if(cdist.lt.xmax) then
246           rewind(95)
247           do while(ok2.eq.0)
248             read(95,*,iostat=ok2)clon,clat
249             call dellipsgc(clat,clon,ev%lat,ev%lon,cdist)
250             if (cdist.le.ev%distcarriere) ev%distcarriere=cdist
251           end do
252           ! -------                                                 ---------     .
253           if ((ok.eq.0).and.(ev%distcarriere.le.xmax)) then
254             write(101,*)ev%lon,ev%lat,-2.0,'catalogue_non_tecto.d'               ! cata_all
255             call JDATE (i,ev%aaaa,1,1)
256             lat2=real(ev%aaaa,wr)+real(ev%jday-i,wr)/365.25_wr-real(refold%aaaa,wr)
```

```
257             write(103,*)ev%heure360,lat2,-2.0,'catalogue_non_tecto.d'      ! Plot polaire 1, SiHEXP1
258             write(104,*)ev%heure360,ev%distcarriere,-2.0,'catalogue_non_tecto.d' ! Plot polaire 2, SiHEXP2
259           endif
260         endif
261         ! -------                                                 --------     .
262       end do
263       ! ---------------------------------------------------------------------     .
264       ! ---------------------------------------------------------------------     .
265       close(95)
266       close(96)
267       close(97)
268       close(98)
269       close(99)
270       close(101)
271       close(103)
272       close(104)
273       ! -------                                                 --------     .
274       12345 format(2x,i6,4x,i2.2,1x,i2.2,1x,i4.4,1x,i2.2,1x,i2.2,1x,f4.1,2x,f7.2,4x,f7.2,8x,f4.1,4x,a8,a2,4x,f3.1) ! catalogue_SiHEX_ke_FR.d
275       12346 format(i6,1x,i4.4,1x,i2.2,1x,i2.2,1x,i2.2,1x,i2.2,1x,f4.1,1x,f8.5,f8.5,f6.2,a4,1x,a2,1x,f4.2) ! catalogue_SiHEX_all_Ma.d
276       12347 format(i2.2,1x,i2.2,1x,i4.4,1x,i2.2,1x,i2.2,1x,i2.2,1x,f10.7,1x,f10.7,1x,f10.7,1x,a2) ! catalogue_non_tecto.d
277       ! -------                                                 --------     .
278     endif
279     ! ---------------------------------------------------------------------     . Plot polaire 1 (heure / année)
280     write(600,*)"file=OUTPUT/GMT/carrieresP1-"//trim(adjustl(numberfile))//".ps"
281     write(600,'(a,i4.4)')"geozone=-R0/360/0/",refnew%aaaa+1-refold%aaaa
282     write(600,'(a)')"geoproj='-JP3i'"
283     write(600,'(2a)')"psxy $geozone $geoproj OUTPUT/GMT/SiHEXP1-"//trim(adjustl(numberfile))//".d ", &
284       " -Sc0.05i -Wthinnest -COUTPUT/GMT/colortir.cpt -Xc -Yc -Bpa45f7.5g15/a10f1g5wens -K > $file"
285     lat1=real(seistps%date%hour,wr)+real(seistps%date%min,wr)/60._wr+seistps%sec/3600._wr
286     call polar2time(lat1)
287     call JDATE (i,seistps%date%year,1,1)
288     call JDATE (seistps%date%Jday,seistps%date%year,seistps%date%month,seistps%date%day)
289     lat2=real(seistps%date%year,wr)+real(seistps%date%Jday-i,wr)/365.25_wr-real(refold%aaaa,wr)
290     ! -------                                                 --------     . cadran
291     cdist=0.0
292     call polar2time(cdist)
293     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 00h'", &
294       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
295     cdist=3.0
296     call polar2time(cdist)
297     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 03h'", &
298       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
299     cdist=6.0
300     call polar2time(cdist)
301     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 06h'", &
302       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
303     cdist=9.0
304     call polar2time(cdist)
305     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 09h'", &
306       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
307     cdist=12.0
308     call polar2time(cdist)
309     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 12h'", &
310       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
311     cdist=15.0
312     call polar2time(cdist)
313     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 15h'", &
314       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
315     cdist=18.0
316     call polar2time(cdist)
317     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 18h'", &
318       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
319     cdist=21.0
320     call polar2time(cdist)
321     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*1.16_wr," 12 0 1 CM 21h'", &
```

```
322       " | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
323     ! -------                                                 --------     .
324     cdist=19.5
325     call polar2time(cdist)
326     write(600,*)"echo '",cdist,0," 12 67.5 1 CM ",refold%aaaa," ' ", &
327     " | pstext $geoproj $geozone -G150/150/150 -O -K -N >> $file"
328     write(600,*)"echo '",cdist,real(refnew%aaaa-refold%aaaa,wr)*0.5_wr," 12 67.5 1 CM ", &
329     int(refold%aaaa+(refnew%aaaa-refold%aaaa)/2),"' | pstext $geoproj $geozone -G150/150/150  -O -K -N >> $file"
330     write(600,*)"echo '",cdist,refnew%aaaa-refold%aaaa," 12 67.5 1 CM ",refnew%aaaa, &
331     "' | pstext $geoproj $geozone -G150/150/150  -O -K -N >> $file"
332     ! -------                                                 --------     .
333     write(600,'(a,2(f10.5,1x),a)')"echo '",lat1,lat2, "'| psxy $geozone $geoproj -Sc0.05i -Wthinnest -Gblack -O >> $file"
334     write(600,*)"ps2raster OUTPUT/GMT/carrieresP1-"//trim(adjustl(numberfile))//".ps -Tf -A -P"
335     write(600,'(2a)')"mv OUTPUT/GMT/carrieresP1-"//trim(adjustl(numberfile))//".pdf ", &
336       "OUTPUT/figures/carrieresP1-"//trim(adjustl(numberfile))//".pdf"
337
338     ! ------------------------------------------------------------------     . Plot polaire 2 (heure / distance)
339     write(600,*)"file=OUTPUT/GMT/carrieresP2-"//trim(adjustl(numberfile))//".ps"
340     write(600,'(a,E13.7)')"geozone=-R0/360/0/",xmax
341     write(600,'(a)')"geoproj='-JP3i'"
342     write(600,'(2a)')"psxy $geozone $geoproj OUTPUT/GMT/SiHEXP2-"//trim(adjustl(numberfile))//".d ", &
343       " -Sc0.05i -Wthinnest -COUTPUT/GMT/colortir.cpt -Xc -Yc -Bpa45f7.5g15/a5f1g5wens -K > $file"
344     lat2=refold%distcarriere
345     ! -------                                                 --------     . cadran
346     cdist=0.0
347     call polar2time(cdist)
348     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 00h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
349     cdist=3.0
350     call polar2time(cdist)
351     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 03h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
352     cdist=6.0
353     call polar2time(cdist)
354     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 06h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
355     cdist=9.0
356     call polar2time(cdist)
357     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 09h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
358     cdist=12.0
359     call polar2time(cdist)
360     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 12h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
361     cdist=15.0
362     call polar2time(cdist)
363     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 15h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
364     cdist=18.0
365     call polar2time(cdist)
366     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 18h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
367     cdist=21.0
368     call polar2time(cdist)
369     write(600,*)"echo '",cdist,xmax*1.16," 12 0 1 CM 21h' | pstext $geoproj $geozone -O -K -N -Wwhite >> $file"
370     ! -------                                                 --------     .
371     cdist=19.5
372     call polar2time(cdist)
373     ! write(600,*)"echo '",cdist,0," 12 67.5 1 CM 0 km'| pstext $geoproj $geozone -O -K -N >> $file"
374     write(600,*)"echo '",cdist,xmax*0.5_wr," 12 67.5 1 CM ",int(xmax*0.5_wr), &
375       " km' | pstext $geoproj $geozone -G150/150/150 -O -K -N >> $file"
376     write(600,*)"echo '",cdist,xmax," 12 67.5 1 CM ",int(xmax), &
377       " km'| pstext $geoproj $geozone -G150/150/150 -O -K -N >> $file"
378     ! -------                                                 --------     .
379     write(600,'(a,2f10.5,a)')"echo '",lat1,lat2, "'| psxy $geozone $geoproj -Sc0.05i -Wthinnest -Gblack -O >> $file"
380     write(600,*)"ps2raster OUTPUT/GMT/carrieresP2-"//trim(adjustl(numberfile))//".ps -Tf -A -P"
381     write(600,'(2a)')"mv OUTPUT/GMT/carrieresP2-"//trim(adjustl(numberfile))//".pdf ", &
382     "OUTPUT/figures/carrieresP2-"//trim(adjustl(numberfile))//".pdf"
383
384     ! ------------------------------------------------------------------     . Plot carte
385     write(600,*)"file=OUTPUT/GMT/carrieres-"//trim(adjustl(numberfile))//".ps"
386     write(600,*)"gmtset BASEMAP_TYPE plain"
```

```
    ! -------                                                  --------     .
    v1 = 2.0_wr * pi * rT / 360.0_wr                                        ! km / degree en lon
    v2 = 2.0_wr * pi * rT * sin((90.0_wr-seislat)/180.0_wr*pi) /360.0_wr    ! km / degree en lat
    ! -------                                                  --------     .
    lon1 = seislon - (xmax / v2 * 1.1_wr)
    lon2 = seislon + (xmax / v2 * 1.1_wr)
    lat1 = seislat - (xmax / v1 * 1.1_wr)
    lat2 = seislat + (xmax / v1 * 1.1_wr)
    ! -------                                                  --------     .
    write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",lon1,"/",lon2,"/",lat1,"/",lat2
    write(600,'(a11,E13.7,a1,E13.7,a3)')"geoproj=-JC",seislon,"/",seislat,"/7i"
    ! -------                                                  --------     .
    write(600,*)"bluef=""0/0/100"" "
    write(600,'(2a)')"pscoast $geozone $geoproj -Df+ -S240/255/255 -G180/238/180 -Ia/$bluef -W1 ", &
        "-Xc -Yc -K -Bpa.1f.005/a.1f.005WeSn > $file"
    ! -------                                                  --------     . cercles et croix
    write(600,*)"######### cercles ########"
    do i=1,int(xmax)
      if (mod(i,5).ne.0) then
        write(600,*)"echo '",seislon,seislat,"0",i*2,i*2,"'| psxy $geozone $geoproj -SE -W1,- -O -K >> $file"
      else
        write(600,*)"echo '",seislon,seislat,"0",i*2,i*2,"'| psxy $geozone $geoproj -SE -W2 -O -K >> $file"
        write(600,*)"echo '",seislon+(real(2*i,wr)/v2)/2.0_wr,seislat, &
            "7 0 1 15 ",i,"km ' | pstext $geoproj $geozone -O -K >> $file"
      endif
    enddo
    ! -------                                                  --------     . croix
    write(600,*)"echo -e '",seislon+(real(2*int(xmax,4)+1,wr)/v2)/2.0_wr,seislat,"\n", &
      seislon-(real(2*int(xmax,4)+1,wr)/v2)/2.0_wr,seislat, &
      "' | psxy $geozone $geoproj -W2 -O -K >> $file"
    write(600,*)"echo -e '",seislon,seislat+(real(2*int(xmax,4)+1,wr)/v1)/2.0_wr,"\n", &
      seislon,seislat-(real(2*int(xmax,4)+1,wr)/v1)/2.0_wr, &
      "' | psxy $geozone $geoproj -W2 -O -K >> $file"
    ! -------                                                  --------     . carrières
    write(600,*)"psxy $geozone DATA/files/carriere.d $geoproj -St.05i -Wthinnest,purple -Gblue -O -K >> $file"
    ! -------                                                  --------     . sismicité tectonique
    write(600,'(2a)')"psxy $geozone OUTPUT/GMT/cata_all-"//trim(adjustl(numberfile))//".d ", &
      " $geoproj -Sc0.05i -COUTPUT/GMT/colortir.cpt -Wthinnest -O -K >> $file"
    ! -------                                                  --------     .
    write(600,*)"echo 'N 3' > OUTPUT/GMT/atirlegend.d"
    write(600,*)"echo 'S 0.i c 0.075i 000/000/000 0.25p 0.3i s\351isme' >> OUTPUT/GMT/atirlegend.d"
    write(600,*)"echo 'S 0.i c 0.075i 170/000/000 0.25p 0.3i me+km catalogue' >> OUTPUT/GMT/atirlegend.d"
    write(600,*)"echo 'S 0.i c 0.075i 255/028/000 0.25p 0.3i me+km Si-Hex' >> OUTPUT/GMT/atirlegend.d"
    write(600,*)"echo 'S 0.i c 0.075i 255/255/000 0.25p 0.3i ke Si-Hex' >> OUTPUT/GMT/atirlegend.d"
    write(600,*)"echo 'S 0.i c 0.075i 255/255/170 0.25p 0.3i ke catalogue' >> OUTPUT/GMT/atirlegend.d"
    write(600,*)"echo 'S 0.i t 0.075i 000/000/255 0.25p,purple 0.3i carri\350res' >> OUTPUT/GMT/atirlegend.d"
    ! -------                                                  --------     .
    write(600,*)"pslegend -Dx3.5i/-0.4i/7i/1.i/TC $geozone $geoproj -O -K OUTPUT/GMT/atirlegend.d >> $file"
    ! -------                                                  --------     .
    write(600,*)"echo '",seislon,seislat,"' | psxy $geozone $geoproj -Sc0.05i -Gblack -Wthinnest -O >> $file"
    ! -------                                                  --------     . fin
    write(600,*)"ps2raster OUTPUT/GMT/carrieres-"//trim(adjustl(numberfile))//".ps -Tf -A -P"
    write(600,'(2a)')"mv OUTPUT/GMT/carrieres-"//trim(adjustl(numberfile))//".pdf ", &
      "OUTPUT/figures/carrieres-"//trim(adjustl(numberfile))//".pdf"
    ! -------                                                  --------     .
    write(600,*)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
      int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------     .
```

```
452   end subroutine GMT_carriere
453
454   ! ------------------------------------------------------------------ .
455
456   subroutine polar2time(une_heure)
457     ! _______                                                  ________ .mh
458     ! transforme coordonnées orloge en polaire (sens inverse)
459     ! _______                                                  ________ .
460     implicit none
461     real(KIND=wr), intent (inout) :: une_heure
462     ! ------------------------------------------------------------------ .
463     une_heure=((une_heure*(-1.0_wr))+24.0_wr)
464     if (une_heure.ge.24.0_wr) une_heure=une_heure-24.0_wr
465     une_heure=une_heure/24.0_wr*360.0_wr+90.0_wr
466     ! ------------------------------------------------------------------ .
467   end subroutine polar2time
468
469   ! ------------------------------------------------------------------ .
470
471 END MODULE figure_GMTcarriere
472
473
474
475 ! ********************************************************************** .
476 ! ********************************************************************** .
```

2.11 SRC/MOD/MOD_GMT/mkwada.f90

```
1  ! permet la création des scripts GMT pour le diagramme de Wadati
2  ! mars 2014
3  ! ********************************************************************** .
4  ! _______ Méric Haugmard meric.haugmard@univ-nantes.fr          ________ .
5  ! ********************************************************************** .
6  ! ------------------------------------------------------------------ .
7
8  MODULE figure_GMTwada
9
10    use modparam
11
12    implicit none
13
14    private
15
16    public  :: GMT_wadati
17
18
19 CONTAINS
20
21 ! ------------------------------------------------------------------ .
22
23   subroutine GMT_wadati(nbtps,D,param,dp)
24     ! _______                                                  ________ .mh
25     ! production d'une partie du script GMT pour le Wadatiplot
26     ! _______                                                  ________ .
27     use typetemps
28     use cpt_temps
29     use time
30     use pb_direct
31     ! _______                                                  ________ .
32     implicit none
33     integer(KIND=wi), intent(in) :: nbtps(nbseismes)
34     type(dataall), intent(in) :: D(nbseismes)
35     type(parametres), intent(in) :: param
36     type(densityplot), intent (in) :: dp
37     ! _______                                                  ________ .
```

```
38      real (KIND=wr) :: aREF, R2REF                                          ! coeficient directeur de la régression
39      real (KIND=wr) :: aDIR, R2DIR                                          ! Chi2
40      real (KIND=wr) :: X, Y, tl
41      real (KIND=wr) :: Xmaxi, Ymaxi
42      ! -------                                                  ---------     .
43      integer (KIND=wi), parameter :: taille=2500
44      type(date_sec) :: one_tps_1,one_tps_2
45      integer(KIND=wi) :: i, j, n, nREF, nDIR, Noldtime, Nnewtime, ratetime
46      real (KIND=wr) :: XY(taille,3),XYREF(taille,3),XYDIR(taille,3)
47      real (KIND=wr) :: sDIR(taille,2),sREF(taille,2)
48      real (KIND=wr) :: x1, y1
49      real (KIND=wr) :: a, R2                                                ! coeficient directeur et Chi2 de la régression
50      character(LEN=7) :: char1, char2
51      character (LEN=5) :: char_nbseismes
52      ! ------------------------------------------------------------------   . diagramme de wadati pour toutes les ondes
53      call Wadatiplot(nbtps,D,param,a=a,R2=R2,XY=XY,nb=n)
54      ! -------                                                  ---------     . diagramme de wadati pour ondes réfractées
55      call Wadatiplot(nbtps,D,param,a=aREF,R2=R2REF,XY=XYREF,nb=nREF,sig=sREF,atype='N')
56      ! -------                                                  ---------     . diagramme de wadati pour ondes directes
57      call Wadatiplot(nbtps,D,param,a=aDIR,R2=R2DIR,XY=XYDIR,nb=nDIR,sig=sDIR,atype='G')
58      ! ------------------------------------------------------------------   . minmax
59      Xmaxi=1.0_wr
60      Ymaxi=1.0_wr
61      do i=1,n
62        if (Xmaxi.lt.XY(i,1)) Xmaxi=XY(i,1)
63        if (Ymaxi.lt.XY(i,2)) Ymaxi=XY(i,2)
64      enddo
65      Xmaxi=1.1_wr*Xmaxi
66      Ymaxi=1.1_wr*Ymaxi
67      ! ------------------------------------------------------------------   .
68      write(600,*)"BEFORE=$SECONDS"
69      call system_clock(Noldtime)
70      write(600,*)"echo 'execution du script GMT Wadatiplot'"
71      write(*,*)"ecriture du script GMT Wadatiplot "
72      ! ------------------------------------------------------------------   .
73      do j=1,nbseismes
74        write(char_nbseismes(1:5),'(i5)')j
75        write(600,*)"##########################################"
76        write(600,*)"################ Wadatiplot ###############"
77        ! ----------------------------------------------------------------   .
78        write(600,*)"geoproj=-JX13i/8i"                                      ! système de projection
79        write(600,'(a12,E13.7,a3,E13.7)')"geozone=-R0/",Xmaxi,"/0/",Ymaxi
80        write(600,*)"file=OUTPUT/GMT/wadatiplot"//trim(adjustl(char_nbseismes))//".ps"
81        if (Xmaxi.gt.60.0_wr) then
82          write(600,*)"psbasemap $geozone $geoproj -Ba10f5:""T@-P@--T@-0@- (s)"":",&
83          "/a10f5:""T@-S@--T@-P@- (s)"":WenS -Xc -Yc -K >  $file"
84        else
85          write(600,*)"psbasemap $geozone $geoproj -Ba2f.5:""T@-P@--T@-0@- (s)"":",&
86          "/a2f.5:""T@-S@--T@-P@- (s)"":WenS -Xc -Yc -K >  $file"
87        endif
88        ! -------                                                ---------     .
89        do i=1,int(Xmaxi,wi)+1
90          ! -------                                              ---------     . traits théoriquea min et max
91          write(600,*)"echo -e '",i-1,(dp%VpVs%themax-1.0_wr)*real(i-1,wr),"\n",i,(dp%VpVs%themax-1.0_wr)*real(i,wr), &
92              "' | psxy $geozone $geoproj -W1,red,- -O -K >>  $file"
93          write(600,*)"echo -e '",i-1,(dp%VpVs%themin-1.0_wr)*real(i-1,wr),"\n",i,(dp%VpVs%themin-1.0_wr)*real(i,wr), &
94              "' | psxy $geozone $geoproj -W1,red,- -O -K >>  $file"
95          ! -------                                              ---------     . traits théoriquea (droite de regression) TOTAL
96          write(600,*)"echo -e "" ",i-1,a*real(i-1,wr),"\n",i,a*real(i,wr), " \"
97          write(600,*)" "" | psxy $geozone $geoproj -W5 -O -K >>  $file"
98          ! -------                                              ---------     . traits théoriquea (droite de regression) DIR
99          write(600,*)"echo -e "" ",i-1,aDIR*real(i-1,wr),"\n",i,aDIR*real(i,wr), " \"
100         write(600,*)" "" | psxy $geozone $geoproj -W5,gray,-.-. -O -K >>  $file"
101         ! -------                                              ---------     . traits théoriquea (droite de regression) REF
102         write(600,*)"echo -e "" ",i-1,aREF*real(i-1,wr),"\n",i,aREF*real(i,wr), " \"
```

```
103         write(600,*)" "" | psxy $geozone $geoproj -W5,gray,-- -O -K >>  $file"
104         ! -------                                                  --------     .
105       enddo
106       ! -------                                                  --------     .
107       do i=1,nDIR                                                             ! points réels des ondes directes
108                                                                               !  modèle de référence : 1000 best modèle
109         write(600,*)"echo",XYDIR(i,1),XYDIR(i,2),XYDIR(i,3), &
110         sqrt(sDIR(i,1)**2.0_wr+dp%Tzero(j)%ec_1000**2.0_wr),sqrt(sDIR(i,1)**2.0_wr+sDIR(i,2)**2.0_wr), " \"
111         write(600,*)" | psxy $geozone $geoproj -O -K -St0.1i -Wthinnest -Exy -COUTPUT/GMT/colorpal3.cpt >>  $file"
112       enddo
113       ! -------                                                  --------     .
114       do i=1,nREF                                                             ! points réels des ondes réfractées
115                                                                               ! modèle de référence : 1000 best modèle
116         write(600,*)"echo",XYREF(i,1),XYREF(i,2),XYREF(i,3), &
117         sqrt(sREF(i,1)**2.0_wr+dp%Tzero(j)%ec_1000**2.0_wr),sqrt(sREF(i,1)**2.0_wr+sREF(i,2)**2.0_wr), " \"
118         write(600,*)" | psxy $geozone $geoproj -O -K -Si0.1i -Wthinnest -Exy -COUTPUT/GMT/colorpal3.cpt >>  $file"
119       enddo
120       ! -------                                                  --------     . POUR CE SEISME
121       if (nbseismes.gt.1) then
122         do i=1,nbtps(j)
123           if(D(j)%datatps(i)%andS.eq.'S') then
124             one_tps_1%date = D(j)%datatps(i)%tpsR%date
125             one_tps_1%sec = D(j)%datatps(i)%tpsR%secP
126             call basetime(one_tps_1)
127             one_tps_2 = dp%temps_ref(j)
128             one_tps_2%sec = dp%Tzero(j)%moy_1000                              !  modèle de référence : 1000 best modèle
129             ! one_tps_2%sec = dp%Tzero(j)%best                                !  modèle de référence : best modèle
130             call basetime(one_tps_2)
131             call difftime(x1,one_tps_1,one_tps_2)
132             one_tps_2%date = D(j)%datatps(i)%tpsR%date
133             one_tps_2%sec = D(j)%datatps(i)%tpsR%secS
134             call basetime(one_tps_2)
135             call difftime(y1,one_tps_2,one_tps_1)
136             write(600,*)"echo",x1,y1," | psxy $geozone $geoproj -O -K -Sc0.3i -Wthinnest >>  $file"
137           endif
138         enddo
139       endif
140       ! -------                                                  --------     . légende
141       write(600,*)"##########################################"
142       X = Xmaxi * 0.1_wr
143       Y = Ymaxi * 0.8_wr
144       if(XYREF(3,1).gt.0.0_wr) then
145         ! si il existe des ondes réfractées :
146         write(600,*)"echo",X,Y," | psxy $geozone $geoproj -O -K -St0.1i -Wthinnest -Gyellow >>  $file"
147         write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5,gray,-.-. >>  $file"
148         write(char1,'(f7.4)')1.0_wr+aDIR
149         write(char2,'(f7.4)')R2DIR
150         write(600,*)"echo """,X+0.1_wr*X,Y,"15 0 4 LM ondes directes : V@-P@-/V@-S@- =",char1," \"
151         write(600,*)" (@~\143@~@-2@- =",char2,")"," \"
152         write(600,*)""" | pstext $geozone $geoproj -O -K >> $file"
153         Y = Ymaxi * 0.75_wr
154         write(char1,'(f7.4)')1.0_wr+aREF
155         write(char2,'(f7.4)')R2REF
156         write(600,*)"echo",X,Y," | psxy $geozone $geoproj -O -K -Si0.1i -Wthinnest -Gyellow >>  $file"
157         write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5,gray,-- >>  $file"
158         write(600,*)"echo """,X+0.1_wr*X,Y,"15 0 4 LM ondes r\351fract\351es : V@-P@-/V@-S@- =",char1," \"
159         write(600,*)" (@~\143@~@-2@- =",char2,")"" | pstext $geozone $geoproj -O -K >> $file"
160         Y = Ymaxi * 0.70_wr
161         write(char1,'(f7.4)')1.0_wr+a
162         write(char2,'(f7.4)')R2
163         write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5 >>  $file"
164         write(600,*)"echo """,X+0.1_wr*X,Y,"15 0 4 LM ensembles : V@-P@-/V@-S@- =","," \"
165         write(600,*)char1," (@~\143@~@-2@- =",char2,")"" | pstext $geozone $geoproj -O -K >> $file"
166       else
167         Y = Ymaxi * 0.75_wr
```

```
168           write(char1,'(f7.4)')1.0_wr+a
169           write(char2,'(f7.4)')R2
170           write(600,*)"echo -e """,X-0.25_wr*X,Y," \n", X-0.5_wr*X,Y,""" | psxy $geozone $geoproj -O -K -W5 >>  $file"
171           write(600,*)"echo """,X,Y,"15 0 4 LM ondes directes : V@-P@-/V@-S@- =",char1," \"
172           write(600,*)" (@~\143@~@-2@- =",char2,")"" | pstext $geozone $geoproj -O -K >> $file"
173         endif
174         ! -------                                                    --------    .
175         write(600,*)"psscale -D1/-1/0.50E+01/0.25ch -B.25:""pond\351ration"": -S -I -COUTPUT/GMT/colorpal3.cpt -O -K >> $file"
176         ! -------                                                    --------    .
177         write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
178         write(600,*)"##########################################"
179         write(600,*)"ps2raster OUTPUT/GMT/wadatiplot"//trim(adjustl(char_nbseismes))//".ps -Tf -A"
180         write(600,'(2a)')"mv OUTPUT/GMT/wadatiplot"//trim(adjustl(char_nbseismes))//".pdf ", &
181           "OUTPUT/figures/wadatiplot"//trim(adjustl(char_nbseismes))//".pdf"
182         write(600,*)"##########################################"
183         write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
184         write(600,*)" echo $ELAPSED secondes"
185         call system_clock(Nnewtime,ratetime)
186         tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
187         write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)') temps : ',int(tl/3600.0_wr,wi), &
188         int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
189         ! ------------------------------------------------------------    .
190       enddo
191       ! ------------------------------------------------------------    .
192     end subroutine GMT_wadati
193
194   END MODULE figure_GMTwada
195
196
197
198   ! *****************************************************************    .
199   ! *****************************************************************    .
```


2.12 SRC/MOD/MOD_GMT/mkposteriori.f90

```
1   ! étude a posteriori des paramètres
2   ! janvier 2016
3   ! *****************************************************************    .
4   ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------    .
5   ! *****************************************************************    .
6   ! ------------------------------------------------------------------    .
7
8   MODULE figure_posteriori
9
10      use modparam
11      use typetemps
12
13      implicit none
14
15      private
16
17      public  :: GMT_posteriori_lonlat
18      public  :: GMT_posteriori
19
20
21  CONTAINS
22
23
24  ! ------------------------------------------------------------------    .
25
26    subroutine GMT_posteriori_lonlat(chaine,dp)
27      ! -------                                                  ---------    .mh
28      ! étude a posteriori des paramètres Lon/Lat
29      ! ------------------------------------------------------------    .
30      use distance_epi
```

```
31      use tri, only : melangetab
32      ! ------                                                  --------     .
33      implicit none
34      ! ------                                                  --------     .
35      type(densityplot), intent (in) :: dp                                   ! modèles retenus par McMC
36      integer(KIND=wi), intent (in) :: chaine
37      ! ------                                                  --------     .
38      integer(KIND=wi), parameter :: abin=16
39      integer(KIND=wi), parameter :: nbptmaxbin=500
40      ! ------                                                  --------     .
41      real(KIND=wr) :: tl
42      real(KIND=wr) :: dmax,dlon,dlat,d,daz
43      real(KIND=wr), dimension(:,:), allocatable :: vect1,vect2,vect3,vect4
44      integer(KIND=wi) :: i,j,k,m,Noldtime, Nnewtime, ratetime
45      integer(KIND=wi) :: minbin
46      integer(KIND=wi) :: ok
47      integer(KIND=wi) :: bin(abin)
48      character (LEN=5) :: numberchaine
49      ! ------------------------------------------------------------     .
50      write(numberchaine(1:5),'(i5)')chaine
51      ! ------------------------------------------------------------     .
52      ! selection des modèles autour de l'épientre et dont la distance au meilleur modèle est grande
53      ! répartition des modèles en fonction de leur azimuth
54      ! ------------------------------------------------------------     .
55      open(unit=19,file="OUTPUT/GMT/post_lonlat_lonlat-"//trim(adjustl(numberchaine))//".bin", &
56        STATUS="replace",access='direct',RECL=24)
57      open(unit=20,file="OUTPUT/GMT/post_lonlat_topdf-"//trim(adjustl(numberchaine))//".bin", &
58        STATUS="replace",access='direct',RECL=24)
59      open(unit=21,file="OUTPUT/GMT/post_lonlat_VCVpVs-"//trim(adjustl(numberchaine))//".bin", &
60        STATUS="replace",access='direct',RECL=24)
61      open(unit=22,file="OUTPUT/GMT/post_lonlat_VMZ_moho-"//trim(adjustl(numberchaine))//".bin", &
62        STATUS="replace",access='direct',RECL=24)
63      ! ------                                                  --------     . le modèle le plus loin
64      dmax=0.0_wr
65      do i=1,dp%nbparam
66         call dellipsgc(dp%lat(chaine)%vec(i),dp%lon(chaine)%vec(i), &
67           dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),d,daz)
68         if(d.gt.dmax)dmax=d
69      enddo
70      ! ------                                                  --------     . binnage
71      bin(:)=0
72      do i=1,dp%nbparam
73         call dellipsgc(dp%lat(chaine)%vec(i),dp%lon(chaine)%vec(i), &
74           dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),d,daz)
75         call doselect(dmax,d,ok)
76         if(ok==0) call dobinnage(daz,bin)
77      enddo
78      ! ------                                                  --------     . binnage max
79      minbin=99999999
80      do i=1,abin
81         if(bin(i).lt.minbin) minbin=bin(i)
82      enddo
83      ! ------                                                  --------     . au moins !
84      if (minbin.lt.10) minbin=10
85      ! ------                                                  --------     . au plus !
86      if (minbin.gt.nbptmaxbin) minbin=nbptmaxbin
87      ! ------                                                  --------     .
88      allocate(vect1(3,abin*minbin))
89      allocate(vect2(3,abin*minbin))
90      allocate(vect3(3,abin*minbin))
91      allocate(vect4(3,abin*minbin))
92      ! ------                                                  --------     .
93      vect1=0.0_wr
94      vect2=0.0_wr
95      vect3=0.0_wr
```

```
96      vect4=0.0_wr
97      ! -------                                                    --------     .
98      j=0
99      k=0
100     bin(:)=0
101     do i=1,dp%nbparam
102       ! -------                                                  --------     .
103       call dellipsgc(dp%lat(chaine)%vec(i),dp%lon(chaine)%vec10000(1,1), &
104         dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),dlat)     ! calcul de dlat
105       call dellipsgc(dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec(i), &
106         dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),dlon)     ! calcul de dlon
107       call dellipsgc(dp%lat(chaine)%vec(i),dp%lon(chaine)%vec(i), &
108         dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),d,daz)    ! calcul de daz
109       ! ------------------------------------------------------------------     .
110       call doselect(dmax,d,ok)
111       ! ------------------------------------------------------------------     .
112       if(ok==0) then
113         call dobinnage(daz,bin,amax=minbin,test=ok)
114       endif
115       ! ------------------------------------------------------------------     .
116       if (ok==0) then
117         ! -------                                                  --------     .
118         j=j+1
119         if ((daz.ge.0.0_wr).and.(daz.le.360.0_wr)) then
120           if (daz.le.90.0_wr) then
121             vect1(1,j)=dlon
122             vect1(2,j)=dlat
123             vect1(3,j)=daz
124           elseif(daz.le.180.0_wr) then
125             vect1(1,j)=dlon
126             vect1(2,j)=-dlat
127             vect1(3,j)=daz
128           elseif(daz.le.270.0_wr) then
129             vect1(1,j)=-dlon
130             vect1(2,j)=-dlat
131             vect1(3,j)=daz
132           elseif(daz.le.360.0_wr) then
133             vect1(1,j)=-dlon
134             vect1(2,j)=dlat
135             vect1(3,j)=daz
136           endif
137         else
138           write(*,*)'problème 1 dans GMT_posteriori_lonlat : daz = ',daz
139           stop
140         endif
141         ! -------                                                  --------     .
142         vect2(1,j)=dp%Zhypo(chaine)%vec(i)
143         vect2(2,j)=dp%Tzero(chaine)%vec(i)
144         vect2(3,j)=daz
145         ! -------                                                  --------     .
146         vect3(1,j)=dp%VC%vec(i)
147         vect3(2,j)=dp%VpVs%vec(i)
148         vect3(3,j)=daz
149         ! -------                                                  --------     .
150         vect4(1,j)=dp%VM%vec(i)
151         vect4(2,j)=dp%Zmoho%vec(i)
152         vect4(3,j)=daz
153         ! -------                                                  --------     .
154       else
155         k=k+1
156         ! -------                                                  --------     .
157         if ((daz.ge.0.0_wr).and.(daz.le.360.0_wr)) then
158           if (daz.le.90.0_wr) then
159             write(19,REC=k)real(dlon,8),real(dlat,8),real(ok,8)*500._8
160           elseif(daz.le.180.0_wr) then
```

```
161             write(19,REC=k)real(dlon,8),-real(dlat,8),real(ok,8)*500._8
162           elseif(daz.le.270.0_wr) then
163             write(19,REC=k)-real(dlon,8),-real(dlat,8),real(ok,8)*500._8
164           elseif(daz.le.360.0_wr) then
165             write(19,REC=k)-real(dlon,8),real(dlat,8),real(ok,8)*500._8
166           endif
167         else
168           write(*,*)'problème 2 dans GMT_posteriori_lonlat : daz = ',daz
169           stop
170         endif
171         ! ------                                                  --------     .
172         write(20,REC=k)real(dp%Zhypo(chaine)%vec(i),8),real(dp%Tzero(chaine)%vec(i),8),real(ok,8)*500._8
173         write(21,REC=k)real(dp%VC%vec(i),8),real(dp%VpVs%vec(i),8),real(ok,8)*500._8
174         write(22,REC=k)real(dp%VM%vec(i),8),real(dp%Zmoho%vec(i),8),real(ok,8)*500._8
175         ! ------                                                  --------     .
176       endif
177       ! ---------------------------------------------------------------     .
178     enddo
179     ! ------                                                    --------     .
180     close(19)
181     close(20)
182     close(21)
183     close(22)
184     ! ---------------------------------------------------------------     .
185     call melangetab(j,vect1)                                              ! randomize le tableau
186     call melangetab(j,vect2)                                              ! randomize le tableau
187     call melangetab(j,vect3)                                              ! randomize le tableau
188     call melangetab(j,vect4)                                              ! randomize le tableau
189     ! ------                                                    --------     .
190     open(unit=15,file="OUTPUT/GMT/post_lonlatok_lonlat-"//trim(adjustl(numberchaine))//".bin", &
191       STATUS="replace",access='direct',RECL=24)
192     open(unit=16,file="OUTPUT/GMT/post_lonlatok_topdf-"//trim(adjustl(numberchaine))//".bin", &
193       STATUS="replace",access='direct',RECL=24)
194     open(unit=17,file="OUTPUT/GMT/post_lonlatok_VCVpVs-"//trim(adjustl(numberchaine))//".bin", &
195       STATUS="replace",access='direct',RECL=24)
196     open(unit=18,file="OUTPUT/GMT/post_lonlatok_VMZ_moho-"//trim(adjustl(numberchaine))//".bin", &
197       STATUS="replace",access='direct',RECL=24)
198     ! ------                                                    --------     .
199     m=0
200     do i=1,j
201       if ((abs(vect1(1,i))+abs(vect1(2,i))+abs(vect1(3,i))).gt.0.00000001_wr) then
202         m=m+1
203         write(15,REC=m)real(vect1(1,i),8),real(vect1(2,i),8),real(vect1(3,i),8)
204         write(16,REC=m)real(vect2(1,i),8),real(vect2(2,i),8),real(vect2(3,i),8)
205         write(17,REC=m)real(vect3(1,i),8),real(vect3(2,i),8),real(vect3(3,i),8)
206         write(18,REC=m)real(vect4(1,i),8),real(vect4(2,i),8),real(vect4(3,i),8)
207       endif
208     enddo
209     ! ------                                                    --------     .
210     close(15)
211     close(16)
212     close(17)
213     close(18)
214     ! ------                                                    --------     .
215     deallocate(vect1,vect2,vect3,vect4)
216     ! ---------------------------------------------------------------     .
217     write(*,*)"ecriture des script GMT_post LonLat"
218     write(600,*)"BEFORE=$SECONDS"
219     call system_clock(Noldtime)
220     write(600,*)"#####################################"
221     write(600,*)"#####################################"
222     write(600,*)
223     write(600,*)"echo 'execution du script GMT post LonLat'"
224     write(600,*)"#########################################"
225     write(600,*)"###########     autocorr       ##########"
```

```
226     write(600,*)"########################################"
227     ! ------------------------------------------------------------ .
228     write(600,'(a)')"makecpt -Cwysiwyg -T0.0/380.0/5.0 -Z -N > OUTPUT/GMT/colorpostlonlat.cpt"
229     write(600,'(a)')"echo 'B 150 150 150 ' >> OUTPUT/GMT/colorpostlonlat.cpt"
230     write(600,'(a)')"echo 'F 220 220 220 ' >> OUTPUT/GMT/colorpostlonlat.cpt"
231     write(600,'(a)')"echo 'N 200 200 200 ' >> OUTPUT/GMT/colorpostlonlat.cpt"
232     ! ------------------------------------------------------------ .
233     write(600,*)"geoproj=-JX4i"
234     write(600,*)"file=OUTPUT/GMT/postLonLat-"//trim(adjustl(numberchaine))//".ps"
235     ! ------------------------------------------------------------ .
236     ! pour lon et lat :
237     write(600,*)"minmax OUTPUT/GMT/post_lonlatok_lonlat-"//trim(adjustl(numberchaine))//".bin -bi3 -I1.5 ", &
238       "> OUTPUT/GMT/geozone.d "
239     write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
240     write(600,*)"psbasemap $geozone $geoproj -Ba0:'"//dp%Lon(chaine)%char//"':/a0:'", &
241       dp%Lat(chaine)%char//"':SWen -K -Xc -Yc >  $file"
242     write(600,*)"psbasemap $geozone $geoproj -Ba0:'longitude (km)':/a0:'latitude (km)':SWen -K >  $file"
243     ! ------------------------------------------------------------ .
244     write(600,*)"gmtset BASEMAP_FRAME_RGB gray"
245     write(600,*)"gmtset GRID_PEN_PRIMARY=black GRID_PEN_SECONDARY=black TICK_PEN=black"
246     ! ------------------------------------------------------------ .
247     write(600,*)"psbasemap $geozone $geoproj -Ba1f.25g100SWen -K -O >>  $file"
248     ! ------------------------------------------------------------ .
249     write(600,*)"gmtset BASEMAP_FRAME_RGB black"
250     write(600,*)"gmtset GRID_PEN_PRIMARY=black GRID_PEN_SECONDARY=black TICK_PEN=black"
251     ! ------------------------------------------------------------ .
252     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlat_lonlat-"//trim(adjustl(numberchaine))//".bin ", &
253       "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
254     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlatok_lonlat-"//trim(adjustl(numberchaine))//".bin ", &
255       "-bi3 -Sa0.05 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
256     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
257     ! -------                                                ---------     . rose données
258     write(600,*)"psrose ./OUTPUT/GMT/baz"//"-"//trim(adjustl(numberchaine))//".d -: -A10 -S.5in", &
259     " -Ggreen -R0/1/0/360 -W1 -F -L'@'/'@'/'@'/'donn\351es' -Y2.5i -X.15i -B.25g0.25/30g30 -O -K -D >> $file"
260     write(600,*)"psrose ./OUTPUT/GMT/baz1_"//"-"//trim(adjustl(numberchaine))//".d -: -A10 -S.5i", &
261     " -Gblue -R0/1/0/360 -W1 -F -O -K -D >> $file "
262     ! ------------------------------------------------------------ .
263     ! pour to et pfd :
264     write(600,*)"minmax OUTPUT/GMT/post_lonlatok_topdf-"//trim(adjustl(numberchaine))//".bin -bi3 -I5/1", &
265       " > OUTPUT/GMT/geozone.d "
266     write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
267     write(600,*)"psbasemap $geozone $geoproj -Ba5f1:'"//dp%Zhypo(chaine)%char//"':/a1f.25:'", &
268       dp%Tzero(chaine)%char//"':SEwn -K -O -X4.35i -Y-2.5i >>  $file"
269     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlat_topdf-"//trim(adjustl(numberchaine))//".bin ", &
270       "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
271     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlatok_topdf-"//trim(adjustl(numberchaine))//".bin ", &
272       "-bi3 -Sa0.05 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
273     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
274     ! ------------------------------------------------------------ .
275     ! pour VC et VpVs :
276     write(600,*)"minmax OUTPUT/GMT/post_lonlatok_VCVpVs-"//trim(adjustl(numberchaine))//".bin -bi3 -I.25/.05", &
277       " > OUTPUT/GMT/geozone.d "
278     write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
279     write(600,*)"psbasemap $geozone $geoproj -Ba.2f.05:'"//dp%VC%char//"':/a.03f.01:'", &
280       dp%VpVs%char//"':NWes -K -O -Y4.5i -X-4.5i >>  $file"
281     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlat_VCVpVs-"//trim(adjustl(numberchaine))//".bin ", &
282       "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
283     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlatok_VCVpVs-"//trim(adjustl(numberchaine))//".bin ", &
284       "-bi3 -Sa0.05 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
285     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
286     ! ------------------------------------------------------------ .
287     ! pour VM et Z_moho :
288     write(600,*)"geoproj=-JX4i/-4i"
289     write(600,*)"minmax OUTPUT/GMT/post_lonlatok_VMZ_moho-"//trim(adjustl(numberchaine))//".bin -bi3 -I.25/5", &
290       " > OUTPUT/GMT/geozone.d "
```

```
291     write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
292     write(600,*)"psbasemap $geozone $geoproj -Ba.2f.05:''//dp%VM%char//'':/a5f1:'', &
293       dp%Zmoho%char//'':NsEw -K -O -X4.5i  >>  $file"
294     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlat_VMZ_moho-"//trim(adjustl(numberchaine))//".bin ", &
295       "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
296     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_lonlatok_VMZ_moho-"//trim(adjustl(numberchaine))//".bin ", &
297       "-bi3 -Sa0.05 -COUTPUT/GMT/colorpostlonlat.cpt -K -O >> $file"
298     ! ------------------------------------------------------------------      .
299     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
300     write(600,*)"ps2raster OUTPUT/GMT/postLonLat-"//trim(adjustl(numberchaine))//".ps -Tg -A -P"
301     write(600,'(2a)')"mv OUTPUT/GMT/postLonLat-"//trim(adjustl(numberchaine))//".png OUTPUT/figures/postLonLat"// &
302       "-"//trim(adjustl(numberchaine))//".png"
303     ! ------------------------------------------------------------------      .
304     write(600,*)
305     write(600,*)"#####################################"
306     write(600,*)"#####################################"
307     write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
308     write(600,*)" echo $ELAPSED secondes"
309     call system_clock(Nnewtime,ratetime)
310     tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
311     write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
312     int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
313
314     ! ------------------------------------------------------------------      .
315     CONTAINS
316     ! ------------------------------------------------------------------      .
317
318     subroutine doselect(dmax,d,ok)
319       ! -------                                           ---------      .mh
320       ! selection des modèles à colorier !
321       ! ------------------------------------------------------------------      .
322       implicit none
323       ! -------                                           ---------      .
324       real(KIND=wr), intent (in)  :: dmax,d
325       integer, intent (out)  :: ok
326       ! -------                                           ---------      .
327       real(KIND=wr) :: minpct,maxpct
328       ! ------------------------------------------------------------------      .
329       minpct=0.05_wr
330       maxpct=0.5_wr
331       ! ------------------------------------------------------------------      .
332       if (d.lt.(maxpct*dmax)) then
333         if (d.gt.(minpct*dmax)) then
334           ok=0
335         else
336           ok=-1
337         endif
338       else
339         ok=1
340       endif
341       ! ------------------------------------------------------------------      .
342     end subroutine doselect
343
344       ! ------------------------------------------------------------------      .
345
346     subroutine dobinnage(az,bin,amax,test)
347       ! -------                                           ---------      .mh
348       ! répartition azimutale homogène des modèles à colorier !
349       ! ------------------------------------------------------------------      .
350       implicit none
351       ! -------                                           ---------      .
352       real(KIND=wr), intent (in)  :: az
353       integer(KIND=wi), intent (inout)  :: bin(abin)
354       integer(KIND=wi), intent (in), optional :: amax
355       integer(KIND=wi), intent (out), optional :: test
```

```
356       ! -------                                                  --------     .
357       integer(KIND=wi) :: i
358       ! ---------------------------------------------------------------     .
359       i=int(az/(360._wr/real(abin,wr))+1.0_wr,wi)
360       ! -------                                                  --------     .
361       if (present(amax)) then
362         if (bin(i).lt.amax) then
363           bin(i)=bin(i)+1
364           if (present(amax)) test=0
365         else
366           if (present(amax)) test=1
367         endif
368       else
369         bin(i)=bin(i)+1
370       endif
371       ! ---------------------------------------------------------------     .
372     end subroutine dobinnage
373
374     ! ---------------------------------------------------------------     .
375   end subroutine GMT_posteriori_lonlat
376
377
378   ! ---------------------------------------------------------------     .
379
380   subroutine GMT_posteriori(chaine,dp,undp)
381     ! -------                                                  --------     .mh
382     ! étude a posteriori d'un paramètre
383     ! ---------------------------------------------------------------     .
384     use distance_epi
385     ! -------                                                  --------     .
386     implicit none
387     ! -------                                                  --------     .
388     type(densityplot), intent (in) :: dp                                   ! modèles retenus par McMC
389     type(densityplot_one), intent (in) :: undp
390     integer(KIND=wi), intent (in) :: chaine
391     ! -------                                                  --------     .
392     real(KIND=wr) :: tl
393     real(KIND=wr) :: dlon,dlat,d,daz
394     real(KIND=wr) ::dpmin,dpmax,a,b,coef
395     integer(KIND=wi) :: i,Noldtime, Nnewtime, ratetime
396     character (LEN=5) :: numberchaine
397     ! ---------------------------------------------------------------     .
398     write(numberchaine(1:5),'(i5)')chaine
399     ! ---------------------------------------------------------------     .
400     dpmin=1.e9_wr
401     dpmax=-1.e9_wr
402     do i=1,dp%nbparam
403       if (dpmin.gt.undp%vec(i)) dpmin=undp%vec(i)
404       if (dpmax.lt.undp%vec(i)) dpmax=undp%vec(i)
405     enddo
406     ! ---------------------------------------------------------------     .
407     open(unit=29,file="OUTPUT/GMT/post_"//undp%name//"_lonlat-"//trim(adjustl(numberchaine))//".bin", &
408       STATUS="replace",access='direct',RECL=24)
409     open(unit=30,file="OUTPUT/GMT/post_"//undp%name//"_topdf-"//trim(adjustl(numberchaine))//".bin", &
410       STATUS="replace",access='direct',RECL=24)
411     open(unit=31,file="OUTPUT/GMT/post_"//undp%name//"_VCVpVs-"//trim(adjustl(numberchaine))//".bin", &
412       STATUS="replace",access='direct',RECL=24)
413     open(unit=32,file="OUTPUT/GMT/post_"//undp%name//"_VMZ_moho-"//trim(adjustl(numberchaine))//".bin", &
414       STATUS="replace",access='direct',RECL=24)
415     ! ---------------------------------------------------------------     .
416     do i=1,dp%nbparam
417       b= (1.0_wr)/(1.0_wr-(dpmax/dpmin))
418       a = -b/dpmin
419       coef = a * undp%vec(i) + b                                       ! redimessionne entre 0 et 1
420       ! -------                                                  --------     .
```

```
421       call dellipsgc(dp%lat(chaine)%vec(i),dp%lon(chaine)%vec10000(1,1), &
422         dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),dlat)     ! calcul de dlat
423       call dellipsgc(dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec(i), &
424         dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),dlon)     ! calcul de dlon
425       call dellipsgc(dp%lat(chaine)%vec(i),dp%lon(chaine)%vec(i), &
426         dp%lat(chaine)%vec10000(1,1),dp%lon(chaine)%vec10000(1,1),d,daz)    ! calcul de daz
427       ! -------                                                   --------    .
428       if ((daz.ge.0.0_wr).and.(daz.le.360.0_wr)) then
429         if (daz.le.90.0_wr) then
430           write(29,REC=i)real(dlon,8),real(dlat,8),real(coef,8)
431         elseif(daz.le.180.0_wr) then
432           write(29,REC=i)real(dlon,8),-real(dlat,8),real(coef,8)
433         elseif(daz.le.270.0_wr) then
434           write(29,REC=i)-real(dlon,8),-real(dlat,8),real(coef,8)
435         elseif(daz.le.360.0_wr) then
436           write(29,REC=i)-real(dlon,8),real(dlat,8),real(coef,8)
437         endif
438       else
439         write(*,*)'problème 2 dans GMT_posteriori_VC : daz = ',daz
440         stop
441       endif
442       ! -------                                                   --------    .
443       write(30,REC=i)real(dp%Zhypo(chaine)%vec(i),8),real(dp%Tzero(chaine)%vec(i),8),real(coef,8)
444       write(31,REC=i)real(dp%VC%vec(i),8),real(dp%VpVs%vec(i),8),real(coef,8)
445       write(32,REC=i)real(dp%VM%vec(i),8),real(dp%Zmoho%vec(i),8),real(coef,8)
446       ! -------                                                   --------    .
447     enddo
448     ! ------------------------------------------------------------------    .
449     close(29)
450     close(30)
451     close(31)
452     close(32)
453     ! ------------------------------------------------------------------    .
454     write(*,*)"ecriture des script GMT_post "//undp%name
455     write(600,*)"BEFORE=$SECONDS"
456     call system_clock(Noldtime)
457     write(600,*)"#####################################"
458     write(600,*)"#####################################"
459     write(600,*)
460     write(600,*)"echo 'execution du script GMT post "//undp%name//" '"
461     write(600,*)"#############################################"
462     write(600,*)"###########     autocorr       ##########"
463     write(600,*)"#############################################"
464     ! ------------------------------------------------------------------    .
465     write(600,'(a)')"makecpt -Cwysiwyg -T0.0/1.0/0.005 -Z -N > OUTPUT/GMT/colorpostvc.cpt"
466     write(600,'(a)')"echo 'B 150 150 150 ' >> OUTPUT/GMT/colorpostvc.cpt"
467     write(600,'(a)')"echo 'F 220 220 220 ' >> OUTPUT/GMT/colorpostvc.cpt"
468     write(600,'(a)')"echo 'N 200 200 200 ' >> OUTPUT/GMT/colorpostvc.cpt"
469     ! ------------------------------------------------------------------    .
470     write(600,*)"geoproj=-JX4i"
471     write(600,*)"file=OUTPUT/GMT/post_"//undp%name//"-"//trim(adjustl(numberchaine))//".ps"
472     ! ------------------------------------------------------------------    .
473     ! pour lon et lat :
474     write(600,*)"minmax OUTPUT/GMT/post_"//undp%name//"_lonlat-"//trim(adjustl(numberchaine))//".bin", &
475       " -bi3 -I1.5 > OUTPUT/GMT/geozone.d "
476     write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
477     write(600,*)"psbasemap $geozone $geoproj -Ba1f.25g100:'longitude (km)':/a1f.25g100:'latitude (km)':SWen -K -Xc  >  $file"
478     write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_"//undp%name//"_lonlat-"//trim(adjustl(numberchaine))//".bin ", &
479       "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostvc.cpt -K -O >> $file"
480     write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
481     ! ------------------------------------------------------------------    .
482     ! pour to et pfd :
483     write(600,*)"minmax OUTPUT/GMT/post_"//undp%name//"_topdf-"//trim(adjustl(numberchaine))//".bin -bi3 -I5/1 ", &
484       "> OUTPUT/GMT/geozone.d "
485     write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
```

```
    write(600,*)"psbasemap $geozone $geoproj -Ba5f1:'"//dp%Zhypo(chaine)%char//"':/a1f.25:'", &
      dp%Tzero(chaine)%char//"':SEwn -K -O -X4.5i >>  $file"
    write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_"//undp%name//"_topdf_"//trim(adjustl(numberchaine))//".bin ", &
      "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostvc.cpt -K -O >> $file"
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
    ! ------------------------------------------------------------------    .
    ! pour VC et VpVs :
    write(600,*)"minmax OUTPUT/GMT/post_"//undp%name//"_VCVpVs_"//trim(adjustl(numberchaine))//".bin -bi3 -I.25/.05 ", &
      "> OUTPUT/GMT/geozone.d "
    write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
    write(600,*)"psbasemap $geozone $geoproj -Ba.2f.05:'"//dp%VC%char//"':/a.03f.01:'", &
      dp%VpVs%char//"':NWes -K -O -Y4.5i -X-4.5i >>  $file"
    write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_"//undp%name//"_VCVpVs_"//trim(adjustl(numberchaine))//".bin ", &
      "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostvc.cpt -K -O >> $file"
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O -K >>  $file"
    ! ------------------------------------------------------------------    .
    ! pour VM et Z_moho :
    write(600,*)"geoproj=-JX4i/-4i"
    write(600,*)"minmax OUTPUT/GMT/post_"//undp%name//"_VMZ_moho_"//trim(adjustl(numberchaine))//".bin -bi3 -I.25/5 ", &
      "> OUTPUT/GMT/geozone.d "
    write(600,*)"read geozone < OUTPUT/GMT/geozone.d "
    write(600,*)"psbasemap $geozone $geoproj -Ba.2f.05:'"//dp%VM%char//"':/a5f1:'", &
      dp%Zmoho%char//"':NsEw -K -O -X4.5i >>  $file"
    write(600,*)"psxy $geozone $geoproj OUTPUT/GMT/post_"//undp%name//"_VMZ_moho_"//trim(adjustl(numberchaine))//".bin ", &
      "-bi3 -Sa0.075 -COUTPUT/GMT/colorpostvc.cpt -K -O >> $file"
    ! ------------------------------------------------------------------    .
    write(600,*)"psbasemap $geozone $geoproj -Ba0 -O >>  $file"
    write(600,*)"ps2raster OUTPUT/GMT/post_"//undp%name//"-"//trim(adjustl(numberchaine))//".ps -Tg -A -P"
    write(600,'(2a)')"mv OUTPUT/GMT/post_"//undp%name//"-"//trim(adjustl(numberchaine))//".png ", &
      "OUTPUT/figures/post_"//undp%name//"-"//trim(adjustl(numberchaine))//".png"
    ! ------------------------------------------------------------------    .
    write(600,*)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
    write(600,*)" echo $ELAPSED secondes"
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
    int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------------    .
  end subroutine GMT_posteriori

  ! ------------------------------------------------------------------    .

END MODULE figure_posteriori



! ********************************************************************    .
! ********************************************************************    .
```

2.13 SRC/MOD/MOD_Geiger/subgeiger.f90

```
! septembre 2013 - fevrier 2014
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr         --------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .
!
! résolution d'un problème inverse :
!
! recherche des parametres hypocentraux en admettant un modèle de
! Terre, par la méthode Geiger (1910, 1912), méthode itérative par
! moindres carrées, pour un ou plusierus seismes
```

```
12
13 MODULE invGEIGER
14
15     use modparam
16     use typetemps, only : date_sec, dataone, dataall, parametre, parametres
17     use typetemps, only : mvPall_2_P1, mvP1_2_Pall, amoho_centroid
18     use mt19937, only : genrand_real3, normal
19     use time, only : basetime, difftime
20
21
22     implicit none
23
24     private
25
26     public  :: dogeiger                                                    ! inversion par la méthode Geiger avec un modèle de terre et hypocentral connu
27     public  :: dogeigerone
28
29     ! ------------------------------------------------------------------   .
30     ! mod (character (LEN=1)):
31     ! -------                                                 --------     .
32     ! 'I' pour un modèle simple                                            ! 2 couches -> pb direct CHEs !
33     ! 'A' pour les modèles de Terre Arroucau, phD 2006                     ! 9 couches
34     ! 'S' pour le modèles de Terre Si-Hex type Haslach                     ! 3 couches
35     ! 'F' pour un modèles de Terre test et fun                            ! x couches
36     ! 'C' pour un modèles de Terre CEA                                    ! 3 couches
37     ! ------------------------------------------------------------------   .
38
39
40 CONTAINS
41
42 ! ------------------------------------------------------------------       .
43
44   subroutine dogeiger(n,D,params,acentroid,mod,chut,amisfit)
45     ! -------                                                 --------     .mh
46     ! dogeiger pour nbseismes seismes
47     ! -------                                                 --------     .
48     implicit none
49     ! -------                                                 --------     .
50     integer(KIND=wi), intent (in) :: n(nbseismes)                          ! nombre de données de temps
51     type(dataall), intent (inout) :: D(nbseismes)                          ! données
52     type(parametres), intent (inout) :: params                             ! paramètres
53     type (amoho_centroid), intent (in) :: acentroid
54     character (LEN=1), intent(in) :: mod
55     logical, intent(in), optional :: chut
56     real(KIND=wr), intent(out), optional :: amisfit(nbseismes)
57     ! -------                                                 --------     .
58     type(parametre) :: param_init                                          ! paramètres
59     integer(KIND=wi) :: j
60     real(KIND=wr) :: onemisfit
61     logical :: con
62     ! ------------------------------------------------------------------   .
63     do j=1,nbseismes
64       call  mvPall_2_P1(param_init,params,j)
65       if(present(chut)) then
66         call dogeigerone(j,n(j),D(j)%datatps,param_init,acentroid,mod,onemisfit,chut=chut,con=con)
67       else
68         call dogeigerone(j,n(j),D(j)%datatps,param_init,acentroid,mod,onemisfit,con=con)
69       endif
70       call mvP1_2_Pall(params,param_init,j)
71       if(present(amisfit)) amisfit(j)=onemisfit
72     enddo
73     ! ------------------------------------------------------------------   .
74     end subroutine dogeiger
75
76 ! ------------------------------------------------------------------       .
```

```
77
78  subroutine dogeigerone(j,nbtps,datatps,param_one,acentroid,mod,onemisfit,chut,con)
79    ! ------                                                    --------    .mh
80    ! applique la méthode Geiger, en admettant un modèle de Terre fixe (param_one), pour un unique séisme.
81    ! ------                                                    --------    .
82    implicit none
83    ! ------                                                    --------    .
84    integer(KIND=wi), intent (in) :: j
85    integer(KIND=wi), intent (in) :: nbtps                                ! nb de données de temps
86    type(dataone), intent (inout) :: datatps(nbtps)                       ! données
87    type(parametre), intent (inout) :: param_one                          ! paramètres
88    type (amoho_centroid), intent (in) :: acentroid
89    character (LEN=1), intent(in) :: mod
90    real(KIND=wr), intent(out) :: onemisfit
91    logical, intent(in), optional :: chut
92    logical, intent(out), optional :: con                                 ! "true" si convergence
93    ! ------                                                    --------    .
94    integer(KIND=wi) i
95    ! ------------------------------------------------------------------    .
96    i=0
97    if(present(chut)) then
98        call geiger(nbtps,datatps,param_one,con,i,acentroid,mod,onemisfit,chut)
99    else
100       call geiger(nbtps,datatps,param_one,con,i,acentroid,mod,onemisfit)
101       if (.not.con) write(*,*)'problème dans dogeiger : données insufiantes, méthode Geiger fixe divergente',j
102   endif
103   ! ------------------------------------------------------------------    .
104 end subroutine dogeigerone
105
106 ! ------------------------------------------------------------------    .
107
108 subroutine geiger(nbtps,datatps,param_init,con,nk,acentroid,mod,onemisfit,chut)
109   ! ------                                                    --------    .mh
110   ! application de la méthode itérative de Geiger après initialisation des paramètres
111   ! con = .true. : si convergence
112   ! con = .false. : si divergence
113   ! ------                                                    --------    .
114   use sub_misfit, only : compute_misfitone
115   use pb_direct
116   ! ------                                                    --------    .
117   implicit none
118   ! ------                                                    --------    .
119   TYPE elementmatrice
120     real(KIND=wr) :: df_Dto,df_Dhypo_P,df_Dhypo_S,df_Dlon_P,df_Dlon_S,df_Dlat_P,df_Dlat_S
121   END TYPE elementmatrice
122   ! ------                                                    --------    .
123   integer(KIND=wi), intent(in) :: nbtps                                 ! nombre de données de temps P et S confondues
124   type(dataone), intent(inout) :: datatps(nbtps)                          ! données
125   type(parametre), intent(inout) :: param_init                          ! paramètres
126   logical, intent(out) :: con                                           ! "true" si convergence
127   integer(KIND=wi), intent(in) :: nk                                    ! nombre de divergences précendantes
128   type (amoho_centroid), intent (in) :: acentroid
129   character (LEN=1), intent(in) :: mod
130   real(KIND=wr), intent(out) :: onemisfit
131   logical, intent(in), optional :: chut
132   ! ------                                                    --------    .
133   integer(KIND=wi) :: i,j,k
134   integer(KIND=wi) :: nbdonnees                                         ! nombre de données de temps P et S non confondues
135   ! ------                                                    --------    .
136   real(KIND=wr), dimension(:,:), allocatable :: mat_A,mat_A_t,invAt_A_At ! matrices cf : SRC/SUB_Geiger/DOC/DOC_Geiger.pdf
137   real(KIND=wr), dimension(4,4) :: At_A,invAt_A                         ! matrices cf : SRC/SUB_Geiger/DOC/DOC_Geiger.pdf
138   real(KIND=wr), dimension(4) :: x_h, delta_x                           ! matrices cf : SRC/SUB_Geiger/DOC/DOC_Geiger.pdf
139   real(KIND=wr), dimension(:), allocatable :: gamma                     ! matrice différence de temps théoriques vs. réel
140   ! ------                                                    --------    .
141   type(elementmatrice) :: mat_el                                        ! matrice avec les dérivées partielles
```

```
142     ! -------                                                  --------     .
143     real(KIND=wr) :: sumdelta_x                                             ! précision à atteindre
144     type(dataone) :: one_data
145     integer(KIND=wi) :: dim                                                 ! dimension, ici dim = 4
146     character(LEN=76) :: prog                                               ! chaîne pour affichage
147     logical :: critique, rechut
148     real(KIND=wr) :: xmin=1000.0_wr, xmax=1500.0_wr                         ! diamètres de pondération, mais ici on s'en fiche ...
149     real(KIND=wr) :: pdfmoho
150     ! ------------------------------------------------------------------    .
151     if (mod=='I') then
152       call tempsTheoDirectone(param_init,datatps(1),critique,acentroid)  ! calcul le temps théorique et la difference théoriques et réels
153       pdfmoho=param_init%Zmoho
154     elseif ((mod=='A').or.(mod=='S').or.(mod=='C').or.(mod=='F')) then
155       call tempsTheoDirectone_AUTRE(param_init,datatps(1),pdfmoho,critique,mod) ! calcul le temps théorique et la difference théoriques et réels
156     else
157       write(*,*)'problème dans geiger : modèle inexistant !'
158       stop
159     endif
160     ! ------------------------------------------------------------------    .
161     con=.true.                                                              ! par défaut
162     ! ------------------------------------------------------------------    .
163     nbdonnees=0                                                             ! taille de la matrice A (nombre de données directes et réfractés, à la fois P et S)
164     do i=1,nbtps
165       if (datatps(i)%coefP.ne.4) then                                       ! ne prend pas en compte les plus mauvaises données
166         nbdonnees=nbdonnees+1
167       endif
168       if (datatps(i)%andS.eq.'S') then                                      ! si ondes S existe
169         if (datatps(i)%coefS.ne.4) then                                     ! ne prend pas en compte les plus mauvaises données
170           nbdonnees=nbdonnees+1
171         endif
172       endif
173     enddo
174     ! -------                                                  --------     .
175     allocate(mat_A(nbdonnees,4),mat_A_t(4,nbdonnees),invAt_A_At(4,nbdonnees),gamma(nbdonnees))
176     ! ------------------------------------------------------------------    .
177     ! ------- solution apriori initiale                        --------     .
178     x_h(1) = param_init%Lon
179     x_h(2) = param_init%Lat
180     if ((param_init%Zhypo.gt.0.0_wr).and.( param_init%Zhypo.lt.(pdfmoho-0.1_wr))) then
181       x_h(3) = param_init%Zhypo                                             ! séisme entre la surface (ici 0.0 masl) et le moho
182     else
183       x_h(3) = pdfmoho/2.0_wr
184     endif
185     x_h(4) = param_init%Tzero%sec
186     ! -------                                                  --------     .
187     ! initialisation
188     sumdelta_x=1.0e9_wr
189     k=0
190     ! -------                                                  --------     .
191     ! processus itératif
192     do while(con.and.(sumdelta_x.gt.1.e-4_wr))                              ! itération en cours
193       gamma=0.0_wr
194       k = k + 1
195       rechut=.true.                                                         ! écriture du nombre d'itération
196       if(present(chut)) then
197         if (chut) rechut=.false.
198       endif
199       if (rechut) then
200         if(nk.eq.1) then                                                    ! premier essai
201           write(prog,'(a51,i3)')" Geiger méthode - nb itération  :                ",k
202         elseif(nk.lt.4) then                                                ! nk essai
203           write(prog,'(a51,i3,a2,i2.2,a)')" Geiger méthode - nb itération  :                ", &
204           k," (",nk-1," fois divergente)"
205         else
206           write(prog,'(a51,i3,a)')" Geiger méthode - nb itération  :                ", &
```

```
        k," (non convergente)"
      endif
      write(*,'(a,a,$)') prog(1:76),char(13)                              ! affichage dynamique
    endif
    ! ------------------------------------------------------------------  .
    ! -------- remplissage de la matrice A                  ----------  . matrice des dérivées partielles
    nbdonnees=0
    do i=1,nbtps
    ! -------                                                ----------  .
      param_init%Tzero%sec=x_h(4)
      call basetime(param_init%Tzero)                                     ! reste en base 60/12/365 ...
      call derivpart(mat_el,datatps(i),param_init,acentroid,mod)          ! calcul les dérivées partielles
      if (datatps(i)%coefP.ne.4) then                                     ! ne prend pas en compte les plus mauvaises données
        nbdonnees=nbdonnees+1
        mat_A(nbdonnees,1)=mat_el%df_Dlon_P
        mat_A(nbdonnees,2)=mat_el%df_Dlat_P
        mat_A(nbdonnees,3)=mat_el%df_Dhypo_P
        mat_A(nbdonnees,4)=mat_el%df_Dto
      endif
      if (datatps(i)%andS.eq.'S') then                                    ! si ondes S existe
        if (datatps(i)%coefS.ne.4) then                                   ! ne prend pas en compte les plus mauvaises données
          nbdonnees=nbdonnees+1
          mat_A(nbdonnees,1)=mat_el%df_Dlon_S
          mat_A(nbdonnees,2)=mat_el%df_Dlat_S
          mat_A(nbdonnees,3)=mat_el%df_Dhypo_S
          mat_A(nbdonnees,4)=mat_el%df_Dto
        endif
      endif
    enddo
    ! ------------------------------------------------------------------  .
    ! -------- remplissage de la matrice A_t, transposée de A ----------  .
    do j=1,nbdonnees
      do i=1,4
        mat_A_t(i,j) = mat_A(j,i)
      enddo
    enddo
    ! ------------------------------------------------------------------  .
    ! -------- remplissage de la matrice At_A, produit de At et A -------  .
    At_A = matmul(mat_A_t,mat_A)
    ! ------------------------------------------------------------------  .
    ! -------- inversion de la matrice At_A                 ----------  .
    dim=4
    call inverse(At_A,invAt_A,dim)
    ! ------------------------------------------------------------------  .
    ! --------  remplissage de la matrice produit de invAt_A et A_t --  .
    invAt_A_At = matmul(invAt_A,mat_A_t)
    ! ------------------------------------------------------------------  .
    ! -------- remplissage de la matrice gamma              ----------  . difference entre temps théoriques et réels
    nbdonnees=0
    do i=1,nbtps
      ! --------                                             ----------  .
      one_data=datatps(i)
      ! --------                                             ----------  .
      if (mod=='I') then
        call tempsTheoDirectone(param_init,one_data,critique,acentroid)   ! calcul le temps théorique et la difference théoriques et réels
      elseif ((mod=='A').or.(mod=='S').or.(mod=='F').or.(mod=='C')) then
        call tempsTheoDirectone_AUTRE(param_init,one_data,pdfmoho,critique,mod) ! calcul le temps théorique et la difference théoriques et réels
      else
        write(*,*)'problème dans geiger : modèle inexistant !'
        stop
      endif
      ! --------                                             ----------  .
      if (datatps(i)%coefP.ne.4) then                                     ! ne prend pas en compte les plus mauvaises données
        nbdonnees=nbdonnees+1
        gamma(nbdonnees)=one_data%dTP
```

```
        endif
        ! -------                                                    --------     .
        if (datatps(i)%andS.eq.'S') then                                        ! si ondes S existe
          if (datatps(i)%coefS.ne.4) then                                       ! ne prend pas en compte les plus mauvaises données
            nbdonnees=nbdonnees+1
            gamma(nbdonnees)=one_data%dTS
          endif
        endif
        ! -------                                                    --------     .
        datatps(i)=one_data
        ! -------                                                    --------     .
      enddo
      ! ------------------------------------------------------------------     .
      call compute_misfitone (nbtps,datatps,onemisfit,xmin,xmax,CorH='C')
      onemisfit=onemisfit/real(nbtps,wr)  ! or : onemisfit=sumdelta_x
      ! -------- remplissage de la matrice delta_x                  --------     .
      delta_x = matmul(invAt_A_At,gamma)
      ! -------- itération suivante                                 --------     .
      x_h=x_h+delta_x
      param_init%Lon = x_h(1)
      param_init%Lat = x_h(2)
      ! -------                                                    --------     . séisme entre la surface et le moho
      if ((x_h(3).gt.0.0_wr).and.(x_h(3).lt.(pdfmoho-0.1_wr))) then
        param_init%Zhypo = x_h(3)
      end if
      ! -------                                                    --------     .
      param_init%Tzero%sec = x_h(4)
      call basetime(param_init%Tzero)                                           ! reste en base 60/12/365 ...
      ! sumdelta_x = abs(delta_x(1))+abs(delta_x(2))+abs(delta_x(3))+abs(delta_x(4)) ! résidus
      sumdelta_x = abs(delta_x(1))+abs(delta_x(2))+abs(delta_x(4)) ! résidus sans pdf ...
      ! -------- divergence                                         --------     .
      if ((k.gt.25).or.(IsNaN(sumdelta_x)).or.(sumdelta_x.gt.1.e5_wr) ) then
        con=.false.                                                             ! c'est pas grave, on repart pour un autre modele de terre ...
        !write(*,*)'divergence après',k,'itérations'
        sumdelta_x=1.0e9_wr
        onemisfit=1.e9_wr
      endif
      ! -------                                                    --------     .
    enddo                                                                       ! fin processus itératif
    ! -------                                                    --------     .
    deallocate (mat_A,mat_A_t,invAt_A_At,gamma)
    ! ------------------------------------------------------------------     .


    ! ------------------------------------------------------------------     .

      CONTAINS !

      subroutine inverse(a_ori,c,n)
        ! -------                                                  --------     .
        ! Inverse matrix, Method: Based on Doolittle LU factorization for Ax=b
        ! Alex G. December 2009
        ! ---------------------------------------------------------------     .
        ! source : http://ww2.odu.edu/~agodunov/computing/programs/book2/Ch06/Inverse.f90
        ! ---------------------------------------------------------------     .
        implicit none
        ! -------                                                  --------     .
        real(KIND=wr), intent (in) :: a_ori(n,n) ! array of coefficients for matrix A
        real(KIND=wr) :: a(n,n)
        real(KIND=wr), intent (out) :: c(n,n)    ! inverse matrix of A
        integer(KIND=wi), intent (in) :: n       ! dimension
        real(KIND=wr) :: L(n,n), U(n,n), b(n), d(n), x(n)
        real(KIND=wr) :: coeff
        integer(KIND=wi) i, j, k
        ! -------                                                  --------     .
```

```
337         a=a_ori
338         ! ------                                                  --------    .
339         ! step 0: initialization for matrices L and U and b
340         ! Fortran 90/95 aloows such operations on matrices
341         L=0.0_wr
342         U=0.0_wr
343         b=0.0_wr
344         do k=1, n-1                                                         ! step 1: forward elimination
345           do i=k+1,n
346             coeff=a(i,k)/a(k,k)
347             L(i,k) = coeff
348             do j=k+1,n
349               a(i,j) = a(i,j)-coeff*a(k,j)
350             end do
351           end do
352         end do
353         ! Step 2: prepare L and U matrices
354         ! L matrix is a matrix of the elimination coefficient
355         ! + the diagonal elements are 1.0
356         do i=1,n
357           L(i,i) = 1.0_wr
358         end do
359         ! U matrix is the upper triangular part of A
360         do j=1,n
361           do i=1,j
362             U(i,j) = a(i,j)
363           end do
364         end do
365         ! Step 3: compute columns of the inverse matrix C
366         do k=1,n
367           b(k)=1.0_wr
368           d(1) = b(1)
369           ! Step 3a: Solve Ld=b using the forward substitution
370           do i=2,n
371             d(i)=b(i)
372             do j=1,i-1
373               d(i) = d(i) - L(i,j)*d(j)
374             end do
375           end do
376           ! Step 3b: Solve Ux=d using the back substitution
377           x(n)=d(n)/U(n,n)
378           do i = n-1,1,-1
379             x(i) = d(i)
380             do j=n,i+1,-1
381               x(i)=x(i)-U(i,j)*x(j)
382             end do
383           x(i) = x(i)/u(i,i)
384           end do
385           ! Step 3c: fill the solutions x(n) into column k of C
386           do i=1,n
387             c(i,k) = x(i)
388           end do
389           b(k)=0.0_wr
390         end do
391         ! ------------------------------------------------------------    .
392       end subroutine inverse
393
394         ! ------------------------------------------------------------    .
395
396       subroutine derivpart(mat_el,one_data,p_i,acentroid,mod)
397         ! ------                                                  --------    .mh
398         ! calcul élements de la matrice A en dérivant les lois de temps théoriques des arrivées des ondes
399         ! dérivations non analytiques : dérivée par différences finies centrées d'ordre 2
400         ! ------                                                  --------    .
401         implicit none
```

```
402         ! --------                                                  ---------     .
403         type(parametre), intent (in) :: p_i                                       ! paramètres de l'inversion
404         type(dataone), intent (in) :: one_data                                    ! pour une donnée
405         type(elementmatrice), intent (out) :: mat_el                              ! élements de la matrice (dérivées partielles)
406         type (amoho_centroid), intent (in) :: acentroid
407         character (LEN=1), intent(in) :: mod
408         ! --------                                                  ---------     .
409         type(dataone) :: donnees                                                  ! pour cette donnée
410         type(parametre) :: param                                                  ! pour ce jeu de paramètres
411         ! --------                                                  ---------     .
412         real(KIND=wr), parameter :: h=0.00001_wr                                  ! précision
413         real(KIND=wr) :: d1p,d2p,d1s,d2s
414         real(KIND=wr) :: pdfmoho
415         logical :: critique
416         ! ------------------------------------------------------------------     .
417         donnees=one_data
418         param=p_i
419         ! ------------------------------------------------------------------     . Tzéro (temps derivé par le temps) :
420         mat_el%df_Dto = 1.0_wr
421         ! --------                                                  ---------     . Zhypo :
422         param%Zhypo = param%Zhypo - h
423         if (mod=='I') then
424           call tempsTheoDirectone(param,donnees,critique,acentroid)
425         elseif ((mod=='A').or.(mod=='S').or.(mod=='F').or.(mod=='C')) then
426           call tempsTheoDirectone_AUTRE(param,donnees,pdfmoho,critique,mod)
427         else
428           write(*,*)'problème dans derivpart : modèle inexistant !'
429           stop
430         endif
431         d1p = donnees%tpsTh%secP
432         d1s = donnees%tpsTh%secS
433         param%Zhypo = param%Zhypo + 2.0_wr*h
434         if (mod=='I') then
435           call tempsTheoDirectone(param,donnees,critique,acentroid)
436         else
437           call tempsTheoDirectone_AUTRE(param,donnees,pdfmoho,critique,mod)
438         endif
439         d2p = donnees%tpsTh%secP
440         d2s = donnees%tpsTh%secS
441         mat_el%df_Dhypo_P = (d2p - d1p)/(2.0_wr*h)
442         mat_el%df_Dhypo_S = (d2s - d1s)/(2.0_wr*h)
443         ! --------                                                  ---------     . Lon :
444         param=p_i
445         param%lon = param%lon - h
446         if (mod=='I') then
447           call tempsTheoDirectone(param,donnees,critique,acentroid)
448         else
449           call tempsTheoDirectone_AUTRE(param,donnees,pdfmoho,critique,mod)
450         endif
451         d1p = donnees%tpsTh%secP
452         d1s = donnees%tpsTh%secS
453         param%lon = param%lon + 2.0_wr*h
454         if (mod=='I') then
455           call tempsTheoDirectone(param,donnees,critique,acentroid)
456         else
457           call tempsTheoDirectone_AUTRE(param,donnees,pdfmoho,critique,mod)
458         endif
459         d2p = donnees%tpsTh%secP
460         d2s = donnees%tpsTh%secS
461         mat_el%df_Dlon_P = (d2p - d1p)/(2.0_wr*h)
462         mat_el%df_Dlon_S = (d2s - d1s)/(2.0_wr*h)
463         ! --------                                                  ---------     . Lat :
464         param=p_i
465         param%lat = param%lat - h
466         if (mod=='I') then
```

```
467           call tempsTheoDirectone(param,donnees,critique,acentroid)
468         else
469           call tempsTheoDirectone_AUTRE(param,donnees,pdfmoho,critique,mod)
470         endif
471         d1p = donnees%tpsTh%secP
472         d1s = donnees%tpsTh%secS
473         param%lat = param%lat + 2.0_wr*h
474         if (mod=='I') then
475           call tempsTheoDirectone(param,donnees,critique,acentroid)
476         else
477           call tempsTheoDirectone_AUTRE(param,donnees,pdfmoho,critique,mod)
478         endif
479         d2p = donnees%tpsTh%secP
480         d2s = donnees%tpsTh%secS
481         mat_el%df_Dlat_P = (d2p - d1p)/(2.0_wr*h)
482         mat_el%df_Dlat_S = (d2s - d1s)/(2.0_wr*h)
483         !-------------------------------------------------------------      .
484       end subroutine derivpart
485
486   end subroutine geiger
487
488 END MODULE invGEIGER
489
490
491
492 !*****************************************************************      .
493 !*****************************************************************      .
```

2.14 SRC/MOD/tracer_rais.f90

```
1  ! octobre 2015
2  ! tracer de rais 1D, couches tabulaires et homogènes
3  ! module non utilise pour la recherche du modèle de Terre dans CHE
4  !*****************************************************************      .
5  !-------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------      .
6  !*****************************************************************      .
7  !-------------------------------------------------------------------      .
8
9  MODULE ray_tracing
10
11    use modparam
12
13    implicit none
14
15    private
16
17    public  :: tracerays
18
19    !-------------------------------------------------------------------      .
20    real (kind=wr), parameter :: eps = 0.000001_wr                           ! le chouïa
21    !-------------------------------------------------------------------      .
22
23 !-------------------------------------------------------------------      .
24 CONTAINS
25 !-------------------------------------------------------------------      .
26
27   subroutine tracerays(distancepi,distancehypo,dcritiqueH,pfdseisme,altidudesta,lon, &
28                        lat,pdfmoho,tdirectP,trefP,tdirectS,trefS,modele)
29     !--------                                                  ---------      . mh
30     ! trace les rais sismiques des ondes Pg,Sg,Pn,Sn pour un couple station-séisme,
31     ! avec calcul du temps de parcours et un modèle de Terre donnée (modele={'A';'S';'F';'C'})
32     !--------                                                  ---------      .
33     implicit none
34     !--------                                                  ---------      .
35     real (kind=wr), intent(in) :: distancepi,pfdseisme                       ! pour un couple station-séisme
```

```
36      real (kind=wr), intent(in) :: altidudesta,lon,lat                    ! pour un couple station-séisme
37      real(KIND=wr), intent (out) :: pdfmoho
38      real (kind=wr), intent(out) :: tdirectP,trefP,tdirectS,trefS         ! temps parcours ondes directes et réfractées
39      real (kind=wr), intent(out) :: distancehypo,dcritiqueH
40      character (LEN=1), intent(in), optional :: modele
41      ! ------------------------------------------------------------------ .
42      integer (kind=wi) :: nbl                                             ! nombre couche du modèle
43      integer (kind=wi) :: nc                                              ! couche dans laquelle est le séisme
44      ! -------                                                 --------     .
45      integer (kind=wi) :: i
46      ! -------                                                 --------     .
47      real (kind=wr), allocatable :: vP(:),vS(:)                           ! vitesse P et S dans chaque couche
48      real (kind=wr), allocatable :: h(:)                                  ! profondeur cumulés du toit la ieme couche
49      real (kind=wr), allocatable :: hpuiss(:)                             ! puissance la ieme couche
50      real (kind=wr), allocatable :: X(:),X2(:,:)                          ! distance, projetée horizontalement, parcourue dans chaque couche
51      logical, parameter :: plotfigures=.false.
52      ! ------------------------------------------------------------------ . choix d'un modèle de terre :
53      if(present(modele)) then
54        ! -------                                               --------     .
55        if (modele=='A') then
56          ! call ModTerrArroucau(nbl,vP,vS,h,lon,lat,forcelettre='a')
57          call ModTerrArroucau(nbl,vP,vS,h,lon,lat,pdfmoho)
58        elseif (modele=='S') then
59          call ModTerrSiHex_Haslach(nbl,vP,vS,h,pdfmoho)
60        elseif (modele=='F') then
61          call ModTerr_fun(nbl,vP,vS,h,pdfmoho)
62        elseif (modele=='C') then
63          call ModTerr_fun(nbl,vP,vS,h,pdfmoho)
64        else
65          write(*,*)'problème dans tracerays2, le modèle de terre : ',modele,' n''existe pas '
66          stop
67        endif
68        ! -------                                               --------     . par défaut
69      else
70        call ModTerrSiHex_Haslach(nbl,vP,vS,h,pdfmoho)
71      endif
72      ! ------------------------------------------------------------------ . VÉRIF modèle de Terre
73      do i=2,nbl
74        if(h(i).le.h(i-1)) then
75          write(*,*)'problème dans tracerays2 : ! ordre profondeur ', h
76          stop
77        endif
78      enddo
79      ! -------                                                 --------     .
80      if (altidudesta.gt.h(2)) then
81        write(*,*)'problème dans tracerays2 : ! altidude de la station non adapté au modèle de terre'
82        stop
83      endif
84      ! -------                                                 --------     . dans quelle couche est le séisme ?
85      do i=1,nbl
86        if(pfdseisme.gt.h(i)) nc=i
87      enddo
88      if (nc.ge.nbl) then
89        ! write(*,*)'problème dans tracerays2 : le séisme est trop profond'
90        tdirectP=-1.e9_wr
91        trefP=1.e9_wr
92        tdirectS=1.e9_wr
93        trefS=1.e9_wr
94        distancehypo=0.0_wr
95        dcritiqueH=0.0_wr
96      else
97        ! -------                                               --------     .
98        allocate(hpuiss(nbl),X(nbl))
99        ! ------------------------------------------------------------------ . puissance de chaque couche
100       hpuiss=0.0_wr
```

```
101       do i=1,nc
102         if(i==nc) then
103           hpuiss(i)=pfdseisme-h(i)
104         else
105           hpuiss(i)=h(i+1)-h(i)
106         endif
107       enddo
108       hpuiss(1)=hpuiss(1)-altidudesta
109       h(1)=altidudesta
110       ! -------                                                  --------     .
111       distancehypo=0.0_wr
112       ! ---------------------------------------------------------------     . pour onde Pg
113       call traceondeGKIM(distancepi,pfdseisme,altidudesta,nbl,nc,vP,hpuiss,X,distancehypo,tdirectP)
114       if ((tdirectP.gt.0.0_wr).and.(plotfigures)) then
115         call printAray(int(distancepi*100.0_wr),nbl,nc,h,X,X2,pfdseisme,distancepi,altidudesta,lettre='Pg')
116       endif
117       ! -------                                                  --------     . pour onde Pn
118       call traceondeN(distancepi,pfdseisme,altidudesta,nbl,nc,vP,h,X,X2,distancehypo,dcritiqueH,trefP)
119       if ((trefP.gt.0.0_wr).and.(plotfigures)) then
120         call printAray(int(distancepi*100.0_wr),nbl,nc,h,X,X2,pfdseisme,distancepi,altidudesta,lettre='Pn')
121       endif
122       ! -------                                                  --------     . pour onde Sg
123       call traceondeGKIM(distancepi,pfdseisme,altidudesta,nbl,nc,vS,hpuiss,X,distancehypo,tdirectS)
124       if ((tdirectS.gt.0.0_wr).and.(plotfigures)) then
125         call printAray(int(distancepi*100.0_wr),nbl,nc,h,X,X2,pfdseisme,distancepi,altidudesta,lettre='Sg')
126       endif
127       ! -------                                                  --------     .  pour onde Sn
128       call traceondeN(distancepi,pfdseisme,altidudesta,nbl,nc,vS,h,X,X2,distancehypo,dcritiqueH,trefS)
129       if ((trefS.gt.0.0_wr).and.(plotfigures)) then
130         call printAray(int(distancepi*100.0_wr),nbl,nc,h,X,X2,pfdseisme,distancepi,altidudesta,lettre='Sn')
131       endif
132       ! -------                                                  --------     .
133       deallocate(hpuiss,X,X2)
134     endif
135     ! ---------------------------------------------------------------     .
136     deallocate(vP,vS,h)
137     ! ---------------------------------------------------------------     .
138   end subroutine tracerays
139
140 ! ---------------------------------------------------------------     .
141
142   subroutine traceondeDichotomie(distancepi,nbl,nc,v,hpuiss,X,theta)
143     ! -------                                                  --------     . mh
144     ! trace onde directe, recherche de theta par dichotomie
145     ! (on retire ici un quart seulement de l'espace, pas la moitié)
146     ! -------                                                  --------     .
147     implicit none
148     ! -------                                                  --------     .
149     real (kind=wr), intent(in) :: distancepi                              ! pour un couple station-séisme
150     real (kind=wr), intent(in) :: hpuiss(nbl)
151     integer (kind=wi), intent(in) :: nbl,nc
152     real (kind=wr), intent(in) :: v(nbl)
153     ! -------                                                  --------     .
154     real (kind=wr), intent(out) :: X(nbl)
155     real (kind=wr), intent(out) :: theta
156     ! ---------------------------------------------------------------     .
157     integer (kind=wi) :: i
158     ! -------                                                  --------     .
159     real (kind=wr) :: thetamin,thetamax                                   ! takeoff angle (clockwise from the upward vertical direction)
160     real (kind=wr) :: sommeX,sommeXmin,sommeXmax                          ! somme des Xi
161     ! ---------------------------------------------------------------     .
162     thetamin=0.0_wr*pi/180.0_wr+eps/10.0_wr
163     thetamax=90.0_wr*pi/180.0_wr-eps/10.0_wr
164     ! ---------------------------------------------------------------     .
165     i=0
```

```
    do while ((abs(sommeX-distancepi).gt.eps).and.(i.lt.20000).and.(thetamin.ne.thetamax))
      i=i+1
      theta=(thetamin+thetamax)/2._wr
      call CalcX(nbl,nc,hpuiss,(thetamin+theta)/2._wr,X,v,sommeXmin)
      call CalcX(nbl,nc,hpuiss,(theta+thetamax)/2._wr,X,v,sommeXmax)
      call CalcX(nbl,nc,hpuiss,theta,X,v,sommeX)
      if(abs(sommeXmin-distancepi).le.abs(sommeXmax-distancepi)) then
        thetamax=(thetamax+theta)/2._wr
      else
        thetamin=(theta+thetamin)/2._wr
      endif
    enddo
    !-------------------------------------------------------------------      .
  end subroutine traceondeDichotomie

!-------------------------------------------------------------------      .

  subroutine traceondeGKIM(distancepi,pfdseisme,altidudesta,nbl,nc,v,hpuiss,X,distancehypo,temps)
    !-------                                                 --------      . mh
    ! ray tracing : d'après Kim and Baag (2002) :
    ! Rapid and Accurate Two-Point Ray Tracing Based on
    ! a Quadratic Equation of Takeoff Angle in Layered Media with Constant (BSSA)
    !-------                                                 --------      .
    implicit none
    !-------                                                 --------      .
    real (kind=wr), intent(in) :: distancepi,pfdseisme,altidudesta          ! pour un couple station-séisme
    real (kind=wr), intent(in) :: hpuiss(nbl)
    integer (kind=wi), intent(in) :: nbl,nc
    real (kind=wr), intent(inout) :: v(nbl)
    !-------                                                 --------      .
    real (kind=wr), intent(out) :: X(nbl),distancehypo
    real (kind=wr), intent(out) :: temps                                    ! temps de parcours des ondes
    !-------------------------------------------------------------------      .
    integer (kind=wi) :: i,j
    !-------                                                 --------      .
    real (kind=wr) :: vx(nbl),vt(nbl),vxtot                                 ! vérif ...
    real (kind=wr) :: thetainit,theta                                       ! takeoff angle (clockwise from the upward vertical direction)
    real (kind=wr) :: deltatheta1,deltatheta2                               ! incrément de theta, angle (2 solutions équation 2nd degré)
    real (kind=wr) :: sommeX                                                ! somme des Xi
    !-------------------------------------------------------------------      .
    sommeX=99999.999_wr
    !-------------------------------------------------------------------      .
    call initialvalue1(nbl,nc,v,hpuiss,thetainit,distancepi,pfdseisme,altidudesta)
    !-------------------------------------------------------------------      .
    ! test pour plusieurs valeurs initales si besoin, avec incrément tous les dégrés, jusqu'à convergence
    j=-1
    do while ((abs(sommeX-distancepi).gt.eps).and.(j.lt.180))
      !-------------------------------------------------------------------      .
      j=j+1
      theta=thetainit+real(j,wr)/0.5_wr*pi/180.0_wr
      !-------------------------------------------------------------------      .
      i=1
      call CalcX(nbl,nc,hpuiss,theta,X,v,sommeX)
      do while ((abs(sommeX-distancepi).gt.eps).and.(i.lt.20))
        i=i+1
        call CalcDeltaTheta(nbl,v,hpuiss,theta,X,deltatheta1,deltatheta2,distancepi,nc)
        theta=theta+deltatheta2
        do while((theta*180.0_wr/pi).ge.89.9_wr)
          theta=theta-1.0_wr
        enddo
        call CalcX(nbl,nc,hpuiss,theta,X,v,sommeX)
        !write(*,*)i,'DELTA : -- ',sommeX-distancepi,deltatheta1*180.0_wr/pi,theta*180.0_wr/pi
      enddo
      !-------------------------------------------------------------------      .
      if (isnan(sommeX))sommeX=99999.999_wr
```

```
231       ! ------------------------------------------------------------------      .
232       if (j.gt.179) then
233         call traceondeDichotomie(distancepi,nbl,nc,v,hpuiss,X,theta)        ! trace onde directe, recherche de theta par dichotomie
234         call CalcX(nbl,nc,hpuiss,theta,X,v,sommeX)
235         j=200
236       endif
237     enddo
238     ! ------------------------------------------------------------------      .VERIF
239     temps=0.0_wr
240     vxtot=0.0_wr
241     distancehypo=0.0_wr
242     vt(nc)=theta
243     do i=nc,1,-1
244       if ((i-1).gt.0) vt(i-1)=asin(sin(vt(i))*v(i-1)/v(i))
245       vx(i)=tan(vt(i))*hpuiss(i)
246       vxtot=vxtot+vx(i)
247       temps=temps+(hpuiss(i)/cos(vt(i)))/v(i)
248       distancehypo=distancehypo+hpuiss(i)/cos(vt(i))
249     enddo
250     ! ------------------------------------------------------------------      .pb cos(x)
251     if (isnan(temps))then
252       v=0.0_wr
253       X=0.0_wr
254       distancehypo=0.0_wr
255       temps=0.0_wr
256     endif
257     ! ------------------------------------------------------------------      .
258     ! write(*,'(a,f15.1,2f15.9,f15.1,f20.10)')'K',distancepi,vxtot-distancepi,sommeX-distancepi,temps,theta*180.0_wr/pi
259     ! ------------------------------------------------------------------      .
260   end subroutine traceondeGKIM
261
262 ! ------------------------------------------------------------------------      .
263
264   subroutine traceondeN(distancepi,pfdseisme,altidudesta,nbl,nc,v,h,X,Xplot,distancehypo,dcritiqueH,temps)
265     ! -------                                                  ---------      . mh
266     ! ray tracing : onde réfracté au moho (toujours la dernière couche du modèle de terre !)
267     ! -------                                                  ---------      .
268     implicit none
269     ! -------                                                  ---------      .
270     real (kind=wr), intent(in) :: distancepi,pfdseisme,altidudesta           ! pour un couple station-séisme
271     real (kind=wr), intent(in) :: h(nbl)
272     integer (kind=wi), intent(in) :: nbl,nc
273     real (kind=wr), intent(inout) :: v(nbl)
274     ! -------                                                  ---------      .
275     real (kind=wr), intent(inout), allocatable :: Xplot(:,:)
276     real (kind=wr), intent(out) :: X(nbl),distancehypo,dcritiqueH
277     real (kind=wr), intent(out) :: temps                                      ! temps de parcours des ondes
278     ! ------------------------------------------------------------------      .
279     integer (kind=wi) :: i,j
280     integer (kind=wi) :: nm                                                   ! couche de la réfraction
281     ! -------                                                  ---------      .
282     real (kind=wr) :: theta(nbl),hpuiss(nbl),hequi(nbl),xcumule
283     ! ------------------------------------------------------------------      .
284     nm=nbl
285     ! ------------------------------------------------------------------      . modèle équivalent pour la première réfracté
286     ! ne prend pas encore en compte : altidudesta
287     ! ------------------------------------------------------------------      .
288     hequi=0.0_wr
289     hpuiss=0.0_wr
290     do i=1,nbl-1
291       hpuiss(i)=h(i+1)-h(i)
292     enddo
293     ! -------                                                  ---------      .
294     distancehypo=0.0_wr
295     ! -------                                                  ---------      .
```

```
    do i=1,nm-1
      if (i.lt.nc) then
        hequi(i)=hpuiss(i)
      elseif(i==nc) then
        hequi(i)=hpuiss(i)+(h(nc+1)-pfdseisme)
      else ! (i.gt.nc)
        hequi(i)=2.0_wr*hpuiss(i)
      endif
    enddo
    ! ------------------------------------------------------------------ .
    theta(nm)=pi/2.0_wr
    do i=nm-1,1,-1
      theta(i)=asin(sin(theta(i+1))*v(i)/v(i+1))
    enddo
    ! ------------------------------------------------------------------ .
    xcumule=0.0_wr
    dcritiqueH=0.0_wr
    X(nm)=0.0_wr
    do i=1,nm-1
      dcritiqueH=dcritiqueH+hequi(i)/cos(theta(i))
      X(i)=tan(theta(i))*hequi(i)
      X(nm)= X(nm)+X(i)
    enddo
    X(nm)= distancepi-X(nm)
    ! ------------------------------------------------------------------ .
    if (allocated(Xplot)) deallocate(Xplot)
    allocate(Xplot(2*int(nbl,4)-nc,2))
    Xplot=0.0_wr
    j=1
    Xplot(j,1)=tan(theta(nc))*(h(nc+1)-pfdseisme)
    Xplot(j,2)=h(nc+1)
    do i=nc+1,nm-1
      j=j+1
      Xplot(j,1)=Xplot(j-1,1)+tan(theta(i))*hpuiss(i)
      Xplot(j,2)=h(i+1)
    enddo
    j=j+1
    Xplot(j,1)=Xplot(j-1,1)+X(nm)
    Xplot(j,2)=Xplot(j-1,2)
    do i=nm-1,1,-1
      j=j+1
      Xplot(j,1)=Xplot(j-1,1)+tan(theta(i))*hpuiss(i)
      Xplot(j,2)=h(i)
    enddo
    ! ------------------------------------------------------------------ .
    distancehypo=dcritiqueH+X(nm)
    temps=0.0_wr
    do i=1,nm-1
      temps=temps+(hequi(i)/cos(theta(i)))/v(i)
    enddo
    if (distancehypo.ge.dcritiqueH) then
        temps=temps+X(nm)/v(nm)
    else
      temps=0.0_wr
      Xplot=0.0_wr
      X=0.0_wr
    endif
    ! ------------------------------------------------------------------ .pb
    if (isnan(temps))then
      temps=0.0_wr
      Xplot=0.0_wr
      X=0.0_wr
    endif
    ! ------------------------------------------------------------------ .
  end subroutine traceondeN
```

```
361
362 ! ------------------------------------------------------------------      .
363
364   subroutine initialvalue1(nbl,nc,vP,h,theta,distancepi,pfdseisme,altidudesta)
365     ! -------                                                  ---------     . mh
366     ! inital value 1, primary estimation : kinematic properties of rays
367     ! -------                                                  ---------     .
368     implicit none
369     ! -------                                                  ---------     .
370     real (kind=wr), intent(in) ::vP(nbl),h(nbl),distancepi,pfdseisme,altidudesta
371     integer (KIND=wi), intent(in) :: nbl,nc
372     real (kind=wr), intent(out) :: theta
373     ! -------                                                  ---------     .
374     integer (KIND=wi) :: i
375     real (kind=wr) :: thetaI,thetaH,thetaM
376     real (kind=wr) :: vmoy,vmax,S,si(nc)
377     ! ------------------------------------------------------------------     .
378     thetaH=atan(distancepi/abs(pfdseisme-altidudesta))
379     ! -------                                                  ---------     . vitesse maximale
380     vmax=-99.99_wr
381     do i=1,nc
382       if(vmax.le.vP(i)) vmax=vP(i)
383     enddo
384     !if(vmax.ne.vP(nc)) write(*,*)'problème dans initialvalue : vitesse on max',vmax,vP(nc)
385     ! -------                                                  ---------     .
386     !vmax=vP(nbl)
387     ! -------                                                  ---------     .
388     vmoy=0.0_wr
389     S=0.0_wr
390     si=0.0_wr
391     ! -------                                                  ---------     .
392     do i=1,nc
393       thetaI=asin(vP(i)/vmax*sin(thetaH))
394       si(i)=h(i)/cos(thetaI)
395       S=S+si(i)
396       vmoy=vmoy+vP(i)*si(i)
397     enddo
398     vmoy=vmoy/S
399     ! ------------------------------------------------------------------     .
400     if((vmax/vmoy*sin(thetaH)).le.1.0_wr) then
401       thetaM=asin(vmax/vmoy*sin(thetaH))
402     else
403       thetaM=thetaH
404     endif
405     ! -------                                                  ---------     .
406     theta=(thetaH+thetaM)/2.0_wr
407     ! ------------------------------------------------------------------     .
408     if (isnan(theta))theta=thetaH*1.1_wr
409     ! -------                                                  ---------     .
410     do while((theta*180.0_wr/pi).ge.90.0_wr)
411       theta=theta-1.5_wr*pi/180.0_wr
412     enddo
413     !write(*,*)'initial value 1 : ',thetainit*180.0_wr/pi
414     ! ------------------------------------------------------------------     .
415   end subroutine initialvalue1
416
417 ! ------------------------------------------------------------------      .
418
419   subroutine CalcDeltaTheta(nbl,vP,h,theta,X,deltatheta1,deltatheta2,distancepi,nc)
420     ! -------                                                  ---------     . mh
421     ! CalcDeltaTheta
422     ! -------                                                  ---------     .
423     implicit none
424     ! -------                                                  ---------     .
425     real (kind=wr), intent(in) ::vP(nbl),h(nbl),distancepi,theta,X(nbl)
```

```
426     integer (KIND=wi), intent(in) :: nbl,nc
427     real (kind=wr), intent(out) :: deltatheta1,deltatheta2
428     ! ------                                                    --------      .
429     integer (KIND=wi) :: i
430     real (kind=wr) :: A,B,C
431     real (kind=wr) :: lambda(nbl)
432     real (kind=wr) :: racine
433     real (kind=wr) :: sommeXi
434     ! ------------------------------------------------------------------      .
435     A=0.0_wr
436     B=0.0_wr
437     sommeXi=0.0_wr
438     ! ------                                                    --------      .
439     do i=1,nc
440       ! ------                                                  --------      .
441       lambda(i)=vP(i)/vP(nc)
442       racine=sqrt(1.0_wr-(lambda(i)**2.0_wr)*(sin(theta)**2.0_wr))
443       ! ------                                                  --------      .
444       A=A+0.5_wr*h(i)*(3.0_wr*(lambda(i)**3.0_wr)*(cos(theta)**2.0_wr)*sin(theta)/(racine**5.0_wr) &
445         - lambda(i)*sin(theta)/(racine**3.0_wr))
446       ! ------                                                  --------      .
447       B=B+h(i)*lambda(i)*cos(theta)/(racine**3.0_wr)
448       ! ------                                                  --------      .
449       sommeXi=sommeXi+X(i)
450       C=sommeXi-distancepi
451       ! ------                                                  --------      .
452     enddo
453     ! ------                                                    --------      .
454     if((B**2.0_wr-4.0_wr*A*C).gt.0.0_wr) then
455       deltatheta1=(-B-sqrt(B**2.0_wr-4.0_wr*A*C))/(2.0_wr*A)
456       deltatheta2=(-B+sqrt(B**2.0_wr-4.0_wr*A*C))/(2.0_wr*A)
457     else
458       deltatheta1=0.0_wr
459       deltatheta2=0.0_wr
460     endif
461     ! ------------------------------------------------------------------      .
462   end subroutine CalcDeltaTheta
463
464 ! ------------------------------------------------------------------------      .
465
466   subroutine CalcX(nbl,nc,h,theta,X,v,sommeX)
467     ! ------                                                    --------      . mh
468     ! CalcX
469     ! ------                                                    --------      .
470     implicit none
471     ! ------                                                    --------      .
472     real (kind=wr), intent(in) ::h(nbl),v(nbl),theta
473     integer (KIND=wi), intent(in) :: nbl,nc
474     real (kind=wr), intent(out) :: X(nbl),sommeX
475     ! ------                                                    --------      .
476     integer (KIND=wi) :: i
477     real (kind=wr) :: lambda(nbl),racine                                    !
478     ! ------------------------------------------------------------------      .
479     sommeX=0.0_wr
480     X=0.0_wr
481     do i=1,nc
482       lambda(i)=v(i)/v(nc)
483       racine=sqrt(1.0_wr-(lambda(i)**2.0_wr)*(sin(theta)**2.0_wr))
484       X(i)=h(i)*lambda(i)*sin(theta)/racine
485       sommeX=sommeX+X(i)
486     enddo
487     ! ------------------------------------------------------------------      .
488   end subroutine CalcX
489
490 ! ------------------------------------------------------------------------      .
```

```
491
492 subroutine printAray(k,nbl,nc,h,X,Xplot,pfdseisme,distancepi,altidudesta,lettre)
493   ! _______                                                    ________    . mh
494   ! printAray
495   ! _______                                                    ________    .
496   implicit none
497   ! _______                                                    ________    .
498   real (kind=wr), intent(in) :: h(nbl),X(nbl),pfdseisme,distancepi,altidudesta
499   integer (KIND=wi), intent(in) :: k,nbl,nc
500   character (LEN=2), intent(in), optional :: lettre
501   real (kind=wr), intent(in), allocatable :: Xplot(:,:)
502   ! _______                                                    ________    .
503   real (kind=wr) :: X2(nbl)
504   integer (KIND=wi) :: i,ok
505   character (LEN=5) :: numberchaine
506   ! ------------------------------------------------------------------    .
507   write(numberchaine(1:5),'(i5.2)')k
508   ! ------------------------------------------------------------------    .
509   if(present(lettre)) then
510     open(105,FILE='ray-'//lettre//'-'//trim(adjustl(numberchaine))//'.d',status='replace',iostat = ok)
511   else
512     open(105,FILE='ray-'//trim(adjustl(numberchaine))//'.d',status='replace',iostat = ok)
513   endif
514   if (ok .ne. 0) then
515     write(*,*)'problème dans printAray : le fichier n''existe pas '
516     stop
517   endif
518   ! _______                                                    ________    .
519   write(105,'(2f15.7)')0.0_wr,pfdseisme
520   ! _______                                                    ________    .
521   if(present(lettre)) then
522   if(lettre(2:)=='n') then
523     ! _______                                                  ________    .
524     do i=1,2*int(nbl-nc,4)
525       write(105,*)Xplot(i,1),Xplot(i,2)
526     enddo
527     ! _______                                                  ________    .
528     elseif(lettre(2:)=='g') then
529       ! _______                                                ________    .
530       X2(nbl)=X(nbl)
531       do i=nbl-1,1,-1
532         X2(i)=X2(i+1)+X(i)
533       enddo
534       do i=nc,1,-1
535         write(105,*)x2(i),h(i)
536       enddo
537       ! _______                                                ________    .
538     else
539       write(*,*)'problème dans printAray : onde ni "g" ni "n" -> ',lettre
540       stop
541      ! _______                                                 ________    .
542     endif
543   else
544     ! _______                                                  ________    .
545     X2(nbl)=X(nbl)
546     do i=nbl-1,1,-1
547       X2(i)=X2(i+1)+X(i)
548     enddo
549     do i=nc,1,-1
550       write(105,*)x2(i),h(i)
551     enddo
552     ! _______                                                  ________    .
553   endif
554   ! _______                                                    ________    .
555   close(105)
```

```
556     ! ------------------------------------------------------------------      .
557     !if(k==0) then
558       open(99, FILE = 'mod.d',status='replace',iostat = ok)
559       do i=1,nbl
560         write(99,*)">"
561         write(99,*)-distancepi-10.0_wr,h(i)
562         write(99,*)distancepi+10.0_wr,h(i)
563       enddo
564       close(99)
565       ! ------------------------------------------------------------------      .
566       open(101, FILE = 'plot.sh',status='replace',iostat = ok)
567       write(101,'(a)')"gmtset LABEL_FONT_SIZE 15"
568       write(101,'(a)')'gmtset HEADER_FONT_SIZE 15'
569       write(101,'(a)')"gmtset ANNOT_FONT_PRIMARY Times-Roman"
570       write(101,'(a)')"gmtset ANNOT_FONT_SECONDARY Times-Roman"
571       write(101,'(a)')"gmtset PAPER_MEDIA A3"
572       write(101,'(a)')"gmtset TIME_LANGUAGE FR"
573       write(101,'(a)')"gmtset CHAR_ENCODING ISOLatin1+"
574       ! ------------------------------------------------------------------      .
575       write(101,'(a)')"geoproj=-JX12i/-1.2i"
576       write(101,'(a)')"geozone=-R-55/550/-7/77"
577       write(101,'(2a)')"psxy $geozone $geoproj mod.d -Wthinnest,green -Ba100f50/a10f2NsWeg100  -m -K > file.ps"
578       write(101,*)"echo '",0.0_wr,pfdseisme,"' | psxy $geozone $geoproj -Sa0.25 -Gorange -Wthinnest,red -O -K >> file.ps"
579       write(101,*)"echo '",distancepi,altidudesta,"' | psxy $geozone $geoproj -St0.25 -Gblue -Wthinnest,blue -O -K -N >> file.ps"
580       write(101,'(2a)')"psxy $geozone $geoproj ray-Pg-*.d -Wthinnest,blue,-- -O -K >> file.ps"
581       write(101,'(2a)')"psxy $geozone $geoproj ray-Sg-*.d -Wthinnest,red,-- -O -K >> file.ps"
582       write(101,'(2a)')"psxy $geozone $geoproj ray-Pn-*.d -Wthinnest,blue,-. -O -K >> file.ps"
583       write(101,'(2a)')"psxy $geozone $geoproj ray-Sn-*.d -Wthinnest,red,-. -O >> file.ps"
584       write(101,*)"ps2raster file.ps -Tf -A -P "
585       ! ------------------------------------------------------------------      .
586       close(101)
587     !endif
588     ! ------------------------------------------------------------------      .
589   end subroutine printAray
590
591 ! ------------------------------------------------------------------      .
592
593   subroutine ModTerrArroucau(nbl,vP,vS,h,lon,lat,pdfmoho,forcelettre)
594     ! -------                                                 ---------      . mh
595     ! Modèles de Terre Arroucau, phD 2006
596     ! -------                                                 ---------      .
597     implicit none
598     ! -------                                                 ---------      .
599     real (kind=wr), intent(inout), allocatable :: h(:),vP(:),vS(:)
600     integer (KIND=wi), intent(out) :: nbl
601     real(KIND=wr), intent (out) :: pdfmoho
602     ! -------                                                 ---------      .
603     real (kind=wr), intent(in) :: lon,lat
604     character (LEN=1), intent(in), optional :: forcelettre
605     ! -------                                                 ---------      .
606     character (LEN=1) :: lettre
607     ! ------------------------------------------------------------------      .
608     if (allocated(vP)) deallocate(vP)
609     if (allocated(vS)) deallocate(vS)
610     if (allocated(h)) deallocate(h)
611     ! ------------------------------------------------------------------      .
612     nbl=9
613     allocate(vP(nbl),vS(nbl),h(nbl))
614     ! ------------------------------------------------------------------      .
615     if(present(forcelettre)) then
616       lettre=forcelettre
617     else
618       if ((lon.ge.-6.0_wr).and.(lon.le.-1.5_wr).and.(lat.ge.45.0_wr).and.(lat.le.47.0_wr)) then
619         lettre='a'
620       elseif ((lon.ge.-6.0_wr).and.(lon.le.-3.0_wr).and.(lat.ge.47.0_wr).and.(lat.le.48.0_wr)) then
```

```
        lettre='b'
      elseif ((lon.ge.-6.0_wr).and.(lon.le.-3.0_wr).and.(lat.ge.48.0_wr).and.(lat.le.50.0_wr)) then
        lettre='c'
      elseif ((lon.ge.-3.0_wr).and.(lon.le.0.0_wr).and.(lat.ge.49.0_wr).and.(lat.le.50.0_wr)) then
        lettre='d'
      elseif ((lon.ge.-1.5_wr).and.(lon.le.0.0_wr).and.(lat.ge.48.0_wr).and.(lat.le.49.0_wr)) then
        lettre='d'
      elseif ((lon.ge.-3.0_wr).and.(lon.le.-1.5_wr).and.(lat.ge.48.0_wr).and.(lat.le.49.0_wr)) then
        lettre='e'
      elseif ((lon.ge.-3.0_wr).and.(lon.le.-1.5_wr).and.(lat.ge.47.0_wr).and.(lat.le.48.0_wr)) then
        lettre='f'
      elseif ((lon.ge.-1.0_wr).and.(lon.le.0.0_wr).and.(lat.ge.47.0_wr).and.(lat.le.48.0_wr)) then
        lettre='g'
      elseif ((lon.ge.-1.0_wr).and.(lon.le.0.0_wr).and.(lat.ge.46.0_wr).and.(lat.le.47.0_wr)) then
        lettre='h'
      elseif ((lon.ge.-1.0_wr).and.(lon.le.1.0_wr).and.(lat.ge.45.0_wr).and.(lat.le.46.0_wr)) then
        lettre='i'
      elseif ((lon.ge.0.0_wr).and.(lon.le.1.0_wr).and.(lat.ge.46.0_wr).and.(lat.le.47.0_wr)) then
        lettre='j'
      elseif ((lon.ge.0.0_wr).and.(lon.le.1.0_wr).and.(lat.ge.47.0_wr).and.(lat.le.50.0_wr)) then
        lettre='k'
      else
        !write(*,*)'problème dans ModTerrArroucau, le séisme est hors zone (a-k)'
        lettre='z'
      endif
    endif
    ! ------------------------------------------------------------------     .
    !write(*,*) 'LETTRE    ',lettre
    ! ------------------------------------------------------------------     .
    h(1)=0.0_wr
    h(2)=4.0_wr
    h(3)=8.0_wr
    h(4)=12.0_wr
    h(5)=16.0_wr
    h(6)=20.0_wr
    h(7)=24.0_wr
    h(8)=28.0_wr
    h(9)=32.0_wr
    ! ------                                                 --------     .
    if (lettre=='a') then
      vP(1)=6.0_wr
      vP(2)=6.1_wr
      vP(3)=6.1_wr
      vP(4)=6.1_wr
      vP(5)=6.1_wr
      vP(6)=6.8_wr
      vP(7)=7.0_wr
      vP(8)=7.8_wr
      vP(9)=8.0_wr
    elseif (lettre=='b') then
      vP(1)=6.0_wr
      vP(2)=6.0_wr
      vP(3)=6.0_wr
      vP(4)=6.1_wr
      vP(5)=6.1_wr
      vP(6)=6.2_wr
      vP(7)=7.0_wr
      vP(8)=7.4_wr
      vP(9)=8.1_wr
    elseif (lettre=='c') then
      vP(1)=6.0_wr
      vP(2)=6.0_wr
      vP(3)=6.1_wr
      vP(4)=6.1_wr
      vP(5)=6.2_wr
```

```
      vP(6)=6.2_wr
      vP(7)=6.9_wr
      vP(8)=7.3_wr
      vP(9)=8.1_wr
    elseif (lettre=='d') then
      vP(1)=6.1_wr
      vP(2)=6.1_wr
      vP(3)=6.1_wr
      vP(4)=6.1_wr
      vP(5)=6.3_wr
      vP(6)=6.5_wr
      vP(7)=7.1_wr
      vP(8)=7.8_wr
      vP(9)=8.2_wr
    elseif (lettre=='e') then
      vP(1)=6.0_wr
      vP(2)=6.0_wr
      vP(3)=6.1_wr
      vP(4)=6.2_wr
      vP(5)=6.3_wr
      vP(6)=6.4_wr
      vP(7)=6.8_wr
      vP(8)=7.3_wr
      vP(9)=8.1_wr
    elseif (lettre=='f') then
      vP(1)=6.0_wr
      vP(2)=6.0_wr
      vP(3)=6.1_wr
      vP(4)=6.1_wr
      vP(5)=6.2_wr
      vP(6)=6.4_wr
      vP(7)=6.8_wr
      vP(8)=7.4_wr
      vP(9)=8.1_wr
    elseif (lettre=='g') then
      vP(1)=6.0_wr
      vP(2)=6.1_wr
      vP(3)=6.2_wr
      vP(4)=6.2_wr
      vP(5)=6.2_wr
      vP(6)=6.4_wr
      vP(7)=6.9_wr
      vP(8)=7.5_wr
      vP(9)=7.9_wr
    elseif (lettre=='h') then
      vP(1)=6.0_wr
      vP(2)=6.1_wr
      vP(3)=6.1_wr
      vP(4)=6.1_wr
      vP(5)=6.2_wr
      vP(6)=6.5_wr
      vP(7)=6.7_wr
      vP(8)=7.3_wr
      vP(9)=8.0_wr
    elseif (lettre=='i') then
      vP(1)=5.2_wr
      vP(2)=6.0_wr
      vP(3)=6.1_wr
      vP(4)=6.1_wr
      vP(5)=6.2_wr
      vP(6)=6.3_wr
      vP(7)=6.7_wr
      vP(8)=7.3_wr
      vP(9)=8.0_wr
    elseif (lettre=='j') then
```

```
751       vP(1)=6.0_wr
752       vP(2)=6.1_wr
753       vP(3)=6.1_wr
754       vP(4)=6.2_wr
755       vP(5)=6.3_wr
756       vP(6)=6.4_wr
757       vP(7)=6.9_wr
758       vP(8)=7.5_wr
759       vP(9)=8.0_wr
760     elseif (lettre=='k') then
761       vP(1)=6.0_wr
762       vP(2)=6.1_wr
763       vP(3)=6.2_wr
764       vP(4)=6.2_wr
765       vP(5)=6.2_wr
766       vP(6)=6.5_wr
767       vP(7)=6.8_wr
768       vP(8)=7.4_wr
769       vP(9)=8.0_wr
770     elseif (lettre=='z') then
771       !write(*,*)'nouveau modèle : ModTerrSiHex_Haslach'
772       lettre='z'
773     else
774       write(*,*)'problème dans ModTerrArroucau, le modèle : ',lettre,' n''existe pas '
775       stop
776     endif
777     ! -------                                                  ---------     .
778     vS=vP/1.68_wr
779     ! -------                                                  ---------     .
780     if (lettre=='z') then
781       Call ModTerrSiHex_Haslach(nbl,vP,vS,h,pdfmoho)
782     endif
783     ! -------                                                  ---------     .
784     pdfmoho=h(nbl)
785     ! ---------------------------------------------------------------     .
786   end subroutine ModTerrArroucau
787
788 ! ---------------------------------------------------------------     .
789
790   subroutine ModTerrSiHex_Haslach(nbl,vP,vS,h,pdfmoho)
791     ! -------                                                  ---------     . mh
792     ! Modèles de Terre SiHex Haslach
793     ! -------                                                  ---------     .
794     implicit none
795     ! -------                                                  ---------     .
796     real (kind=wr), intent(inout), allocatable :: h(:),vP(:),vS(:)
797     integer (KIND=wi), intent(out) :: nbl
798     real(KIND=wr), intent (out) :: pdfmoho
799     ! ---------------------------------------------------------------     .
800     if (allocated(vP)) deallocate(vP)
801     if (allocated(vS)) deallocate(vS)
802     if (allocated(h)) deallocate(h)
803     ! ---------------------------------------------------------------     .
804     nbl=3
805     allocate(vP(nbl),vS(nbl),h(nbl))
806     ! ---------------------------------------------------------------     .
807     vP(1)=5.9_wr
808     vP(2)=6.5_wr
809     vP(3)=8.2_wr
810     ! -------                                                  ---------     .
811     vS(1)=3.4_wr
812     vS(2)=3.7_wr
813     vS(3)=4.4_wr
814     ! -------                                                  ---------     .
815     h(1)=0.0_wr
```

```
816       h(2)=20.0_wr
817       h(3)=30.0_wr
818       ! -------                                                    ---------     .
819       pdfmoho=h(nbl)
820       ! ---------------------------------------------------------------------     .
821     end subroutine ModTerrSiHex_Haslach
822
823   ! ---------------------------------------------------------------------     .
824
825     subroutine ModTerrCEA(nbl,vP,vS,h,pdfmoho)
826       ! -------                                                    ---------     . mh
827       ! Modèles de Terre urilisé au CEA, globale à toute la france.
828       ! Le moho est à 25.9 km.
829       ! pour les inversions, et pour chaque station : le modèle 1D est moyenné à partir du modèle 3D CRUST2.0.
830       ! -------                                                    ---------     .
831       implicit none
832       ! -------                                                    ---------     .
833       real (kind=wr), intent(inout), allocatable :: h(:),vP(:),vS(:)
834       integer (KIND=wi), intent(out) :: nbl
835       real(KIND=wr), intent (out) :: pdfmoho
836       ! ---------------------------------------------------------------------     .
837       if (allocated(vP)) deallocate(vP)
838       if (allocated(vS)) deallocate(vS)
839       if (allocated(h)) deallocate(h)
840       ! ---------------------------------------------------------------------     .
841       nbl=3
842       allocate(vP(nbl),vS(nbl),h(nbl))
843       ! ---------------------------------------------------------------------     .
844       vP(1)=3.00_wr
845       vP(2)=6.03_wr
846       vP(3)=8.16_wr
847       ! -------                                                    ---------     .
848       vS(1)=1.73_wr
849       vS(2)=3.56_wr
850       vS(3)=4.65_wr
851       ! -------                                                    ---------     .
852       h(1)=0.0_wr
853       h(2)=0.9_wr
854       h(3)=25.0_wr
855       ! -------                                                    ---------     .
856       pdfmoho=h(nbl)
857       ! ---------------------------------------------------------------------     .
858     end subroutine ModTerrCEA
859
860   ! ---------------------------------------------------------------------     .
861
862     subroutine ModTerr_fun(nbl,vP,vS,h,pdfmoho)
863       ! -------                                                    ---------     . mh
864       ! Modèles de Terre fun
865       ! -------                                                    ---------     .
866       implicit none
867       ! -------                                                    ---------     .
868       real (kind=wr), intent(inout), allocatable :: h(:),vP(:),vS(:)
869       integer (KIND=wi), intent(out) :: nbl
870       real(KIND=wr), intent (out) :: pdfmoho
871       ! -------                                                    ---------     .
872       integer (KIND=wi) :: i
873       ! ---------------------------------------------------------------------     .
874       if (allocated(vP)) deallocate(vP)
875       if (allocated(vS)) deallocate(vS)
876       if (allocated(h)) deallocate(h)
877       ! ---------------------------------------------------------------------     .
878       nbl=12
879       allocate(vP(nbl),vS(nbl),h(nbl))
880       ! ---------------------------------------------------------------------     .
```

```
    vP(1)=5.2_wr
    vS(1)=vP(1)/1.65_wr
    h(1)=0.0
    do i=2,nbl
        vP(i)=vP(i-1)+1._wr
        h(i)=5.0_wr+h(i-1)
        vS(i)=vP(i)/(1.60+h(i)/100._wr)
    enddo
    ! -------                                                    --------    .
    vP(7)=vP(1)
    vP(10)=vP(7)
    !vS(1)=vP(1)/1.60
    ! -------                                                    --------    .
    pdfmoho=h(nbl)
    ! ------------------------------------------------------------------    .
end subroutine ModTerr_fun

END MODULE ray_tracing



! ********************************************************************    .
! ********************************************************************    .
```

2.15 SRC/MOD/MOD_LaTeX/mklatex.f90

```
! subroutine permettant la création d'un script LaTeX
! octobre 2013
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .

MODULE latexscript

    use modparam
    use typetemps
    use time
    use datalecture, only : catalogue
    use sub_param, only : moy_ec
    use figure_GMT, only : affiche_temps_ref

    implicit none

    private

    public  :: latexfull


    ! ------------------------------------------------------------------    .
    ! on défini _FILE_DIR_ en fonction du compilateur
    ! variable permettant de tester l'existance du dossier

#ifdef __INTEL_COMPILER
#define _FILE_DIR_ DIRECTORY
#elif __GFORTRAN__
#define _FILE_DIR_ FILE
#endif

    ! ------------------------------------------------------------------    .

CONTAINS

! ------------------------------------------------------------------    .
  subroutine latexfull(dc,dp,xmin,xmax,nbChaineMV,acceptance,pbest,misfit,E,nomsta,acentroid,nbtps,D)
```

```
    ! -------                                                    --------     .mh
    ! création de n scripts LaTeX pour la sortie des tableaux et des figures ...
    ! -------                                                    --------     .
    use sub_param, only : lectparam
    ! -------                                                    --------     .
    implicit none
    type(densityplot), intent (in) :: dp                                    ! modèles retenus par McMC
    type(coldmoy), intent (in) :: dc                                        ! modèles du coldrun
    real(KIND=wr), intent (inout) :: xmin(nbseismes), xmax(nbseismes)       ! cercles pond.
    integer(KIND=wi), intent (in) :: nbChaineMV
    type(accept), intent (in) :: acceptance(nbChaineMV)                     ! acceptance
    type(parametres), intent (inout) :: pbest(nbChaineMV)
    type(fcout), intent (in) :: misfit (nbChaineMV)                         ! fonction coût
    type(ellip), intent(in) :: E(nbseismes)
    character(LEN=4), dimension(:), intent(in) :: nomsta
    type(amoho_centroid), intent(in)  :: acentroid                          ! si moho non tabulaire
    integer(KIND=wi), intent(in) :: nbtps(nbseismes)                        ! nombre de données de temps
    type(dataall), intent(inout) :: D(nbseismes)                            ! données
    ! -------                                                    --------     .
    integer(KIND=wi) :: i
    integer(KIND=wi) :: nbChaineMVCold, nbChaineMVhot, maxiterhot, maxitercold
    ! ------------------------------------------------------------------     .
    call lectparam(nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold,chut=.true.)
    ! -------                                                    --------     .
    do i=1, nbseismes
      call latexone(i,dc,dp,xmin,xmax,nbChaineMVCold,nbChaineMVhot,maxiterhot,maxitercold, &
        acceptance,pbest,misfit,E(i),nbtps,D,acentroid)
    enddo
    if (FLAGresSTA) call mk_sta(nomsta)                                     ! nécessite GMT_resSTA dans mkGMT.f90
    ! ------------------------------------------------------------------     .
  end subroutine latexfull

! ------------------------------------------------------------------------     .

  subroutine mk_sta(nomsta)
    ! -------                                                    --------     .mh
    ! création d'un script LaTeX pour la sortie des résidus aux stations ...
    ! -------                                                    --------     .
    implicit none
    character(LEN=4), dimension(:), intent(in) :: nomsta
    ! -------                                                    --------     .
    integer(KIND=wi) :: i, ok
    character (LEN=500) :: a_char
    ! ------------------------------------------------------------------     .
    open(unit=905,file="OUTPUT/LATEX/stations.tex",STATUS="replace")
    write(905,*)"%latex plan.tex ; pdflatex plan.tex"
    write(905,*)"\documentclass[11pt,a4paper]{article}"
    write(905,*)"\usepackage[T1]{fontenc} \usepackage[frenchb]{babel}"
    write(905,*)"\usepackage{geometry}  \geometry{top=20mm, bottom=20mm, left=20mm, right=20mm}"
    write(905,*)"\usepackage{lscape, tabularx,array,multirow,pdflscape}"
    write(905,*)"\newcommand{\kms}{km$\cdotp$s{\footnotesize $^{-1}$}}"
    write(905,*)"\usepackage{gensymb} % degree"
    write(905,*)"\usepackage[pdftex]{graphicx} \DeclareGraphicsExtensions{.jpg,.pdf,.png,.bmp,.jpeg,.ps,.eps}"
    write(905,*)"\usepackage[clockwise]{rotating}"
    write(905,*)"\usepackage{longtable}"
    write(905,*)"\usepackage[np]{numprint} \npdecimalsign{,}"
    write(905,*)"\usepackage{datetime}"
    write(905,*)"\usepackage{lastpage}"
    write(905,*)"\usepackage{fancyhdr}"
    write(905,*)"\usepackage{color}"
    write(905,*)"\definecolor{dkblue}{rgb}{0,0,0.55}"
    write(905,*)"\pagestyle{fancy}"
    write(905,*)"\fancyhead[R]{\today ~ -- ~\currenttime}"
    write(905,*)"\fancyhead[L]{M\'eric Haugmard (\href{mailto:meric.haugmard@univ-nantes.fr}{meric.haugmard@univ-nantes.fr})}"
    write(905,*)"\fancyfoot[L] {\thepage /\pageref{LastPage}}"
```

```
105     write(905,*)"\fancyfoot[C] {}"
106     write(905,*)"\fancyfoot[R] {\href{http://lpgnantes.fr/haugmard-m}{http://lpgnantes.fr/haugmard-m} ; tel : 02 51 12 54 31 }"
107     write(905,*)"\usepackage{hyperref}   % pdf interactif"
108     write(905,*)"\hypersetup"
109     write(905,*)"{"
110     write(905,*)"pdftitle={Th\'ese de Doctorat},"
111     write(905,*)"pdfauthor={M\'eric Haugmard},"
112     write(905,*)"pdfsubject={sismicit\'e du Massif armoricain}"
113     write(905,*)"pdfKeywords={sismicit\'e, Massif armoricain}"
114     write(905,*)"pdfproducer={Laboratoire de Plan\'etologie et G\'eodynamique de Nantes}"
115     write(905,*)"pdftoolbar=true, %barre d'outils non visible"
116     write(905,*)"pdfmenubar=true, %barre de menu visible"
117     write(905,*)"pdfpagelayout=TwoColumnLeft,"
118     write(905,*)"pdfpagemode=UseThumbs,"
119     write(905,*)"colorlinks=true,"
120     write(905,*)"linkcolor= dkblue,"
121     write(905,*)"filecolor= dkblue,"
122     write(905,*)"urlcolor= dkblue"
123     write(905,*)"}"
124     write(905,*)"\usepackage[labelformat=empty]{caption}"
125     write(905,*)"\newcommand{\PN}{$P_{n}$}"
126     write(905,*)"\newcommand{\PG}{$P_{g}$}"
127     write(905,*)"\newcommand{\SN}{$S_{n}$}"
128     write(905,*)"\newcommand{\SG}{$S_{g}$}"
129     ! ------------------------------------------------------------------    .
130     write(905,*)"\begin{document}"
131     do i=1,size(nomsta)
132       write(905,*)"\centering \begin{landscape} \centering"
133       write(905,*)"\vfill"
134       write(905,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE reshisto %%%%%%%%%"
135       write(905,*)"\centerline{\includegraphics[width=0.65\linewidth,angle=-90]{../figures/resSTA-"//nomsta(i)//".pdf}}"
136        write(905,*)"\caption{\large r\'esidus pour l'ensemble des \np{1000} meilleurs mo\'eles, par station : "//nomsta(i)//"}"
137       write(905,*)"\end{figure}"
138       write(905,*)"\vfill"
139       write(905,*)"\end{landscape}"
140     enddo
141     ! ------------------------------------------------------------------    .
142     if (FLAGresSTA) then
143       do i=1,size(nomsta)
144         write(905,*)"\centering \begin{landscape} \centering"
145         write(905,*)"\vfill"
146         write(905,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE reshisto %%%%%%%%%"
147         write(905,*)"\centerline{\includegraphics[width=0.65\linewidth,angle=-90]{../figures/resSTA_TOT-"//nomsta(i)//".pdf}}"
148         write(905,*)"\caption{\large r\'esidus pour toutes les it\'erations : "//nomsta(i)//"}"
149         write(905,*)"\end{figure}"
150         write(905,*)"\vfill"
151         write(905,*)"\end{landscape}"
152       enddo
153       ! ------------------------------------------------------------------    .
154       write(905,'(a)')"\begin{landscape} \centering"
155       write(905,'(a)')"\renewcommand{\arraystretch}{1.75}"
156       write(905,'(a)')"\begin{longtable}[!ht]"
157       write(905,'(3a)')"{|>{\centering}p{2.5cm}|>{\centering}p{2.9cm}|>{\centering}p{1.5cm}|>{\centering}", &
158         "p{2.9cm}|>{\centering}p{1.5cm}|>{\centering}p{2.9cm}|>{\centering}p{1.5cm}|", &
159         ">{\centering}p{2.9cm}|p{1.5cm}<{\centering}|}"
160       write(905,'(a)')"% Entete de la première page"
161       write(905,'(2a)')"\hline"
162       write(905,'(2a)')"\multirow{2}{*}{station} & \multicolumn{2}{c|}{\PG} & \multicolumn{2}{c|}{\PN} & ", &
163         " \multicolumn{2}{c|}{\SG} & \multicolumn{2}{c|}{\SN} \\ "
164       write(905,'(a)')"\cline{2-9} "
165       write(905,'(2a)')" & moy. {\small($\pm2\sigma$)} & m\'ed. & moy. {\small($\pm2\sigma$)} & m\'ed. & ", &
166         " moy. {\small($\pm2\sigma$)} & m\'ed. & moy. {\small($\pm2\sigma$)} & m\'ed. \\ "
167       write(905,'(a)')"\hline"
168       write(905,'(a)')"\bf & & & & & & & & \\"
169       write(905,'(a)')"\endfirsthead"
```

```
      write(905,'(a)')"% Entête de toutes les pages"
      write(905,'(a)')"\hline"
      write(905,'(2a)')"\multirow{2}{*}{station} & \multicolumn{2}{c|}{\PG} & \multicolumn{2}{c|}{\PN} & ", &
        " \multicolumn{2}{c|}{\SG} & \multicolumn{2}{c|}{\SN} \\ "
      write(905,'(a)')"\cline{2-9} "
      write(905,'(2a)')" & moy. {\small($\pm2\sigma$)} & m\'ed. & moy. {\small($\pm2\sigma$)} & m\'ed. & ", &
        " moy. {\small($\pm2\sigma$)} & m\'ed. & moy. {\small($\pm2\sigma$)} & m\'ed. \\ "
      write(905,'(a)')"\hline"
      write(905,'(a)')"\bf & & & & & & & & \\"
      write(905,'(a)')"\bf  ... & ... & ... & ... & ... & ... & ... & ... & ... \\"
      write(905,'(a)')"\bf & & & & & & & & \\"
      write(905,'(a)')"\endhead"
      write(905,'(a)')"% Bas de toutes les pages"
      write(905,'(a)')"\bf  ... & ... & ... & ... & ... & ... & ... & ... & ... \\"
      write(905,'(a)')"\bf & & & & & & & & \\"
      write(905,'(a)')"\hline"
      write(905,'(a)')"\endfoot"
      write(905,'(a)')"\endlastfoot"
      write(905,'(a)')"% Contenu du tableau"
      ! ------------------------------------------------------------     .
      ok=0
      open(902, FILE = "OUTPUT/GMT/sta_RES_TOT2latex.txt",status='old',iostat = ok)
      if (ok .ne. 0) then
        write(*,*)"problème dans mk_sta : le fichier OUTPUT/GMT/sta_RES_TOT2latex.txt n''existe pas "
        stop
      endif
      do while(ok .eq. 0)
        read(902,'(a500)',iostat = ok)a_char
        if(ok .eq. 0) write(905,'(a)')a_char
      enddo
      write(905,*)"\hline "
      write(905,*)"\end{longtable}"
      write(905,*)"\end{landscape}"
      close(902)
      ! ------------------------------------------------------------     . affiche la carte des résidus aux stations, moy.
      write(905,*)"\begin{landscape} \centering"
      write(905,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE residus %%%%%%%%%"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
        "{../figures/resTOTPgmoy.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\hfill"
       write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
        "{../figures/resTOTSgmoy.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\vfill"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
        "{../figures/resTOTPnmoy.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\hfill"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
        "{../figures/resTOTSnmoy.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\caption{moyenne ($\pm2\sigma$) des r\'esidus pour tous les s\'eismes ", &
        "-- la taille de l'histogramme est proportionnelle au crit\'ere R$_p$ ", &
        "(R$_p$ = 1, si la distribution est gaussienne, 0 sinon)}"
      write(905,*)"\end{figure}"
      write(905,*)"\end{landscape}"
      ! ------------------------------------------------------------     . affiche la carte des résidus aux stations, med.
      write(905,*)"\begin{landscape} \centering"
      write(905,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE residus %%%%%%%%%"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
```

```
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
        "{../figures/resTOTPgmed.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\hfill"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
      "{../figures/resTOTSgmed.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\vfill"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
      "{../figures/resTOTPnmed.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\hfill"
      write(905,*)"\begin{minipage}[t]{0.47\linewidth}"
      write(905,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
        "{../figures/resTOTSnmed.pdf}}"
      write(905,*)"\end{minipage}"
      write(905,*)"\caption{m\'ediane des r\'esidus pour tous les s\'eismes}"
      write(905,*)"\end{figure}"
      write(905,*)"\end{landscape}"

      ! ------------------------------------------------------------------     . histo des résidus aux stations
      write(905,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE matCorr %%%%%%%%%"
      write(905,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/Allres.pdf}}"
      write(905,*)"\definecolor{ss}{rgb}{1,0,1}"
      write(905,*)"\definecolor{pp}{rgb}{0,0.67,0.92}"
      write(905,*)"\caption{\large histogramme emplil\'e des r\'esidus pour tous les s\'eismes (ondes {\textcolor{pp}{\PG}}, ", &
        "{\textcolor{blue}{\PN}}, {\textcolor{ss}{\SG}} et {\textcolor{red}{\SN}}). }"
      write(905,*)"\end{figure}"

    endif
    write(905,*)"\end{document}"
    ! ------------------------------------------------------------------     .
    close(905)
    ! ------------------------------------------------------------------     .
  end subroutine mk_sta

! ------------------------------------------------------------------     .

  subroutine latexone(i,dc,dp,xmin,xmax,nbChaineMVCold,nbChaineMV,maxiter,maxitercold, &
                      acceptance,pbest,misfit,E,nbtps,D,acentroid)
    ! _______                                                 ________    .mh
    ! création d'un script LaTeX pour la sortie des tableaux et des figures (pour un unique séisme)
    ! _______                                                 ________    .
    use invGEIGER
    use sub_misfit
    use pb_direct
    ! _______                                                 ________    .
    implicit none
    integer(KIND=wi), intent (in) :: i
    type(densityplot), intent (in) :: dp                                  ! modèles retenus par McMC
    type(coldmoy), intent (in) :: dc                                      ! modèles du coldrun
    real(KIND=wr), intent (inout) :: xmin(nbseismes), xmax(nbseismes)       ! cercles pond.
    integer(KIND=wi), intent (in) :: nbChaineMVCold,nbChaineMV,maxiter,maxitercold
    type(accept), intent (in) :: acceptance(nbChaineMV)                   ! acceptance
    type(parametres), intent (inout) :: pbest(nbChaineMV)
    type(fcout), intent (in) :: misfit (nbChaineMV)                       ! fonction coût
    type(ellip), intent(in) :: E
    type(amoho_centroid), intent(in)  :: acentroid                        ! si moho non tabulaire
    integer(KIND=wi), intent(in) :: nbtps(nbseismes)                      ! nombre de données de temps
    type(dataall), intent(inout) :: D(nbseismes)                          ! données
    ! _______                                                 ________    .
    type(seismes) :: refseisme(2)
    type(parametre) :: onep
```

```
300     type(parametres) :: param_best
301     type(date_sec) :: tpsref
302     logical :: existe1, chut, critique
303     character(LEN=30) :: char_0
304     character(LEN=20) :: charname
305     character(LEN=4) :: staname
306     character(LEN=2) :: ondetype
307     character(LEN=33) :: nomfile
308     integer(KIND=wi) :: j,k,l,ok,x1,x2,Noldtime,Nnewtime,ratetime,find
309     real(KIND=wr) :: moy,val,val2,ec,vec(nbChaineMV),a_ponderation,residu,atime
310     real(KIND=wr) :: pct,atimemoy,atimeec,tl,dist,valec,valec2,tps,depi
311     real(KIND=wr) :: misfitval
312     character (LEN=5) :: numberfile
313     character (LEN=1) :: mod
314     ! ------------------------------------------------------------------      .
315     write(numberfile(1:5),'(i5)')i
316     write(*,*)"ecriture du script LaTeX"
317     call system_clock(Noldtime)
318     ! -------                                                 --------      .
319     write(nomfile(1:13),'(a13)')"OUTPUT/LATEX/"
320     write(nomfile(14:18),'(i4.4)')dp%temps_ref(i)%date%year
321     write(nomfile(18:19),'(a1)')"."
322     write(nomfile(19:21),'(i2.2)')dp%temps_ref(i)%date%month
323     write(nomfile(21:22),'(a1)')"."
324     write(nomfile(22:24),'(i2.2)')dp%temps_ref(i)%date%day
325     write(nomfile(24:25),'(a1)')"T"
326     write(nomfile(25:27),'(i2.2)')dp%temps_ref(i)%date%hour
327     write(nomfile(27:28),'(a1)')"h"
328     write(nomfile(28:30),'(i2.2)')dp%temps_ref(i)%date%min
329     write(nomfile(30:31),'(a1)')"."
330     write(nomfile(31:33),'(i2.2)')int(dp%Tzero(i)%best,wi)
331     open(unit=900,file=trim(adjustl(nomfile))//"-"//trim(adjustl(numberfile))//".tex",STATUS="replace")
332     write(900,*)"%latex plan.tex ; pdflatex plan.tex"
333     write(900,*)"%",dp%temps_ref(i)
334     write(900,*)"\documentclass[11pt,a4paper]{article}"
335     write(900,*)"\usepackage[T1]{fontenc} \usepackage[frenchb]{babel}"
336     write(900,*)"\usepackage{geometry}  \geometry{top=20mm, bottom=20mm, left=20mm, right=20mm}"
337     write(900,*)"\usepackage{lscape, tabularx,array,multirow,pdflscape}"
338     write(900,*)"\newcommand{\kms}{km$\cdotp$s{\footnotesize $^{-1}$}}"
339     write(900,*)"\usepackage{gensymb} % degree"
340     write(900,*)"\usepackage[pdftex]{graphicx} \DeclareGraphicsExtensions{.jpg,.pdf,.png,.bmp,.jpeg,.ps,.eps}"
341     write(900,*)"\usepackage[clockwise]{rotating}"
342     write(900,*)"\usepackage[labelformat=empty]{caption}"
343     write(900,*)"\usepackage{longtable}"
344     write(900,*)"\usepackage[np]{numprint} \npdecimalsign{,}"
345     write(900,*)"\usepackage{datetime}"
346     write(900,*)"\usepackage{lastpage}"
347     write(900,*)"\usepackage{fancyhdr}"
348     write(900,*)"\usepackage{color}"
349     write(900,*)"\definecolor{dkblue}{rgb}{0,0,0.55}"
350     write(900,*)"\pagestyle{fancy}"
351     write(900,*)"\fancyhead[R]{\today ~ -- ~\currenttime}"
352     write(900,*)"\fancyhead[L]{M\'eric Haugmard (\href{mailto:meric.haugmard@univ-nantes.fr}{meric.haugmard@univ-nantes.fr})}"
353     write(900,*)"\fancyfoot[L] {\thepage /\pageref{LastPage}}"
354     write(900,*)"\fancyfoot[C] {}"
355     write(900,*)"\fancyfoot[R] {\href{http://lpgnantes.fr/haugmard-m}{http://lpgnantes.fr/haugmard-m} ; tel : 02 51 12 54 31 }"
356     write(900,*)"\usepackage{hyperref}  % pdf interactif"
357     write(900,*)"\hypersetup"
358     write(900,*)"{"
359     write(900,*)"pdftitle={},"
360     write(900,*)"pdfauthor={M\'eric Haugmard},"
361     write(900,*)"pdfsubject={Th\`ese - sismicit\'e du Massif armoricain }"
362     write(900,*)"pdfKeywords={sismicit\'e, Vannes, Massif armoricain}"
363     write(900,*)"pdfproducer={Laboratoire de Plan\'etologie et G\'eodynamique de Nantes}"
364     write(900,*)"pdftoolbar=true, %barre d'outils non visible"
```

```
    write(900,*)"pdfmenubar=true, %barre de menu visible"
    write(900,*)"pdfpagelayout=TwoColumnLeft,"
    write(900,*)"pdfpagelayout=TwoColumnLeft,"
    write(900,*)"pdfpagemode=UseThumbs,"
    write(900,*)"linkcolor= dkblue,"
    write(900,*)"filecolor= dkblue,"
    write(900,*)"urlcolor= dkblue"
    write(900,*)"}"
    write(900,*)"\newcommand{\PN}{$P_{n}$}"
    write(900,*)"\newcommand{\PG}{$P_{g}$}"
    write(900,*)"\newcommand{\SN}{$S_{n}$}"
    write(900,*)"\newcommand{\SG}{$S_{g}$}"
    ! ------------------------------------------------------------------    .
    write(900,*)"\begin{document}"
    ! ------------------------------------------------------------------    .
    write(900,*)
    call affiche_temps_ref(dp%temps_ref(i),char_0,1)
    write(900,*)"\centering{\NoAutoSpaceBeforeFDP \Huge S\'eisme du ",char_0,"} \\"
    write(900,*)" "
    write(900,*)"\vspace{1.0cm}"
    write(900,*)" "
    write(900,*)"\sloppy \raggedright"
    ! ------------------------------------------------------------------    .
    do j=1,nbChaineMV
      vec(j)=acceptance(j)%val
    enddo
    call moy_ec (vec,nbChaineMV,nbChaineMV,moy,ec)
    write(900,*)"Param\'etres de l'inversion:"
    write(900,'(a,f9.2,a,f5.2,a)')"moyenne des acceptances : \np{",moy,"} $\pm$ \np{",ec,"} \\"
    write(900,'(a,i12,a)')"nombre de cha\^ines de Markov (coldrun) : \np{",nbChaineMVCold,"}\\"
    write(900,'(a,i12,a)')"nombre d'it\'erations par cha\^ine (coldrun) : \np{",maxitercold,"}\\"
    write(900,'(a,i12,a)')"nombre de cha\^ines de Markov (hotrun) : \np{",nbChaineMV,"}\\"
    write(900,'(a,i12,a)')"nombre d'it\'erations par cha\^ine (hotrun) : \np{",maxiter,"}\\"
    write(900,'(a,i12,a)')"nombre mod\'eles test\'es : \np{",maxiter*nbChaineMV+nbChaineMVCold*maxitercold,"}\\"
    write(900,'(a,i12,a)')"nombre de mod\'eles retenus : \np{",dp%nbparam,"}\\"
    write(900,'(a,i6,a)')"discr\'etisation pour le diagramme de densit\'e : \np{",dp%deltaxy,"}\\"
    x1=int(xmin(i),wi)
    x2=int(xmax(i),wi)
    write(900,'(a,i5,a,i5,a)')"cercles de pond\'erations (km) : \np{",x1,"} et \np{",x2,"}\\"
    ! ------------------------------------------------------------------    . ellipse
    write(900,'(a)')"~\\"
    write(900,'(a)')"ellipse (1$\sigma$) des 1000 meilleurs mod\'eles"
    write(900,'(a,f8.2,a)')" azimuth : \np[\degree]{",E%ang,"}\\"
    write(900,'(a,f10.2,a)')" demi axe a : \np[m]{",E%axeA*1000.0_wr,"}\\"
    write(900,'(a,f10.2,a)')" demi axe b : \np[m]{",E%axeB*1000.0_wr,"}\\"
    write(900,'(a,f10.2,a,f13.2,a)')" Aire : \np[km^2]{",E%axeB*E%axeA*pi,"} {\small (\np[ha]{",E%axeB*E%axeA*pi/0.01_wr,"})} \\"
    write(900,'(a)')"~\\"
    ! ------------------------------------------------------------------    . recherche meilleur modèle
    moy=100000.0_wr
    do j=1,nbChaineMV
      if (misfit(j)%best.lt.moy) then
      moy = misfit(j)%best
      k = j
      endif
    enddo
    call mvPall_2_P1(onep,pbest(k),i)
    call catalogue(onep,refseisme,find)                                   ! comparaison avec le catalogue
    ! ------                                                  ---------    .
    if (find==1) then
      write(900,*)"{ \small S\'eisme pr\'esent dans le catalogue : "//refseisme(1)%name//"\\"
      write(900,*)"\begin{itemize}"
      write(900,'(a,f9.2,a3)')"\item magnitude $M_l$ : \np{",refseisme(1)%mag,"}\\"
      write(900,'(a55,f9.2,a3)')"\item longitude : \np[\degree]{",refseisme(1)%lon,"}\\"
      write(900,'(a55,f9.2,a3)')"\item latitude : \np[\degree]{",refseisme(1)%lat,"}\\"
      write(900,'(a55,f9.2,a3)')"\item profondeur hypocentre : \np[km]{",refseisme(1)%pfd,"}\\"
```

```
      call affiche_temps_ref(refseisme(1)%tps_init,char_0,1)
      write(900,'(a45,a30,a8,f4.1,a13)')"\item temps initial : \NoAutoSpaceBeforeFDP",char_0," et \np{", &
        refseisme(1)%tps_init%sec,"} secondes \\"
      write(900,'(a65,f9.3,a4)')"\item diff\'erence de temps avec le meilleur mod\`ele : \np[s]{",refseisme(1)%d_t,"} \\"
      write(900,'(a75,f9.2,a4)')"\item diff\'erence de profondeur avec le meilleur mod\`ele : \np[km]{",refseisme(1)%d_p,"} \\"
      write(900,'(a65,f12.3,a4)')"\item distance \'epicentrale : \np[m]{",refseisme(1)%d_epi*1000.0_wr,"} \\"
      write(900,*)"\end{itemize}}"

    elseif (find==2) then
      write(900,*)"{ \small S\'eisme pr\'esent dans le catalogue 2 fois : \\"
      write(900,*)"catalogue 1 : "//refseisme(1)%name//"\\"
      write(900,*)"\begin{itemize}"
      write(900,'(a,f9.2,a3)')"\item magnitude $M_l$ : \np{",refseisme(1)%mag,"}\\"
      write(900,'(a55,f9.2,a3)')"\item longitude : \np[\degree]{",refseisme(1)%lon,"}\\"
      write(900,'(a55,f9.2,a3)')"\item latitude : \np[\degree]{",refseisme(1)%lat,"}\\"
      write(900,'(a55,f9.2,a3)')"\item profondeur hypocentre : \np[km]{",refseisme(1)%pfd,"}\\"
      call affiche_temps_ref(refseisme(1)%tps_init,char_0,1)
      write(900,'(a45,a30,a8,f4.1,a13)')"\item temps initial : \NoAutoSpaceBeforeFDP",char_0," et \np{", &
        refseisme(1)%tps_init%sec,"} secondes \\"
      write(900,'(a65,f9.3,a4)')"\item diff\'erence de temps avec le meilleur mod\`ele : \np[s]{",refseisme(1)%d_t,"} \\"
      write(900,'(a75,f9.2,a4)')"\item diff\'erence de profondeur avec le meilleur mod\`ele : \np[km]{",refseisme(1)%d_p,"} \\"
      write(900,'(a65,f12.3,a4)')"\item distance \'epicentrale : \np[m]{",refseisme(1)%d_epi*1000.0_wr,"} \\"
      write(900,*)"\end{itemize}"
      write(900,*)"~\\"
      write(900,*)"catalogue 2 : "//refseisme(2)%name//"\\"
      write(900,*)"\begin{itemize}"
      write(900,'(a,f9.2,a3)')"\item magnitude $M_l$ : \np{",refseisme(2)%mag,"}\\"
      write(900,'(a55,f9.2,a3)')"\item longitude : \np[\degree]{",refseisme(2)%lon,"}\\"
      write(900,'(a55,f9.2,a3)')"\item latitude : \np[\degree]{",refseisme(2)%lat,"}\\"
      write(900,'(a55,f9.2,a3)')"\item profondeur hypocentre : \np[km]{",refseisme(2)%pfd,"}\\"
      call affiche_temps_ref(refseisme(2)%tps_init,char_0,1)
      write(900,'(a45,a30,a8,f4.1,a13)')"\item temps initial : \NoAutoSpaceBeforeFDP",char_0," et \np{", &
        refseisme(2)%tps_init%sec,"} secondes \\"
      write(900,'(a65,f9.3,a4)')"\item diff\'erence de temps avec le meilleur mod\`ele : \np[s]{",refseisme(2)%d_t,"} \\"
      write(900,'(a75,f9.2,a4)')"\item diff\'erence de profondeur avec le meilleur mod\`ele : \np[km]{",refseisme(2)%d_p,"} \\"
      write(900,'(a65,f12.3,a4)')"\item distance \'epicentrale : \np[m]{",refseisme(2)%d_epi*1000.0_wr,"} \\"
      write(900,*)"\end{itemize}}"
    else
      write(900,*)"S\'eisme absent du catalogue"
    endif
    write(900,*)
    ! ------------------------------------------------------------------    .
    write(900,*)"\begin{figure}[!ht] \centering         %%%%%%%% FIGURE MAP init %%%%%%%"
    write(900,*)"\centerline{\includegraphics[width=0.99\linewidth]", &
      "{../figures/init-"//trim(adjustl(numberfile))//".pdf}}"
    write(900,*)"\caption{\large prior sur l'\'epicentre, m\'ethode des arriv\'ees les plus proches}"
    write(900,*)"\end{figure}"
    write(900,*)
    ! ------------------------------------------------------------------     . affiche le diagramme de chatelain
    write(900,*)"\begin{landscape} \centering"
    write(900,*)"\begin{figure}[!ht]    %%%%%%%%%% FIGURE chatelain %%%%%%%%"
    write(900,*)"\centerline{\includegraphics[width=.6\linewidth,angle=-90]", &
      "{../figures/chatelainplot"//trim(adjustl(numberfile))//".pdf}}"
    write(900,*)"\caption{\large diagramme de Ch\^atelain }"
    write(900,*)"\end{figure}"
    write(900,*)"\end{landscape}"
    write(900,*)
    ! ------------------------------------------------------------------     .
    write(900,*)"\begin{landscape} \centering"
    write(900,*)"\begin{figure}[!ht] \centering         %%%%%%%%%% FIGURE MAP %%%%%%%%%"
    write(900,*)"\centerline{\includegraphics[width=0.99\textwidth,angle=-90]", &
      "{../figures/map"//"-"//trim(adjustl(numberfile))//".pdf}}"
    write(900,*)"\end{figure}"
    write(900,*)"\end{landscape}"
    write(900,*)
```

```
    ! ------------------------------------------------------------------      .
    write(900,*)"\begin{landscape} \centering           %%%%%%%% Table données %%%%%%%%"

    write(900,*)"\begin{table}[!ht] \scriptsize \centering  \renewcommand{\arraystretch}{1.5}"
    write(900,*)"\begin{tabular}{|>{\centering}m{2.5cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
    write(900,*)"|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
    write(900,*)"|>{\centering}m{2.2cm}|>{\centering}m{2.2cm}|m{2.1cm}<{\centering}|}"
    write(900,*)"\hline "
    write(900,*)"{\bf \large coldruns} & {fonction co\^ut} & {V$_C$ (\kms)} & {V$_M$ (\kms)} & ", &
      " {Z$_{moho}$ (km)} & {V$_{P}$V$_{S}$} & {Z$_{hypo}$ (km)} & {longitude (\degree)} & ", &
      " {latitude (\degree)} & {T$_{z\acute{e}ro}$ (s)} \\"
    write(900,*)"\hline "
    call difftime(atime,dc%tempsrefcold(i),dp%temps_ref(i)) !!!!!!!!!
    atimeec=dc%ectot%par%Tzero(i)%sec
    atimemoy=dc%moytot%par%Tzero(i)%sec + atime
    write(900,4997)" {moyenne ($\pm2\sigma$) des meilleurs mod\'eles de toutes les cha\^ines} & \np{",dc%moytot%mis, &
      "} $\pm$ \np{",2.0_wr*dc%ectot%mis,"} & \np{",dc%moytot%par%VC,"} $\pm$ \np{",2.0_wr*dc%ectot%par%VC, &
      "} & \np{",dc%moytot%par%VM,"} $\pm$ \np{",2.0_wr*dc%ectot%par%VM,"} & \np{",dc%moytot%par%Zmoho, &
      "} $\pm$ \np{",2.0_wr*dc%ectot%par%Zmoho,"} & \np{",dc%moytot%par%VpVs,"} $\pm$ \np{",2.0_wr*dc%ectot%par%VpVs, &
      "} & \np{",dc%moytot%par%Zhypo(i),"} $\pm$ \np{",2.0_wr*dc%ectot%par%Zhypo(i),"} & \np{",dc%moytot%par%lon(i), &
      "} $\pm$ \np{",2.0_wr*dc%ectot%par%lon(i),"} & \np{",dc%moytot%par%lat(i),"} $\pm$ \np{",2.0_wr*dc%ectot%par%lat(i), &
      "} & \np{",atimemoy,"} $\pm$ \np{",2.0_wr*atimeec,"} \\ "
    atimeec=dc%ecselect%par%Tzero(i)%sec
    atimemoy=dc%moyselect%par%Tzero(i)%sec + atime
    write(900,*)"\hline "
    write(900,4998)" {moyenne ($\pm2\sigma$) des meilleurs mod\'eles de chaque cha\^ine s\'electionn\'ee} & \np{", &
      dc%moyselect%mis,"} $\pm$ \np{",2.0_wr*dc%ecselect%mis,"} & \np{",dc%moyselect%par%VC,"} $\pm$ \np{", &
      2.0_wr*dc%ecselect%par%VC,"} & \np{",dc%moyselect%par%VM,"} $\pm$ \np{",2.0_wr*dc%ecselect%par%VM,"} & \np{", &
      dc%moyselect%par%Zmoho,"} $\pm$ \np{",2.0_wr*dc%ecselect%par%Zmoho,"} & \np{",dc%moyselect%par%VpVs, &
      "} $\pm$ \np{",2.0_wr*dc%ecselect%par%VpVs,"} & \np{",dc%moyselect%par%Zhypo(i),"} $\pm$ \np{", &
      2.0_wr*dc%ecselect%par%Zhypo(i),"} & \np{",dc%moyselect%par%lon(i),"} $\pm$ \np{",2.0_wr*dc%ecselect%par%lon(i), &
      "} & \np{",dc%moyselect%par%lat(i),"} $\pm$ \np{",2.0_wr*dc%ecselect%par%lat(i),"} & \np{",atimemoy, &
      "} $\pm$ \np{",2.0_wr*atimeec,"} \\ "
    4997 format (a128,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f10.4,a12,f10.3,a8,f10.4,a12,f10.3,a8,f10.2,a12,f10.2,a4)
    4998 format (a140,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f7.4,a12,f9.3,a8,f7.4,a12,f9.3,a8,f9.2,a12,f9.2,a4)
    write(900,*)"\hline "
    write(900,*)"\end{tabular}"
    write(900,*)"\end{table}"

    write(900,*)"\begin{table}[!ht] \scriptsize \centering  \renewcommand{\arraystretch}{1.5}"
    write(900,*)"\begin{tabular}{|>{\centering}m{2.5cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
    write(900,*)"|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
    write(900,*)"|>{\centering}m{2.2cm}|>{\centering}m{2.2cm}|m{2.1cm}<{\centering}|}"
    write(900,*)"\hline "
    write(900,*)" {\bf \large hotruns} & {fonction co\^ut} & {V$_C$ (\kms)} & {V$_M$ (\kms)} & {Z$_{moho}$ (km)} ", &
      " & {V$_{P}$V$_{S}$} & {Z$_{hypo}$ (km)} & {longitude (\degree)} & {latitude (\degree)} & {T$_{z\acute{e}ro}$ (s)} \\"
    write(900,*)"\hline "
    write(900,4999)" {mode} & \np{",dp%mis%mode,"} & \np{",dp%VC%mode,"} & \np{",dp%VM%mode, &
      "} & \np{",dp%Zmoho%mode,"} & \np{",dp%VpVs%mode,"} & \np{",dp%Zhypo(i)%mode,"} & \np{",dp%lon(i)%mode, &
      "} & \np{",dp%lat(i)%mode,"} & \np{",dp%Tzero(i)%mode,"} \\ "
    write(900,*)"\hline "
    write(900,5000)" {m\'ediane} & \np{",dp%mis%mediane,"} & \np{",dp%VC%mediane,"} & \np{",dp%VM%mediane, &
      "} & \np{",dp%Zmoho%mediane,"} & \np{",dp%VpVs%mediane,"} & \np{",dp%Zhypo(i)%mediane,"} & \np{",dp%lon(i)%mediane, &
      "} & \np{",dp%lat(i)%mediane,"} & \np{",dp%Tzero(i)%mediane,"} \\ "
    write(900,*)"\hline "
    write(900,5001)" {meilleur mod\'ele} & \np{",dp%mis%best,"} & \np{",dp%VC%best,"} & \np{",dp%VM%best, &
      "} & \np{",dp%Zmoho%best,"} & \np{",dp%VpVs%best,"} & \np{",dp%Zhypo(i)%best,"} & \np{",dp%lon(i)%best, &
      "} & \np{",dp%lat(i)%best,"} & \np{",dp%Tzero(i)%best,"} \\ "
    write(900,*)"\hline "
    write(900,5003)" {moyenne ($\pm2\sigma$) des 100 meilleurs mod\'eles} & \np{",dp%mis%moy_100,"} $\pm$ \np{", &
      2.0_wr*dp%mis%ec_100,"} & \np{",dp%VC%moy_100,"} $\pm$ \np{",2.0_wr*dp%VC%ec_100,"} & \np{",dp%VM%moy_100, &
      "} $\pm$ \np{",2.0_wr*dp%VM%ec_100,"} & \np{",dp%Zmoho%moy_100,"} $\pm$ \np{",2.0_wr*dp%Zmoho%ec_100, &
      "} & \np{",dp%VpVs%moy_100,"} $\pm$ \np{",2.0_wr*dp%VpVs%ec_100,"} & \np{",dp%Zhypo(i)%moy_100,"} $\pm$ \np{", &
```

```
      2.0_wr*dp%Zhypo(i)%ec_100,"} & \np{",dp%lon(i)%moy_100,"} $\pm$ \np{",2.0_wr*dp%lon(i)%ec_100,"} & \np{",dp%lat(i)%moy_100, &
      "} $\pm$ \np{",2.0_wr*dp%lat(i)%ec_100,"} & \np{",dp%Tzero(i)%moy_100,"} $\pm$ \np{",2.0_wr*dp%Tzero(i)%ec_100,"} \\ "
    write(900,*)"\hline "
    write(900,5004)" {moyenne ($\pm2\sigma$) des 1000 meilleurs mod\'eles} & \np{",dp%mis%moy_1000,"} $\pm$ \np{", &
      2.0_wr*dp%mis%ec_1000,"} & \np{",dp%VC%moy_1000,"} $\pm$ \np{",2.0_wr*dp%VC%ec_1000,"} & \np{",dp%VM%moy_1000, &
      "} $\pm$ \np{",2.0_wr*dp%VM%ec_1000,"} & \np{",dp%Zmoho%moy_1000,"} $\pm$ \np{",2.0_wr*dp%Zmoho%ec_1000,"} & \np{", &
      dp%VpVs%moy_1000,"} $\pm$ \np{",2.0_wr*dp%VpVs%ec_1000,"} & \np{",dp%Zhypo(i)%moy_1000,"} $\pm$ \np{", &
      2.0_wr*dp%Zhypo(i)%ec_1000,"} & \np{",dp%lon(i)%moy_1000,"} $\pm$ \np{",2.0_wr*dp%lon(i)%ec_1000,"} & \np{", &
      dp%lat(i)%moy_1000,"} $\pm$ \np{",2.0_wr*dp%lat(i)%ec_1000,"} & \np{",dp%Tzero(i)%moy_1000,"} $\pm$ \np{", &
      2.0_wr*dp%Tzero(i)%ec_1000,"} \\ "
    write(900,*)"\hline "
    write(900,5005)" {moyenne ($\pm2\sigma$) des 10000 meilleurs mod\'eles} & \np{",dp%mis%moy_10000,"} $\pm$ \np{", &
      2.0_wr*dp%mis%ec_10000,"} & \np{",dp%VC%moy_10000,"} $\pm$ \np{",2.0_wr*dp%VC%ec_10000,"} & \np{",dp%VM%moy_10000, &
      "} $\pm$ \np{",2.0_wr*dp%VM%ec_10000,"} & \np{",dp%Zmoho%moy_10000,"} $\pm$ \np{",2.0_wr*dp%Zmoho%ec_10000, &
      "} & \np{",dp%VpVs%moy_10000,"} $\pm$ \np{",2.0_wr*dp%VpVs%ec_10000,"} & \np{",dp%Zhypo(i)%moy_10000,"} $\pm$ \np{", &
      2.0_wr*dp%Zhypo(i)%ec_10000,"} & \np{",dp%lon(i)%moy_10000,"} $\pm$ \np{",2.0_wr*dp%lon(i)%ec_10000,"} & \np{", &
      dp%lat(i)%moy_10000,"} $\pm$ \np{",2.0_wr*dp%lat(i)%ec_10000,"} & \np{",dp%Tzero(i)%moy_10000, &
      "} $\pm$ \np{",2.0_wr*dp%Tzero(i)%ec_10000,"} \\ "
    write(900,*)"\hline "
    write(900,5006)" {moyenne ($\pm2\sigma$) des meilleurs mod\'eles de chaque cha\^ine} & \np{",dp%mis%moy_bestchaine, &
      "} $\pm$ \np{",2.0_wr*dp%mis%ec_bestchaine,"} & \np{",dp%VC%moy_bestchaine,"} $\pm$ \np{",2.0_wr*dp%VC%ec_bestchaine, &
      "} & \np{",dp%VM%moy_bestchaine,"} $\pm$ \np{",2.0_wr*dp%VM%ec_bestchaine,"} & \np{",dp%Zmoho%moy_bestchaine, &
      "} $\pm$ \np{",2.0_wr*dp%Zmoho%ec_bestchaine,"} & \np{",dp%VpVs%moy_bestchaine,"} $\pm$ \np{", &
      2.0_wr*dp%VpVs%ec_bestchaine,"} & \np{",dp%Zhypo(i)%moy_bestchaine,"} $\pm$ \np{",2.0_wr*dp%Zhypo(i)%ec_bestchaine, &
      "} & \np{",dp%lon(i)%moy_bestchaine,"} $\pm$ \np{",2.0_wr*dp%lon(i)%ec_bestchaine,"} & \np{", &
      dp%lat(i)%moy_bestchaine,"} $\pm$ \np{",2.0_wr*dp%lat(i)%ec_bestchaine,"} & \np{",dp%Tzero(i)%moy_bestchaine, &
      "} $\pm$ \np{",2.0_wr*dp%Tzero(i)%ec_bestchaine,"} \\ "
    write(900,*)"\hline "
      write(900,5002)" {moyenne ($\pm2\sigma$) totale} & \np{",dp%mis%moy_tot,"} $\pm$ \np{",2.0_wr*dp%mis%ec_tot, &
      "} & \np{",dp%VC%moy_tot,"} $\pm$ \np{",2.0_wr*dp%VC%ec_tot,"} & \np{",dp%VM%moy_tot, &
      "} $\pm$ \np{",2.0_wr*dp%VM%ec_tot,"} & \np{",dp%Zmoho%moy_tot,"} $\pm$ \np{",2.0_wr*dp%Zmoho%ec_tot, &
      "} & \np{",dp%VpVs%moy_tot,"} $\pm$ \np{",2.0_wr*dp%VpVs%ec_tot,"} & \np{",dp%Zhypo(i)%moy_tot, &
      "} $\pm$ \np{",2.0_wr*dp%Zhypo(i)%ec_tot,"} & \np{",dp%lon(i)%moy_tot,"} $\pm$ \np{",2.0_wr*dp%lon(i)%ec_tot, &
      "} & \np{",dp%lat(i)%moy_tot,"} $\pm$ \np{",2.0_wr*dp%lat(i)%ec_tot,"} & \np{",dp%Tzero(i)%moy_tot, &
      "} $\pm$ \np{",2.0_wr*dp%Tzero(i)%ec_tot,"} \\ "
    write(900,*)"\hline "
    4999 format (a14,f9.2,a8,f9.2,a8,f9.2,a8,f9.2,a8,f9.3,a8,f9.2,a8,f7.4,a8,f7.4,a8,f9.2,a4)
    5000 format (a19,f9.2,a8,f9.2,a8,f9.2,a8,f9.2,a8,f9.3,a8,f9.2,a8,f7.4,a8,f7.4,a8,f9.2,a4)
    5001 format (a27,f9.2,a8,f9.2,a8,f9.2,a8,f9.2,a8,f9.3,a8,f9.2,a8,f7.4,a8,f7.4,a8,f9.2,a4)
    5002 format (a87,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f7.4,a12,f9.3,a8,f7.4,a12,f9.3,a8,f9.2,a12,f9.2,a4)
    5003 format (a128,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f7.4,a12,f9.3,a8,f7.4,a12,f9.3,a8,f9.2,a12,f9.2,a4)
    5004 format (a129,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f7.4,a12,f9.3,a8,f7.4,a12,f9.3,a8,f9.2,a12,f9.2,a4)
    5005 format (a80,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f7.4,a12,f9.3,a8,f7.4,a12,f9.3,a8,f9.2,a12,f9.2,a4)
    5006 format (a123,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.2,a8,f9.2,a12,f9.1,a8,f9.3,a12,f9.2,a8,f9.2,a12,f9.1,a8, &
    f7.4,a12,f9.3,a8,f7.4,a12,f9.3,a8,f9.2,a12,f9.2,a4)
    write(900,*)"\end{tabular}"
    write(900,*)"\end{table}"


    ! ------------------------------------------------------------------    .
    ! modèle Arroucau
    ! ------------------------------------------------------------------    .
    write(900,*)" "
    write(900,*)"\input{../GMT/Arroucau_all-"//trim(adjustl(numberfile))//"}"
    write(900,*)" "
    ! ------------------------------------------------------------------    .
    ! modèle Si-Hex
    ! ------------------------------------------------------------------    .
    write(900,*)" "
    write(900,*)"\input{../GMT/SiHex_all-"//trim(adjustl(numberfile))//"}"
    write(900,*)" "
```

```
625     ! ------------------------------------------------------------------      .
626     ! modèle CEA
627     ! ------------------------------------------------------------------      .
628     write(900,*)" "
629     write(900,*)"\input{../GMT/CEA_all-"//trim(adjustl(numberfile))//"}"
630     write(900,*)" "
631     ! ------------------------------------------------------------------      .
632
633
634
635     ! ------------------------------------------------------------------      .
636     ! -------- verif avec méthode de Geiger                  ---------      .
637     ! ------------------------------------------------------------------      .
638     param_best%VC=dp%VC%best ! moy_100
639     param_best%VM=dp%VM%best ! moy_100
640     param_best%Zmoho=dp%Zmoho%best ! moy_100
641     param_best%VpVs=dp%VpVs%best ! moy_100
642     do j=1,nbseismes
643       param_best%Zhypo(j)=dp%Zhypo(j)%best ! moy_100
644       param_best%lon(j)=dp%lon(j)%best ! moy_100
645       param_best%lat(j)=dp%lat(j)%best ! moy_100
646       param_best%Tzero(j) = dp%temps_ref(j)
647       param_best%Tzero(j)%sec = dp%Tzero(j)%best ! 100
648       call basetime(param_best%Tzero(j))
649     enddo
650     call tempsTheoDirect(nbtps,param_best,D,critique,acentroid)
651     call compute_misfit(nbtps,D,misfitval,xmin,xmax,'H')
652     ! -------                                              ---------      .
653     write(900,*)"\begin{table}[!ht] \scriptsize \centering  \renewcommand{\arraystretch}{1.5}"
654     write(900,*)"\begin{tabular}{|>{\centering}m{2.5cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
655     write(900,*)"|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
656     write(900,*)"|>{\centering}m{2.2cm}|>{\centering}m{2.2cm}|m{2.1cm}<{\centering}|}"
657     write(900,*)"\hline "
658     write(900,*)"{\bf \large geiger} & {fonction co\^ut} & {V$_C$ (\kms)} & {V$_M$ (\kms)} & ", &
659       " {Z$_{moho}$ (km)} & {V$_{P}$V$_{S}$} & {Z$_{hypo}$ (km)} & {longitude (\degree)} & ", &
660       " {latitude (\degree)} & {T$_{z\acute{e}ro}$ (s)} \\"
661     write(900,*)"\hline "
662     ! -------                                              ---------      .
663     write(900,5010)" {entr\'ee} & \np{",misfitval,"} & \np{",param_best%VC,"} & \np{",param_best%VM, &
664       "} & \np{",param_best%Zmoho,"} & \np{",param_best%VpVs,"} & \np{",param_best%Zhypo(i),"} & \np{",param_best%lon(i), &
665       "} & \np{",param_best%lat(i),"} & \np{",param_best%Tzero(i)%sec,"} \\ "
666     ! -------                                              ---------      .
667     chut=.true.
668     mod='I'
669     call dogeiger(nbtps,D,param_best,acentroid,mod,chut)
670     call tempsTheoDirect(nbtps,param_best,D,critique,acentroid)
671     call compute_misfit(nbtps,D,misfitval,xmin,xmax,'H')
672     ! -------                                              ---------      .
673     write(900,*)"\hline "
674
675     if (misfitval.lt.1000.0_wr) then
676       write(900,5010)" {sortie} & \np{",misfitval,"} & \np{",param_best%VC,"} & \np{",param_best%VM, &
677         "} & \np{",param_best%Zmoho,"} & \np{",param_best%VpVs,"} & \np{",param_best%Zhypo(i),"} & \np{",param_best%lon(i), &
678         "} & \np{",param_best%lat(i),"} & \np{",param_best%Tzero(i)%sec,"} \\ "
679     else
680       write(900,5010)" {sortie} & \np{",0.0_wr,"} & \np{",param_best%VC,"} & \np{",param_best%VM, &
681         "} & \np{",param_best%Zmoho,"} & \np{",param_best%VpVs,"} & \np{",0.0_wr,"} & \np{",0.0_wr, &
682         "} & \np{",0.0_wr,"} & \np{",0.0_wr,"} \\ "
683     endif
684     5010 format (a25,f9.2,a8,f9.2,a8,f9.2,a8,f9.2,a8,f9.3,a8,f9.2,a8,f7.4,a8,f7.4,a8,f9.2,a4)
685     write(900,*)"\hline "
686     write(900,*)"\end{tabular}"
687     write(900,*)"\end{table}"
688     write(900,*)"\end{landscape}"
689     write(900,*)
```

```
690     ! ------------------------------------------------------------------     .
691     ! ------------------------------------------------------------------     .
692     write(900,*)"\begin{longtable}[!ht]"
693     write(900,*)"{|>{\centering}m{2.5cm}|>{\centering}m{1.5cm}|>{\centering}m{2.5cm}|>{\centering}m{2.5cm}", &
694                 "|>{\centering}m{2.5cm}|m{2.5cm}<{\centering}|}"
695     write(900,*)"% Entete de la première page"
696     write(900,*)"\hline \multicolumn{6}{c}{\bf r\'esidus aux stations} \\ \hline"
697     write(900,*)"\hline   non station & onde & r\'esidus (s) & pond\'eration & r\'esidus / temps total ", &
698                 "& distance hypocentrale (km) \\ \hline"
699     write(900,*)"\endfirsthead"
700     write(900,*)"% Entête de toutes les pages"
701     write(900,*)"\hline   non station & onde & r\'esidus (s) & pond\'eration & r\'esidus / temps total ", &
702                 "& distance hypocentrale (km) \\ \hline"
703     write(900,*)"\endhead"
704     write(900,*)"% Bas de toutes les pages"
705     write(900,*)"\bf ... & ... & ... & ... & ... & ... \\ \hline"
706     write(900,*)"\endfoot"
707     write(900,*)"\endlastfoot"
708     write(900,*)"% Contenu du tableau"
709     ok=0
710     j=0
711     val=0.0_wr
712     val2=0.0_wr
713     open(901, FILE = "OUTPUT/residus"//"-"//trim(adjustl(numberfile))//".d",status='old',iostat = ok)
714     if (ok .ne. 0) then
715       write(*,*)"problème dans latexfull 1 : le fichier OUTPUT/residus"//"-"//trim(adjustl(numberfile))//".d n''existe pas "
716       stop
717     endif
718     do while(ok .eq. 0)
719       read(901,*,iostat = ok)ondetype,staname,moy,moy,residu,a_ponderation,pct, dist
720       if(ok .eq. 0) then
721         j=j+1
722         val = val + abs(residu)
723         val2 = val2 + (residu)
724         if (a_ponderation.gt.0.001_wr) write(900,'(a4,a4,a2,a7,f8.4,a8,f8.4,a8,f8.2,a,f10.3,a)')staname," & \", &
725         ondetype," & \np{",residu,"} & \np{",a_ponderation,"} & \np{",pct,"}\% & \np{",dist,"} \\"
726       endif
727     end do
728     close(901)
729     val = val / real(j,wr)
730     val2 = val2 / real(j,wr)
731     ok=0
732     valec=0.0_wr
733     valec2=0.0_wr
734     open(904, FILE = "OUTPUT/residus"//"-"//trim(adjustl(numberfile))//".d",status='old',iostat = ok)
735     if (ok .ne. 0) then
736       write(*,*)"problème dans latexfull 2 : le fichier OUTPUT/residus"//"-"//trim(adjustl(numberfile))//".d n''existe pas "
737       stop
738     endif
739     do k=1,j
740       read(904,*,iostat = ok)ondetype,staname,moy,moy,residu,a_ponderation,pct,dist
741       valec = valec + (abs(residu)-val)**2.0_wr
742       valec2 = valec2 + (residu-val2)**2.0_wr
743     end do
744     close(904)
745     valec = sqrt(valec/real(j,wr))
746     valec2 = sqrt(valec2/real(j,wr))
747     write(900,*)"\hline "
748     write(900,*)"\end{longtable}"
749     write(900,'(a,f18.5,a,f18.5,a)')"moyenne des r\'esidus : \np{",val2,"} $\pm$ \np{",1.0_wr*valec2,"} ($\pm1\sigma$) \\"
750     write(900,'(a,f18.5,a,f18.5,a)')"moyenne des r\'esidus absolus : \np{",val,"} $\pm$ \np{",1.0_wr*valec,"} ($\pm1\sigma$) \\"
751
752     ! ------------------------------------------------------------------     . affiche magnitude si calculée
753
754     inquire (file="OUTPUT/files/mag-"//trim(adjustl(numberfile))//".d",exist=existe1)
```

```
    open(907, FILE = "OUTPUT/GMT/magfin-"//trim(adjustl(numberfile))//".d",status='replace')

    if ((tracessac).and.(existe1)) then

      open(906, FILE = "OUTPUT/files/mag-"//trim(adjustl(numberfile))//".d",status='old')
      write(900,*)"\vspace{1cm}"
      write(900,*)" {\Large \bf magnitude :\\}"
      write(900,*)"\vspace{1cm}"
      write(900,*)"formule de Lee et al. (1972) : "
      write(900,*)"\begin{equation}"
      write(900,*)"M_d =  -\np{0,87}  + 2 log_{10}(coda) + \np{0,0035} \Delta ,"
      write(900,*)"\end{equation}"
      write(900,'(2a)')"avec $ \Delta$, la distance \'epicentrale (en km) et {$coda$}, la dur\'ee du sigal depuis ", &
        " la premi\`ere arriv\'ee de de l'onde {\em P} jusqu'\`a la fin du signal. \\"
      write(900,*)"\vspace{1cm}"

      write(900,*)"\begin{longtable}[!ht]"
      write(900,*)"{>{\centering}m{2.5cm}>{\centering}m{2.5cm}>{\centering}m{2.5cm}m{2.5cm}<{\centering}}"
      write(900,*)"\hline"
      write(900,*)"station & M$_d$ & dur\'ee (s) & distance \'epicentrale (km)\\"
      write(900,*)"\hline"
      write(900,*)"\endfirsthead"
      write(900,*)"% Entête de toutes les pages"
      write(900,*)"\hline"
      write(900,*)"station & M$_d$ & dur\'ee (s) & distance \'epicentrale (km)\\"
      write(900,*)"\hline"
      write(900,*)"\endhead"
      write(900,*)"% Bas de toutes les pages"
      write(900,*)"\bf ... & ... & ... & ... \\ \hline"
      write(900,*)"\endfoot"
      write(900,*)"\endlastfoot"
      write(900,*)"% Contenu du tableau"
      ok=0
      j=0
      val=0.0_wr
      do while(ok .eq. 0)
        read(906,*,iostat = ok)staname,moy,tps,depi
        if (ok .eq. 0) then
          val=val+moy
          j=j+1
          write(900,'(a4,a,f10.2,a,f10.2,a,f10.2,a)')staname," & \np{",moy,"} & \np{",tps,"} & \np{",depi,"} \\"
        endif
      end do
      moy=val/real(j,wr)
      rewind(906)
      ok=0
      j=0
      ec=0.0_wr
      do while(ok .eq. 0)
        read(906,*,iostat = ok)staname,val,tps,depi
        if (ok .eq. 0) then
          ec=ec+(moy-val)**2.0_wr
          j=j+1
        endif
      end do
      close(906)
      ec = sqrt(ec/real(j,wr))
      write(900,*)"\hline"
      write(900,'(a,f10.2,a,f10.2,a)')"moyenne & \multicolumn{3}{l}{\np{",moy,"} {\small $\pm$ \np{",ec,"} (1$\sigma$)}} \\"
      ! ------------------------------------------------------------ .
      write(907,"(i4.4,4(1a,i2.2),f10.2,f10.2,2f10.2)")dp%temps_ref(i)%date%year,"-",dp%temps_ref(i)%date%month, &
        "-",dp%temps_ref(i)%date%day,"T",dp%temps_ref(i)%date%hour,":",dp%temps_ref(i)%date%min,moy,ec, &
        refseisme(1)%mag,refseisme(2)%mag
      ! ------------------------------------------------------------ .
      write(900,*)"\hline"
```

```
      write(900,*)"\end{longtable}"
    endif
    close(907)
    ! ------------------------------------------------------------------     . affiche les parametres de Terre si fixes
    if (FLAGterre) then
      write(900,*)"\newpage"
      write(900,*)"{\bf Param\'etres de Terre fixes} {\small (PARAM/paramTerre.d)} : \\"
      write(900,*)"\vspace{0.5cm}"
      ok=0
      open(unit=50,file="PARAM/paramTerre.d",STATUS="old",iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans latexone : le fichier PARAM/paramTerre.d n''magn pas '
        stop
      endif
      read(50,*)moy,ec
      write(900,'(a,f18.2,a,f18.3,a)')"V$_C$ (\kms) = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
      read(50,*)moy,ec
      write(900,'(a,f18.2,a,f18.3,a)')"V$_M$ (\kms) = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
      read(50,*)moy,ec
      write(900,'(a,f18.2,a,f18.3,a)')"Z$_{moho}$ (km) = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
      read(50,*)moy,ec
      write(900,'(a,f18.2,a,f18.3,a)')"V$_{P}$V$_{S}$ = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
      close(50)
    endif
    ! ------------------------------------------------------------------     . affiche les parametres hypocentaux si fixes
    if (FLAGhypo) then
      if(FLAGterre)write(900,*)"\vspace{1cm}"
      if(.not.FLAGterre) write(900,*)"\newpage"
      write(900,*)"{\bf Param\'etres hypocentaux fixes} {\small (PARAM/paramHypo.d)} : \\"
      write(900,*)"\vspace{0.5cm}"
      ok=0
      open(unit=51,file="PARAM/paramHypo.d",STATUS="old",iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans latexone : le fichier PARAM/paramHypo.d n''existe pas '
        stop
      endif
      do j=1,nbseismes
        write(900,*)"s\'eisme {\tiny \#}",j,"\\"
        read(51,*)moy,ec
        write(900,'(a,f18.3,a,f18.4,a)')"longitude (\degree) = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
        read(51,*)moy,ec
        write(900,'(a,f18.3,a,f18.4,a)')"latitude (\degree) = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
        read(51,*)moy,ec
        write(900,'(a,f18.2,a,f18.3,a)')"Z$_{hypo}$ (km)= \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
        read(51,*)tpsref
        read(51,*)moy,ec
        call affiche_temps_ref(tpsref,char_0,1)
        write(900,'(a)')"{\NoAutoSpaceBeforeFDP"//char_0//"} \\"
        write(900,'(a,f18.2,a,f18.3,a)')"T$_{z\acute{e}ro}$ (s) = \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} ($\pm2\sigma$)\\"
        write(900,*)"\vspace{0.5cm}"
      enddo
        write(900,*)"\newpage"
      close(51)
    endif
    ! ------------------------------------------------------------------     . affiche les figures param1 VS param2
    ok = 0
    open(unit=103,file="OUTPUT/GMT/files.txt",STATUS="old",iostat = ok)
    if (ok .ne. 0) then
      write(*,*)'problème dans latexfull : le fichier OUTPUT/GMT/files.txt n''existe pas '
      stop
    endif
    do while(ok .eq. 0)
      read(103,'(i6,1x,a20)',iostat = ok)l,charname
      if (l==i) then
        if (ok .eq. 0) then
```

```
885           write(900,*)"\begin{landscape} \centering "
886           write(900,*)"\begin{figure}[!ht] %%%%%%%% FIGURE _vc__vm"//"-"//trim(adjustl(numberfile))//".pdf %%%%%%%"
887           ! write(900,*)"\centerline{\includegraphics[width=1.0\linewidth]{../figures/"//charname//"}}"
888             write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/"//trim(adjustl(charname))//"}}"
889           write(900,*)"\end{figure}"
890           write(900,*)"\end{landscape}"
891         endif
892       endif
893     end do
894     close(103)
895     ! ------------------------------------------------------------------     . affiche la carte des résidus aux stations : ondes directes
896     write(900,*)"\begin{landscape} \centering"
897     write(900,*)"\begin{figure}[!ht]    %%%%%%%%%%% FIGURE residus %%%%%%%%%"
898     write(900,*)"\begin{minipage}[t]{0.45\linewidth}"
899     write(900,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
900       "{../figures/resPg"//"-"//trim(adjustl(numberfile))//".pdf}}"
901     write(900,*)"\end{minipage}"
902     write(900,*)"\hfill"
903     write(900,*)"\begin{minipage}[t]{0.45\linewidth}"
904     write(900,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
905       "{../figures/resSg"//"-"//trim(adjustl(numberfile))//".pdf}}"
906     write(900,*)"\end{minipage}"
907     write(900,*)"\vfill"
908     write(900,*)"\begin{minipage}[t]{0.45\linewidth}"
909     write(900,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
910       "{../figures/resPn"//"-"//trim(adjustl(numberfile))//".pdf}}"
911     write(900,*)"\end{minipage}"
912     write(900,*)"\hfill"
913     write(900,*)"\begin{minipage}[t]{0.45\linewidth}"
914     write(900,*)"\centerline{\includegraphics[width=.7\linewidth,angle=-90]", &
915       "{../figures/resSn"//"-"//trim(adjustl(numberfile))//".pdf}}"
916     write(900,*)"\end{minipage}"
917     write(900,*)"\caption{\large r\'esidus aux stations (s) -- ", &
918       "la taille de l'histogramme est proportionnelle aux coefficient de pond\'eration}"
919     write(900,*)"\end{figure}"
920     write(900,*)"\end{landscape}"
921     ! ------------------------------------------------------------------     . affiche le histogrammes des résidus
922     write(900,*)"\begin{landscape} \centering"
923     write(900,*)"\begin{figure}[!ht]    %%%%%%%%%%% FIGURE reshisto %%%%%%%%%"
924     write(900,*)"\centerline{\includegraphics[width=.6\linewidth,angle=-90]", &
925       "{../figures/reshisto"//"-"//trim(adjustl(numberfile))//".pdf}}"
926     write(900,*)"\end{figure}"
927     write(900,*)"\end{landscape}"
928     write(900,*)
929     ! ------------------------------------------------------------------     . affiche le diagramme de Wadati
930     write(900,*)"\begin{landscape} \centering"
931     write(900,*)"\begin{figure}[!ht]    %%%%%%%%%%% FIGURE Wadati %%%%%%%%%"
932     write(900,*)"\centerline{\includegraphics[width=.6\linewidth,angle=-90]", &
933       "{../figures/wadatiplot"//trim(adjustl(numberfile))//".pdf}}"
934     write(900,*)"\caption{\large diagramme de Wadati }"
935     write(900,*)"\end{figure}"
936     write(900,*)"\end{landscape}"
937     write(900,*)
938     ! ------------------------------------------------------------------     . affiche l'hodochrone
939     write(900,*)"\begin{landscape} \centering"
940     write(900,*)"\begin{figure}[!ht]    %%%%%%%%%%% FIGURE Hodo %%%%%%%%%"
941     write(900,*)"\centerline{\includegraphics[width=.6\linewidth,angle=-90]", &
942       "{../figures/hodochrone"//"-"//trim(adjustl(numberfile))//".pdf}}"
943     write(900,*)"\caption{\large hodochrone}"
944     write(900,*)"\end{figure}"
945     write(900,*)"\end{landscape}"
946     write(900,*)
947     ! ------------------------------------------------------------------     . affiche le plot coda/magnitude
948     inquire (_FILE_DIR_="DATA/sac-"//trim(adjustl(numberfile)),exist=existe1) ! option différente selon compilo !
949     if ((existe1).and.(tracessac)) then
```

```
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE coda %%%%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=.6\linewidth,angle=-90]", &
        "{../figures/coda-"//trim(adjustl(numberfile))//".pdf}}"
      write(900,*)"\caption{\large dur\'ee de la coda et magnitude de dur\'ee, $M_d$}"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
      write(900,*)
      ! ------------------------------------------------------------------    . affiche le plot coda/magnitude -> map
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE coda %%%%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=.6\linewidth,angle=-90]", &
        "{../figures/coda_map-"//trim(adjustl(numberfile))//".pdf}}"
      write(900,*)"\caption{\large magnitude de dur\'ee, $M_d$, calcul\'ee \`a chaque station, pour ce s\'eisme}"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
      write(900,*)
    endif
    ! ------------------------------------------------------------------    . affiche l'histogramme de la fonction coût
    if (plotgraph) then
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE mishisto %%%%%%%%%"
      !     write(900,*)"\centerline{\includegraphics[width=1.0\linewidth]{../figures/mishisto.pdf}}"
      write(900,*)"\centerline{\includegraphics[width=.66\textwidth,angle=-90]{../figures/mishisto.pdf}}"
      write(900,*)"\caption{\large cha\^ines de Markov : fonction co\^ut des mod\`eles s\'electionn\'es}"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
    endif
    ! ------------------------------------------------------------------    . affiche VC et VM en fonction des itérations
    if (plotgraph) then
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE VC et VM %%%%%%%%%"
      !     write(900,*)"\centerline{\includegraphics[width=1.0\linewidth]{../figures/VCVM_histo.pdf}}"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/VCVM_histo.pdf}}"
      write(900,*)"\caption{\large cha\^ines de Markov : $V_c$ et $V_m$ }"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
    endif
    ! ------------------------------------------------------------------    . affiche  VpVs et Zmoho en fonction des itérations
    if (plotgraph) then
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE VpVs Zmo %%%%%%%%%"
      !     write(900,*)"\centerline{\includegraphics[width=1.0\linewidth]{../figures/VpVsZmoho_histo.pdf}}"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/VpVsZmoho_histo.pdf}}"
      write(900,*)"\caption{\large cha\^ines de Markov : $\frac{V_P}{V_S}$ et $Z_{moho}$}"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
    endif
    ! ------------------------------------------------------------------    . affiche Lat et Lon en fonction des itérations
    if (plotgraph) then
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE lon lat  %%%%%%%%%"
      !     write(900,*)"\centerline{\includegraphics[width=1.0\linewidth]{../figures/LatLon_histo.pdf}}"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/LatLon_histo"// &
        "-"//trim(adjustl(numberfile))//".pdf}}"
      write(900,*)"\caption{\large cha\^ines de Markov : $lon$ et $lat$}"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
    endif
    ! ------------------------------------------------------------------    . affiche Zhypo et Tzero en fonction des itérations
    if (plotgraph) then
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%%% FIGURE Zhypo To %%%%%%%%%"
      ! write(900,*)"\centerline{\includegraphics[width=1.0\linewidth]{../figures/ZhypoTzero_histo.pdf}}"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/ZhypoTzero_histo"// &
```

```
1015          "-"//trim(adjustl(numberfile))//".pdf}}"
1016        write(900,*)"\caption{\large cha\^ines de Markov : $T_{z\acute ero}$ et $Z_{hypo}$}"
1017        write(900,*)"\end{figure}"
1018        write(900,*)"\end{landscape}"
1019      endif
1020      ! ------------------------------------------------------------------     . affiche fonction d'autovariance de VC et VM
1021      if (plotgraph) then
1022        write(900,*)"\begin{landscape} \centering"
1023        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE VC et VM %%%%%%%%"
1024        write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/autoVCVM_histo.pdf}}"
1025        write(900,*)"\caption{\large fonction d'autovariance pour $V_c$ et $V_m$ }"
1026        write(900,*)"\end{figure}"
1027        write(900,*)"\end{landscape}"
1028      endif
1029      ! ------------------------------------------------------------------     . affiche fonction d'autovariance de VpVs et Zmoho
1030      if (plotgraph) then
1031        write(900,*)"\begin{landscape} \centering"
1032        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE VpVs Zmo %%%%%%%%"
1033        write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/autoVpVsZmoho_histo.pdf}}"
1034        write(900,*)"\caption{\large fonction d'autovariance pour $\frac{V_P}{V_S}$ et $Z_{moho}$}"
1035        write(900,*)"\end{figure}"
1036        write(900,*)"\end{landscape}"
1037      endif
1038      ! ------------------------------------------------------------------     . affiche fonction d'autovariance de Lat et Lon
1039      if (plotgraph) then
1040        write(900,*)"\begin{landscape} \centering"
1041        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE lon lat  %%%%%%%%"
1042        write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/autoLatLon_histo"// &
1043          "-"//trim(adjustl(numberfile))//".pdf}}"
1044        write(900,*)"\caption{\large fonction d'autovariance pour $lon$ et $lat$}"
1045        write(900,*)"\end{figure}"
1046        write(900,*)"\end{landscape}"
1047      endif
1048      ! ------------------------------------------------------------------     . affiche fonction d'autovariance de Zhypo et Tzero
1049      if (plotgraph) then
1050        write(900,*)"\begin{landscape} \centering"
1051        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE Zhypo To %%%%%%%%"
1052        write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/autoZhypoTzero_histo"// &
1053          "-"//trim(adjustl(numberfile))//".pdf}}"
1054        write(900,*)"\caption{\large fonction d'autovariance pour $T_{z\acute ero}$ et $Z_{hypo}$}"
1055        write(900,*)"\end{figure}"
1056        write(900,*)"\end{landscape}"
1057      endif
1058      ! ------------------------------------------------------------------     . matrice correlation parametres
1059      write(900,*)"\begin{figure}[!ht]    %%%%%%%%% FIGURE matCorr %%%%%%%%"
1060      write(900,*)"\centerline{\includegraphics[width=.9\textwidth,angle=-90]{../figures/matCorr.pdf}}"
1061      write(900,*)"\caption{\large matrice de corr\'elation}"
1062      write(900,*)"\end{figure}"
1063      ! ------------------------------------------------------------------     . étude a posteriori
1064      if (plotposteriori) then
1065        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
1066        write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/postLonLat"// &
1067          "-"//trim(adjustl(numberfile))//".png}}"
1068        write(900,*)"\caption{\large \'Etude {\em a posteriori} des param\`etres Lon et Lat}"
1069        write(900,*)"\end{figure}"
1070        ! -------                                                 --------     .
1071        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
1072        write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/post__vc"// &
1073          "-"//trim(adjustl(numberfile))//".png}}"
1074        write(900,*)"\caption{\large \'Etude {\em a posteriori} du param\`etre $V_{c}$ }"
1075        write(900,*)"\end{figure}"
1076        ! -------                                                 --------     .
1077        write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
1078        write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/post_vps"// &
1079          "-"//trim(adjustl(numberfile))//".png}}"
```

```
      write(900,*)"\caption{\large \'Etude {\em a posteriori} du param\'etre $\frac{V_P}{V_S}$ }"
      write(900,*)"\end{figure}"
      ! -------                                                  --------     .
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/post_vm"// &
        "-"//trim(adjustl(numberfile))//".png}}"
      write(900,*)"\caption{\large \'Etude {\em a posteriori} du param\'etre $V_{m}$ }"
      write(900,*)"\end{figure}"
      ! -------                                                  --------     .
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/post_zm"// &
        "-"//trim(adjustl(numberfile))//".png}}"
      write(900,*)"\caption{\large \'Etude {\em a posteriori} du param\'etre $Z_{moho}$ }"
      write(900,*)"\end{figure}"
      ! -------                                                  --------     .
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/post_to"// &
        "-"//trim(adjustl(numberfile))//".png}}"
      write(900,*)"\caption{\large \'Etude {\em a posteriori} du param\'etre $T_{z\acute ero}$ }"
      write(900,*)"\end{figure}"
      ! -------                                                  --------     .
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE post %%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/post_zh"// &
        "-"//trim(adjustl(numberfile))//".png}}"
      write(900,*)"\caption{\large \'Etude {\em a posteriori} du param\'etre $Z_{hypo}$ }"
      write(900,*)"\end{figure}"
      ! -------                                                  --------     .
    endif
    ! ------------------------------------------------------------------     . carrieres
    write(900,*)"\begin{figure}[!ht]    %%%%%%%%% FIGURE carrieres %%%%%%%"
    write(900,*)"\vspace{1cm}"
    write(900,*)"\centering"
    write(900,*)"\begin{minipage}{.45\textwidth}"
    write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/carrieresP1-"//trim(adjustl(numberfile))//".pdf}}"
    write(900,*)"\end{minipage}"
    write(900,*)"\hfill"
    write(900,*)"\begin{minipage}{.45\textwidth}"
    write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/carrieresP2-"//trim(adjustl(numberfile))//".pdf}}"
    write(900,*)"\end{minipage}"
    write(900,*)
    write(900,*)"\vspace{1cm}"
    write(900,*)
    write(900,*)"\centerline{\includegraphics[width=.9\textwidth]{../figures/carrieres-"//trim(adjustl(numberfile))//".pdf}}"
    write(900,*)"\caption{\large discrimination des tirs de carri\'eres}"
    write(900,*)"\end{figure}"
    write(900,*)"\vspace{1cm}"
    ! ------------------------------------------------------------------     . moho
    if (FLAG_non_tabulaire) then
      write(900,*)"\begin{landscape} \centering"
      write(900,*)"\begin{figure}[!ht]          %%%%%%%%% FIGURE moho_inc %%%%%%%%"
      write(900,*)"\centerline{\includegraphics[width=0.8\linewidth]{../figures/topo_moho"// &
        "-"//trim(adjustl(numberfile))//".pdf}}"
      write(900,*)"\caption{\large mohographie}"
      write(900,*)"\end{figure}"
      write(900,*)"\end{landscape}"
    endif
    ! ------------------------------------------------------------------     .
    write(900,*)"\end{document}"
    ! ------------------------------------------------------------------     .
    write(900,*)
    close(900)
    ! ------------------------------------------------------------------     .
    call system_clock(Nnewtime,ratetime)
    tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
    write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
```

```
    int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
    ! ------------------------------------------------------------------     .
  end subroutine latexone

END MODULE latexscript



! ********************************************************************     .
! ********************************************************************     .
```

2.16 SRC/MOD/MOD_rand/mt19937ar.f90

```
! ------------------------------------------------------------------     .
! --------  modification début 27/03/2014                 ---------     .1_mh
! ------------------------------------------------------------------     .
!   compilation :
!       mt19937ar.o : MOD/MOD_rand/mt19937ar.f90 modparam.o
!       gfortran -c -fno-range-check $(OPTIONC) $(FFLAGS) $<
!
! ------------------------------------------------------------------     .
! Le statisticien William Youden (en) écrit69 en 1962 une explication      !
! du but et de la position de la loi normale dans les sciences.            !
! Il la présente sous forme de courbe en cloche :                          !
!                                                                          !
!                                    THE                                   !
!                                  NORMAL                                  !
!                               LAW OF ERROR                               !
!                             STANDS OUT IN THE                            !
!                           EXPERIENCE OF MANKIND                          !
!                         AS ONE OF THE BROADEST                           !
!                       GENERALIZATIONS OF NATURAL                         !
!                     PHILOSOPHY . IT SERVES AS THE                        !
!                   GUIDING INSTRUMENT IN RESEARCHES                       !
!                IN THE PHYSICAL AND SOCIAL SCIENCES AND                   !
!              IN MEDICINE AGRICULTURE AND ENGINEERING .                   !
!         IT IS AN INDISPENSABLE TOOL FOR THE ANALYSIS AND THE             !
! INTERPRETATION OF THE BASIC DATA OBTAINED BY OBSERVATION AND EXPERIMENT  !
!                                                                          !
! ------------------------------------------------------------------     .
! --------  modification fin                              ---------     .1_mh
! ------------------------------------------------------------------     .

! A C-program for MT19937, with initialization improved 2002/1/26.
! Coded by Takuji Nishimura and Makoto Matsumoto.

! Code converted to Fortran 95 by Josi Rui Faustino de Sousa
! Date: 2002-02-01

! Before using, initialize the state by using init_genrand(seed)
! or init_by_array(init_key, key_length).

! This library is free software.
! This library is distributed in the hope that it will be useful,
! but WITHOUT ANY WARRANTY; without even the implied warranty of
! MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

! Copyright (C) 1997, 2002 Makoto Matsumoto and Takuji Nishimura.
! Any feedback is very welcome.
! http://www.math.keio.ac.jp/matumoto/emt.html
! email: matumoto@math.keio.ac.jp

      MODULE mt19937

    ! ------------------------------------------------------------------     .
```

```
53    ! --------  modification début 27/03/2014                          ---------     . 1_mh
54    ! ------------------------------------------------------------------------     .
55          use modparam
56    ! ------------------------------------------------------------------------     .
57    ! --------  modification fin                                       ---------     . 1_mh
58    ! ------------------------------------------------------------------------     .
59
60          implicit none
61
62          intrinsic :: bit_size
63
64          private
65
66    ! ------------------------------------------------------------------------     .
67    ! --------  modification début 27/03/2014                          ---------     . 2_mh
68    ! ------------------------------------------------------------------------     .
69          public  :: initseed
70    ! ------------------------------------------------------------------------     .
71    ! --------  modification fin                                       ---------     . 2_mh
72    ! ------------------------------------------------------------------------     .
73
74          public  :: init_genrand, init_by_array
75          public  :: genrand_int32, genrand_int31
76          public  :: genrand_real1, genrand_real2, genrand_real3, genrand_res53
77          public  :: normal
78
79    ! ------------------------------------------------------------------------     .
80    ! --------  modification début 27/03/2014                          ---------     . 3_mh
81    ! ------------------------------------------------------------------------     .
82    !     integer,  parameter  :: intg = selected_int_kind( 9 )
83    !     integer,  parameter  :: long = selected_int_kind( 18 )
84    !     integer,  parameter  :: flot = selected_real_kind( 6, 37 )
85    !     integer,  parameter  :: dobl = selected_real_kind( 15, 307 )
86    !     integer,  public, parameter :: wi = intg
87    !     integer,  public, parameter :: wl = long
88    !     integer,  public, parameter :: wr = dobl
89    ! ------------------------------------------------------------------------     .
90    ! --------  modification fin                                       ---------     . 3_mh
91    ! ------------------------------------------------------------------------     .
92
93    ! Period parameters
94
95          integer( kind = wi ), parameter :: n = 624_wi
96          integer( kind = wi ), parameter :: m = 397_wi
97          integer( kind = wi ), parameter :: hbs = bit_size( n ) / 2_wi
98          integer( kind = wi ), parameter :: qbs = hbs / 2_wi
99          integer( kind = wi ), parameter :: tbs = 3_wi * qbs
100
101         integer( kind = wi )  :: mt(n)                                  ! the array for the state vector
102         logical( kind = wi )  :: mtinit = .false._wi                    ! means mt[N] is not initialized
103         integer( kind = wi )  :: mti = n + 1_wi                         ! mti==N+1 means mt[N] is not initialized
104
105       contains
106
107   ! ------------------------------------------------------------------------     .
108   ! --------  modification début 27/03/2014                          ---------     . 4_mh
109   ! ------------------------------------------------------------------------     .
110   subroutine initseed(lib)
111     ! --------                                                       ---------     . mh
112     ! initialisation de la graine du générateur de nb aleatoire
113     ! fixe (lib=0) ou non (lib=1)
114     ! --------                                                       ---------     .
115     implicit none
116     ! --------                                                       ---------     .
117     logical( kind = wi ), intent (in)  :: lib
```

```
    integer( kind = wi )  :: i,seed,countn,count_rate,count_max
    ! ------------------------------------------------------------------ .
    if(lib) then
      call system_clock(countn,count_rate,count_max)
      do while (countn > 10000000)
        i = int(10000123*genrand_real1())
        countn = countn - i
      end do
      if (countn < 0) then
        write(*,*)'probleme dans initseed : mauvaise génération de graine'
        stop
      endif
      seed = countn
    else
      seed = 12345678
    endif
    call init_genrand (seed)
    write(*,*)'générateur de nombre aléatoire :       ',seed
    ! ------------------------------------------------------------------ .
  end subroutine initseed
  ! ------------------------------------------------------------------ .
  ! --------  modification fin                            ---------    . 4_mh
  ! ------------------------------------------------------------------ .

        elemental function uiadd( a, b ) result( c )

          implicit none
          intrinsic :: ibits, ior, ishft
          integer( kind = wi ), intent( in )  :: a, b
          integer( kind = wi )  :: c
          integer( kind = wi )  :: a1, a2, b1, b2, s1, s2

          a1 = ibits( a, 0, hbs )
          a2 = ibits( a, hbs, hbs )
          b1 = ibits( b, 0, hbs )
          b2 = ibits( b, hbs, hbs )
          s1 = a1 + b1
          s2 = a2 + b2 + ibits( s1, hbs, hbs )
          c  = ior( ishft( s2, hbs ), ibits( s1, 0, hbs ) )
          return
        end function uiadd

        elemental function uisub( a, b ) result( c )

          implicit none
          intrinsic :: ibits, ior, ishft
          integer( kind = wi ), intent( in )  :: a, b
          integer( kind = wi )  :: c
          integer( kind = wi )  :: a1, a2, b1, b2, s1, s2

          a1 = ibits( a, 0, hbs )
          a2 = ibits( a, hbs, hbs )
          b1 = ibits( b, 0, hbs )
          b2 = ibits( b, hbs, hbs )
          s1 = a1 - b1
          s2 = a2 - b2 + ibits( s1, hbs, hbs )
          c  = ior( ishft( s2, hbs ), ibits( s1, 0, hbs ) )
          return
        end function uisub

        elemental function uimlt( a, b ) result( c )

          implicit none
          intrinsic :: ibits, ior, ishft
          integer( kind = wi ), intent( in )  :: a, b
```

```
        integer( kind = wi )  :: c
        integer( kind = wi )  :: a0, a1, a2, a3
        integer( kind = wi )  :: b0, b1, b2, b3
        integer( kind = wi )  :: p0, p1, p2, p3

        a0 = ibits( a, 0, qbs )
        a1 = ibits( a, qbs, qbs )
        a2 = ibits( a, hbs, qbs )
        a3 = ibits( a, tbs, qbs )
        b0 = ibits( b, 0, qbs )
        b1 = ibits( b, qbs, qbs )
        b2 = ibits( b, hbs, qbs )
        b3 = ibits( b, tbs, qbs )
        p0 = a0 * b0
        p1 = a1 * b0 + a0 * b1 + ibits( p0, qbs, tbs )
        p2 = a2 * b0 + a1 * b1 + a0 * b2 + ibits( p1, qbs, tbs )
        p3 = a3 * b0 + a2 * b1 + a1 * b2 + a0 * b3 + ibits( p2, qbs, tbs )
        c  = ior( ishft( p1, qbs ), ibits( p0, 0, qbs ) )
        c  = ior( ishft( p2, hbs ), ibits( c, 0, hbs ) )
        c  = ior( ishft( p3, tbs ), ibits( c, 0, tbs ) )
        return
      end function uimlt

! initializes mt[N] with a seed
      subroutine init_genrand( s )

        implicit none
        intrinsic :: iand, ishft, ieor, ibits
        integer( kind = wi ), intent( in )  :: s
        integer( kind = wi )  :: i, mult_a

        data mult_a /z'6C078965'/

        mtinit = .true._wi
        mt(1) = ibits( s, 0, 32 )
        do i = 2, n, 1
           mt(i) = ieor( mt(i-1), ishft( mt(i-1), -30 ) )
           mt(i) = uimlt( mt(i), mult_a )
           mt(i) = uiadd( mt(i), uisub( i, 1_wi ) )
    ! See Knuth TAOCP Vol2. 3rd Ed. P.106 for multiplier.
    ! In the previous versions, MSBs of the seed affect
    ! only MSBs of the array mt[].
    ! 2002/01/09 modified by Makoto Matsumoto
           mt(i) = ibits( mt(i), 0, 32 )
    ! for >32 bit machines
        end do
        return
      end subroutine init_genrand

! initialize by an array with array-length
! init_key is the array for initializing keys
! key_length is its length
      subroutine init_by_array( init_key )

        implicit none
        intrinsic :: iand, ishft, ieor
        integer( kind = wi ), intent( in )  :: init_key(:)
        integer( kind = wi )  :: i, j, k, tp, key_length
        integer( kind = wi )  :: seed_d, mult_a, mult_b, msb1_d

        data seed_d /z'12BD6AA'/
        data mult_a /z'19660D'/
        data mult_b /z'5D588B65'/
        data msb1_d /z'80000000'/

```

```
          key_length = size( init_key, dim = 1 )
          call init_genrand( seed_d )
          i = 2_wi
          j = 1_wi
          do k = max( n, key_length ), 1, -1
             tp = ieor( mt(i-1), ishft( mt(i-1), -30 ) )
             tp = uimlt( tp, mult_a )
             mt(i) = ieor( mt(i), tp )
             mt(i) = uiadd( mt(i), uiadd( init_key(j), uisub( j, 1_wi ) ) ) ! non linear
             mt(i) = ibits( mt(i), 0, 32 ) ! for WORDSIZE > 32 machines
             i = i + 1_wi
             j = j + 1_wi
             if ( i > n ) then
                mt(1) = mt(n)
                i = 2_wi
             end if
             if ( j > key_length) j = 1_wi
          end do
          do k = n-1, 1, -1
             tp = ieor( mt(i-1), ishft( mt(i-1), -30 ) )
             tp = uimlt( tp, mult_b )
             mt(i) = ieor( mt(i), tp )
             mt(i) = uisub( mt(i), uisub( i, 1_wi ) ) ! non linear
             mt(i) = ibits( mt(i), 0, 32 ) ! for WORDSIZE > 32 machines
             i = i + 1_wi
             if ( i > n ) then
                mt(1) = mt(n)
                i = 2_wi
             end if
          end do
          mt(1) = msb1_d ! MSB is 1; assuring non-zero initial array
          return
        end subroutine init_by_array

  ! generates a random number on [0,0xffffffff]-interval
        function genrand_int32( ) result( y )

          implicit none
          intrinsic :: iand, ishft, ior, ieor, btest, ibset, mvbits
          integer( kind = wi )  :: y
          integer( kind = wi )  :: kk
          integer( kind = wi )  :: seed_d, matrix_a, matrix_b, temper_a,temper_b

          data seed_d   /z'5489'/
          data matrix_a /z'9908B0DF'/
          data matrix_b /z'0'/
          data temper_a /z'9D2C5680'/
          data temper_b /z'EFC60000'/

          if ( mti > n ) then ! generate N words at one time
             if ( .not. mtinit ) call init_genrand( seed_d ) ! if init_genrand() has not been called, a default initial seed is used
             do kk = 1, n-m, 1
                y = ibits( mt(kk+1), 0, 31 )
                call mvbits( mt(kk), 31, 1, y, 31 )
                if ( btest( y, 0 ) ) then
                   mt(kk) = ieor( ieor( mt(kk+m), ishft( y, -1 ) ), matrix_a )
                else
                   mt(kk) = ieor( ieor( mt(kk+m), ishft( y, -1 ) ), matrix_b )
                end if
             end do
             do kk = n-m+1, n-1, 1
                y = ibits( mt(kk+1), 0, 31 )
                call mvbits( mt(kk), 31, 1, y, 31 )
                if ( btest( y, 0 ) ) then
                   mt(kk) = ieor( ieor( mt(kk+m-n), ishft( y, -1 ) ), matrix_a )
```

```
                else
                   mt(kk) = ieor( ieor( mt(kk+m-n), ishft( y, -1 ) ), matrix_b )
                end if
             end do
             y = ibits( mt(1), 0, 31 )
             call mvbits( mt(n), 31, 1, y, 31 )
             if ( btest( y, 0 ) ) then
                mt(kk) = ieor( ieor( mt(m), ishft( y, -1 ) ), matrix_a )
             else
                mt(kk) = ieor( ieor( mt(m), ishft( y, -1 ) ), matrix_b )
             end if
             mti = 1_wi
          end if
          y = mt(mti)
          mti = mti + 1_wi
          ! Tempering
          y = ieor( y, ishft( y, -11) )
          y = ieor( y, iand( ishft( y, 7 ), temper_a ) )
          y = ieor( y, iand( ishft( y, 15 ), temper_b ) )
          y = ieor( y, ishft( y, -18 ) )
          return
        end function genrand_int32

  ! generates a random number on [0,0x7fffffff]-interval
        function genrand_int31( ) result( i )

          implicit none
          intrinsic :: ishft
          integer( kind = wi )  :: i
          i = ishft( genrand_int32( ), -1 )
          return
        end function genrand_int31

  ! generates a random number on [0,1]-real-interval
        function genrand_real1( ) result( r )

          implicit none
          real( kind = wr )  :: r
          integer( kind = wi )  :: a, a1, a0

          a = genrand_int32( )
          a0 = ibits( a, 0, hbs )
          a1 = ibits( a, hbs, hbs )
           r = real( a0, kind = wr ) * 2.32830643708079737543l5e-10_wr
!          r = real( a0, kind = wr ) / 4294967295.0_wr
           r = real( a1, kind = wr ) * 1.525878906605271367963e-5_wr    + r
!          r = real( a1, kind = wr ) * ( 65536.0_wr / 4294967295.0_wr ) + r
    ! divided by 2^32-1
          return
        end function genrand_real1

  ! generates a random number on [0,1)-real-interval
        function genrand_real2( ) result( r )

          implicit none
          intrinsic :: ibits
          real( kind = wr )  :: r
          integer( kind = wi )  :: a, a1, a0

          a = genrand_int32( )
          a0 = ibits( a, 0, hbs )
          a1 = ibits( a, hbs, hbs )
          r = real( a0, kind = wr ) * 2.3283064365386962890625e-10_wr
!          r = real( a0, kind = wr ) / 4294967296.0_wr
          r = real( a1, kind = wr ) * 1.52587890625e-5_wr + r
```

```
!          r = real( a1, kind = wr ) / 65536.0_wr + r
   ! divided by 2^32
         return
       end function genrand_real2

  ! generates a random number on (0,1)-real-interval
       function genrand_real3( ) result( r )

         implicit none
         real( kind = wr )  :: r
         integer( kind = wi )  :: a, a1, a0

         a = genrand_int32( )
         a0 = ibits( a, 0, hbs )
         a1 = ibits( a, hbs, hbs )
         r = ( real( a0, kind = wr ) + 0.5_wr ) * 2.3283064365386962890625e-10_wr
!         r = ( real( a0, kind = wr ) + 0.5_wr ) / 4294967296.0_wr
         r = real( a1, kind = wr ) * 1.52587890625e-5_wr + r
!         r = real( a1, kind = wr ) / 65536.0_wr + r
   ! divided by 2^32
         return
       end function genrand_real3

  ! generates a random number on [0,1) with 53-bit resolution
       function genrand_res53( )  result( r )

         implicit none
         intrinsic :: ishft
         real( kind = wr )  :: r
         integer( kind = wi )  :: a, a0, a1
         integer( kind = wi )  :: b, b0, b1

         a = ishft( genrand_int32( ), -5 )
         a0 = ibits( a, 0, hbs )
         a1 = ibits( a, hbs, hbs )
         b = ishft( genrand_int32( ), -6 )
         b0 = ibits( b, 0, hbs )
         b1 = ibits( b, hbs, hbs )
         r = real( a1, kind = wr ) * 4.8828125e-4_wr
!         r = real( a1, kind = wr ) / 2048.0_wr
         r = real( a0, kind = wr ) * 7.450580596923828125e-9_wr + r
!         r = real( a0, kind = wr ) / 134217728.0_wr + r
         r = real( b1, kind = wr ) * 7.27595761418342590332e-12_wr + r
!         r = real( b1, kind = wr ) / 137438953472.0_wr + r
         r = real( b0, kind = wr ) * 1.1102230246251565404236e-16_wr + r
!         r = real( b0, kind = wr ) / 9007199254740992.0_wr + r
         return
       end function genrand_res53
  ! These real versions are due to Isaku Wada, 2002/01/09 added

  ! tirage au sort selon une loi normale

      function normal0()

         implicit none
         real(kind=wr)::r1,r2,rsq,normal0
         real(wr),save::g
         logical,save::gaus_stored=.false.
             if (gaus_stored) then
                     normal0=g
                     gaus_stored=.false.
             else
             do
                     r1=2._wr*genrand_real1()-1._wr
                     r2=2._wr*genrand_real1()-1._wr
```

```
                        rsq=r1**2+r2**2
                        if (rsq >0.0_wr .and. rsq < 1.0) exit
                        end do
                        rsq=sqrt(-2.0_wr*log(rsq)/rsq)
                        normal0=r1*rsq
                        g=r2*rsq
                        gaus_stored=.true.
                    end if
                return
            end function normal0

            function normal(m,s)

                implicit none
                real(kind=wr):: normal
                real(wr),intent(in)::m,s
                    normal=s*normal0()+m
                return
            end function normal

        END MODULE mt19937
```

2.17 SRC/MOD/MOD_sac/mod_sac_io.f90

```
MODULE sac_i_o

!-----------------------------------------------------------------------

! This module allows the reading and writing of sac files in either
! alphanumeric or binary format.  All the header info is included,
! allowing for exact duplication of a read in/written out record.

! 1) Compile with:
!    f95 mod_sac_io.f90 "your_program" -kind=byte -o sac.x

! This typically produces the following warning (at least on my machine):
! Extension: mod_sac_io.f90, line 209: ALLOCATABLE dummy argument YARRAY
!            detected at ::@YARRAY
! Extension: mod_sac_io.f90, line 460: ALLOCATABLE dummy argument YARRAY
!            detected at ::@YARRAY
!
! This can be safely ignored.

! 2) Make sure to include the following line in your programs
!    USE sac_i_o

! Variables
! ========================================
! Sac header variables

! REALS: (32-bit or 4 bytes each)
! delta, depmin, depmax, scale, odelta
! b, e, o, a, internal1
! t0, t1, t2, t3, t4
! t5, t6, t7, t8, t9
! f, resp0, resp1, resp2, resp3
! resp4, resp5, resp6, resp7, resp8
! resp9, stla, stlo, stel, stdp
! resp9, stla, stlo, stel, stdp
! evla, evlo, evel, evdp, mag
! user0, user1, user2, user3, user4
! user5, user6, user7, user8, user9
! dist, az, baz, gcarc, internal2
! internal3,  depmen, cmpaz, cmpinc, xminimum
! xmaximum,   yminimum,   ymaximum,   unused1, unused2
```

```
! unused3, unused4, unused5, unused6, unused7

! INTEGERS: (32-bit or 4 bytes each)
! nzyear, nzjday, nzhour, nzmin, nzsec
! nzmsec, nvhdr, norid, nevid, npts
! internal4,    nwfid, nxsize, nysize, unused8
! iftype, idep, iztype, unused9, iinst
! istreg, ievreg, ievtyp, iqual, isynth
! imagtyp, imagsrc, unused10, unused11, unused12
! unused13, unused14, unused15, unused16, unused17

! LOGICALS: (32-bit or 4 bytes each)
! leven, lpspol, lovrok, lcalda, unused18

! CHARACTERS: (64-bit or 8 bytes each)
! kstnm, kevnm*
! khole,ko,ka
! kt0,kt1,kt2
! kt3,kt4,kt5
! kt6,kt7,kt8
! kt9,kf,kuser0
! kuser1,kuser2,kcmpnm
! knetwk,kdatrd,kinst
! yarray

! *128 bit or 16 bytes

! Subroutines Included:
!=====================================
! rbsac: reads binary sacfile
! rasac: reads alphanumeric sacfile
! wbsac: writes binary sacfile
! wasac: writes alphanumeric sacfile


! To call the subroutines use (and replace rbsac w/ rasac, wasac, or wbsac):
!=====================================
! CALL rbsac(infile,delta,depmin,depmax,scale,odelta,b,e,o,a,internal1,          &
! t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,     &
! resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,     &
! user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
! internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
! unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
! nzsec,nzmsec,nvhdr,norid,nevid,npts,internal4,nwfid,nxsize,nysize,unused8,    &
! iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
! imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
! unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
! kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
! kinst,yarray)

! To undefine a header in your program:
!=====================================
! real_header_variable = rundef
! intg_header_variable = iundef
! logi_header_variable = ltrue -or- lfalse
! char_header_variable = kundef

!-----------------------------------------------------------------------

IMPLICIT NONE

! Header variables

! Select kind type (byte)
   INTEGER,PARAMETER :: k=4_4
```

```
! Real sac header variables
  REAL(k),PARAMETER :: rundef=-12345.0_4
  REAL(k) :: delta,depmin,depmax,scale,odelta
  REAL(k) :: b,e,o,a,internal1
  REAL(k) :: t0,t1,t2,t3,t4
  REAL(k) :: t5,t6,t7,t8,t9
  REAL(k) :: f,resp0,resp1,resp2,resp3
  REAL(k) :: resp4,resp5,resp6,resp7,resp8
  REAL(k) :: resp9,stla,stlo,stel,stdp
  REAL(k) :: evla,evlo,evel,evdp,mag
  REAL(k) :: user0,user1,user2,user3,user4
  REAL(k) :: user5,user6,user7,user8,user9
  REAL(k) :: dist,az,baz,gcarc,internal2
  REAL(k) :: internal3,depmen,cmpaz,cmpinc,xminimum
  REAL(k) :: xmaximum,yminimum,ymaximum,unused1,unused2
  REAL(k) :: unused3,unused4,unused5,unused6,unused7
! Integer sac header variables
  INTEGER(k),PARAMETER :: iundef=-12345_4
  INTEGER(k) :: nzyear,nzjday,nzhour,nzmin,nzsec
  INTEGER(k) :: nzmsec,nvhdr,norid,nevid,npts
  INTEGER(k) :: internal4,nwfid,nxsize,nysize,unused8
  INTEGER(k) :: iftype,idep,iztype,unused9,iinst
  INTEGER(k) :: istreg,ievreg,ievtyp,iqual,isynth
  INTEGER(k) :: imagtyp,imagsrc,unused10,unused11,unused12
  INTEGER(k) :: unused13,unused14,unused15,unused16,unused17
! Logical sac header variables
  INTEGER(k),PARAMETER :: ltrue=1_4,lfalse=0_4
  INTEGER(k) :: leven,lpspol,lovrok,lcalda,unused18
! Character sac header variables
  CHARACTER(LEN=8),PARAMETER :: kundef='-12345'
  CHARACTER(LEN=16) :: kevnm
  CHARACTER(LEN=8) :: kstnm
  CHARACTER(LEN=8) :: khole,ko,ka
  CHARACTER(LEN=8) :: kt0,kt1,kt2
  CHARACTER(LEN=8) :: kt3,kt4,kt5
  CHARACTER(LEN=8) :: kt6,kt7,kt8
  CHARACTER(LEN=8) :: kt9,kf,kuser0
  CHARACTER(LEN=8) :: kuser1,kuser2,kcmpnm
  CHARACTER(LEN=8) :: knetwk,kdatrd,kinst
! Data array
  REAL(k), ALLOCATABLE :: yarray(:)

CONTAINS

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

 SUBROUTINE rbsac(infile,delta,depmin,depmax,scale,odelta,b,e,o,a,internal1,  &
t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
nzsec,nzmsec,nvhdr,norid,nevid,npts,internal4,nwfid,nxsize,nysize,unused8,    &
iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
kinst,yarray)

! Read in sac binary format file with ALL the header information
! ========================================
! Header values are as described on
! the sac homepage http://www.llnl.gov/sac/

```

```
! Note: With NAGWare Fortran 95 Compiler
! must compile with -kind=byte;
! has not been tested on non-Macintosh/Unix machines

  IMPLICIT NONE

  ! Real header variables
  INTEGER, PARAMETER :: k=4_4  !32 bit (4 byte) words
  REAL(k), INTENT(OUT) :: delta,depmin,depmax,scale,odelta
  REAL(k), INTENT(OUT) :: b,e,o,a,internal1
  REAL(k), INTENT(OUT) :: t0,t1,t2,t3,t4
  REAL(k), INTENT(OUT) :: t5,t6,t7,t8,t9
  REAL(k), INTENT(OUT) :: f,resp0,resp1,resp2,resp3
  REAL(k), INTENT(OUT) :: resp4,resp5,resp6,resp7,resp8
  REAL(k), INTENT(OUT) :: resp9,stla,stlo,stel,stdp
  REAL(k), INTENT(OUT) :: evla,evlo,evel,evdp,mag
  REAL(k), INTENT(OUT) :: user0,user1,user2,user3,user4
  REAL(k), INTENT(OUT) :: user5,user6,user7,user8,user9
  REAL(k), INTENT(OUT) :: dist,az,baz,gcarc,internal2
  REAL(k), INTENT(OUT) :: internal3,depmen,cmpaz,cmpinc,xminimum
  REAL(k), INTENT(OUT) :: xmaximum,yminimum,ymaximum,unused1,unused2
  REAL(k), INTENT(OUT) :: unused3,unused4,unused5,unused6,unused7

  ! Integer header variables
  INTEGER(k), INTENT(OUT) :: nzyear,nzjday,nzhour,nzmin,nzsec
  INTEGER(k), INTENT(OUT) :: nzmsec,nvhdr,norid,nevid,npts
  INTEGER(k), INTENT(OUT) :: internal4,nwfid,nxsize,nysize,unused8
  INTEGER(k), INTENT(OUT) :: iftype,idep,iztype,unused9,iinst
  INTEGER(k), INTENT(OUT) :: istreg,ievreg,ievtyp,iqual,isynth
  INTEGER(k), INTENT(OUT) :: imagtyp,imagsrc,unused10,unused11,unused12
  INTEGER(k), INTENT(OUT) :: unused13,unused14,unused15,unused16,unused17

  ! Logical header variables (1=true 0=false)
  INTEGER(k), INTENT(OUT) :: leven,lpspol,lovrok,lcalda,unused18

  ! Character header values
  CHARACTER(LEN=100), INTENT(IN) :: infile
  CHARACTER(LEN=16), INTENT(OUT) :: kevnm
  CHARACTER(LEN=8), INTENT(OUT) :: kstnm
  CHARACTER(LEN=8), INTENT(OUT) :: khole,ko,ka
  CHARACTER(LEN=8), INTENT(OUT) :: kt0,kt1,kt2
  CHARACTER(LEN=8), INTENT(OUT) :: kt3,kt4,kt5
  CHARACTER(LEN=8), INTENT(OUT) :: kt6,kt7,kt8
  CHARACTER(LEN=8), INTENT(OUT) :: kt9,kf,kuser0
  CHARACTER(LEN=8), INTENT(OUT) :: kuser1,kuser2,kcmpnm
  CHARACTER(LEN=8), INTENT(OUT) :: knetwk,kdatrd,kinst
  ! Data array
  REAL(k), ALLOCATABLE,INTENT(OUT) :: yarray(:)

  ! Internal variables
  INTEGER(k) :: io,j,N,tot


     ! Read in float and integer data (each variable 4 units)
     OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=20)

     ! Position      Header Variables
     ! 1-20         DELTA DEPMIN DEPMAX SCALE ODELTA
     read(99,REC=1) delta, depmin, depmax, scale, odelta

     ! 21-40        B  E  O  A       INTERNAL
     read(99,REC=2) b, e, o, a, internal1

     ! 41-60             T0      T1      T2      T3      T4
     read(99,REC=3) t0, t1, t2, t3, t4
```

```
      ! 61-80              T5       T6       T7       T8       T9
      read(99,REC=4) t5, t6, t7, t8, t9

      ! 81-100              F        RESP0    RESP1    RESP2    RESP3
      read(99,REC=5) f, resp0, resp1, resp2, resp3

      ! 101-120             RESP4    RESP5    RESP6    RESP7    RESP8
      read(99,REC=6) resp4, resp5, resp6, resp7, resp8

      ! 121-140             RESP9    STLA   STLO   STEL   STDP
      read(99,REC=7) resp9, stla, stlo, stel, stdp

      ! 141-160             EVLA   EVLO   EVEL     EVDP   MAG
      read(99,REC=8) evla, evlo, evel, evdp, mag

      ! 161-180             USER0    USER1    USER2    USER3    USER4
      read(99,REC=9) user0, user1, user2, user3, user4

      ! 181-200             USER5    USER6    USER7    USER8    USER9
      read(99,REC=10) user5, user6, user7, user8, user9

      ! 201-220             DIST     AZ       BAZ      GCARC    INTERNAL
      read(99,REC=11) dist, az, baz, gcarc, internal2

      ! 221-240             INTERNAL DEPMEN   CMPAZ    CMPINC   XMINIMUM
      read(99,REC=12) internal3,  depmen, cmpaz, cmpinc, xminimum

      ! 241-260             XMAXIMUM   YMINIMUM   YMAXIMUM   UNUSED   UNUSED
      read(99,REC=13) xmaximum,   yminimum,   ymaximum,   unused1, unused2

      ! 261-280             UNUSED   UNUSED   UNUSED   UNUSED   UNUSED
      read(99,REC=14) unused3, unused4, unused5, unused6, unused7

      !read integer header variables
      !-----------------------------------------------------
      ! 281-300             NZYEAR   NZJDAY   NZHOUR   NZMIN    NZSEC
      read(99,REC=15) nzyear, nzjday, nzhour, nzmin, nzsec

      ! 301-320             NZMSEC   NVHDR    NORID    NEVID    NPTS
      read(99,REC=16) nzmsec, nvhdr, norid, nevid, npts

      ! 321-340             INTERNAL   NWFID    NXSIZE   NYSIZE   UNUSED
      read(99,REC=17) internal4,   nwfid, nxsize, nysize, unused8

      ! 341-360             IFTYPE   IDEP     IZTYPE   UNUSED   IINST
      read(99,REC=18) iftype, idep, iztype, unused9, iinst

      ! 361-380             ISTREG   IEVREG   IEVTYP   IQUAL    ISYNTH
      read(99,REC=19) istreg, ievreg, ievtyp, iqual, isynth

      ! 381-400             IMAGTYP IMAGSRC UNUSED   UNUSED   UNUSED
      read(99,REC=20) imagtyp, imagsrc, unused10, unused11, unused12

      ! 401-420             UNUSED   UNUSED   UNUSED   UNUSED   UNUSED
      read(99,REC=21) unused13, unused14, unused15, unused16, unused17

      !read logical header variables
      !-----------------------------------------------------
      ! 421-440             LEVEN   LPSPOL   LOVROK LCALDA UNUSED
      read(99,REC=22) leven, lpspol, lovrok, lcalda, unused18
      CLOSE(99)

      ! Character variable (size=8)*    *kevnm=16
      ! Here it gets tricky; we have to read in each k-header separately
```

```
302       ! due to their different size... so I deviate from a straight up
303       ! copy of the alpha numeric format
304
305       !read character header variables
306       !-------------------------------------------------------
307       ! 440-448               KSTNM
308       OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=8)
309       read(99,REC=56) kstnm
310       CLOSE(99)
311
312       ! 449-464               KEVNM
313       OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=16)
314       read(99,REC=29) kevnm
315
316       ! 465-480               KHOLE    KO
317       read(99,REC=30) khole, ko
318       CLOSE(99)
319
320       ! 480-504               KA    KT0      KT1
321       OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=24)
322       read(99,REC=21) ka, kt0, kt1
323
324       ! 504-528               KT2      KT3      KT4
325       read(99,REC=22) kt2, kt3, kt4
326       ! 529-552                 KT5      KT6      KT7
327       read(99,REC=23) kt5, kt6, kt7
328
329       ! 553-576               KT8   KT9    KF
330       read(99,REC=24) kt8, kt9, kf
331
332       ! 577-600               KUSER0 KUSER1   KUSER2
333       read(99,REC=25) kuser0, kuser1, kuser2
334
335       ! 601-624               KCMPNM KNETWK   KDATRD
336       read(99,REC=26) kcmpnm, knetwk, kdatrd
337       CLOSE(99)
338
339       ! 625-632               KINST
340       OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=8)
341       read(99,REC=79) kinst
342       CLOSE(99)
343
344       !read data
345       ! 633 ->  npts*4
346       !-------------------------------------------------------
347       ALLOCATE (yarray(npts))
348
349
350       ! Efficiency counts; I've worked out a way of reading in multiple
351       ! binary values at once
352
353       ! 1) If there are less than 158 points, then a simple do loop is
354       ! best
355       ! 2) If there are more than 158 points, you want to read in blocks
356       ! of 158 pts each (Direct access requires you keep track of where
357       ! you are in the file; multiples of 158 are good as there are 158
358       ! header variables, so you count them as record 1).
359       ! However, you also need to read in the rest of the points as it
360       ! is unlikely files will always have a multiple of 158 for npts
361
362       ! Case 1: less than 158 pts in the sacfile
363       IF (npts < 158_4) THEN
364 !WRITE(*,*) "npts < 158"
365
366         OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=4)
```

```
367         DO j=159_4,(159_4+npts)
368
369            READ(99,REC=j,iostat=io) yarray(j-158_4)
370            IF (io /= 0) EXIT
371
372         END DO
373         CLOSE(99)
374
375       ! Case 2:
376       ELSE IF (npts >= 158_4) THEN
377 !WRITE(*,*) "npts > 158"
378
379         OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=632)
380
381         N=int(real(npts)/158_4)+1_4
382
383 !WRITE(*,*) N,npts
384
385         ! Start reading in the large record blocks (158*4 blocks each)
386         DO j=2_4,N
387
388            READ(99,REC=j,iostat=io) yarray(158_4*(j-2_4)+1_4:158_4*(j-1_4))
389 !WRITE(*,*) io
390            IF (io /= 0) EXIT
391         END DO
392         CLOSE(99)
393
394         tot=npts-158_4*int(real(npts)/158_4)
395
396 !WRITE(*,*) tot
397
398         ! Case 3: Read in the rest
399         OPEN(UNIT=99,FILE=infile,FORM='unformatted',ACCESS='DIRECT',RECL=4)
400         DO j=1,tot
401            READ(99,REC=((158_4*N)+j),iostat=io) yarray(158_4*(N-1_4)+j)
402 ! WRITE(*,*) 158_4*(N)+j,yarray(158_4*(N)+j),io
403 !            IF (io /= 0) EXIT
404         END DO
405         CLOSE(99)
406
407       END IF
408
409
410 ! For Debugging:
411 !       write(*,201) delta, depmin, depmax, scale, odelta
412 !       write(*,201) b, e, o, a, internal1
413 !       write(*,201) t0, t1, t2, t3, t4
414 !       write(*,201) t5, t6, t7, t8, t9
415 !       write(*,201) f, resp0, resp1, resp2, resp3
416 !       write(*,201) resp4, resp5, resp6, resp7, resp8
417 !       write(*,201) resp9, stla, stlo, stel, stdp
418 !       write(*,201) resp9, stla, stlo, stel, stdp
419 !       write(*,201) evla, evlo, evel, evdp, mag
420 !       write(*,201) user0, user1, user2, user3, user4
421 !       write(*,201) user5, user6, user7, user8, user9
422 !       write(*,201) dist, az, baz, gcarc, internal2
423 !       write(*,201) internal3,  depmen, cmpaz, cmpinc, xminimum
424 !       write(*,201) xmaximum,   yminimum,   ymaximum,   unused1, unused2
425 !       write(*,201) unused3, unused4, unused5, unused6, unused7
426 !       write(*,202) nzyear, nzjday, nzhour, nzmin, nzsec
427 !       write(*,202) nzmsec, nvhdr, norid, nevid, npts
428 !       write(*,202) internal4,   nwfid, nxsize, nysize, unused8
429 !       write(*,202) iftype, idep, iztype, unused9, iinst
430 !       write(*,202) istreg, ievreg, ievtyp, iqual, isynth
431 !       write(*,202) imagtyp, imagsrc, unused10, unused11, unused12
```

```
!       write(*,202) unused13, unused14, unused15, unused16, unused17
!       write(*,202) leven, lpspol, lovrok, lcalda, unused18
!       write(*,203) kstnm, kevnm
!       write(*,204) khole,ko,ka
!       write(*,204) kt0,kt1,kt2
!       write(*,204) kt3,kt4,kt5
!       write(*,204) kt6,kt7,kt8
!       write(*,204) kt9,kf,kuser0
!       write(*,204) kuser1,kuser2,kcmpnm
!       write(*,204) knetwk,kdatrd,kinst
!       write(*,205) yarray

!     201 FORMAT(G15.7,G15.7,G15.7,G15.7,G15.7)
!     202 FORMAT(I10,I10,I10,I10,I10)
!     203 FORMAT(A8,A16)
!     204 FORMAT(A8,A8,A8)
!     205 FORMAT(G15.7,G15.7,G15.7,G15.7,G15.7)

  END SUBROUTINE rbsac

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SUBROUTINE rasac(infile,delta,depmin,depmax,scale,odelta,b,e,o,a,internal1,&
t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
nzsec,nzmsec,nvhdr,norid,nevid,npts,internal4,nwfid,nxsize,nysize,unused8,    &
iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
kinst,yarray)

! Read in sac alphanumeric format file with ALL the header information
! ====================================
! Header values are as described on
! the sac homepage http://www.llnl.gov/sac/

! Note: With NAGWare Fortran 95 Compiler
! must compile with -kind=byte;
! has not been tested on non-Macintosh/Unix machines

    IMPLICIT NONE

    ! Real header variables
    REAL, INTENT(OUT) :: delta,depmin,depmax,scale,odelta
    REAL, INTENT(OUT) :: b,e,o,a,internal1
    REAL, INTENT(OUT) :: t0,t1,t2,t3,t4
    REAL, INTENT(OUT) :: t5,t6,t7,t8,t9
    REAL, INTENT(OUT) :: f,resp0,resp1,resp2,resp3
    REAL, INTENT(OUT) :: resp4,resp5,resp6,resp7,resp8
    REAL, INTENT(OUT) :: resp9,stla,stlo,stel,stdp
    REAL, INTENT(OUT) :: evla,evlo,evel,evdp,mag
    REAL, INTENT(OUT) :: user0,user1,user2,user3,user4
    REAL, INTENT(OUT) :: user5,user6,user7,user8,user9
    REAL, INTENT(OUT) :: dist,az,baz,gcarc,internal2
    REAL, INTENT(OUT) :: internal3,depmen,cmpaz,cmpinc,xminimum
    REAL, INTENT(OUT) :: xmaximum,yminimum,ymaximum,unused1,unused2
    REAL, INTENT(OUT) :: unused3,unused4,unused5,unused6,unused7

    ! integer header variables
    INTEGER, INTENT(OUT) :: nzyear,nzjday,nzhour,nzmin,nzsec
```

```
497     INTEGER, INTENT(OUT) :: nzmsec,nvhdr,norid,nevid,npts
498     INTEGER, INTENT(OUT) :: internal4,nwfid,nxsize,nysize,unused8
499     INTEGER, INTENT(OUT) :: iftype,idep,iztype,unused9,iinst
500     INTEGER, INTENT(OUT) :: istreg,ievreg,ievtyp,iqual,isynth
501     INTEGER, INTENT(OUT) :: imagtyp,imagsrc,unused10,unused11,unused12
502     INTEGER, INTENT(OUT) :: unused13,unused14,unused15,unused16,unused17
503
504     ! logical header variables (1=true 0=false)
505     INTEGER, INTENT(OUT) :: leven,lpspol,lovrok,lcalda,unused18
506
507     ! character header values
508     CHARACTER(LEN=90), INTENT(IN) :: infile
509     CHARACTER(LEN=16), INTENT(OUT) :: kevnm
510     CHARACTER(LEN=8), INTENT(OUT) :: kstnm
511     CHARACTER(LEN=8), INTENT(OUT) :: khole,ko,ka
512     CHARACTER(LEN=8), INTENT(OUT) :: kt0,kt1,kt2
513     CHARACTER(LEN=8), INTENT(OUT) :: kt3,kt4,kt5
514     CHARACTER(LEN=8), INTENT(OUT) :: kt6,kt7,kt8
515     CHARACTER(LEN=8), INTENT(OUT) :: kt9,kf,kuser0
516     CHARACTER(LEN=8), INTENT(OUT) :: kuser1,kuser2,kcmpnm
517     CHARACTER(LEN=8), INTENT(OUT) :: knetwk,kdatrd,kinst
518
519     ! Data array
520     REAL, ALLOCATABLE,INTENT(OUT) :: yarray(:)
521
522     ! status='replace' removed
523     OPEN(UNIT=777,FILE=infile)
524
525         !read in real header variables
526         !-----------------------------------------------------
527         !               DELTA   DEPMIN   DEPMAX   SCALE    ODELTA
528         read(777,201) delta, depmin, depmax, scale, odelta
529
530         !               B   E   O   A        INTERNAL
531         read(777,201) b, e, o, a, internal1
532
533         !               T0       T1       T2       T3       T4
534         read(777,201) t0, t1, t2, t3, t4
535
536         !               T5       T6       T7       T8       T9
537         read(777,201) t5, t6, t7, t8, t9
538
539         !               F        RESP0    RESP1    RESP2    RESP3
540         read(777,201) f, resp0, resp1, resp2, resp3
541
542         !               RESP4    RESP5    RESP6    RESP7    RESP8
543         read(777,201) resp4, resp5, resp6, resp7, resp8
544
545         !               RESP9    STLA   STLO   STEL   STDP
546         read(777,201) resp9, stla, stlo, stel, stdp
547
548         !               EVLA   EVLO   EVEL     EVDP   MAG
549         read(777,201) evla, evlo, evel, evdp, mag
550
551         !               USER0    USER1    USER2    USER3    USER4
552         read(777,201) user0, user1, user2, user3, user4
553
554         !               USER5    USER6    USER7    USER8    USER9
555         read(777,201) user5, user6, user7, user8, user9
556
557         !               DIST     AZ       BAZ      GCARC    INTERNAL
558         read(777,201) dist, az, baz, gcarc, internal2
559
560         !               INTERNAL DEPMEN   CMPAZ   CMPINC   XMINIMUM
561         read(777,201) internal3,  depmen, cmpaz, cmpinc, xminimum
```

```
      !              XMAXIMUM   YMINIMUM   YMAXIMUM   UNUSED   UNUSED
      read(777,201) xmaximum,    yminimum,    ymaximum,    unused1, unused2

      !              UNUSED   UNUSED   UNUSED   UNUSED   UNUSED
      read(777,201) unused3, unused4, unused5, unused6, unused7


      !read integer header variables
      !-------------------------------------------------------
      !              NZYEAR   NZJDAY   NZHOUR   NZMIN   NZSEC
      read(777,202) nzyear, nzjday, nzhour, nzmin, nzsec

      !              NZMSEC   NVHDR   NORID    NEVID    NPTS
      read(777,202) nzmsec, nvhdr, norid, nevid, npts

      !              INTERNAL   NWFID    NXSIZE   NYSIZE   UNUSED
      read(777,202) internal4,    nwfid, nxsize, nysize, unused8

      !              IFTYPE   IDEP     IZTYPE   UNUSED   IINST
      read(777,202) iftype, idep, iztype, unused9, iinst

      !              ISTREG   IEVREG   IEVTYP   IQUAL    ISYNTH
      read(777,202) istreg, ievreg, ievtyp, iqual, isynth

      !              IMAGTYP IMAGSRC UNUSED   UNUSED   UNUSED
      read(777,202) imagtyp, imagsrc, unused10, unused11, unused12

      !              UNUSED   UNUSED   UNUSED   UNUSED   UNUSED
      read(777,202) unused13, unused14, unused15, unused16, unused17

      !read logical header variables
      !-------------------------------------------------------
      !              LEVEN   LPSPOL   LOVROK LCALDA UNUSED
      read(777,202) leven, lpspol, lovrok, lcalda, unused18

      !read character header variables
      !-------------------------------------------------------
      !              KSTNM   KEVNM
      read(777,203) kstnm, kevnm

      !              KHOLE    KO       KA
      read(777,204) khole, ko, ka

      !              KT0      KT1      KT2
      read(777,204) kt0, kt1, kt2

      !              KT3      KT4      KT5
      read(777,204) kt3, kt4, kt5

      !              KT6      KT7      KT8
      read(777,204) kt6, kt7, kt8

      !              KT9      KF       KUSER0
      read(777,204) kt9, kf, kuser0

      !              KUSER1   KUSER2  KCMPNM
      read(777,204) kuser1, kuser2, kcmpnm

      !              KNETWK   KDATRD   KINST
      read(777,204) knetwk, kdatrd, kinst


      !read data
      !-------------------------------------------------------
```

```fortran
      ALLOCATE (yarray(npts))

      read(777,205) yarray

    CLOSE(777)

    201 FORMAT(G15.7,G15.7,G15.7,G15.7,G15.7)
    202 FORMAT(I10,I10,I10,I10,I10)
    203 FORMAT(A8,A16)
    204 FORMAT(A8,A8,A8)
    205 FORMAT(G15.7,G15.7,G15.7,G15.7,G15.7)

  END SUBROUTINE rasac

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SUBROUTINE wbsac(outfile,delta,depmin,depmax,scale,odelta,b,e,o,a,internal1,&
t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,    &
resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,    &
user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,    &
unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,   &
nzsec,nzmsec,nvhdr,norid,nevid,npts,internal4,nwfid,nxsize,nysize,unused8,    &
iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,   &
imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,       &
unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1,&
kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd,&
kinst,yarray)

! Write out sac binary format file with ALL the header information
! ================================
! Header values are as described on
! the sac homepage http://www.llnl.gov/sac/

! Note: With NAGWare Fortran 95 Compiler
! must compile with -kind=byte;
! has not been tested on non-Macintosh/Unix machines

   IMPLICIT NONE

    ! Real header variables
   INTEGER, PARAMETER :: k=4_4   !32 bit (4 byte) words
   REAL(k), INTENT(IN):: delta,depmin,depmax,scale,odelta
   REAL(k), INTENT(IN):: b,e,o,a,internal1
   REAL(k), INTENT(IN):: t0,t1,t2,t3,t4
   REAL(k), INTENT(IN):: t5,t6,t7,t8,t9
   REAL(k), INTENT(IN):: f,resp0,resp1,resp2,resp3
   REAL(k), INTENT(IN):: resp4,resp5,resp6,resp7,resp8
   REAL(k), INTENT(IN):: resp9,stla,stlo,stel,stdp
   REAL(k), INTENT(IN):: evla,evlo,evel,evdp,mag
   REAL(k), INTENT(IN):: user0,user1,user2,user3,user4
   REAL(k), INTENT(IN):: user5,user6,user7,user8,user9
   REAL(k), INTENT(IN):: dist,az,baz,gcarc,internal2
   REAL(k), INTENT(IN):: internal3,depmen,cmpaz,cmpinc,xminimum
   REAL(k), INTENT(IN):: xmaximum,yminimum,ymaximum,unused1,unused2
   REAL(k), INTENT(IN):: unused3,unused4,unused5,unused6,unused7

    ! integer header variables
   INTEGER(k), INTENT(IN):: nzyear,nzjday,nzhour,nzmin,nzsec
   INTEGER(k), INTENT(IN):: nzmsec,nvhdr,norid,nevid,npts
   INTEGER(k), INTENT(IN):: internal4,nwfid,nxsize,nysize,unused8
   INTEGER(k), INTENT(IN):: iftype,idep,iztype,unused9,iinst
   INTEGER(k), INTENT(IN):: istreg,ievreg,ievtyp,iqual,isynth
   INTEGER(k), INTENT(IN):: imagtyp,imagsrc,unused10,unused11,unused12
```

```
692     INTEGER(k), INTENT(IN):: unused13,unused14,unused15,unused16,unused17
693
694     ! logical header variables (1=true 0=false)
695     INTEGER(k), INTENT(IN):: leven,lpspol,lovrok,lcalda,unused18
696
697     ! character header values
698     CHARACTER(LEN=100), INTENT(IN) :: outfile
699     CHARACTER(LEN=16), INTENT(IN):: kevnm
700     CHARACTER(LEN=8), INTENT(IN):: kstnm
701     CHARACTER(LEN=8), INTENT(IN):: khole,ko,ka
702     CHARACTER(LEN=8), INTENT(IN):: kt0,kt1,kt2
703     CHARACTER(LEN=8), INTENT(IN):: kt3,kt4,kt5
704     CHARACTER(LEN=8), INTENT(IN):: kt6,kt7,kt8
705     CHARACTER(LEN=8), INTENT(IN):: kt9,kf,kuser0
706     CHARACTER(LEN=8), INTENT(IN):: kuser1,kuser2,kcmpnm
707     CHARACTER(LEN=8), INTENT(IN):: knetwk,kdatrd,kinst
708     ! Data array
709     REAL(k),INTENT(IN) :: yarray(npts)
710
711     ! Internal variables
712     INTEGER(k) :: io,j,N,tot
713
714         ! Write out float and integer data (4 units/variable)
715         OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=20)
716
717         ! Position       Header Variables
718         ! 1-20          DELTA DEPMIN DEPMAX SCALE ODELTA
719         write(20,REC=1) delta, depmin, depmax, scale, odelta
720
721         ! 21-40         B  E  O  A        INTERNAL
722         write(20,REC=2) b, e, o, a, internal1
723
724         ! 41-60              T0       T1       T2       T3       T4
725         write(20,REC=3) t0, t1, t2, t3, t4
726
727         ! 61-80              T5       T6       T7       T8       T9
728         write(20,REC=4) t5, t6, t7, t8, t9
729
730         ! 81-100              F        RESP0    RESP1    RESP2    RESP3
731         write(20,REC=5) f, resp0, resp1, resp2, resp3
732
733         ! 101-120               RESP4    RESP5    RESP6    RESP7    RESP8
734         write(20,REC=6) resp4, resp5, resp6, resp7, resp8
735
736         ! 121-140               RESP9    STLA   STLO   STEL   STDP
737         write(20,REC=7) resp9, stla, stlo, stel, stdp
738
739         ! 141-160               EVLA   EVLO   EVEL     EVDP   MAG
740         write(20,REC=8) evla, evlo, evel, evdp, mag
741
742         ! 161-180               USER0    USER1    USER2    USER3    USER4
743         write(20,REC=9) user0, user1, user2, user3, user4
744
745         ! 181-200               USER5    USER6    USER7    USER8    USER9
746         write(20,REC=10) user5, user6, user7, user8, user9
747
748         ! 201-220               DIST     AZ       BAZ      GCARC    INTERNAL
749         write(20,REC=11) dist, az, baz, gcarc, internal2
750
751         ! 221-240               INTERNAL DEPMEN   CMPAZ   CMPINC   XMINIMUM
752         write(20,REC=12) internal3,  depmen, cmpaz, cmpinc, xminimum
753
754         ! 241-260               XMAXIMUM   YMINIMUM   YMAXIMUM   UNUSED   UNUSED
755         write(20,REC=13) xmaximum,    yminimum,    ymaximum,    unused1, unused2
756
```

```
      ! 261-280              UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
      write(20,REC=14) unused3, unused4, unused5, unused6, unused7

      !write integer header variables
      !-------------------------------------------------------
      ! 281-300              NZYEAR  NZJDAY  NZHOUR  NZMIN   NZSEC
      write(20,REC=15) nzyear, nzjday, nzhour, nzmin, nzsec

      ! 301-320              NZMSEC  NVHDR   NORID   NEVID   NPTS
      write(20,REC=16) nzmsec, nvhdr, norid, nevid, npts

      ! 321-340              INTERNAL NWFID   NXSIZE  NYSIZE  UNUSED
      write(20,REC=17) internal4,    nwfid, nxsize, nysize, unused8

      ! 341-360              IFTYPE  IDEP    IZTYPE  UNUSED  IINST
      write(20,REC=18) iftype, idep, iztype, unused9, iinst

      ! 361-380              ISTREG  IEVREG  IEVTYP  IQUAL   ISYNTH
      write(20,REC=19) istreg, ievreg, ievtyp, iqual, isynth

      ! 381-400              IMAGTYP IMAGSRC UNUSED  UNUSED  UNUSED
      write(20,REC=20) imagtyp, imagsrc, unused10, unused11, unused12

      ! 401-420              UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
      write(20,REC=21) unused13, unused14, unused15, unused16, unused17

      !write logical header variables
      !-------------------------------------------------------
      ! 421-440              LEVEN   LPSPOL  LOVROK LCALDA UNUSED
      write(20,REC=22) leven, lpspol, lovrok, lcalda, unused18
      CLOSE(20)

      ! Character variable (size=8)*    *kevnm=16
      ! Here it gets tricky; we have to write out each k-header separately
      ! due to their different size... so I deviate from a straight up
      ! copy of the alpha numeric format, and do it my own way

      !write character header variables
      !-------------------------------------------------------
      ! 440-448              KSTNM
      OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=8)
      write(20,REC=56) kstnm
      CLOSE(20)
      ! 449-464              KEVNM
      OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=16)
      write(20,REC=29) kevnm

      ! 465-480              KHOLE   KO
      write(20,REC=30) khole, ko
      CLOSE(20)

      ! 480-504              KA    KT0     KT1
      OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=24)
      write(20,REC=21) ka, kt0, kt1

      ! 504-528              KT2     KT3     KT4
      write(20,REC=22) kt2, kt3, kt4

      ! 529-552               KT5     KT6     KT7
      write(20,REC=23) kt5, kt6, kt7

      ! 553-576              KT8   KT9   KF
      write(20,REC=24) kt8, kt9, kf

      ! 577-600              KUSER0 KUSER1  KUSER2
```

```
822         write(20,REC=25) kuser0, kuser1, kuser2
823
824         ! 601-624                KCMPNM KNETWK  KDATRD
825         write(20,REC=26) kcmpnm, knetwk, kdatrd
826         CLOSE(20)
827
828         ! 625-632                KINST
829         OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=8)
830         write(20,REC=79) kinst
831         CLOSE(20)
832
833         ! write data
834         ! 633 ->  npts*4
835         !-------------------------------------------------------
836
837 !       OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=4)
838 !
839 !       DO j=159_4,(159_4+npts)
840 !
841 !          WRITE(20,REC=j,iostat=io) yarray(j-158_4)
842 !          IF (io /= 0) EXIT
843 !
844 !       END DO
845 !
846 !       CLOSE(20)
847
848         ! Efficiency counts; I've worked out a way of reading in multiple
849         ! binary values at once
850
851         ! 1) If there are less than 158 points, then a simple do loop is
852         ! best
853         ! 2) If there are more than 158 points, you want to read in blocks
854         ! of 158 pts each (Direct access requires you keep track of where
855         ! you are in the file; multiples of 158 are good as there are 158
856         ! header variables, so you count them as record 1).
857         ! However, you also need to read in the rest of the points as it
858         ! is unlikely files will always have a multiple of 158 for npts
859
860         ! Case 1: less than 158 pts in the sacfile
861         IF (npts < 158_4) THEN
862
863           OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=4)
864           DO j=159_4,(159_4+npts)
865
866              WRITE(20,REC=j,iostat=io) yarray(j-158_4)
867              IF (io /= 0) EXIT
868
869           END DO
870           CLOSE(20)
871
872         ! Case 2:
873         ELSE IF (npts >= 158_4) THEN
874
875           OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=632)
876
877           N=int(real(npts)/158_4)+1_4
878
879           ! Start reading in the large record blocks (158*4 blocks each)
880           DO j=2_4,N
881
882              WRITE(20,REC=j,iostat=io) yarray(158_4*(j-2_4)+1_4:158_4*(j-1_4))
883              IF (io /= 0) EXIT
884           END DO
885           CLOSE(20)
886
```

```fortran
        tot=npts-158_4*int(real(npts)/158_4)

        ! Case 3: Read in the rest
        OPEN(UNIT=20,FILE=outfile,FORM='unformatted',ACCESS='DIRECT',RECL=4)
        DO j=1,tot
           WRITE(20,REC=((158_4*N)+j),iostat=io) yarray(158_4*(N-1_4)+j)
!            WRITE(20,REC=((158_4*N)+j),iostat=io) yarray(158_4*(N-1)+j:npts)
           IF (io /= 0) EXIT
        END DO

      END IF

  END SUBROUTINE wbsac

!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
!%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

  SUBROUTINE wasac(outfile,delta,depmin,depmax,scale,odelta,b,e,o,a,internal1,&
t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6,     &
resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1,     &
user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2,  &
internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1,     &
unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin,    &
nzsec,nzmsec,nvhdr,norid,nevid,npts,internal4,nwfid,nxsize,nysize,unused8,     &
iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp,    &
imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16,        &
unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1, &
kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd, &
kinst,yarray)

! Write out sac alphanumeric format file with ALL the header information
! ==================================
! Header values are as described on
! the sac homepage http://www.llnl.gov/sac/

! Note: With NAGWare Fortran 95 Compiler
! must compile with -kind=byte;
! has not been tested on non-Macintosh/Unix machines

    IMPLICIT NONE

    ! Real header variables
    REAL, INTENT(IN):: delta,depmin,depmax,scale,odelta
    REAL, INTENT(IN):: b,e,o,a,internal1
    REAL, INTENT(IN):: t0,t1,t2,t3,t4
    REAL, INTENT(IN):: t5,t6,t7,t8,t9
    REAL, INTENT(IN):: f,resp0,resp1,resp2,resp3
    REAL, INTENT(IN):: resp4,resp5,resp6,resp7,resp8
    REAL, INTENT(IN):: resp9,stla,stlo,stel,stdp
    REAL, INTENT(IN):: evla,evlo,evel,evdp,mag
    REAL, INTENT(IN):: user0,user1,user2,user3,user4
    REAL, INTENT(IN):: user5,user6,user7,user8,user9
    REAL, INTENT(IN):: dist,az,baz,gcarc,internal2
    REAL, INTENT(IN):: internal3,depmen,cmpaz,cmpinc,xminimum
    REAL, INTENT(IN):: xmaximum,yminimum,ymaximum,unused1,unused2
    REAL, INTENT(IN):: unused3,unused4,unused5,unused6,unused7

    ! integer header variables
    INTEGER, INTENT(IN):: nzyear,nzjday,nzhour,nzmin,nzsec
    INTEGER, INTENT(IN):: nzmsec,nvhdr,norid,nevid,npts
    INTEGER, INTENT(IN):: internal4,nwfid,nxsize,nysize,unused8
    INTEGER, INTENT(IN):: iftype,idep,iztype,unused9,iinst
    INTEGER, INTENT(IN):: istreg,ievreg,ievtyp,iqual,isynth
    INTEGER, INTENT(IN):: imagtyp,imagsrc,unused10,unused11,unused12
    INTEGER, INTENT(IN):: unused13,unused14,unused15,unused16,unused17
```

```
952
953     ! logical header variables (1=true 0=false)
954     INTEGER, INTENT(IN):: leven,lpspol,lovrok,lcalda,unused18
955
956     ! character header values
957     CHARACTER(LEN=90), INTENT(IN) :: outfile
958     CHARACTER(LEN=16), INTENT(IN):: kevnm
959     CHARACTER(LEN=8), INTENT(IN):: kstnm
960     CHARACTER(LEN=8), INTENT(IN):: khole,ko,ka
961     CHARACTER(LEN=8), INTENT(IN):: kt0,kt1,kt2
962     CHARACTER(LEN=8), INTENT(IN):: kt3,kt4,kt5
963     CHARACTER(LEN=8), INTENT(IN):: kt6,kt7,kt8
964     CHARACTER(LEN=8), INTENT(IN):: kt9,kf,kuser0
965     CHARACTER(LEN=8), INTENT(IN):: kuser1,kuser2,kcmpnm
966     CHARACTER(LEN=8), INTENT(IN):: knetwk,kdatrd,kinst
967
968     ! Data array
969     REAL, INTENT(IN) :: yarray(npts)
970
971     ! Open the new sac file
972     OPEN(UNIT=777,FILE=outfile,status='replace')
973
974         !read in real header variables
975         !----------------------------------------------------
976         !               DELTA   DEPMIN   DEPMAX   SCALE    ODELTA
977         write(777,201) delta, depmin, depmax, scale, odelta
978
979         !               B   E   O   A        INTERNAL
980         write(777,201) b, e, o, a, internal1
981
982         !               T0       T1       T2       T3       T4
983         write(777,201) t0, t1, t2, t3, t4
984
985         !               T5       T6       T7       T8       T9
986         write(777,201) t5, t6, t7, t8, t9
987
988         !               F        RESP0    RESP1    RESP2    RESP3
989         write(777,201) f, resp0, resp1, resp2, resp3
990
991         !               RESP4    RESP5    RESP6    RESP7    RESP8
992         write(777,201) resp4, resp5, resp6, resp7, resp8
993
994         !               RESP9    STLA   STLO   STEL   STDP
995         write(777,201) resp9, stla, stlo, stel, stdp
996
997         !               EVLA   EVLO   EVEL     EVDP   MAG
998         write(777,201) evla, evlo, evel, evdp, mag
999
1000        !               USER0    USER1    USER2    USER3    USER4
1001        write(777,201) user0, user1, user2, user3, user4
1002
1003        !               USER5    USER6    USER7    USER8    USER9
1004        write(777,201) user5, user6, user7, user8, user9
1005
1006        !               DIST     AZ       BAZ      GCARC    INTERNAL
1007        write(777,201) dist, az, baz, gcarc, internal2
1008
1009        !               INTERNAL DEPMEN   CMPAZ   CMPINC   XMINIMUM
1010        write(777,201) internal3,  depmen, cmpaz, cmpinc, xminimum
1011
1012        !               XMAXIMUM   YMINIMUM   YMAXIMUM   UNUSED   UNUSED
1013        write(777,201) xmaximum,   yminimum,   ymaximum,   unused1, unused2
1014
1015        !               UNUSED   UNUSED   UNUSED   UNUSED   UNUSED
1016        write(777,201) unused3, unused4, unused5, unused6, unused7
```

```
1017
1018
1019        !write integer header variables
1020        !--------------------------------------------------------
1021        !               NZYEAR  NZJDAY  NZHOUR  NZMIN   NZSEC
1022        write(777,202) nzyear, nzjday, nzhour, nzmin, nzsec
1023
1024        !               NZMSEC  NVHDR   NORID   NEVID   NPTS
1025        write(777,202) nzmsec, nvhdr, norid, nevid, npts
1026
1027        !               INTERNAL NWFID   NXSIZE  NYSIZE  UNUSED
1028        write(777,202) internal4,   nwfid, nxsize, nysize, unused8
1029
1030        !               IFTYPE  IDEP    IZTYPE  UNUSED  IINST
1031        write(777,202) iftype, idep, iztype, unused9, iinst
1032
1033        !               ISTREG  IEVREG  IEVTYP  IQUAL   ISYNTH
1034        write(777,202) istreg, ievreg, ievtyp, iqual, isynth
1035
1036        !               IMAGTYP IMAGSRC UNUSED  UNUSED  UNUSED
1037        write(777,202) imagtyp, imagsrc, unused10, unused11, unused12
1038
1039        !               UNUSED  UNUSED  UNUSED  UNUSED  UNUSED
1040        write(777,202) unused13, unused14, unused15, unused16, unused17
1041
1042        !write logical header variables
1043        !--------------------------------------------------------
1044        !               LEVEN   LPSPOL  LOVROK LCALDA UNUSED
1045        write(777,202) leven, lpspol, lovrok, lcalda, unused18
1046
1047        !write character header variables
1048        !--------------------------------------------------------
1049        !               KSTNM   KEVNM
1050        write(777,203) kstnm, kevnm
1051
1052        !               KHOLE   KO      KA
1053        write(777,204) khole, ko, ka
1054
1055        !               KT0     KT1     KT2
1056        write(777,204) kt0, kt1, kt2
1057
1058        !               KT3     KT4     KT5
1059        write(777,204) kt3, kt4, kt5
1060
1061        !               KT6     KT7     KT8
1062        write(777,204) kt6, kt7, kt8
1063
1064        !               KT9     KF      KUSER0
1065        write(777,204) kt9, kf, kuser0
1066
1067        !               KUSER1  KUSER2  KCMPNM
1068        write(777,204) kuser1, kuser2, kcmpnm
1069
1070        !              KNETWK  KDATRD  KINST
1071        write(777,204) knetwk, kdatrd, kinst
1072
1073
1074        !write data
1075        !--------------------------------------------------------
1076
1077        write(777,205) yarray
1078
1079      CLOSE(777)
1080
1081      201 FORMAT(G15.7,G15.7,G15.7,G15.7,G15.7)
```

```
    202 FORMAT(I10,I10,I10,I10,I10)
    203 FORMAT(A8,A16)
    204 FORMAT(A8,A8,A8)
    205 FORMAT(G15.7,G15.7,G15.7,G15.7,G15.7)

END SUBROUTINE wasac

SUBROUTINE initsac(outfile,delta,depmin,depmax,scale,odelta,b,e,o,a,internal1, &
   t0,t1,t2,t3,t4,t5,t6,t7,t8,t9,f,resp0,resp1,resp2,resp3,resp4,resp5,resp6, &
   resp7,resp8,resp9,stla,stlo,stel,stdp,evla,evlo,evel,evdp,mag,user0,user1, &
   user2,user3,user4,user5,user6,user7,user8,user9,dist,az,baz,gcarc,internal2, &
   internal3,depmen,cmpaz,cmpinc,xminimum,xmaximum,yminimum,ymaximum,unused1, &
   unused2,unused3,unused4,unused5,unused6,unused7,nzyear,nzjday,nzhour,nzmin, &
   nzsec,nzmsec,nvhdr,norid,nevid,npts,internal4,nwfid,nxsize,nysize,unused8, &
   iftype,idep,iztype,unused9,iinst,istreg,ievreg,ievtyp,iqual,isynth,imagtyp, &
   imagsrc,unused10,unused11,unused12,unused13,unused14,unused15,unused16, &
   unused17,leven,lpspol,lovrok,lcalda,unused18,kevnm,kstnm,khole,ko,ka,kt0,kt1, &
   kt2,kt3,kt4,kt5,kt6,kt7,kt8,kt9,kf,kuser0,kuser1,kuser2,kcmpnm,knetwk,kdatrd, &
   kinst,yarray)

 ! Real header variables
 INTEGER, PARAMETER :: k=4_4  !32 bit (4 byte) words
 REAL(k), INTENT(OUT) :: delta,depmin,depmax,scale,odelta
 REAL(k), INTENT(OUT) :: b,e,o,a,internal1
 REAL(k), INTENT(OUT) :: t0,t1,t2,t3,t4
 REAL(k), INTENT(OUT) :: t5,t6,t7,t8,t9
 REAL(k), INTENT(OUT) :: f,resp0,resp1,resp2,resp3
 REAL(k), INTENT(OUT) :: resp4,resp5,resp6,resp7,resp8
 REAL(k), INTENT(OUT) :: resp9,stla,stlo,stel,stdp
 REAL(k), INTENT(OUT) :: evla,evlo,evel,evdp,mag
 REAL(k), INTENT(OUT) :: user0,user1,user2,user3,user4
 REAL(k), INTENT(OUT) :: user5,user6,user7,user8,user9
 REAL(k), INTENT(OUT) :: dist,az,baz,gcarc,internal2
 REAL(k), INTENT(OUT) :: internal3,depmen,cmpaz,cmpinc,xminimum
 REAL(k), INTENT(OUT) :: xmaximum,yminimum,ymaximum,unused1,unused2
 REAL(k), INTENT(OUT) :: unused3,unused4,unused5,unused6,unused7
 REAL(k), PARAMETER   :: rundef=-12345.0

 ! Integer header variables
 INTEGER(k), INTENT(OUT) :: nzyear,nzjday,nzhour,nzmin,nzsec
 INTEGER(k), INTENT(OUT) :: nzmsec,nvhdr,norid,nevid,npts
 INTEGER(k), INTENT(OUT) :: internal4,nwfid,nxsize,nysize,unused8
 INTEGER(k), INTENT(OUT) :: iftype,idep,iztype,unused9,iinst
 INTEGER(k), INTENT(OUT) :: istreg,ievreg,ievtyp,iqual,isynth
 INTEGER(k), INTENT(OUT) :: imagtyp,imagsrc,unused10,unused11,unused12
 INTEGER(k), INTENT(OUT) :: unused13,unused14,unused15,unused16,unused17
 INTEGER(k), PARAMETER   :: iundef=-12345

 ! Logical header variables (1=true 0=false)
 INTEGER(k), INTENT(OUT) :: leven,lpspol,lovrok,lcalda,unused18

 ! Character header values
 CHARACTER(LEN=100), INTENT(OUT) :: outfile
 CHARACTER(LEN=16), INTENT(OUT) :: kevnm
 CHARACTER(LEN=8), INTENT(OUT) :: kstnm
 CHARACTER(LEN=8), INTENT(OUT) :: khole,ko,ka
 CHARACTER(LEN=8), INTENT(OUT) :: kt0,kt1,kt2
 CHARACTER(LEN=8), INTENT(OUT) :: kt3,kt4,kt5
 CHARACTER(LEN=8), INTENT(OUT) :: kt6,kt7,kt8
 CHARACTER(LEN=8), INTENT(OUT) :: kt9,kf,kuser0
 CHARACTER(LEN=8), INTENT(OUT) :: kuser1,kuser2,kcmpnm
 CHARACTER(LEN=8), INTENT(OUT) :: knetwk,kdatrd,kinst
 CHARACTER(LEN=8)              :: kundef='-12345'
 ! Data array
 REAL(k), ALLOCATABLE,INTENT(OUT) :: yarray(:)
```

```
1147
1148    outfile = 'initialized.sac'
1149
1150    delta=rundef; depmin=rundef; depmax=rundef; scale=rundef; odelta=rundef;
1151    b=rundef; e=rundef; o=rundef; a=rundef; internal1=rundef; t0=rundef;
1152    t1=rundef; t2=rundef; t3=rundef; t4=rundef; t5=rundef; t6=rundef;
1153    t7=rundef; t8=rundef; t9=rundef; f=rundef; resp0=rundef; resp1=rundef;
1154    resp2=rundef; resp3=rundef; resp4=rundef; resp5=rundef; resp6=rundef;
1155    resp7=rundef; resp8=rundef; resp9=rundef; stla=rundef; stlo=rundef;
1156    stel=rundef; stdp=rundef; evla=rundef; evlo=rundef; evel=rundef;
1157    evdp=rundef; mag=rundef; user0=rundef; user1=rundef; user2=rundef;
1158    user3=rundef; user4=rundef; user5=rundef; user6=rundef; user7=rundef;
1159    user8=rundef; user9=rundef; dist=rundef; az=rundef; baz=rundef;
1160    gcarc=rundef; internal2=rundef; internal3=rundef; depmen=rundef;
1161    cmpaz=rundef; cmpinc=rundef; xminimum=rundef; xmaximum=rundef;
1162    yminimum=rundef; ymaximum=rundef; unused1=rundef; unused2=rundef;
1163    unused3=rundef; unused4=rundef; unused5=rundef; unused6=rundef;
1164    unused7=rundef;
1165
1166    nzyear=iundef; nzjday=iundef; nzhour=iundef; nzmin=iundef;
1167    nzsec=iundef; nzmsec = iundef;
1168    nvhdr=6; norid=iundef; nevid=iundef; npts=iundef; internal4=iundef;
1169    nwfid=iundef; nxsize=iundef; nysize=iundef; unused8=iundef
1170    iftype=1; idep=iundef; iztype=iundef; unused9=iundef; iinst=iundef;
1171    istreg=iundef; ievreg=iundef; ievtyp=iundef; iqual=iundef; isynth=iundef;
1172    imagtyp=iundef; imagsrc=iundef; unused10=iundef; unused11=iundef;
1173    unused12=iundef; unused13=iundef; unused14=iundef; unused15=iundef;
1174    unused16=iundef; unused17=iundef;
1175
1176    leven=1; lpspol=0; lovrok=1; lcalda=1; unused18=0;
1177
1178    kevnm=kundef; kstnm=kundef; khole=kundef; ko=kundef;
1179    ka=kundef; kt0=kundef; kt1=kundef; kt2=kundef; kt3=kundef;
1180    kt4=kundef; kt5=kundef; kt6=kundef; kt7=kundef; kt8=kundef
1181    kt9=kundef; kf=kundef; kuser0=kundef; kuser1=kundef; kuser2=kundef
1182    kcmpnm=kundef; knetwk=kundef; kdatrd=kundef; kinst=kundef
1183
1184    ALLOCATE(yarray(1))
1185    yarray = 0.0;
1186
1187
1188  END SUBROUTINE initsac
1189
1190 END MODULE sac_i_o
```


2.18 SRC/MOD/McMC.f90

```
1  ! Librairie de subroutine concernant le McMC : algorithme de Métropolis-Hastings
2  ! septembre 2013
3  ! ******************************************************************     .
4  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
5  ! ******************************************************************     .
6  ! ------------------------------------------------------------------     .
7
8  MODULE algo_metropolis
9
10     use modparam
11     use typetemps, only : parametresinv,parametres,fcout,accept
12     use mt19937, only : genrand_real1
13
14     implicit none
15
16     private
17
18     public  :: metropolis
```

```
19
20
21 CONTAINS
22
23 ! ------------------------------------------------------------------     .
24
25   subroutine metropolis(p,param_best,misfit,acceptance,mod_save,newkeep)
26     ! -------                                                 --------     .mh
27     ! permet l'acceptation et le rejet des modèles selon un critère d'acceptation
28     ! -------                                                 --------     .
29     implicit none
30     type(parametresinv),intent(inout) :: p
31     type(parametres),intent(inout) :: param_best
32     type(fcout),intent(inout) :: misfit
33     type(accept),intent(inout) :: acceptance
34     logical,intent(in) :: mod_save
35     logical, intent(out) :: newkeep
36     ! -------                                                 --------     .
37     real(KIND=wr) :: aleatoire,h,delta
38     ! ------------------------------------------------------------------     .
39     h=-700._wr                                                             ! exp(h)< 10^307 (_wr)
40     ! -------                                                 --------     .
41     newkeep = .true.
42     if (misfit%new.gt.misfit%old) then                                     ! pire
43       aleatoire = genrand_real1()
44       delta=max(misfit%old-misfit%new,h)                                   ! pb exec sinon car déjà : exp(-700)=9.9e-305
45       if (exp(delta).le.aleatoire) then                                    ! repêchage ?
46         newkeep = .false.                                                  ! repêchage : non
47         misfit%new = misfit%old
48         p%valNew=p%valOld
49         acceptance%N = acceptance%N+int(1,wl)
50       else
51         acceptance%NO = acceptance%NO+int(1,wl)                            ! repêchage : oui
52       endif
53     else
54       acceptance%O = acceptance%O+int(1,wl)                                ! meilleur
55     endif
56     !------- if newkeep = .true. -> new parameters           --------     .
57     if (newkeep) then
58       p%valOld = p%valNew
59       misfit%old = misfit%new
60     end if
61     !------- the lowest misfit is updated                    --------     .
62     if ((misfit%new.le.misfit%best).and.(mod_save)) then
63       misfit%best=misfit%new
64       param_best=p%valNew
65     end if
66     ! ------------------------------------------------------------------     .
67   end subroutine metropolis
68
69 END MODULE algo_metropolis
70
71
72
73 ! ********************************************************************     .
74 ! ********************************************************************     .
```

2.19 SRC/MOD/avancement.f90

```
1 ! module indiquant l'avancement d'une boucle avec une barre de progression
2 ! septembre 2013
3 ! ********************************************************************     .
4 ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
5 ! ********************************************************************     .
6 ! ------------------------------------------------------------------     .
```

```
7
8  MODULE cpt_temps
9
10     use modparam
11
12     implicit none
13
14     private
15
16     public  :: progress
17
18
19 CONTAINS
20
21 ! ------------------------------------------------------------------     .
22
23   subroutine progress(n,ntotal,one_string)
24     ! -------                                                 --------     .mh
25     implicit none
26     integer(KIND=wi), intent(in) :: n                                    ! itération en cours
27     integer(KIND=wi), intent(in) :: ntotal                               ! itérations totales
28     character(LEN=*), intent(in) :: one_string                           ! chaîne à afficher avant la barre [##### 100 % #####]
29     ! -------                                                 --------     .
30     character(LEN=255) :: prog,oldprog
31     real(KIND=wr) :: tl
32     integer(KIND=wi) :: i,oldtime,newtime,ratetime,cptmax
33     ! -------                                                 --------     .
34     save :: oldprog                                                      ! garde en mémoire l'ancienne chaîne et le temps initial
35     save :: oldtime
36     ! ------------------------------------------------------------------     .
37     if (n.eq.1) then
38       write(oldprog,'(a255)')"x"                                         ! chaîne vide, première itération
39       call system_clock(oldtime)                                         ! temps à la première itération
40     endif
41     call system_clock(newtime,ratetime,cptmax)                           ! temps à l'itération courante
42     ! -------                                                 --------     . différence de temps
43     tl = real(newtime-oldtime,wr)
44     if(tl.lt.0.0_wr) tl = real(newtime-oldtime+cptmax,wr)
45     tl = tl/real(ratetime,wr)
46     ! -------                                                 --------     .
47     if ((n.gt.0).and.(n.lt.ntotal)) then
48       tl=(1.0_wr*real(ntotal,wr)/real(n,wr))*tl-tl                       ! temps restant en sec
49     else
50       tl=tl                                                              ! temps total en sec
51     endif
52     if (tl.le.0.0_wr) tl=0.0_wr                                          ! première(s) itération(s)
53     ! -------                                                 --------     .
54     write(prog,'(a30,1x,''['')')one_string//"    "                      ! construction de la chaîne de caractère
55     do i=1,40
56       prog(32+i:32+i)=' '
57     enddo
58     write(prog(48:56),'(f7.1,''%'')') 100.0_wr*real(n,wr)/real(ntotal,wr)  ! pourcentage restant
59     do i=1,40
60       if ((1.0_wr*real(n,wr)/real(ntotal,wr)).gt.(1.0_wr*real(i,wr)/40.0_wr)) then
61         if (prog(32+i:32+i).eq.' ') prog(32+i:32+i)='#'                  ! barre de progression ##
62       endif
63     enddo
64     prog(72:72)=']'
65     write(prog(75:77),'(i2.2,'':'')')int(tl/3600.0_wr)                   ! temps restant en heure
66
67     write(prog(78:80),'(i2.2,'':'')')int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi) ! temps restant en min
68
69     write(prog(81:82),'(i2.2)')int((tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr),wi) ! temps restant en sec
70     write(prog(83:86),'(a4)')" sec"
71     if (prog.ne.oldprog) write(*,'(a,a,$)') prog(1:86),char(13)          ! permet de rester sur la ligne courante et n'écrit que si nécessaire
```

```
    oldprog=prog
    if (n.eq.ntotal) write(*,*)                                    ! fin,char(13)
    ! ------------------------------------------------------------  .
  end subroutine progress

END MODULE cpt_temps



! ********************************************************************  .
! ********************************************************************  .
```

2.20 SRC/MOD/dellipsgc.f90

```
! novembre 2013
! calcul la distance épicentrale et le back-azimuth (en degrés)
! ********************************************************************  .mh
! -------- Eric Beucler eric.beucler@univ-nantes.fr (12.07.2005) -------  .eb
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr ----------------  .mh
! ********************************************************************  .
! ------------------------------------------------------------------  .

MODULE distance_epi

    use modparam

    implicit none

    private

    public  :: dellipsgc


CONTAINS

! ------------------------------------------------------------------  .

  subroutine dellipsgc(dlat1,dlon1,dlat2,dlon2,d,bazfin)
    ! -------                                          ---------  .eb
    ! calcul plus complexe de la distance épicentrale et du back-azimuth
    ! -------                                          ---------  .mh
    implicit none
    real (kind=wr), intent(in) :: dlat1,dlat2,dlon1,dlon2          ! coordonnées des points [en degrés]
    real (kind=wr), intent(out), optional :: bazfin                ! back-azimuth [en degrés :0;360]
    real (kind=wr), intent(out) :: d                               ! distance épicentrale [en kilomètres]
    ! -------                                          ---------  .
    real (kind=wr) :: val, a, f, az, baz
    real (kind=wr) :: alph0,alpha,b,bb,d2r,delta,dphi,eps,i1,i2,m
    real (kind=wr) :: nx,ny,nz,p,rlat1,rlat2,rlatn,rlon1,rlon2
    ! ------------------------------------------------------------  .
    val=abs(dlat1-dlat2)+abs(dlon1-dlon2)
    if(val.gt.(real(2,wr)*spacing(dlat1))) then                    ! pb si distance épicentre => zéro
    ! ------------------------------------------------------------  .
      az = 0.0_wr
      baz = 0.0_wr
      d = 0.0_wr
      eps = real(2,wr)*spacing(dlat1)
      d2r = pi/180.0_wr
      if ((abs(dlat1-dlat2) < eps).and.&
           ((abs(dlon1-dlon2) < eps).or.(abs(dlon1-dlon2) == 360.0_wr))) then
        write(*,*)'problème dans dellipsgc : mauvaise coordonnées'
      endif
      a = rT * 1000.0_wr
      f = 0.0_wr                                                   ! sphère sans aplatissement au pôles
      b = a*(1._wr-f)
```

```
      ! ------------------------------------------------------------------
      ! Conversion to radians and verification of non nullity:
      ! Latitude  1:
      rlat1 = dlat1*d2r
      if (abs(rlat1) < eps) then
        if (rlat1 < 0.0_wr) rlat1 = -1.0_wr*eps
        if (rlat1 >= 0.0_wr) rlat1 = eps
      end if
      ! Longitude 1:
      if (dlon1 > (180.0_wr+eps)) then
        rlon1 = (dlon1-360.0_wr)*d2r
      else if (dlon1 < (-180.0_wr-eps)) then
        rlon1 = (dlon1+360.0_wr)*d2r
      else
        rlon1 = dlon1*d2r
      end if
      if (abs(rlon1) < eps) then
        if (rlon1 < 0.0_wr) rlon1 = -1.0_wr*eps
        if (rlon1 >= 0.0_wr) rlon1 = eps
      end if
      if ((dlat1 == 90.0_wr).or.(dlat1 == -90.0_wr)) rlon1 = eps
      ! Latitude  2:
      rlat2 = dlat2*d2r
      if (abs(rlat2) < eps) then
        if (rlat2 < 0.0_wr) rlat2 = -1.0_wr*eps
        if (rlat2 >= 0.0_wr) rlat2 = eps
      end if
      ! Longitude 1:
      if (dlon2 > (180.0_wr+eps)) then
        rlon2 = (dlon2-360.0_wr)*d2r
      else if (dlon2 < (-180.0_wr-eps)) then
        rlon2 = (dlon2+360.0_wr)*d2r
      else
        rlon2 = dlon2*d2r
      end if
      if (abs(rlon2) < eps) then
        if (rlon2 < 0.0_wr) rlon2 = -1.0_wr*eps
        if (rlon2 >= 0.0_wr) rlon2 = eps
      end if
      if ((dlat2 == 90.0_wr).or.(dlat2 == -90.0_wr)) rlon2 = eps
      dphi = rlon2-rlon1
      ! ------------------------------------------------------------------
      ! Computes path angle (identical to the spherical case):
      if (((cos(rlat1)*cos(rlat2)*cos(rlon1-rlon2))+ &
           (sin(rlat1)*sin(rlat2))) >= 1.0_wr) then
        delta = eps
      elseif (((cos(rlat1)*cos(rlat2)*cos(rlon1-rlon2))+ &
           (sin(rlat1)*sin(rlat2))) <= -1.0_wr) then
        delta = pi-eps
      else
        delta = acos((cos(rlat1)*cos(rlat2)*cos(rlon1-rlon2))+ &
           (sin(rlat1)*sin(rlat2)))
      end if
      ! ------------------------------------------------------------------
      ! Computes azimuth (angle between north and path 1->2):
      if (((sin(dphi)*cos(rlat2))/sin(delta)) > 1.0_wr) then
        az = pi/2.0_wr
      elseif (((sin(dphi)*cos(rlat2))/sin(delta)) < -1.0_wr) then
        az = -1.0_wr*pi/2.0_wr
      else
        az = asin((sin(dphi)*cos(rlat2))/sin(delta))
        if (((sin(rlat2)-(sin(rlat1)*cos(delta)))/&
          (cos(rlat1)*sin(delta))) < 0.0_wr) az = pi-az
      end if
      if ((abs(dphi) < eps).and.(rlat1 > rlat2)) az = 180.0_wr*d2r
```

```
      if (abs(az) < eps) then
        az = 0.0_wr
      elseif (az >= (2.0_wr*pi)) then
        az = (2.0_wr*pi)-az
      elseif (az < 0.0_wr) then
        az = (2.0_wr*pi)+az
      end if
      ! ------------------------------------------------------------------ .
      ! Computes back-azimuth (angle between north and path 2->1):
      dphi = -1.0_wr*dphi
      if (((sin(dphi)*cos(rlat1))/sin(delta)) > 1.0_wr) then
        baz = pi/2.0_wr
      elseif (((sin(dphi)*cos(rlat1))/sin(delta)) < -1.0_wr) then
        baz = -1.0_wr*pi/2.0_wr
      else
        baz = asin(sin(dphi)*cos(rlat1)/sin(delta))
        if (((sin(rlat1)-(sin(rlat2)*cos(delta)))/&
          (cos(rlat2)*sin(delta))) < 0.0_wr) baz = pi-baz
      end if
      if ((abs(dphi) < eps).and.(rlat2 > rlat1)) baz = 180.0_wr*d2r
      if (abs(baz) < eps) then
        baz = 0.0_wr
      elseif (baz >= (2.0_wr*pi)) then
        baz = (2.0_wr*pi)-baz
      elseif (baz < 0.0_wr) then
        baz = (2.0_wr*pi)+baz
      end if
      ! ------------------------------------------------------------------ .
      ! Computes the angle between great circle and the equatorial plane:
      nx = (cos(rlat1)*sin(rlon1)*sin(rlat2))- &
       (sin(rlat1)*cos(rlat2)*sin(rlon2))
      ny = (sin(rlat1)*cos(rlat2)*cos(rlon2))- &
       (cos(rlat1)*cos(rlon1)*sin(rlat2))
      nz = (cos(rlat1)*cos(rlon1)*cos(rlat2)*sin(rlon2))- &
       (cos(rlat1)*sin(rlon1)*cos(rlat2)*cos(rlon2))
      if ((sqrt(nx**2+ny**2) < eps).and.(abs(nz) > eps)) then
        rlatn = pi/2.0_wr
      elseif ((sqrt(nx**2+ny**2) < eps).and.(abs(nz) < eps)) then
        rlatn = 0.0_wr
      else
        rlatn = atan(nz/sqrt(nx**2+ny**2))
      end if
      rlatn = (pi/2.0_wr)+rlatn
      if (rlatn > (pi/2.0_wr)) rlatn = pi-rlatn
      ! ------------------------------------------------------------------ .
      ! Computes the new length of the half axe:
      bb = (a*b)/sqrt(((b*cos(rlatn))**2)+((a*sin(rlatn))**2))
      if (bb < b) bb = b
      if (bb > a) bb = a
      ! ------------------------------------------------------------------ .
      ! Computes the initial angle in the great circle plane:
      if (abs(rlatn) < eps) then
        alph0 = 0.0_wr
      elseif (abs(rlatn-pi) < eps) then
        alph0 = pi
      else
        if (rlat1 <= rlat2) then
          if (abs(sin(rlat1)/sin(rlatn)) >= 1.0_wr) then
            if ((rlat1*rlatn) < 0.0_wr) then
              alph0 = -1.0_wr*(pi/2.0_wr)
            else
              alph0 = pi/2.0_wr
            end if
          else
            alph0 = asin(sin(rlat1)/sin(rlatn))
```

```
        end if
      else
        if (abs(sin(rlat2)/sin(rlatn)) >= 1.0_wr) then
          if ((rlat2*rlatn) < 0.0_wr) then
            alph0 = -1.0_wr*(pi/2.0_wr)
          else
            alph0 = pi/2.0_wr
          end if
        else
         alph0 = asin(sin(rlat2)/sin(rlatn))
        end if
      end if
    end if
    ! ------------------------------------------------------------------    .
    ! Computes the curvilign integral along the elliptic path:
    m = 1.0_wr-(a**2/bb**2)
    if (abs(m) < eps) m = 0.0_wr
    p = 2.0_wr*pi*sqrt(0.5_wr*(a**2+bb**2)) ! Perimeter of the ellipse.
    ! ------------------------------------------------------------------    .
    ! Second integral between 0 and alph0:
    i2 = 0.0_wr
    alpha = alph0
    if (abs(1.0_wr-sin(alpha)**2) < eps) then
      i2 = p/4.0_wr
      if (alpha < 0.0_wr) i2 = -i2
    else
      if (alpha > (pi/2.0_wr)) then
        i2 = p/2.0_wr
        alpha = pi-alpha
        i2 = i2-(a*sqrt(1.0_wr-sin(alpha)**2)*asin(sin(alpha))/&
             sqrt((1.0_wr-(m*sin(alpha)**2))*(1.0_wr-sin(alpha)**2)))
      else
        i2 = a*sqrt(1.0_wr-sin(alpha)**2)*asin(sin(alpha))/&
             sqrt((1.0_wr-(m*sin(alpha)**2))*(1.0_wr-sin(alpha)**2))
      end if
    end if
    ! ------------------------------------------------------------------    .
    ! First integral between 0 and (alph0+delta):
    i1 = 0.0_wr
    alpha = alph0+delta
    if (abs(1.0_wr-sin(alpha)**2) < eps) then
      i1 = p/4.0_wr
      if (alpha < 0.0_wr) i1 = -i1
    else
      if (alpha > (pi/2.0_wr)) then
        i1 = p/2.0_wr
        alpha = pi-alpha
        i1 = i1-(a*sqrt(1.0_wr-sin(alpha)**2)*asin(sin(alpha))/&
             sqrt((1.0_wr-(m*sin(alpha)**2))*(1.0_wr-sin(alpha)**2)))
      else
        i1 = a*sqrt(1.0_wr-sin(alpha)**2)*asin(sin(alpha))/&
             sqrt((1.0_wr-(m*sin(alpha)**2))*(1.0_wr-sin(alpha)**2))
      end if
    end if
    d = i1-i2
    ! ------------------------------------------------------------------    .
    ! Back to kilometers and degrees:
    d = d/1000.0_wr
    az = az/d2r
    baz = baz/d2r
  else
    ! write(*,*)" problème dans dellipsgc : l'épicentre est trop porche de la station ! ",val
    d = 0.000000001_wr
    az = 0.000000001_wr
  endif
```

```
247     ! ------------------------------------------------------------------    . BAZ, output in present
248     if(present(bazfin)) bazfin=baz
249     ! ------------------------------------------------------------------    .
250   end subroutine dellipsgc
251
252 END MODULE distance_epi
253
254
255
256 ! ********************************************************************    .
257 ! ********************************************************************    .
```

2.21 SRC/MOD/intersect.f90

```
1  ! module permetant le calcul des points d'intersection entre deux cercles
2  ! quelconques sur une sphère.
3  ! fevrier 2014
4  ! ********************************************************************    .mh
5  ! -------- Philippe Cance philippe.cance@univ-nantes.fr          --------    .pc
6  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .mh
7  ! -------- Ianis Gaudot ianis.gaudot@univ-nantes.fr              --------    .ig
8  ! ********************************************************************    .
9  ! ------------------------------------------------------------------    .
10
11
12 MODULE dist_cercle
13
14     use modparam
15
16     implicit none
17
18     private
19
20     public  :: dist2c
21
22     real(KIND=wr), parameter :: d2r=pi/180._wr
23     real(KIND=wr), parameter :: r2d=180._wr/pi
24
25
26 CONTAINS
27
28 ! ------------------------------------------------------------------    .
29
30
31   subroutine dist2c(t1,t2,p1,p2,a1,a2,t_1,p_1,t_2,p_2)
32     ! -------                                                 --------    .mh
33     ! calcul des points d'intersection entre deux cercles quelconques sur une sphère
34     ! t (theta) : latitudes [-90;90]
35     ! p (phi) : longitudes [-180;180]
36     ! t et p représente le centre du cecle, à la surface (Rt=6371 km)
37     ! a : demi-angle d'ouveture du cône [0;180], depuis le centre
38     ! -------                                                 --------    .
39     implicit none
40     ! -------                                                 --------    .
41     real(KIND=wr), intent(in) :: t1,t2,p1,p2,a1,a2
42     real(KIND=wr), intent(out) :: t_1,p_1,t_2,p_2
43     ! -------                                                 --------    .
44     real(KIND=wr) :: t11,t21,p11,p21
45     ! ------------------------------------------------------------------    . conversion lat -> colat
46     t11=90.0_wr-t1
47     t21=90.0_wr-t2
48     ! ------------------------------------------------------------------    . conversion lon -> colon
49     p11=p1
50     p21=p2
51     if (p1.lt.0.0_wr) then
```

```
      p11=360.0_wr+p1
    endif
    if (p2.lt.0.0_wr) then
      p21=360.0_wr+p2
    endif
    !-------------------------------------------------------------------     . calcul des points d'intersection
    call dist_2sc(t11,t21,p11,p21,a1,a2,t_1,p_1,t_2,p_2)
    !-------------------------------------------------------------------     . if Okay
    if ((p_1.lt.1000.0_wr).and.(p_2.lt.1000.0_wr).and.(t_1.lt.1000.0_wr).and.(t_2.lt.1000.0_wr)) then
      !-------                                                  ---------     . conversion colat -> lat
      t_1=90.0_wr-t_1
      t_2=90.0_wr-t_2
      !-------                                                  ---------     . conversion colon -> lon
      if (p_1.gt.180.0_wr) then
        p_1=p_1-360.0_wr
      endif
      if (p_2.gt.180.0_wr) then
        p_2=p_2-360.0_wr
      endif
      !-------                                                  ---------     . else
      if ((p_1.lt.-170.0_wr).or.(p_2.lt.-170.0_wr).or.(p_1.gt.170.0_wr).or.(p_2.gt.170.0_wr)) then
        write(*,*)'problème dans dist2c : longitude actuelle trop près des coutures !',p_1,p_2
        stop
      endif
      if ((t_1.lt.-80.0_wr).or.(t_2.lt.-80.0_wr).or.(t_1.gt.80.0_wr).or.(t_2.gt.80.0_wr)) then
        write(*,*)'problème dans dist2c : latitude actuelle trop près des coutures !',t1,t2
        stop
      endif
      !-------                                                  ---------     .
    endif
    !-------------------------------------------------------------------     .
  end subroutine dist2c

!-----------------------------------------------------------------------     .

    subroutine dist_2sc(theta1,theta2,phi1,phi2,alpha1,alpha2,thetaA,phiA,thetaB,phiB)
    !-------                                                  ---------     .pc
    ! calcul des points d'intersection entre deux cercles quelconques sur une sphère
    ! Philippe Cance
    !-------                                                  ---------     .
    implicit none
    !-------                                                  ---------     .
    real(KIND=wr), intent(in) :: theta1, theta2, phi1, phi2, alpha1, alpha2
    real(KIND=wr), intent(out) :: thetaA, phiA, thetaB, phiB
    !-------                                                  ---------     .
    real(KIND=wr) :: heisenberg
    real(KIND=wr) :: v1(3), v2(3), v3(3), M0(3), M1(3)
    real(KIND=wr) :: a(2),b(2),c(2),alpha(2),tmp1(2)
    real(KIND=wr) :: aA,bB,cC,alphaA,alphaB,alphaC
    real(KIND=wr) :: detla, t
    !-------------------------------------------------------------------     .
    heisenberg=real(3,wr)*spacing(pi)
    !-------------------------------------------------------------------     .
    if ((abs(theta1-theta2).lt.heisenberg).and.(abs(sin(phi1-phi2)).lt.heisenberg)) then
      write(*,*)'problème dans dist_2sc : cercles paralleles !'
      thetaA=10000.0_wr
      phiA=10000.0_wr
      thetaB=thetaA
      phiB=phiA
      stop
    endif
    !-------------------------------------------------------------------     .
    v1 = (/ cos(d2r*phi1)*sin(d2r*theta1), sin(d2r*phi1)*sin(d2r*theta1), cos(d2r*theta1) /) ! v1=vec(OO1)
    v2 = (/ cos(d2r*phi2)*sin(d2r*theta2), sin(d2r*phi2)*sin(d2r*theta2), cos(d2r*theta2) /)! v2=vec(OO2)
    call vec_prod(v1, v2, v3) ! v3=vec(OO1)^vec(OO2)
```

```
    a=(/v1(1),v2(1)/)
    b=(/v1(2),v2(2)/)
    c=(/v1(3),v2(3)/)
    alpha=(/cos(d2r*alpha1),cos(d2r*alpha2)/)
    aA=det2(b,c)
    bB=det2(c,a)
    cC=det2(a,b)
    alphaA=det2(alpha,a)
    alphaB=det2(alpha,b)
    alphaC=det2(alpha,c)
    ! ------------------------------------------------------------------    .
    v1 = (/ alphaA, alphaB, alphaC /)
    v2 = (/ aA, bB, cC /)
    call vec_prod(v1,v2,M0)
    M0 = M0 / dot_product(v2,v2) ! M0 = P1 inter P2 inter (O,O1,O2)
    detla=dot_product(M0,M0)
    ! ------------------------------------------------------------------    .
    if (detla.gt.(1.0_wr)) then
      ! write(*,*)'problème dans dist_2sc : cercles non secants !',detla
      thetaA=10000.0_wr
      phiA=10000.0_wr
      thetaB=thetaA
      phiB=phiA
      ! ------------------------------------------------------------------    . un unique point d'intersection
    else if (abs(detla-1.0_wr).lt.heisenberg) then
      M1 = M0
      thetaA = acos(M1(3))
      tmp1 = M1(1:2) / sin(thetaA)
      thetaA = r2d * thetaA
      phiA= r2d * acos(tmp1(1))
      if (tmp1(2).lt.0.0_wr) phiA= phiA+ 180.0_wr
      thetaB=thetaA
      phiB=phiA
      ! ------------------------------------------------------------------    . deux points d'intersection
    else
      ! -------                                            ---------    . solution 1
      t = sqrt((1.0_wr - detla) / dot_product(v3,v3))
      M1 = M0 + t * v3
      thetaA = acos(M1(3))
      tmp1 = M1(1:2) / sin(thetaA)
      thetaA = r2d * thetaA
      phiA= r2d * acos(tmp1(1))
      if (abs(tmp1(2)) .gt. heisenberg) then
        if (tmp1(2).lt.0.0_wr) phiA= 360.0_wr - phiA
      endif
      ! -------                                            ---------    . solution 2
      M1 = M0 - t * v3
      thetaB = acos(M1(3))
      tmp1 = M1(1:2) / sin(thetaB)
      thetaB = r2d * thetaB
      phiB= r2d * acos(tmp1(1))
      if (abs(tmp1(2)) .gt. heisenberg) then
        if (tmp1(2).lt.0.0_wr) phiB= 360.0_wr - phiB
      endif
    endif
    ! ------------------------------------------------------------------    .
end subroutine dist_2sc

! ------------------------------------------------------------------    .

  function det2(a1,a2) result(res)
    ! -------                                            ---------    .pc
    ! determinant
    ! -------                                            ---------    .
    implicit none
```

```
    ! -------                                                    --------    .
    real(KIND=wr),dimension(2),intent(in) :: a1,a2
    real(KIND=wr) :: res
    ! ------------------------------------------------------------------    .
    res = a1(1)*a2(2)-a1(2)*a2(1)
    ! ------------------------------------------------------------------    .
  end function det2

! ------------------------------------------------------------------    .

  subroutine vec_prod(u1,u2,res)
    ! -------                                                    --------    .pc
    ! produit vectoriel
    ! -------                                                    --------    .
    implicit none
    ! -------                                                    --------    .
    real(KIND=wr),dimension(3),intent(in):: u1,u2
    real(KIND=wr),dimension(3),intent(out)::res
    real(KIND=wr),dimension(2) :: tmp1, tmp2
    ! ------------------------------------------------------------------    .
    tmp1 = u1(2:3)
    tmp2 = u2(2:3)
    res(1) = det2(tmp1,tmp2)
    tmp1 = (/ u1(3), u1(1) /)
    tmp2 = (/ u2(3), u2(1) /)
    res(2) = det2(tmp1,tmp2)
    tmp1 = u1(1:2)
    tmp2 = u2(1:2)
    res(3) = det2(tmp1,tmp2)
    ! ------------------------------------------------------------------    .
  end subroutine vec_prod

! ------------------------------------------------------------------    .

END MODULE dist_cercle


! ********************************************************************    .
! ********************************************************************    .
```

2.22 SRC/MOD/lectdata.f90

```
! Librairie de subroutines permettant la lecture des données
! septembre 2013
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .

MODULE datalecture

    use modparam

    implicit none

    private

    public  :: lectnbdata
    public  :: lectdata
    public  :: mksynth,mksynthallsta
    public  :: cerclespond
    public  :: catalogue
    public  :: initR

CONTAINS
```

```
24
25 ! ------------------------------------------------------------------     .
26
27   subroutine lectnbdata(nbsta,nbtps)
28     ! -------                                                 --------     .mh
29     ! lecture du nombre de données (par séisme) et du nombre de stations connues
30     ! -------                                                 --------     .
31     implicit none
32     integer(KIND=wi), intent (out) :: nbsta,nbtps(nbseismes)
33     ! -------                                                 --------     .
34     integer(KIND=wi) :: i,ok
35     character(len=35) :: seismeNom(nbseismes)
36     ! ------------------------------------------------------------------     . nombre de station connues
37     nbsta = 0
38     ok = 0
39     open(500, FILE = 'DATA/sta.d',status='old',iostat = ok)
40     if (ok .ne. 0) then
41       open(500, FILE = 'sta.d',status='old',iostat = ok)
42       if (ok .ne. 0) then
43         write(*,*)'problème dans lectnbdata : le fichier DATA/sta.d n''existe pas '
44         write(*,*)'problème dans lectnbdata : le fichier sta.d n''existe pas '
45         stop
46       endif
47     endif
48     do while(ok .eq. 0)                                             ! boucle pour compter le nombre de lignes du fichier
49       read(500,*,iostat = ok)
50       if (ok .eq. 0) nbsta = nbsta + 1
51     end do
52     close(500)
53     ! ------------------------------------------------------------------     . nombre de séismes
54     call nomseismefichier(seismeNom)                                ! lit les différents noms de phases list
55     ! -------                                                 --------     . nombre de données par séismes
56     do i=1,nbseismes
57       nbtps(i) = 0
58       ok = 0
59       open(501, FILE = 'DATA/'//seismeNom(i),status='old',iostat = ok)
60       if (ok .ne. 0) then
61         open(501, FILE = seismeNom(i),status='old',iostat = ok)
62         if (ok .ne. 0) then
63           write(*,*)'problème dans lectnbdata : le fichier DATA/',seismeNom(i),' n''existe pas '
64           write(*,*)'problème dans lectnbdata : le fichier ',seismeNom(i),' n''existe pas '
65           stop
66         endif
67       endif
68       read(501,*,iostat = ok)
69       do while(ok .eq. 0)                                           ! boucle pour compter le nombre de lignes du fichier
70         read(501,*,iostat = ok)
71         if (ok .eq. 0) nbtps(i) = nbtps(i) + 1
72       end do
73       close(501)
74       if (nbtps(i).lt.3) then
75         write(*,*)'problème dans lectnbdata : le fichier DATA/',seismeNom(i),' contient trop peu de données : nb lignes < 4 '
76         stop
77       endif
78     enddo
79     ! ------------------------------------------------------------------     .
80   end subroutine lectnbdata
81
82     ! ------------------------------------------------------------------     .
83
84   subroutine lectdata(nbsta,nbtps,D)
85     ! -------                                                 --------     .mh
86     ! lecture des données dans DATA/xxxx.xx.xx.xx.xx.d et DATA/sta-x.d
87     ! modification des données, si résidus (station correction) aux stations disponibles dans DATA/sta.d
88     ! -------                                                 --------     .
```

```
89      use typetemps, only : dataall, stations
90      use time, only : JDATE, GDATE, basetime
91      use tri, only : tridata
92      ! ------                                                  --------    .
93      implicit none
94      integer(KIND=wi), intent (in) :: nbsta,nbtps(nbseismes)
95      type(dataall), intent (inout) :: D(nbseismes)
96      ! ------                                                  --------    .
97      integer(KIND=wi) :: i,j,k,ok,count
98      type(stations) :: datasta(nbsta)
99      character(len=35) :: seismeNom(nbseismes)
100     ! ---------------------------------------------------------------    . lecture des données stations
101     open(503, FILE = 'DATA/sta.d',status='old',iostat = ok)
102     if (ok .ne. 0) then
103       write(*,*)'problème dans lectdata : le fichier DATA/sta.d n''existe pas '
104       stop
105     endif
106     do i=1,nbsta
107       read(503,*,iostat = ok) datasta(i) ! nouveau -> (i)
108     enddo
109     close(503)
110     ! ---------------------------------------------------------------    . nom des fichier par séismes
111     call nomseismefichier(seismeNom)                                    ! lit les différents noms de phases list
112     ! ------                                                  --------    . données par séismes
113     a_event : do k=1,nbseismes
114       !write(*,*)'DATA/'//seismeNom(k)
115       open(504, FILE = 'DATA/'//seismeNom(k),status='old',iostat = ok)
116       if (ok .ne. 0) then
117         write(*,*)'problème dans lectdata : le fichier DATA',seismeNom(i),' n''existe pas '
118         stop
119       endif
120       read(504,*,iostat = ok)
121       a_sta : do i=1,nbtps(k)                                           ! lecture des données temps
122         read(504,1000,iostat = ok)D(k)%datatps(i)%sta%staname,D(k)%datatps(i)%typeonde, &
123         D(k)%datatps(i)%coefP,D(k)%datatps(i)%tpsR%date%year,D(k)%datatps(i)%tpsR%date%month, &
124         D(k)%datatps(i)%tpsR%date%day,D(k)%datatps(i)%tpsR%date%hour,D(k)%datatps(i)%tpsR%date%min, &
125         D(k)%datatps(i)%tpsR%secP,D(k)%datatps(i)%tpsR%secS,D(k)%datatps(i)%andS,D(k)%datatps(i)%coefS, &
126         D(k)%datatps(i)%sigP,D(k)%datatps(i)%sigS
127
128         ! ------                                                --------    . incertitudes en secondes sur les données
129         if((D(k)%datatps(i)%sigP.lt.0.0001_wr).and.(D(k)%datatps(i)%sigP.gt.10._wr).and.(IsNaN(D(k)%datatps(i)%sigP))) then
130           write(*,*)'problème dans lectdata : les incertitudes sur les données P n''existent pas ',D(k)%datatps(i)%sigP
131           stop
132         endif
133         if ((D(k)%datatps(i)%andS.ne."S").and.(D(k)%datatps(i)%sigS.gt.10._wr) &
134           .and.(D(k)%datatps(i)%sigS.lt.0.0001_wr).and.(IsNaN(D(k)%datatps(i)%sigS))) then
135           write(*,*)'problème dans lectdata : les incertitudes sur les données S n''existent pas ',D(k)%datatps(i)%sigS
136           stop
137         endif
138         ! ------                                                --------    . si pas d'ondes S
139         if(D(k)%datatps(i)%andS.ne."S") then
140           D(k)%datatps(i)%andS="X"
141           D(k)%datatps(i)%tpsR%secS=0.0_wr
142           D(k)%datatps(i)%coefS=4
143         endif
144         ! ------                                                --------    .
145         D(k)%datatps(i)%tpsR%date%year=D(k)%datatps(i)%tpsR%date%year+2000
146         ! ------                                                --------    . respect du decoupage des années en mois et jours avec prise en compte des années
                bisextiles
147         call basetime(D(k)%datatps(i)%tpsR)
148         call JDATE(D(k)%datatps(i)%tpsR%date%Jday,D(k)%datatps(i)%tpsR%date%year, &
149           D(k)%datatps(i)%tpsR%date%month,D(k)%datatps(i)%tpsR%date%day)
150         call GDATE (D(k)%datatps(i)%tpsR%date%Jday,D(k)%datatps(i)%tpsR%date%year, &
151           D(k)%datatps(i)%tpsR%date%month,D(k)%datatps(i)%tpsR%date%day)
152         call basetime(D(k)%datatps(i)%tpsR)                             ! respect du decoupage temps dans la base composite 60/12/365 ...
```

```
153         ! -------                                                  --------     .
154         count=0
155         do j=1,nbsta
156           if(D(k)%datatps(i)%sta%staname.eq.datasta(j)%staname) then
157             D(k)%datatps(i)%sta=datasta(j)                                    ! attribution d'une station
158             count=count+1
159           endif
160         enddo
161         if(count.gt.1) then                                                   ! vérification de l'absence de doublons
162           write(*,*)'problème dans lectdata : station ',D(k)%datatps(i)%sta%staname,' en double in file : DATA/sta.d'
163           stop
164         endif
165         if(count.eq.0) then
166           write(*,*)'problème dans lectdata : station ',D(k)%datatps(i)%sta%staname,' non répertoriée'
167           stop
168         endif
169         ! -------                                                  --------     .
170         ! ajout des résidus aux station (DATA/sta.d)
171         ! les données sont donc modifié en amont
172         ! -------                                                  --------     .
173         if (D(k)%datatps(i)%typeonde.eq.'G') then                             ! ondes directes
174           D(k)%datatps(i)%tpsR%secP=D(k)%datatps(i)%tpsR%secP-D(k)%datatps(i)%sta%res_Pg
175           if(D(k)%datatps(i)%andS.eq.'S') D(k)%datatps(i)%tpsR%secS=D(k)%datatps(i)%tpsR%secS-D(k)%datatps(i)%sta%res_Sg
176         elseif (D(k)%datatps(i)%typeonde.eq.'N') then                         ! ondes réfractées
177           ! -------                                                --------     . ondes réfractées moins pondérées !!!!!!!!!!!!!!!
178           D(k)%datatps(i)%coefP= min(D(k)%datatps(i)%coefP+2,4)
179           D(k)%datatps(i)%coefS= min(D(k)%datatps(i)%coefS+2,4)
180           ! -------                                                --------     . ondes réfractées moins pondérées !!!!!!!!!!!!!!!
181           D(k)%datatps(i)%tpsR%secP=D(k)%datatps(i)%tpsR%secP-D(k)%datatps(i)%sta%res_Pn
182           if(D(k)%datatps(i)%andS.eq.'S') D(k)%datatps(i)%tpsR%secS=D(k)%datatps(i)%tpsR%secS-D(k)%datatps(i)%sta%res_Sn
183         else
184           write(*,*)'problème dans lectdata : onde ni directe ni réfractée ... ? '
185           write(*,*)D(k)%datatps(i)%sta%staname
186           write(*,*)
187           write(*,*)D(k)%datatps(i)
188           write(*,*)
189           stop
190         endif
191         call basetime(D(k)%datatps(i)%tpsR)
192         call modifPY42(D(k)%datatps(i))                                       ! gestion de cas particuliers ...
193         ! -------                                                  --------     .
194       enddo a_sta
195       close(504)
196       ! -------                                                  --------     .
197       call tridata(nbtps(k),D(k)%datatps)                                     ! tri des donnée selon un temps d'arrivée des ondes P croissant
198     enddo a_event
199     ! -------                                                    --------     . format de lecture des données
200     1000 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
201     ! ------------------------------------------------------------------     .
202
203     CONTAINS
204
205       ! ------------------------------------------------------------------     .
206       subroutine modifPY42(adata)
207       ! ------------------------------------------------------------------     . mh
208       ! modification des données, gestion de cas particuliers ...
209       ! ici modification de localisation de la station PY42 de PyrOPE
210       ! PY42A et PY42B
211       ! ------------------------------------------------------------------     .
212       use typetemps, only : dataone,date_secPS
213       use time, only : difftime
214       ! ------------------------------------------------------------------     .
215       implicit none
216       type(dataone), intent (inout) :: adata
217       ! ------------------------------------------------------------------     .
```

```
218       type(date_secPS) :: datespe
219       real(KIND=wr) :: diff1,diff2
220       real(KIND=wr) :: deltatime
221       ! ------------------------------------------------------------------      .
222       if (adata%sta%staname=='PY42') then
223         ! -------                                                  --------      .
224         datespe%date%year=2012
225         datespe%date%month=10
226         datespe%date%day=16
227         datespe%date%hour=12
228         datespe%date%min=30
229         call JDATE (datespe%date%Jday,datespe%date%year,datespe%date%month,datespe%date%day)
230         call basetime(datespe)
231         datespe%secP=0.0_wr
232         datespe%secS=0.0_wr
233         ! -------                                                  --------      .
234         call difftime(diff1,diff2,adata%tpsR,datespe)
235         deltatime=real(60*60*24*2,wr)                                          ! 2 jours
236         ! -------                                                  --------      .
237         if (diff1.gt.deltatime) then                                           ! PY42-B
238           adata%sta%lon=-1.20256_wr
239           adata%sta%lat=46.39462_wr
240           adata%sta%alti=53.0_wr
241         elseif (diff1.lt.(-deltatime)) then                                    ! PY42-A
242           adata%sta%lon=-1.20075_wr
243           adata%sta%lat=46.41012_wr
244           adata%sta%alti=50.0_wr
245         else                                                                   ! n'existe pas !
246           adata%sta%lon=0.0_wr
247           adata%sta%lat=0.0_wr
248           adata%sta%alti=0.0_wr
249           write(*,*)'problème dans modifPY42 (lectdata) : avec PY42 ',adata
250           stop
251         endif
252         ! -------                                                  --------      .
253       endif
254       ! ------------------------------------------------------------------      .
255       end subroutine modifPY42
256       ! ------------------------------------------------------------------      .
257
258   end subroutine lectdata
259
260     ! ------------------------------------------------------------------      .
261
262   subroutine initR(D,R,maxiter,nbtps,nbsta,nbofSta)
263     ! ------------------------------------------------------------------      .mh
264     ! initialise le calcul des résidus (si FLAGresSTA=.true.), en allouant R
265     ! pour chaque station, chaque itération on sauve le résidus
266     ! afin de définir des tendenses -> station correction
267     ! sub inR et outR -> dans MOD/subparam.f90
268     ! ------------------------------------------------------------------      .
269     use typetemps
270     ! -------                                                  --------      .
271     implicit none
272     type(dataall), intent (in) :: D(nbseismes)                                 ! données
273     integer(KIND=wi), intent (in) :: maxiter, nbtps(nbseismes),nbsta
274     integer(KIND=wi), intent (out) :: nbofSta                                  ! nb de phases
275     ! -------                                                  --------      .
276     type(residus), dimension(:), allocatable, intent (out) :: R
277     character (LEN=4) :: nomDesta(2*nbsta+2)
278     ! -------                                                  --------      .
279     integer(KIND=wi) :: i,j,k,l
280     logical :: test
281     ! ------------------------------------------------------------------      . nb de station différente pour ondes directes et refractées
282     do i=1,2*nbsta+2
```

```
      nomDesta(i)='1234'
    enddo
    ! -------                                                --------    . stations ?
    nbofSta=0
    do i=1,nbseismes
      do j=1,nbtps(i)
        test=.true.
        do k=1,nbofSta+2
          if (D(i)%datatps(j)%sta%staname==nomDesta(k)) then             ! déjà présente ?
            test=.false.
          endif
        enddo
        if (test) then
          nbofSta=nbofSta+1
          nomDesta(nbofSta)=D(i)%datatps(j)%sta%staname
        endif
      enddo
    enddo
    ! -------                                                --------    .
    if (nbsta.lt.nbofSta) then
      write(*,*)'problème dans initR : ',nbsta,'<',nbofSta
      stop
    endif
    ! ----------------------------------------------------------------    .
    allocate(R(nbofSta))
    ! -------                                                --------    .
    do i=1,nbofSta
      R(i)%nbPg=-1000
      R(i)%nbSg=-1000
      R(i)%nbPn=-1000
      R(i)%nbSn=-1000
      R(i)%nbPgT=-1000
      R(i)%nbSgT=-1000
      R(i)%nbPnT=-1000
      R(i)%nbSnT=-1000
      R(i)%staname=nomDesta(i)
    enddo
    ! -------                                                --------    . existe-il des Pg, Pn, Sg et Sn pour ces stations ?
    do k=1,nbofSta                                                       ! pour chaque station
      ! -------                                              --------    .
      l=0
      do i=1,nbseismes                                                   ! Pg ?
        do j=1,nbtps(i)
          if ((D(i)%datatps(j)%sta%staname==R(k)%staname).and.(D(i)%datatps(j)%typeonde=='G')) l=l+1
        enddo
      enddo
      if (l.gt.0) R(k)%nbPg=0
      if (l.gt.0) R(k)%nbPgT=0
      if ((l.gt.0).and.(.not.allocated(R(k)%resPg))) then
        allocate(R(k)%resPg(maxiter*l,2))
        R(k)%resPg(:,1)=0.0_wr ! résidu
        R(k)%resPg(:,2)=1.e9_wr ! fonction coût
      endif
      if (l.gt.nbseismes) then
        write(*,*)'problème dans initR Pg : trop de ok ',l,nbseismes
        stop
      endif
      ! -------                                              --------    .
      l=0
      do i=1,nbseismes                                                   ! Pn ?
        do j=1,nbtps(i)
          if ((D(i)%datatps(j)%sta%staname==R(k)%staname).and.(D(i)%datatps(j)%typeonde=='N')) l=l+1
        enddo
      enddo
      if (l.gt.0) R(k)%nbPn=0
```

```
348       if (l.gt.0) R(k)%nbPnT=0
349       if ((l.gt.0).and.(.not.allocated(R(k)%resPn))) then
350         allocate(R(k)%resPn(maxiter*l,2))
351         R(k)%resPn(:,1)=0.0_wr ! résidu
352         R(k)%resPn(:,2)=1.e9_wr ! fonction coût
353       endif
354       if (l.gt.nbseismes) then
355         write(*,*)'problème dans initR Pn : trop de ok ',l,nbseismes
356         stop
357       endif
358       ! -------                                                 --------     .
359       l=0
360       do i=1,nbseismes                                                       ! Sg ?
361         do j=1,nbtps(i)
362           if ((D(i)%datatps(j)%sta%staname==R(k)%staname).and.(D(i)%datatps(j)%typeonde=='G') &
363             .and.(D(i)%datatps(j)%andS=='S')) l=l+1
364         enddo
365       enddo
366       if (l.gt.0) R(k)%nbSg=0
367       if (l.gt.0) R(k)%nbSgT=0
368       if ((l.gt.0).and.(.not.allocated(R(k)%resSg))) then
369         allocate(R(k)%resSg(maxiter*l,2))
370         R(k)%resSg(:,1)=0.0_wr ! résidu
371         R(k)%resSg(:,2)=1.e9_wr ! fonction coût
372       endif
373       if (l.gt.nbseismes) then
374         write(*,*)'problème dans initR Sg : trop de ok ',l,nbseismes
375         stop
376       endif
377       ! -------                                                 --------     .
378       l=0
379       do i=1,nbseismes                                                       ! Sn ?
380         do j=1,nbtps(i)
381           if ((D(i)%datatps(j)%sta%staname==R(k)%staname).and.(D(i)%datatps(j)%typeonde=='N') &
382             .and.(D(i)%datatps(j)%andS=='S')) l=l+1
383         enddo
384       enddo
385       if (l.gt.0) R(k)%nbSn=0
386       if (l.gt.0) R(k)%nbSnT=0
387       if ((l.gt.0).and.(.not.allocated(R(k)%resSn))) then
388         allocate(R(k)%resSn(maxiter*l,2))
389         R(k)%resSn(:,1)=0.0_wr ! résidu
390         R(k)%resSn(:,2)=1.e9_wr ! fonction coût
391         endif
392       if (l.gt.nbseismes) then
393         write(*,*)'problème dans initR Sn : trop de ok ',l,nbseismes
394         stop
395       endif
396       ! -------                                                 --------     .
397     enddo
398     ! ------------------------------------------------------------------     .
399   end subroutine initR
400
401     ! ------------------------------------------------------------------     .
402
403   subroutine mksynth(nbtps,D,acentroid)
404     ! -------                                                 --------     .mh
405     ! calcul données synthétiques bruitées ou non
406     ! ------------------------------------------------------------------     .
407     use typetemps
408     use time, only : basetime
409     use mt19937, only : genrand_real1, normal
410     use pb_direct
411     use sub_param
412     use sub_misfit
```

```
    ! _______                                                    ________     .
    implicit none
    integer(KIND=wi), intent (in) :: nbtps(nbseismes)                        ! nombre de données de temps
    type(dataall), intent (in) :: D(nbseismes)                               ! données
    type (amoho_centroid), intent (in) :: acentroid
    ! _______                                                    ________     .
    type(parametre) :: pmod                                                  ! paramètres d'inv. modifiés
    type(dataone), dimension(:), allocatable :: datatempsmod                 ! données modifiés
    integer(KIND=wi) :: i, j, mil_1, mil_2, ok
    real(KIND=wr) :: val_1, val_2, aleatoire, bestval, xmin, xmax
    logical :: critique
    type(parametres) :: paramall
    ! ------------------------------------------------------------------     .
    open(505, FILE = 'DATA/newtemps.d',status='replace')
    ! _______                                                    ________     .
    nb_seismes : do j=1,nbseismes
      allocate(datatempsmod(nbtps(j)))
      datatempsmod=D(j)%datatps
      ! _______                                                  ________     .
      ! un MODÈLE À DÉFINIR :
      pmod%VC=6.0_wr
      pmod%VM=8.0_wr
      pmod%Zmoho=30._wr
      pmod%VpVs=1.710_wr
      pmod%Tzero%date=D(j)%datatps(1)%tpsR%date
      pmod%Tzero%date%year=2012
      pmod%Tzero%date%month=12
      pmod%Tzero%date%day=15
      pmod%Tzero%date%hour=12
      pmod%Tzero%date%min=0
      pmod%Tzero%sec=30.00_wr
      pmod%Lat=48.25_wr
      pmod%Lon=-2.25_wr
      pmod%Zhypo=15.00_wr
      ! _______                                                  ________     .
      write(505,*)'FICHIER synthétique : Lat=',pmod%Lat,'Lon=',pmod%Lon,'Zhypo=',pmod%Zhypo
      write(505,*)
      do i=1,nbtps(j)
        call tempsTheoDirectone(pmod,datatempsmod(i),critique,acentroid)    ! pour le jeu de paramètre tiré et chaque donnée
        if((datatempsmod(i)%typeonde.eq.'G').or.((datatempsmod(i)%typeonde.eq.'N').and.(.not.critique))) then ! distance critique
          if(datatempsmod(i)%andS.eq.'S') then
            write(505,1001)datatempsmod(i)%sta%staname,datatempsmod(i)%typeonde,datatempsmod(i)%coefP, &
            datatempsmod(i)%tpsTh%date%year-2000,datatempsmod(i)%tpsTh%date%month,datatempsmod(i)%tpsTh%date%day, &
            datatempsmod(i)%tpsTh%date%hour,datatempsmod(i)%tpsTh%date%min,datatempsmod(i)%tpsTh%secP, &
            datatempsmod(i)%tpsTh%secS,datatempsmod(i)%andS,datatempsmod(i)%coefS, &
            datatempsmod(i)%sigP,datatempsmod(i)%sigS
          else
            write(505,1002)datatempsmod(i)%sta%staname,datatempsmod(i)%typeonde,datatempsmod(i)%coefP, &
            datatempsmod(i)%tpsTh%date%year-2000,datatempsmod(i)%tpsTh%date%month,datatempsmod(i)%tpsTh%date%day, &
            datatempsmod(i)%tpsTh%date%hour,datatempsmod(i)%tpsTh%date%min,datatempsmod(i)%tpsTh%secP,datatempsmod(i)%sigP
          endif
        endif
      enddo
      ! _______                                                  ________     .
      write(505,*)
      write(505,*)' ! _______                                    ________     .'
      write(505,*)'synthétique bruité : '
      write(505,*)
      do i=1,nbtps(j)
        ! _______                                                ________     .
        aleatoire = genrand_real1()
        mil_1 = int(aleatoire*999.99999999999_wr)
        aleatoire = genrand_real1()
        mil_2 = int(aleatoire*999.99999999999_wr)
        select case(mil_1)
```

```
        case(0:555)
          val_1=0.025_wr                                       ! 55,5 % d'être bruité avec une gaussienne de 0.025 s d'écart-type
        case(556:835)
          val_1=0.05_wr                                        ! 28 % d'être bruité avec une gaussienne de 0.05 s d'écart-type
        case(836:905)
          val_1=0.10_wr                                        ! 7 % d'être bruité avec une gaussienne de 0.1 s d'écart-type
        case(906:910)
          val_1=0.25_wr                                        ! 0,5 % d'être bruité avec une gaussienne de 0.25 s d'écart-type
        case(911:1000)
          val_1=0.3_wr                                         ! 9 % d'être bruité avec une gaussienne de 0.3 s d'écart-type
        case default                                           ! pour les pct -> Golle, 2013 et Haugmard, 2013 (rapport M2, Annexes, p I.)
          write(*,*)'problème dans mksynth : mil_1 ',mil_1
        end select
        ! -------                                      --------    .
        select case(mil_2)
        case(0:555)
          val_2=0.025_wr
        case(556:835)
          val_2=0.05_wr
        case(836:905)
          val_2=0.10_wr
        case(906:910)
          val_2=0.25_wr
        case(911:1000)
          val_2=0.3_wr
        case default
          write(*,*)'problème dans mksynth : mil_2 ',mil_2
        end select
        ! -------                                      --------    .
        datatempsmod(i)%tpsR = datatempsmod(i)%tpsTh
        aleatoire=val_1 ! aleatoire=normal(0.0_wr,val_1)
        datatempsmod(i)%tpsR%secP = datatempsmod(i)%tpsTh%secP + aleatoire
        datatempsmod(i)%sigP=abs(aleatoire)
        call basetime(datatempsmod(i)%tpsR)                    ! respect du decoupage temps dans la base composite 60/12/365
        if((datatempsmod(i)%typeonde.eq.'G').or.((datatempsmod(i)%typeonde.eq.'N').and.(.not.critique))) then ! distance critique
          if(datatempsmod(i)%andS.eq.'S') then
            aleatoire=val_2 ! aleatoire=normal(0.0_wr,val_2)
            datatempsmod(i)%tpsR%secS = datatempsmod(i)%tpsTh%secS + aleatoire
            datatempsmod(i)%sigS=abs(aleatoire)
            call basetime(datatempsmod(i)%tpsR)                ! respect du decoupage temps dans la base composite 60/12/365
            write(505,1001)datatempsmod(i)%sta%staname,datatempsmod(i)%typeonde,datatempsmod(i)%coefP, &
            datatempsmod(i)%tpsR%date%year-2000,datatempsmod(i)%tpsR%date%month,datatempsmod(i)%tpsR%date%day, &
            datatempsmod(i)%tpsR%date%hour,datatempsmod(i)%tpsR%date%min,datatempsmod(i)%tpsR%secP, &
            datatempsmod(i)%tpsR%secS,datatempsmod(i)%andS,datatempsmod(i)%coefS, &
            datatempsmod(i)%sigP,datatempsmod(i)%sigS
          else
            write(505,1002)datatempsmod(i)%sta%staname,datatempsmod(i)%typeonde,datatempsmod(i)%coefP, &
            datatempsmod(i)%tpsR%date%year-2000,datatempsmod(i)%tpsR%date%month,datatempsmod(i)%tpsR%date%day, &
            datatempsmod(i)%tpsR%date%hour,datatempsmod(i)%tpsR%date%min,datatempsmod(i)%tpsR%secP, datatempsmod(i)%sigP
          endif
        endif
      enddo
      ! -------                                        --------    . calcul du misfit
      ok=0
      open(507, FILE = 'PARAM/cerclesponderation.d',status='old',iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans mksynth : le fichier PARAM/cerclesponderation.d n''existe pas '
        stop
      endif
      i=0
      do while (i.ne.j)
        i=i+1
        read(507,*,iostat = ok)xmin,xmax
      enddo
      close(507)
```

```
543       call cerclespondone(nbtps(j),datatempsmod,pmod,xmin,xmax,acentroid,chut=.true.)
544       ! ------                                                    --------    .
545       do i=1,nbtps(j)
546         call tempsTheoDirectone(pmod,datatempsmod(i),critique,acentroid)
547       enddo
548       call mvP1_2_Pall(paramall,pmod,j)
549       call compute_misfitone(nbtps(j),datatempsmod,bestval,xmin,xmax,'H')
550       call mvPall_2_P1(pmod,paramall,j)
551       ! ------                                                    --------    .
552       write(505,*)
553       write(505,*)'meilleur misfit :', bestval/real(nbtps(j),wr)          ! valeur de la fonction coût pour ces donnés bruités et les paramètres fixés (minimum)
              pour ce séisme
554       ! ------                                                    --------    .
555       deallocate(datatempsmod)
556     enddo nb_seismes
557     ! ------                                                    --------    .
558     close(505)
559     ! ------                                                    --------    .
560     1001 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
561     1002 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,13x,f6.3)
562     ! ------------------------------------------------------------------    .
563   end subroutine mksynth
564
565     ! ------------------------------------------------------------------    .
566
567   subroutine mksynthallsta(acentroid,dp)
568     ! ------                                                    --------    .mh
569     ! calcul données synthétiques pour toutes les station connues
570     ! ------------------------------------------------------------------    .
571     use typetemps, only : amoho_centroid,parametres,stations,parametre,dataone,densityplot,mvPall_2_P1
572     use pb_direct, only : tempsTheoDirectone
573     use time, only : tempszero, basetime
574     ! ------                                                    --------    .
575     implicit none
576     type (amoho_centroid), intent (in) :: acentroid
577     type(densityplot), intent (in) :: dp
578     ! ------                                                    --------    .
579     type(parametres) :: param_best
580     type(stations), dimension(:), allocatable :: datasta
581     type(parametre) :: pmod                                             ! paramètres d'inv. modifiés
582     type(dataone) :: datatempsmod
583     integer(KIND=wi) :: nbsta,i, j, ok
584     logical :: critique
585     character (LEN=5) :: numberchaine
586     ! ------------------------------------------------------------------    .
587     open(509, FILE = 'DATA/sta.d',status='old',iostat = ok)
588     if (ok .ne. 0) then
589       write(*,*)'problème dans mksynthallsta : le fichier DATA/sta.d n''existe pas '
590       stop
591     endif
592     ! ------                                                    --------    .
593     nbsta=0
594     do while(ok .eq. 0)                                                 ! boucle pour compter le nombre de lignes du fichier
595       read(509,*,iostat = ok)
596       if (ok .eq. 0) nbsta = nbsta + 1
597     end do
598     ! ------                                                    --------    .
599     rewind(509)
600     allocate(datasta(nbsta))
601     ! ------                                                    --------    .
602     do i=1,nbsta
603       read(509,*,iostat = ok) datasta(i) ! nouveau -> (i)
604     enddo
605     close(509)
606     ! ------------------------------------------------------------------    .
```

```
607     param_best%VC=dp%VC%moy_100
608     param_best%VM=dp%VM%moy_100
609     param_best%Zmoho=dp%Zmoho%moy_100
610     param_best%VpVs=dp%VpVs%moy_100
611     do i=1,nbseismes
612       param_best%Zhypo(i)=dp%Zhypo(i)%moy_100
613       param_best%lon(i)=dp%lon(i)%moy_100
614       param_best%lat(i)=dp%lat(i)%moy_100
615       param_best%Tzero(i) = dp%temps_ref(i)
616       param_best%Tzero(i)%sec = dp%Tzero(i)%moy_100
617       call basetime(param_best%Tzero(i))
618     enddo
619     ! ------------------------------------------------------------------      .
620     nb_seismes : do j=1,nbseismes
621       write(numberchaine(1:5),'(i5)')j
622       ! -------                                                  --------      .
623       open(508, FILE = 'OUTPUT/files/tempsTheoOUT_'//trim(adjustl(numberchaine))//'.d',status='replace')
624       ! -------                                                  --------      .
625       call mvPall_2_P1(pmod,param_best,j)
626       ! -------                                                  --------      .
627       write(508,*)'FICHIER synthétique : ',j
628       do i=1,nbsta
629         ! -------                                                --------      .
630         datatempsmod%sta=datasta(i)
631         call tempszero(datatempsmod%tpsR%date)
632         datatempsmod%tpsR%secP=0.0_wr
633         datatempsmod%tpsR%secS=0.0_wr
634         call tempszero(datatempsmod%tpsTh%date)
635         datatempsmod%tpsTh%secP=0.0_wr
636         datatempsmod%tpsTh%secS=0.0_wr
637         datatempsmod%typeonde='G'                                    ! Pas de N, pour le moment
638         datatempsmod%coefP=0
639         datatempsmod%coefS=0
640         datatempsmod%andS='S'
641         datatempsmod%dTP=0.0_wr
642         datatempsmod%dTS=0.0_wr
643         datatempsmod%ws=1.0_wr
644         datatempsmod%wp=1.0_wr
645         datatempsmod%tpsparcP=0.0_wr
646         datatempsmod%tpsparcS=0.0_wr
647         datatempsmod%depi=0.0_wr
648         datatempsmod%dhypo=0.0_wr
649         datatempsmod%dcritiqueH=0.0_wr
650         datatempsmod%baz=0.0_wr
651         datatempsmod%sigP=0.2_wr
652         datatempsmod%sigS=0.5_wr
653         ! -------                                                --------      .
654         call tempsTheoDirectone(pmod,datatempsmod,critique,acentroid)      ! pour le jeu de paramètre tiré et chaque donnée
655         ! -------                                                --------      .
656         if((datatempsmod%typeonde.eq.'G').or.((datatempsmod%typeonde.eq.'N').and.(.not.critique))) then ! distance critique
657           if(datatempsmod%andS.eq.'S') then
658             write(508,1003)datatempsmod%sta%staname,datatempsmod%typeonde,datatempsmod%coefP, &
659               datatempsmod%tpsTh%date%year-2000,datatempsmod%tpsTh%date%month,datatempsmod%tpsTh%date%day, &
660               datatempsmod%tpsTh%date%hour,datatempsmod%tpsTh%date%min,datatempsmod%tpsTh%secP, &
661               datatempsmod%tpsTh%secS,datatempsmod%andS,datatempsmod%coefS, &
662               datatempsmod%sigP,datatempsmod%sigS
663           else
664             write(508,1004)datatempsmod%sta%staname,datatempsmod%typeonde,datatempsmod%coefP, &
665               datatempsmod%tpsTh%date%year-2000,datatempsmod%tpsTh%date%month,datatempsmod%tpsTh%date%day, &
666               datatempsmod%tpsTh%date%hour,datatempsmod%tpsTh%date%min,datatempsmod%tpsTh%secP,datatempsmod%sigP
667           endif
668         endif
669       enddo
670       ! -------                                                  --------      .
671       close(508)
```

```
672     enddo nb_seismes
673     ! ------------------------------------------------------------------      .
674     1003 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,f7.3,1x,a1,i1.1,3x,f6.3,4x,f6.3)
675     1004 format(a4,1x,a1,1x,i1,1x,5i2.2,f6.3,5x,13x,f6.3)
676     ! ------------------------------------------------------------------      .
677   end subroutine mksynthallsta
678
679     ! ------------------------------------------------------------------      .
680
681   subroutine cerclespond(nbtps,D,param_init,xmin,xmax,acentroid)
682     ! _______                                                 ________      .mh
683     ! calcul xmin et xmax, diamètres cercles de pondération, pour tous les séismes
684     ! lecture dans un fichier PARAM/cerclesponderation.d ou determiné par
685     ! cacul, si val=-1
686     ! _______                                                 ________      .
687     use typetemps, only : dataall, parametre, parametres, amoho_centroid
688     use typetemps, only : mvPall_2_P1, mvP1_2_Pall
689     use pb_direct
690     ! _______                                                 ________      .
691     implicit none
692     integer(KIND=wi), intent(in) :: nbtps(nbseismes)
693     type(dataall), intent(inout) :: D(nbseismes)
694     type(parametres), intent(inout) :: param_init
695     real(KIND=wr), intent(out) :: xmin(nbseismes),xmax(nbseismes)
696     type (amoho_centroid), intent (in) :: acentroid
697     ! _______                                                 ________      .
698     integer(KIND=wi) :: i, ok
699     type(parametre) :: param
700     ! ------------------------------------------------------------------      .
701     ok=0
702     open(507, FILE = 'PARAM/cerclesponderation.d',status='old',iostat = ok)
703     if (ok .ne. 0) then
704       write(*,*)'problème dans cerclespond : le fichier PARAM/cerclesponderation.d n''existe pas '
705       stop
706     endif
707     do i=1,nbseismes
708       read(507,*,iostat = ok)xmin(i),xmax(i)
709       call mvPall_2_P1(param,param_init,i)
710       call cerclespondone(nbtps(i),D(i)%datatps,param,xmin(i),xmax(i),acentroid)
711       call mvP1_2_Pall(param_init,param,i)
712     enddo
713     close(507)
714     ! ------------------------------------------------------------------      .
715    end subroutine cerclespond
716
717     ! ------------------------------------------------------------------      .
718
719   subroutine cerclespondone(nbtps,datatps,param_init,xmin,xmax,acentroid,chut)
720     ! _______                                                 ________      .mh
721     ! calcul xmin et xmax, diamètres cercles de pondération pour un séisme
722     ! _______                                                 ________      .
723     use typetemps, only : dataone, parametre, amoho_centroid
724     use pb_direct
725     ! _______                                                 ________      .
726     implicit none
727     integer(KIND=wi), intent(in) :: nbtps
728     type(dataone), intent(inout) :: datatps(nbtps)
729     type(parametre), intent(in) :: param_init
730     real(KIND=wr), intent(out) :: xmin,xmax
731     type (amoho_centroid), intent (in) :: acentroid
732     logical, intent(in), optional :: chut
733     ! _______                                                 ________      .
734     integer(KIND=wi) :: i,n
735     real(KIND=wr) :: valmax
736     logical :: critique
```

```
737     ! ------------------------------------------------------------------     .
738     if (xmax.lt.xmin) then
739       valmax=xmax
740       xmax=xmin
741       xmin=valmax
742       write(*,*)'problème dans cerclespondone : xmax < xmin !'
743     endif
744       ! -------                                                 --------     .
745     if ((xmin.le.5.0_wr).or.(xmax.le.10.0_wr)) then
746       xmin = 0.0_wr
747       xmax = 0.0_wr
748       valmax = -1.0_wr
749       ! -------                                                 --------     .
750       do i=1,nbtps
751         call tempsTheoDirectone(param_init,datatps(i),critique,acentroid)  ! permet le calcul des distances épicentrales
752         if (datatps(i)%depi.gt.valmax) valmax = datatps(i)%depi            ! sauve la plus lointaine station
753       enddo
754       ! -------                                                 --------     .
755       n = 0
756       do while (n.lt.(nbtps/2))
757         xmin = xmin + 50.0_wr                                              ! on élargit le cercle tant qu'au mois la moitié des stations s'y trouvent
758         n = 0                                                              ! nombre des stations dans le petit cercle
759         do i=1,nbtps
760           if (datatps(i)%depi.lt.xmin) n=n+1
761         enddo
762       enddo
763       ! -------                                                 --------     .
764       if (valmax.lt.100.0_wr) valmax = 100.0_wr                            ! limite inférieur
765       if (valmax.gt.1000.0_wr) valmax = 1000.0_wr                          ! limite supérieur
766       do while (xmax.lt.valmax)
767         xmax = xmax + 50.0_wr                                              ! toutes les stations sont dans le grand cercle
768       enddo
769       ! -------                                                 --------     .
770       xmax = xmax*2.0_wr                                                   ! ce sont des diamètres
771       xmin = xmin*2.0_wr                                                   ! ce sont des diamètres
772       ! -------                                                 --------     .
773       if (xmax.le.xmin) then
774         xmax = xmax + 0.1_wr*xmax
775       endif
776       if(.not.present(chut)) then
777         write(*,'(a34,f7.2,1x,f7.2)')" cercles de pondération libres  : ",xmin,xmax
778       else
779         if(.not.chut) write(*,'(a34,f7.2,1x,f7.2)')" cercles de pondération libres  : ",xmin,xmax
780       endif
781     else
782       if(.not.present(chut)) then
783         write(*,'(a34,f7.2,1x,f7.2)')" cercles de pondération fixes   : ",xmin,xmax
784       else
785         if(.not.chut) write(*,'(a34,f7.2,1x,f7.2)')" cercles de pondération fixes   : ",xmin,xmax
786       endif
787     endif
788     ! ------------------------------------------------------------------     .
789   end subroutine cerclespondone
790
791     ! ------------------------------------------------------------------     .
792
793   subroutine catalogue(param,theseisme,find)
794     ! -------                                                 --------     .mh
795     ! permet la lecture d'un catalogue et de trouver la référence ReNaSS et/ou LDG du séisme
796     ! -------                                                 --------     .
797     use typetemps, only : parametre, seismes
798     use distance_epi
799     use time, only :  JDATE, GDATE, difftime, tempszero
800     ! -------                                                 --------     .
801     implicit none
```

```
    type(parametre), intent(in) :: param                                    ! paramètres issus de l'inversion McMC
    type(seismes), intent(out) :: theseisme(2)                              ! paramètres du catalogue
    integer(KIND=wi), intent(out) :: find                                   ! ce séisme est présent dans le catalogue, 0-1-2 fois
    ! -------                                                 --------      .
    type(seismes) :: S                                                      ! pour chaque événement du catalogue
    integer(KIND=wi) :: i,ok
    ! ---------------------------------------------------------------       .
    ok = 0
    i = 0
    find = 0
    ! -------                                                 --------      . initialise
    theseisme(1)%lon=0.0_wr
    theseisme(1)%lat=0.0_wr
    theseisme(1)%pfd=0.0_wr
    call tempszero(theseisme(1)%tps_init%date)
    theseisme(1)%tps_init%sec=0.0_wr
    theseisme(1)%mag=0.0_wr
    theseisme(1)%d_t=0.0_wr
    theseisme(1)%d_epi=0.0_wr
    theseisme(1)%d_p=0.0_wr
    theseisme(1)%name="      "
    ! -------                                                 --------      .
    theseisme(2)%lon=0.0_wr
    theseisme(2)%lat=0.0_wr
    theseisme(2)%pfd=0.0_wr
    call tempszero(theseisme(2)%tps_init%date)
    theseisme(2)%tps_init%sec=0.0_wr
    theseisme(2)%mag=0.0_wr
    theseisme(2)%d_t=0.0_wr
    theseisme(2)%d_epi=0.0_wr
    theseisme(2)%d_p=0.0_wr
    theseisme(2)%name="      "
    ! -------                                                 --------      . lecture catalogue
    open(506, FILE = 'DATA/files/catalogue.d',status='old',iostat = ok)
    if (ok .ne. 0) then
      open(506, FILE = 'files/catalogue.d',status='old',iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans catalogue : le fichier DATA/files/catalogue.d n''existe pas '
        write(*,*)'problème dans catalogue : le fichier files/catalogue.d n''existe pas '
        stop
      endif
    endif
    do while(ok .eq. 0)
      ! -------                                                 --------      .
      read(506,*,iostat = ok)S%tps_init%date%year,S%tps_init%date%month,S%tps_init%date%day, &
        S%tps_init%date%hour,S%tps_init%date%min,S%tps_init%sec,S%lat,S%lon,S%mag,S%pfd,S%name
      if (ok==0) then
        ! -------                                               --------      .
        call JDATE(S%tps_init%date%Jday,S%tps_init%date%year,S%tps_init%date%month,S%tps_init%date%day)
        call GDATE(S%tps_init%date%Jday,S%tps_init%date%year,S%tps_init%date%month,S%tps_init%date%day)
        call difftime(S%d_t,param%Tzero,S%tps_init)
        ! -------                                               --------      .
        if (abs(S%d_t).lt.60.00_wr) then                                    ! il existe un événement proche (à plus ou moins 1 min)
          i=i+1
          call dellipsgc(param%lat,param%lon,S%Lat,S%Lon,S%d_epi)           ! distance entre les deux epicentres (km)
          S%d_p = param%Zhypo - S%pfd                                       ! différence entre les deux hypocentres (km)
          if (i.le.2) theseisme(i)=S
          find = i
        endif
      endif
    enddo
    if (i.gt.2) then                                                        ! deux séismes max !
      write(*,*)'problème dans catalogue : + de 2 séismes au catalogue'
      find = 2
    endif
```

```
867     close(506)
868     ! ------------------------------------------------------------------     .
869   end subroutine catalogue
870
871     ! ------------------------------------------------------------------     .
872
873   subroutine nomseismefichier(Nom)
874     ! -------                                                 ---------     .mh
875     ! nom des nbseismes fichier (xxxx.xx.xx.xx.xx.d) de données dans DATA/
876     ! sélectionne nbseisme séismes aléatoirement dans ce fichier
877     ! -------                                                 ---------     .
878     use mt19937, only : genrand_real1
879     ! -------                                                 ---------     .
880     implicit none
881     integer (KIND=wi) :: i,j,l,ok
882     integer (KIND=wi) :: deja(nbseismes)
883     integer (KIND=wi) :: nbfile
884     character(len=35) :: Nom(nbseismes)
885     logical :: conv
886     !logical, save :: present=.false.
887     ! ------------------------------------------------------------------     . nombre de lignes
888     !if (present) then
889     !present=.true.
890     open(501, FILE = 'DATA/seismes.d',status='old',iostat = ok)
891     if (ok .ne. 0) then
892       open(501, FILE = 'seismes.d',status='old',iostat = ok)
893       if (ok .ne. 0) then
894         write(*,*)'problème dans nomseismefichier : le fichier DATA/seismes.d n''existe pas '
895         stop
896       endif
897     endif
898     nbfile=0
899     do while(ok .eq. 0)                                             ! boucle pour compter le nombre de lignes du fichier
900       read(501,*,iostat = ok)
901       if (ok .eq. 0) nbfile = nbfile + 1
902     end do
903     ! -------                                                 ---------     .
904     rewind(501)
905     ! -------                                                 ---------     . lecture de nbseisme séismes, sans doublons
906     if (nbfile.eq.nbseismes) then
907       do i=1,nbseismes
908         read(501,*)Nom(i)
909       enddo
910       close(501)
911     ! -------                                                 ---------     .
912     elseif(nbfile.lt.nbseismes) then
913       write(*,*)'problème dans nomseismefichier : nbfile < nb_seismes, manque fichier .dat avec les phases ?'
914       stop
915     ! -------                                                 ---------     . trop de fichier
916     else
917       ! -------                                               ---------     . tire au sort nbseisme sans doublons
918       do i=1,nbseismes
919         l=int(genrand_real1()*real(nbfile,wr)+1.0_wr)                 ! aléatoire de 1 à nbfile
920         if (i==1) then
921           deja(1)=l
922         else
923           conv=.true.
924           do while (conv)
925             conv=.false.
926             do j=1,i-1
927               if (deja(j)==l) then
928                 l=int(genrand_real1()*real(nbfile,wr)+1.0_wr)         ! aléatoire de 1 à nbfile
929                 conv=.true.
930               endif
931             enddo
```

```
932             enddo
933           endif
934           deja(i)=1
935         enddo
936         ! -------                                                 --------    . lecture des nom de fichier
937         do i=1,nbseismes
938           do j=1,deja(i)
939             read(501,*)Nom(i)
940           enddo
941           rewind(501)
942         enddo
943         close(501)
944         ! -------                                                 --------    . réécriture pour après nbfile=nbseismes
945         open(502, FILE = 'DATA/seismes.d',status='replace',iostat = ok)
946         if (ok .ne. 0) then
947           open(502, FILE = 'seismes.d',status='replace')
948         endif
949         do i=1,nbseismes
950           write(502,'(a)')trim(adjustl(Nom(i)))
951         enddo
952         close(502)
953       endif
954       !endif
955       ! ---------------------------------------------------------------    .
956     end subroutine nomseismefichier
957
958 END MODULE datalecture
959
960
961
962 ! ******************************************************************    .
963 ! ******************************************************************    .
```


2.23 SRC/MOD/misfit.f90

```
1  ! Librairie de subroutines concernant le calcul de la fonction coût
2  ! septembre 2013
3  ! ******************************************************************    .
4  ! ------- Méric Haugmard meric.haugmard@univ-nantes.fr            --------    .
5  ! ******************************************************************    .
6  ! ---------------------------------------------------------------    .
7
8  MODULE sub_misfit
9
10    use modparam
11
12    implicit none
13
14    private
15
16    public  :: ponderation
17    public  :: compute_misfitone, compute_misfit
18
19 CONTAINS
20
21    ! ---------------------------------------------------------------    .
22
23   subroutine ponderation (nbtps,datatps,xmin,xmax,w)
24     ! -------                                                 --------    .mh
25     ! calcul de la ponderation des données (distance et qualité)
26     ! -------                                                 --------    .
27     use typetemps
28     ! -------                                                 --------    .
29     implicit none
30     type(dataone), intent (inout) :: datatps(nbtps)                       ! données de temps
```

```
    integer(KIND=wi), intent (in) :: nbtps                                  ! nb de données de temps P et S confondues
    real(KIND=wr), intent (in) :: xmin,xmax                                 ! diamètres cercles pondération
    type(pond), intent (out) :: w                                           ! pondération
    ! ------                                                  ---------     .
    integer(KIND=wi) :: i
    real(KIND=wr) :: one_coef_P, one_coef_S, two_coef                       ! coeficients
    real(KIND=wr) :: a, b                                                   ! droite de régression
    ! ------------------------------------------------------------------    . initialistaion
    w%S_Pg = 0.0_wr                                                         ! pondération
    w%S_Sg = 0.0_wr
    w%S_Pn = 0.0_wr
    w%S_Sn = 0.0_wr
    w%nPg = 0                                                               ! nb de données Pg
    w%nPn = 0                                                               ! nb de données Pn
    w%nSg = 0                                                               ! nb de données Sg
    w%nSn = 0                                                               ! nb de données Sn
    ! ------                                                  ---------     .
    do i=1,nbtps
      ! ------------------------------------------------------------------    .
      ! -------- coeficient 1 : distance épicentrale          ---------     .
      if (FLAGcercles) then
        if (datatps(i)%depi.lt.xmin/2.0_wr) then
          two_coef = 1.0_wr                                                 ! distance faible -> gros coef [1]
        elseif(datatps(i)%depi.gt.xmax/2.0_wr) then                         ! distance forte -> petit coef [0.1]
          two_coef = 0.000001_wr                                            ! très faible mais non nulle -> sinon fonction cout -> NaN (parfois)
        else
          b = (1.0_wr-0.1_wr*(xmin/xmax))/(1.0_wr-(xmin/xmax))
          a = (0.1_wr-b)/(xmax/2.0_wr)
          two_coef = a * datatps(i)%depi + b                                ! entre deux [0.1(loin);1(près)]
        endif
      else
          two_coef = 1.0_wr
      endif
      ! ------------------------------------------------------------------    .
      ! -------- coeficient 2 : qualité des données           ---------     . transforme [0(bon);4(mauvais)] -> [0(mauvais);1(bon)]
      one_coef_P = -0.25_wr*real(datatps(i)%coefP,wr) + 1.0_wr
      one_coef_S = -0.25_wr*real(datatps(i)%coefS,wr) + 1.0_wr
      ! ------------------------------------------------------------------    .
      ! ------                                                  ---------     . ONDE P
      if(datatps(i)%typeonde.eq.'G') then                                   ! si onde directe compressive
        w%nPg = w%nPg+1
        datatps(i)%wp = (one_coef_P * two_coef)
        w%S_Pg = w%S_Pg + datatps(i)%wp                                     ! somme de la pondération pour Pg
      elseif(datatps(i)%typeonde.eq.'N') then                               ! si onde réfractée compressive
        w%nPn = w%nPn+1
        datatps(i)%wp = (one_coef_P * two_coef)
        w%S_Pn = w%S_Pn + datatps(i)%wp                                     ! somme de la pondération pour Pn
      else
        write(*,*)'problème 1 dans ponderation : onde compressive ni directe ni réfractée '
        stop
      endif
      ! ------                                                  ---------     . ONDE S
      if(datatps(i)%andS.eq.'S') then                                       ! si S existe
        if(datatps(i)%typeonde.eq.'G') then                                 ! si onde directe cisaillante
          w%nSg = w%nSg+1
          datatps(i)%ws = (one_coef_S * two_coef)
          w%S_Sg = w%S_Sg + datatps(i)%ws                                   ! somme de la pondération pour Sg
        elseif(datatps(i)%typeonde.eq.'N') then                             ! si onde réfractée cisaillante
          w%nSn = w%nSn+1
          datatps(i)%ws = (one_coef_S * two_coef)
          w%S_Sn = w%S_Sn + datatps(i)%ws                                   ! somme de la pondérationpour Sn
        else
          write(*,*)'problème 2 dans ponderation : onde cisaillante ni directe ni réfractée '
          stop
        endif
```

```
96        endif
97      enddo
98      ! ------------------------------------------------------------------     .
99    end subroutine ponderation
100
101     ! ------------------------------------------------------------------     .
102
103   subroutine compute_misfit (nbtps,D,misfit,xmin,xmax,CorH,div)
104     ! -------                                                 ---------     .mh
105     ! calcul de la fonction coût, pour tous les séismes
106     ! -------                                                 ---------     .
107     use typetemps
108     ! -------                                                 ---------     .
109     implicit none
110     type(dataall), intent (inout) :: D(nbseismes)                           ! données de temps
111     integer(KIND=wi), intent (in) :: nbtps(nbseismes)                       ! nb de données de temps P et S confondues
112     real(KIND=wr), intent (inout) :: xmin(nbseismes),xmax(nbseismes)        ! diamètres cercles pondération
113     real(KIND=wr), intent (out) :: misfit                                   ! valeur de la fonction coût
114     character (LEN=1), intent (in) :: CorH                                  ! cold or Hot runs
115     logical, intent (inout), optional :: div
116     ! -------                                                 ---------     .
117     real(KIND=wr) :: misfit_one
118     integer(KIND=wi) :: i,N
119     ! ------------------------------------------------------------------     .
120     misfit = 0.0_wr
121     do i=1, nbseismes
122       if (present(div)) then
123         ! div : chaine divergente ? -> stop en coldruns
124         call compute_misfitone (nbtps(i),D(i)%datatps,misfit_one,xmin(i),xmax(i),CorH,div)
125       else
126         ! div : chaine divergente ? -> pas de stop en hotruns
127         call compute_misfitone (nbtps(i),D(i)%datatps,misfit_one,xmin(i),xmax(i),CorH)
128       endif
129       misfit = misfit + misfit_one
130     enddo
131     ! -------                                                 ---------     . nombre de données
132     N=0
133     do i=1, nbseismes
134       N = N + nbtps(i)
135     enddo
136     misfit = misfit/real(N,wr)
137     ! ------------------------------------------------------------------     .
138   end subroutine compute_misfit
139
140     ! ------------------------------------------------------------------     .
141
142   subroutine compute_misfitone (nbtps,datatps,misfit,xmin,xmax,CorH,div)
143     ! -------                                                 ---------     .mh
144     ! calcul de la fonction coût, par séisme
145     ! -------                                                 ---------     .
146     use typetemps
147     use pb_direct
148     ! -------                                                 ---------     .
149     implicit none
150     ! -------                                                 ---------     .
151     real(KIND=wr), parameter :: alpha=3.0_wr
152     ! -------                                                 ---------     .
153     type(dataone), intent (inout) :: datatps(nbtps)                         ! données de temps
154     integer(KIND=wi), intent (in) :: nbtps                                  ! nb de données de temps P et S confondues
155     real(KIND=wr), intent (inout) :: xmin,xmax                              ! diamètres cercles pondération
156     real(KIND=wr), intent (out) :: misfit                                   ! valeur de la fonction coût
157     character (LEN=1), intent (in) :: CorH                                  ! cold or Hot runs
158     ! -------                                                 ---------     .
159     logical, intent (inout), optional :: div
160     ! -------                                                 ---------     .
```

```
161     integer(KIND=wi) :: i
162     real(KIND=wr) :: mis_Pg, mis_Sg, mis_Pn, mis_Sn
163     real(KIND=wr) :: moy
164     type(pond) :: w                                                 ! coeficients
165     logical :: FLAGnormale
166     ! -------                                                 --------    .
167     integer(KIND=wi), save :: nbdoublement=0
168     ! ------------------------------------------------------------------    . si divergent :
169 123 continue
170     ! ------------------------------------------------------------------    . initialistaion
171     mis_Pg = 0.0_wr                                                 ! misfits différents par ondes
172     mis_Pn = 0.0_wr
173     mis_Sg = 0.0_wr
174     mis_Sn = 0.0_wr
175     ! -------                                                 --------    .
176     call ponderation (nbtps,datatps,xmin,xmax,w)                    ! calcul de la pondération (qualité des données / distance)
177     ! -------                                                 --------    .
178     do i=1,nbtps
179       ! ------------------------------------------------------------    .
180       ! ------------------------------------------------------------    .
181       FLAGnormale=.true.
182       ! ------------------------------------------------------------    .
183       if(FLAGnormale) then
184         ! ------------------------------------------------------------    .
185         ! les données ont des distributions d'incertitudes de types normales symetriques
186         ! norme L2
187         ! pondérées
188         ! ------------------------------------------------------------    .
189         ! -------                                                 --------    . ONDE P
190         if(datatps(i)%typeonde.eq.'G') then                         ! si onde directe compressive
191
192           mis_Pg = mis_Pg + datatps(i)%wp/2.0_wr * (datatps(i)%dTP/datatps(i)%sigP)**2.0_wr
193
194         elseif(datatps(i)%typeonde.eq.'N') then                     ! si onde réfractée compressive
195
196           mis_Pn = mis_Pn + datatps(i)%wp/2.0_wr * (datatps(i)%dTP/datatps(i)%sigP)**2.0_wr
197
198         else
199           write(*,*)'problème dans compute_misfit : onde compressive ni directe ni réfractée '
200           stop
201         endif
202         if(IsNaN(mis_Pg).or.IsNaN(mis_Pn)) then
203           write(*,*)'problème dans compute_misfit : misfit_P = NaN'
204           stop
205         endif
206         ! ------------------------------------------------------------    .
207         ! -------                                                 --------    . ONDE S
208         if(datatps(i)%andS.eq.'S') then                             ! si S existe
209           if(datatps(i)%typeonde.eq.'G') then                       ! si onde directe cisaillante
210
211             mis_Sg = mis_Sg + datatps(i)%ws/2.0_wr * (datatps(i)%dTS/datatps(i)%sigS)**2.0_wr
212
213           elseif(datatps(i)%typeonde.eq.'N') then                   ! si onde réfractée cisaillante
214
215             mis_Sn = mis_Sn + datatps(i)%ws/2.0_wr * (datatps(i)%dTS/datatps(i)%sigS)**2.0_wr
216
217           else
218             write(*,*)'problème dans compute_misfit : onde cisaillante ni directe ni réfractée '
219             stop
220           endif
221           if(IsNaN(mis_Sg).or.IsNaN(mis_Sn)) then
222             write(*,*)'problème dans compute_misfit : misfit_S = NaN'
223             stop
224           endif
225         endif
```

```
226         ! ------------------------------------------------------------------     .
227         ! ------------------------------------------------------------------     .
228       else
229         ! ------------------------------------------------------------------     .
230         ! les données ont des distributions d'incertitudes de types normales assymetriques
231         ! norme L2
232         ! pondérées
233         ! ------------------------------------------------------------------     .
234         write(*,*)'problème dans compute_misfit : incertitudes de types LOG-normales non codée'
235         stop
236         ! ------------------------------------------------------------------     .
237         moy=100.0_wr
238         ! -------                                          ---------     . ONDE P
239         if(datatps(i)%typeonde.eq.'G') then                             ! si onde directe compressive
240
241           mis_Pg = mis_Pg + datatps(i)%wp*( 1.0_wr/2.0_wr * (datatps(i)%dTP/datatps(i)%sigP)**2.0_wr + &
242             log((1._wr+erf((alpha*datatps(i)%dTP)/(datatps(i)%sigP*sqrt(2.0_wr))))/(datatps(i)%sigP*sqrt(2.0_wr*pi))))
243
244         elseif(datatps(i)%typeonde.eq.'N') then                         ! si onde réfractée compressive
245
246           mis_Pn = mis_Pn + datatps(i)%wp*( 1.0_wr/2.0_wr * (datatps(i)%dTP/datatps(i)%sigP)**2.0_wr + &
247             log((1._wr+erf((alpha*datatps(i)%dTP)/(datatps(i)%sigP*sqrt(2.0_wr))))/(datatps(i)%sigP*sqrt(2.0_wr*pi))))
248
249         else
250           write(*,*)'problème dans compute_misfit : onde compressive ni directe ni réfractée '
251           stop
252         endif
253         if(IsNaN(mis_Pg).or.IsNaN(mis_Pn)) then
254           write(*,*)'problème dans compute_misfit : misfit_P = NaN'
255           stop
256         endif
257         ! ------------------------------------------------------------------     .
258         ! -------                                          ---------     . ONDE S
259         if(datatps(i)%andS.eq.'S') then                                 ! si S existe
260           if(datatps(i)%typeonde.eq.'G') then                           ! si onde directe cisaillante
261
262             mis_Sg = mis_Sg + datatps(i)%ws*( 1.0_wr/2.0_wr * (datatps(i)%dTS/datatps(i)%sigS)**2.0_wr + &
263               log((1._wr+erf((alpha*datatps(i)%dTS)/(datatps(i)%sigS*sqrt(2.0_wr))))/(datatps(i)%sigS*sqrt(2.0_wr*pi))))
264
265           elseif(datatps(i)%typeonde.eq.'N') then                       ! si onde réfractée cisaillante
266
267             mis_Sn = mis_Sn +  datatps(i)%ws*( 1.0_wr/2.0_wr * (datatps(i)%dTS/datatps(i)%sigS)**2.0_wr + &
268               log((1._wr+erf((alpha*datatps(i)%dTS)/(datatps(i)%sigS*sqrt(2.0_wr))))/(datatps(i)%sigs*sqrt(2.0_wr*pi))))
269
270           else
271             write(*,*)'problème dans compute_misfit : onde cisaillante ni directe ni réfractée '
272             stop
273           endif
274           if(IsNaN(mis_Sg).or.IsNaN(mis_Sn)) then
275             write(*,*)'problème dans compute_misfit : misfit_S = NaN'
276             stop
277           endif
278         endif
279         ! ------------------------------------------------------------------     .
280         ! ------------------------------------------------------------------     .
281       endif
282
283       ! ------------------------------------------------------------------     .
284       ! ------------------------------------------------------------------     .
285     enddo
286     ! -------                                          ---------     . calcul de la fonction coût
287     misfit = mis_Pg + mis_Pn + mis_Sg + mis_Sn
288     ! ------------------------------------------------------------------     .
289
290     ! ------------------------------------------------------------------     .
```

```
    ! si divergent, séisme loin de son épicentre : w tend vers 0 et misfit vers NaN
    ! sauf si w, jamais nul car : two_coef = 0.000001_wr
    ! -------                                                  --------     .
    if(IsNaN(misfit).or.(misfit.gt.1.e99_wr)) then                           ! w tends vers 0 (car la loc. est divergente pour ce séisme)
      ! -------                                                --------     . COLDRUNS
      if (CorH.eq."C") then
        if (present(div)) then                                               ! STOP LA CHAINE
          div=.true.
        else
          write(*,*)'problème dans compute_misfit avec "C" : misfit = NaN '
          stop
        endif
        ! -------                                              --------     . HOTRUNS
        elseif (CorH.eq."H") then
          if (present(div)) then                                             ! STOP LA CHAINE
            div=.true.
          else
            if(nbdoublement.lt.nbseismes) then
              write(*,*)'attention dans compute_misfit avec "H" : doublement des cercles de pondération (xmax et xmin) '
              nbdoublement=nbdoublement+1
              xmin=xmin*2._wr
              xmax=xmax*2._wr
            else
              write(*,*)'PB :'
              write(*,*)xmin,xmax
              write(*,*)'problème dans compute_misfit avec "H" : misfit = NaN, Pg ->',mis_Pg,w%S_Pg
              write(*,*)'problème dans compute_misfit avec "H" : misfit = NaN, Pn ->',mis_Pn,w%S_Pn
              write(*,*)'problème dans compute_misfit avec "H" : misfit = NaN, Sg ->',mis_Sg,w%S_Sg
              write(*,*)'problème dans compute_misfit avec "H" : misfit = NaN, Sn ->',mis_Sn,w%S_Sn
              do i=1,nbtps
                write(*,*)datatps(i)
              enddo
              write(*,*)w
              stop
            endif
            go to 123
          endif
        endif
      endif
      ! -------                                                --------     .
      if(xmax.lt.xmin) then
        write(*,*)'problème dans compute_misfit : xmax < xmin :: ', xmax," ? < ? ",xmin
        stop
      endif
      ! ------------------------------------------------------------------     .
    end subroutine compute_misfitone

END MODULE sub_misfit



! ********************************************************************     .
! ********************************************************************     .
```

2.24 SRC/MOD/modparam.f90

```
! Librairies pour le programme CHEs
! septembre 2013
! ********************************************************************     .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
! ********************************************************************     .
! ------------------------------------------------------------------     .

MODULE modparam

```

```
10      implicit none
11
12      private
13
14      public :: wi, wl, wr
15      public :: nbseismes
16      public :: pi, rT, Tminsec
17      public :: FLAGterre, FLAGterrefixe
18      public :: FLAGhypo, FLAGhypofixe
19      public :: FLAGresSTA
20      public :: FLAGcercles
21      public :: plotgraph,plotposteriori
22      public :: libre
23      public :: tracessac
24      public :: autocorr
25      public :: FLAG_non_tabulaire,moho_lon,moho_lat,moho_NS,moho_EO
26      public :: printnbseismes
27
28      ! _______   parametres ForTran precision numérique :       ________    .
29      integer, parameter :: intg = selected_int_kind(r=9)                  ! |int| < 10^9
30      integer, parameter :: long = selected_int_kind(r=18)                 ! |int| < 10^18
31      integer, parameter :: dobl = selected_real_kind(p=15,r=307)          ! |int| < 10^307, 15 chiffres significatifs au moins
32      integer, parameter :: wi = intg
33      integer, parameter :: wl = long
34      integer, parameter :: wr = dobl
35      ! ------------------------------------------------------------------    .
36
37      ! ****************************************************************    .
38      ! ------------------------------------------------------------------    .
39      ! nombre de séismes
40      integer(KIND=wi), parameter :: nbseismes=1
41      ! ------------------------------------------------------------------    .
42      ! ****************************************************************    .
43      ! ------------------------------------------------------------------    .
44      ! _______                                                 ________    . paramètres de terre
45      ! tirage aléatoire selon une loi normale dont la moyenne est fixe :
46      logical, parameter :: FLAGterrefixe=.false.
47      ! prior resserré : (lu dans PARAM/paramTerre.d)
48      logical, parameter :: FLAGterre=.false.
49      ! _______                                                 ________    . paramètres hypocentraux
50      ! tirage aléatoire selon une loi normale dont la moyenne est fixe :
51      logical, parameter :: FLAGhypofixe=.false.
52      ! prior resserré : (lu dans PARAM/paramHypo.d)
53      logical, parameter :: FLAGhypo=.false.
54      ! _______                                                 ________    . résidus
55      ! calcul résidus aux stations (pour chaque modèle sauvé en hotruns,
56      ! les différence temps théoriques-réels sont stockés par types d'onde et
57      ! par stations afin de voir un décalage, ou résidus à la station)
58      logical, parameter :: FLAGresSTA=.true.
59      ! _______                                                 ________    . nombre aléatoire
60      ! générateur aléatoire fixé (libre=vrai)
61      logical, parameter :: libre=.true.
62      ! _______                                                 ________    .
63      ! plot les traces sac sur hodochrones et calcul de la magnitude Ml
64      logical, parameter :: tracessac=.true.
65      ! _______                                                 ________    .
66      ! prise en compte des cercles de pondération xmin et xmax
67      ! attention : parfois l'écipentre se déplace et aucune station se rouve dans le cercle
68      logical, parameter :: FLAGcercles=.true.
69      ! _______                                                 ________    .
70      ! plot les graphs (chaine + autocorr + ... ) pour chaque param
71      logical, parameter :: plotgraph=.true.
72      ! _______                                                 ________    .
73      ! plot les graphs (posteriori )
74      logical, parameter :: plotposteriori=.true.
```

```
    ! -------                                                    --------    .
    ! écart minimal de temps entre deux stations pour la recherche initiale
    real(KIND=wr), parameter :: Tminsec=1.00_wr
    ! -------                                                    --------    .
    ! vrai si moho penche ; faux sinon
    logical, parameter :: FLAG_non_tabulaire=.false.
    ! centre ou est estimée la profondeur du moho
    real(KIND=wr), parameter :: moho_lon=-2.499999999_wr
    real(KIND=wr), parameter :: moho_lat=47.499999999_wr
    ! angle entre 0 (horizontal) et ~90 (vertical), le moho penche positivement vers l'Est et le Sud
    real(KIND=wr), parameter :: moho_NS=0.0_wr
    real(KIND=wr), parameter :: moho_EO=0.0_wr

    ! -------                                                    --------    .
    integer(KIND=wi), parameter :: autocorr=7500                           ! nombre du modèles maximal pour le calcul de l'autocorrelation

    ! ---------------------------------------------------------------    .
    ! ***************************************************************    .
    ! ---------------------------------------------------------------    .

    real(KIND=wr), parameter :: pi = 3.141592653589793238_wr
    real(KIND=wr), parameter :: rT = 6371.0_wr                             ! rayon terrestre moyen

    ! ---------------------------------------------------------------    .
    ! ***************************************************************    .
    ! ---------------------------------------------------------------    .

CONTAINS

  ! ---------------------------------------------------------------------    .

  subroutine printnbseismes
    ! -------                                                    --------    .mh
    ! écriture du nombre de séismes
    ! -------                                                    --------    .
    implicit none
    integer(KIND=wi) :: ok
    ! -------                                                    --------    .
    open(1999, FILE = 'OUTPUT/GMT/nbseisme.d',status='replace',iostat = ok)
    if (ok .ne. 0) then
      write(*,*)'problème dans printnbseismes : le fichier OUTPUT/GMT/nbseisme.d n''existe pas '
      stop
    endif
    ! -------                                                    --------    .
    write(1999,*)nbseismes
    ! -------                                                    --------    .
    close(1999)
    ! ---------------------------------------------------------------    .
  end subroutine printnbseismes

END MODULE modparam



! ********************************************************************    .
! ********************************************************************    .
```

2.25 SRC/MOD/pbdirect.f90

```
! Librairie de subroutines pour le problème direct
! septembre 2013
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! ********************************************************************    .
! ---------------------------------------------------------------    .
```

```
7
8  MODULE pb_direct
9
10     use modparam
11
12     implicit none
13
14     private
15
16     public  :: tempsTheoDirectone, tempsTheoDirect, tempsTheoDirectone_AUTRE
17     public  :: Wadatiplot, chatelainplot
18     public  :: pPn_sSn, reflechie2, reflechie, refracte, refracte_mohovar, directe ! à completer ...
19
20
21 CONTAINS
22
23 ! ------------------------------------------------------------------     .
24
25   subroutine tempsTheoDirect(nbtps,p,D,critique,acentroid)
26     ! ------------------------------------------------------------     .
27     ! Calcul les temps théoriques des arrivées des ondes pour tous les séismes
28     ! ------------------------------------------------------------     .
29     use typetemps
30     ! -------                                              --------     .
31     implicit none
32     integer(KIND=wi), intent (in) :: nbtps(nbseismes)
33     type(parametres), intent (inout) :: p                               ! paramètres d'inv.
34     type(dataall), intent (inout) :: D(nbseismes)                       ! données
35     logical, intent (out) :: critique                                   ! .true. si distance hypo + 5 km < distance hypo critique pour la réfraction
36     type (amoho_centroid), intent (in) :: acentroid
37     ! -------                                              --------     .
38     integer(KIND=wi) :: i,j
39     type(parametre) :: param
40     ! ------------------------------------------------------------     .
41     do i=1, nbseismes
42       do j=1,nbtps(i)
43         call mvPall_2_P1(param,p,i)
44         call tempsTheoDirectone(param,D(i)%datatps(j),critique,acentroid)
45         call mvP1_2_Pall(p,param,i)
46       enddo
47     enddo
48     ! ------------------------------------------------------------     .
49   end subroutine tempsTheoDirect
50
51   ! ------------------------------------------------------------     .
52
53   subroutine tempsTheoDirectone(param,datatemps,critique,acentroid)
54     ! -------                                              --------     .mh
55     ! Calcul les temps théoriques des arrivées des ondes
56     ! pour un couple staion-hypocentre sur un seul séisme
57     ! -------                                              --------     .
58     use typetemps
59     use distance_epi
60     use time
61     ! -------                                              --------     .
62     implicit none
63     type(parametre), intent (in) :: param                               ! paramètres d'inv.
64     type(dataone), intent (inout) :: datatemps                          ! données
65     logical, intent (out) :: critique                                   ! .true. si distance hypo + 5 km < distance hypo critique pour la réfraction
66     type (amoho_centroid), intent (in) :: acentroid
67     ! ------------------------------------------------------------     . calcul de distance épicentrale
68     call dellipsgc(datatemps%sta%lat,datatemps%sta%lon,param%Lat,param%Lon,datatemps%depi,datatemps%baz)
69     ! -------                                              --------     .
70     ! calcul de distance hypocentrale avec prise en compte de l'altitude de la station
71     ! -------                                              --------     . niveau zéro : celui de la mer
```

```
72    datatemps%dhypo = sqrt(datatemps%depi**2.0_wr + (param%Zhypo+datatemps%sta%alti/1000.0_wr)**2.0_wr )
73    ! --------                                                    ---------    .
74    ! calcul de temps de parcours des ondes théoriques et les temps théoriques d'arrivées des ondes
75    ! --------                                                    ---------    .
76    critique=.false.                                                           ! cas normal
77    if (datatemps%typeonde.eq.'G') then                                        ! ondes directes
78      ! --------                                                  ---------    .
79      call directe(param,datatemps%dhypo,datatemps%tpsparcS,datatemps%tpsparcP)
80      ! --------                                                  ---------    .
81    elseif (datatemps%typeonde.eq.'N') then                                    ! ondes réfractées
82      ! if(datatemps%depi.gt.2500.0_wr) datatemps%coefS=max(3,datatemps%coefS)
83      ! car Vp/Vs constant pour le profil, mais peut diverger pour les réfracté à longue distance
84      ! --------                                                  ---------    .
85      if(FLAG_non_tabulaire) then                                              ! moho incliné
86        call refracte_mohovar(acentroid,param,datatemps%sta,datatemps%dhypo,datatemps%tpsparcS,datatemps%tpsparcP, &
87        alti = datatemps%sta%alti, dcritique = datatemps%dcritiqueH)
88      else
89        call refracte(param,datatemps%dhypo,datatemps%tpsparcS,datatemps%tpsparcP, &
90        alti = datatemps%sta%alti, dcritique = datatemps%dcritiqueH)           ! moho non incliné
91      endif
92      ! --------                                                  ---------    .
93      if((datatemps%dhypo+5.0_wr).lt.datatemps%dcritiqueH) then
94        critique=.true. ! distance hypo + 5 km < distance hypo critique pour la réfraction
95      endif
96    else
97      write(*,*)'problème dans tempsTheoDirectone : onde ni directe ni réfractée ... ? ',datatemps%typeonde
98      stop
99    endif
100   ! --------                                                    ---------    .
101   ! calcul du temps d'arrivée théorique absolu
102   ! --------                                                    ---------    .
103   datatemps%tpsTh%date=param%Tzero%date                                      ! même date
104   datatemps%tpsTh%secP = param%Tzero%sec + datatemps%tpsparcP                ! change les secondes
105   datatemps%tpsTh%secS = param%Tzero%sec + datatemps%tpsparcS                ! change les secondes
106   call basetime(datatemps%tpsTh)                                             ! reste en base 60/12/365 ... pour P et S
107   ! --------                                                    ---------    .
108   ! calcul de la difference de temps d'arrivée entre données et modèle
109   ! --------                                                    ---------    .
110   call difftime(datatemps%dTP,datatemps%dTS,datatemps%tpsR,datatemps%tpsTh)
111   if(datatemps%andS.eq.'X') datatemps%dTS=0.0_wr                             ! si il n'existe pas d'ondes S
112   if(IsNaN(datatemps%dTP)) then
113     write(*,*)'problème dans tempsTheoDirectone 1 : datatemps%dTP = NaN'
114     write(*,*)param
115     write(*,*)datatemps
116     stop
117   endif
118   if(IsNaN(datatemps%dTS)) then
119     write(*,*)'problème dans tempsTheoDirectone 2 : datatemps%dTS = NaN'
120     write(*,*)param
121     write(*,*)datatemps
122     stop
123   endif
124   ! ------------------------------------------------------------------    .
125 end subroutine tempsTheoDirectone
126
127   ! ------------------------------------------------------------------    .
128
129 subroutine tempsTheoDirectone_AUTRE(param,datatemps,pdfmoho,critique,m)
130   ! --------                                                    ---------    .mh
131   ! Calcul les temps théoriques des arrivées des ondes
132   ! pour un couple staion-hypocentre sur un seul séisme
133   ! selon un modèle de terre tabulaire à n couches
134   ! --------                                                    ---------    .
135   use typetemps
136   use distance_epi
```

```
    use time
    use ray_tracing
    ! ________                                                    ________    .
    implicit none
    type(parametre), intent (in) :: param                                     ! paramètres d'inv.
    type(dataone), intent (inout) :: datatemps                                ! données
    logical, intent (out) :: critique                                         ! .true. si distance hypo + 5 km < distance hypo critique pour la réfraction
    character (LEN=1), intent(in), optional :: m
    real(KIND=wr), intent (out) :: pdfmoho
    ! ________                                                    ________    .
    real (kind=wr) :: altidudesta,tdirectP,trefP,tdirectS,trefS
    ! ------------------------------------------------------------------    .
    altidudesta=-datatemps%sta%alti/1000._wr
    ! ________                                                    ________    . calcul de distance épicentrale
    call dellipsgc(datatemps%sta%lat,datatemps%sta%lon,param%Lat,param%Lon,datatemps%depi,datatemps%baz)
    ! ________                                                    ________    .
    ! calcul de temps de parcours des ondes théoriques et les temps théoriques d'arrivées des ondes
    ! ________                                                    ________    .
    if(present(m)) then
      call tracerays(datatemps%depi,datatemps%dhypo,datatemps%dcritiqueH,param%Zhypo,altidudesta, &
        param%Lon,param%Lat,pdfmoho,tdirectP,trefP,tdirectS,trefS,m)
    else
      call tracerays(datatemps%depi,datatemps%dhypo,datatemps%dcritiqueH,param%Zhypo,altidudesta, &
        param%Lon,param%Lat,pdfmoho,tdirectP,trefP,tdirectS,trefS)
    endif
    ! ________                                                    ________    .
    critique=.false.                                                          ! cas normal
    ! ________                                                    ________    .
    if (datatemps%typeonde.eq.'G') then                                       ! ondes directes
      datatemps%tpsparcP=tdirectP
      datatemps%tpsparcS=tdirectS
    elseif (datatemps%typeonde.eq.'N') then                                   ! ondes réfractées
      ! if(datatemps%depi.gt.2500.0_wr) datatemps%coefS=max(3,datatemps%coefS)
      ! car Vp/Vs constant pour le profil, mais peut diverger pour les réfracté à longue distance
      datatemps%tpsparcP=trefP
      datatemps%tpsparcS=trefS
      if((datatemps%dhypo+5.0_wr).lt.datatemps%dcritiqueH) then
        critique=.true. ! distance hypo + 5 km < distance hypo critique pour la réfraction
      endif
    else
      write(*,*)'problème dans tempsTheoDirectone_AUTRE : onde ni directe ni réfractée ... ? '
      stop
    endif
    ! ________                                                    ________    .
    ! calcul du temps d'arrivée théorique absolu
    ! ________                                                    ________    .
    datatemps%tpsTh%date=param%Tzero%date                                     ! même date
    datatemps%tpsTh%secP = param%Tzero%sec + datatemps%tpsparcP               ! change les secondes
    datatemps%tpsTh%secS = param%Tzero%sec + datatemps%tpsparcS               ! change les secondes
    call basetime(datatemps%tpsTh)                                            ! reste en base 60/12/365 ... pour P et S
    ! ________                                                    ________    .
    ! calcul de la difference de temps d'arrivée entre données et modèle
    ! ________                                                    ________    .
    call difftime(datatemps%dTP,datatemps%dTS,datatemps%tpsR,datatemps%tpsTh)
    ! ------------------------------------------------------------------    .
    if(datatemps%andS.eq.'X') datatemps%dTS=0.0_wr                            ! si il n'existe pas d'ondes S
    if(IsNaN(datatemps%dTP)) then
      write(*,*)'problème dans tempsTheoDirectone_AUTRE 1 : datatemps%dTP = NaN'
      write(*,*)param
      write(*,*)datatemps
      stop
    endif
    if(IsNaN(datatemps%dTS)) then
      write(*,*)'problème dans tempsTheoDirectone_AUTRE 2 : datatemps%dTS = NaN'
      write(*,*)param
```

```
      write(*,*)datatemps
      stop
    endif
    ! ------------------------------------------------------------------     .
  end subroutine tempsTheoDirectone_AUTRE

    ! ------------------------------------------------------------------     .

  subroutine Wadatiplot(nbtps,D,param_best,vpvs,a,R2,XY,nb,sig,OK,atype)
    ! -------                                                  ---------     .mh
    ! Calcul la regression sur le Wadati plot (Ts-Tp en fonction de Tp-To)
    ! Calcul une bonne estimation de VpVS si les autres paramètres sont déja pas mauvais
    ! wadati (1933)
    ! -------                                                  ---------     .
    use typetemps
    use time
    use statistiques, only : correlationaffpond
    ! -------                                                  ---------     .
    implicit none
    integer(KIND=wi), intent(in) :: nbtps(nbseismes)
    type(dataall), intent(in) :: D(nbseismes)
    type(parametres), intent(in) :: param_best
    ! -------                                                  ---------     .
    integer (KIND=wi), parameter :: taille=2500
    real (KIND=wr), intent(out), optional :: vpvs
    real (KIND=wr), intent(out), optional :: a
    real (KIND=wr), intent(out), optional :: R2
    real (KIND=wr), intent(out), optional :: XY(taille,3)
    real (KIND=wr), intent(out), optional :: sig(taille,2)
    integer (KIND=wi), intent(out), optional :: nb
    logical, intent(out), optional :: OK
    character(len=1), intent(in), optional :: atype
    ! -------                                                  ---------     .
    integer (KIND=wi) :: i,j,n
    type(date_sec) :: one_tps_1,one_tps_2
    real (KIND=wr) :: XY_coef(taille,3),a_coef,R2_coef
    real (KIND=wr), dimension(:,:), allocatable :: XYbis
    real (KIND=wr) :: sig1(taille,2)
    logical :: test,tropval
    ! ------------------------------------------------------------------     . initialisation
    do i=1,taille
      XY_coef(i,1)=-1.0_wr
      XY_coef(i,2)=-1.0_wr
      XY_coef(i,3)=-1.0_wr
      sig1(i,1)=-1.0_wr
      sig1(i,2)=-1.0_wr
    enddo
    ! ------------------------------------------------------------------     .
    n=0
    tropval=.true.
    do j=1,nbseismes
      do i=1,nbtps(j)
        if(tropval) then                                                    ! n < taille=2500
          if(D(j)%datatps(i)%andS.eq.'S') then
           ! -------                                                  ---------     . onde G, N ou les 2
           if(present(atype)) then
             if (D(j)%datatps(i)%typeonde.eq.atype) then
               test = .true.
             else
               test=.false.
             endif
           else
             test = .true.
           endif
           if (test) then
```

```
267                 n=n+1
268                 ! -------                                                  ---------     . onde P
269                 one_tps_1%date = D(j)%datatps(i)%tpsR%date
270                 one_tps_1%sec = D(j)%datatps(i)%tpsR%secP
271                 call difftime(XY_coef(n,1),one_tps_1,param_best%Tzero(j))
272                 ! -------                                                  ---------     . onde S
273                 one_tps_2%date = D(j)%datatps(i)%tpsR%date
274                 one_tps_2%sec = D(j)%datatps(i)%tpsR%secS
275                 call difftime(XY_coef(n,2),one_tps_2,one_tps_1)
276                 ! -------                                                  ---------     . incertitudes sur les données -> pour les figures
277                 sig1(n,1)=D(j)%datatps(i)%sigP
278                 sig1(n,2)=D(j)%datatps(i)%sigS                             ! GMT_wadati prend en compte la propagation de erreurs
279                 ! -------                                                  ---------     .
280                 XY_coef(n,3)= D(j)%datatps(i)%ws * D(j)%datatps(i)%wp
281                 ! -------                                                  ---------     .
282                 if (n.eq.(taille)) then
283                   !do o=1,taille
284                     !write(*,*)o,XY_coef(o,1),XY_coef(o,2),sig1(o,1), sig1(o,2)
285                   !enddo
286                   write(*,*)'problème dans Wadatiplot : trop de données : ',j,i,n
287                   tropval=.false.
288                 endif
289                 ! -------                                                  ---------     .
290               endif
291             endif
292           endif
293         enddo
294       enddo
295       ! -------                                                            ---------     .
296       allocate(XYbis(n,3))
297       do i=1,n
298         XYbis(i,1) = XY_coef(i,1)
299         XYbis(i,2) = XY_coef(i,2)
300         XYbis(i,3) = XY_coef(i,3)
301       enddo
302       ! -------                                                            ---------     .
303       call correlationaffpond(a_coef,R2_coef,n,XYbis)
304       ! -------                                                            ---------     .
305       if (present(a)) a = a_coef
306       if (present(R2)) R2 = R2_coef
307       if (present(vpvs)) vpvs = 1.0_wr+a_coef
308       if (present(XY)) XY=XY_coef
309       if (present(nb)) nb = n
310       if (present(sig)) sig = sig1
311       ! -------                                                            ---------     .
312       deallocate(XYbis)
313       if (n.gt.2) then
314         if (present(OK)) OK = .false.
315       else
316         if (present(OK)) OK = .true.
317       endif
318       ! -------                                                            ---------     .
319       if (present(vpvs)) then
320         if (IsNaN(vpvs)) then
321           write(*,*)'problème dans Wadatiplot : IsNaN(vpvs)',a_coef
322         endif
323       endif
324       ! ------------------------------------------------------------------------------     .
325     end subroutine Wadatiplot
326
327       ! ------------------------------------------------------------------------------     .
328
329     subroutine chatelainplot(nbtps,D,vpvs,a,R2,XY,nb,sig,OK,atype,nom_sta)
330       ! -------                                                            ---------     .mh
331       ! Calcul la regression sur le châtelain plot
```

```
    ! avec : Ts1-Ts2 en fonction de Tp1-Tp2
    ! _______                                                    ________    .
    ! le diagramme de Châtelain (ou Wadati modifié) est indépendant des parametres
    ! et peux ainsi identifier une erreur de pointé sur les ondes
    ! _______                                                    ________    .
    ! Châtelain (1978) : "Etude fine de la sismicité en zone de collision continentale au moyen
    ! d'un réseau de stations portables : la région Hindu-Kush Pamir" these de doctorat
    ! _______                                                    ________    .
    use typetemps
    use time
    use statistiques, only : correlationaffpond
    ! _______                                                    ________    .
    implicit none
    integer(KIND=wi), intent(in) :: nbtps(nbseismes)
    type(dataall), intent(in) :: D(nbseismes)
    ! _______                                                    ________    .
    integer (KIND=wi), parameter :: taille=5000
    ! _______                                                    ________    .
    real (KIND=wr), intent(out), optional :: vpvs
    real (KIND=wr), intent(out), optional :: a
    real (KIND=wr), intent(out), optional :: R2
    real (KIND=wr), intent(out), optional :: XY(taille,3)
    real (KIND=wr), intent(out), optional :: sig(taille,2)
    integer (KIND=wi), intent(out), optional :: nb
    logical, intent(out), optional :: OK
    character(len=1), intent(in), optional :: atype
    character(len=4), intent(out), optional :: nom_sta(taille,2)
    ! _______                                                    ________    .
    integer (KIND=wi) :: i,j,k,l,m,n
    type(date_sec) :: one_tps_1,one_tps_2
    real (KIND=wr) :: val1,val2,XY_coef(taille,3),a_coef,R2_coef
    real (KIND=wr), dimension(:,:), allocatable :: XYbis
    real (KIND=wr) :: sig1(taille,2)
    logical :: test, deja, tropval
    ! ____________________________________________________________    .
    if(present(nom_sta)) nom_sta(:,:)='123_'
    ! _______                                                    ________    .
    do i=1,taille
      XY_coef(i,1)=-1.0_wr
      XY_coef(i,2)=-1.0_wr
      XY_coef(i,3)=-1.0_wr
      sig1(i,1)=-1.0_wr
      sig1(i,2)=-1.0_wr
    enddo
    n=0
    tropval=.true.
    do j=1,nbseismes
      do i=1,nbtps(j)
        do k=1,nbtps(j)
          if(tropval) then                                          ! n < taille=1000

            ! _______                                              ________    . onde G, N ou les 2
            if(present(atype)) then
              if ((D(j)%datatps(i)%typeonde.eq.atype).and.(D(j)%datatps(k)%typeonde.eq.atype)) then
                test = .true.
              else
                test=.false.
              endif
            else
              if (D(j)%datatps(i)%typeonde.eq.D(j)%datatps(k)%typeonde) then
                test = .true.
              else
                test = .false.
              endif
            endif
```

```
397             if ((test).and.(D(j)%datatps(i)%andS=='S').and.(D(j)%datatps(k)%andS=='S')) then
398               ! -------                                                 --------     . onde P et S
399               n=n+1
400               one_tps_1%date = D(j)%datatps(i)%tpsR%date
401               one_tps_1%sec = D(j)%datatps(i)%tpsR%secP
402               one_tps_2%date = D(j)%datatps(k)%tpsR%date
403               one_tps_2%sec = D(j)%datatps(k)%tpsR%secP
404               call difftime(val1,one_tps_1,one_tps_2)
405               one_tps_1%date = D(j)%datatps(i)%tpsR%date
406               one_tps_1%sec = D(j)%datatps(i)%tpsR%secS
407               one_tps_2%date = D(j)%datatps(k)%tpsR%date
408               one_tps_2%sec = D(j)%datatps(k)%tpsR%secS
409               call difftime(val2,one_tps_1,one_tps_2)
410               ! -------                                                 --------     . déja présent ?
411               deja=.true.
412               do l=1,n
413                 if((abs(val1)==XY_coef(l,1)).and.(abs(val2)==XY_coef(l,2))) then
414                   deja=.false.
415                 endif
416               enddo
417               if(deja)then
418                 XY_coef(n,1)=abs(val1)
419                 XY_coef(n,2)=abs(val2)
420                 ! -------                                           --------     . incertitudes sur les données -> pour les figures
421                 if((abs(val1)==0.0_wr).and.(abs(val2)==0.0_wr)) then
422                   sig1(n,1)=0.0_wr
423                   sig1(n,2)=0.0_wr
424                 else
425                   sig1(n,1)=sqrt(D(j)%datatps(i)%sigP**2.0_wr+D(j)%datatps(k)%sigP**2.0_wr)
426                   sig1(n,2)=sqrt(D(j)%datatps(i)%sigS**2.0_wr+D(j)%datatps(k)%sigS**2.0_wr)
427                 endif
428                 ! -------                                                 --------     .
429                 XY_coef(n,3)=(D(j)%datatps(i)%ws+D(j)%datatps(i)%wp+D(j)%datatps(k)%ws+D(j)%datatps(k)%wp)/4.0_wr
430                 if(present(nom_sta)) then
431                   nom_sta(n,1)=D(j)%datatps(i)%sta%staname
432                   nom_sta(n,2)=D(j)%datatps(k)%sta%staname
433                 endif
434               else
435                 n=n-1
436               endif
437               ! -------                                                 --------     .
438               if (n.eq.(taille)) then
439                 write(*,*)'problème dans chatelainplot : trop de données : ',j,i,m,k,n
440                 !do l=1,taille
441                   !write(*,*)l,XY_coef(l,1),XY_coef(l,2),sig1(l,1), sig1(l,2)
442                 !enddo
443                 tropval=.false.
444               endif
445               ! -------                                                 --------     .
446             endif
447           endif
448         enddo
449       enddo
450     enddo
451     ! -------                                                           --------     .
452     allocate(XYbis(n,3))
453     do i=1,n
454       XYbis(i,1) = XY_coef(i,1)
455       XYbis(i,2) = XY_coef(i,2)
456       XYbis(i,3) = XY_coef(i,3)
457     enddo
458     ! -------                                                           --------     .
459     call correlationaffpond(a_coef,R2_coef,n,XYbis)
460     ! -------                                                           --------     .
461     if (present(a)) a = a_coef
```

```
462     if (present(R2)) R2 = R2_coef
463     if (present(vpvs)) vpvs = a_coef
464     if (present(XY)) XY=XY_coef
465     if (present(nb)) nb = n
466     if (present(sig)) sig = sig1
467     ! _______                                                    ________    .
468     deallocate(XYbis)
469     if (n.gt.2) then
470       if (present(OK)) OK = .false.
471     else
472       if (present(OK)) OK = .true.
473     endif
474     ! _______                                                    ________    .
475     if (present(vpvs)) then
476       if (IsNaN(vpvs)) then
477         write(*,*)'problème dans chatelainplot : IsNaN(vpvs)',a_coef
478       endif
479     endif
480     ! ------------------------------------------------------------------    .
481   end subroutine chatelainplot
482
483     ! ------------------------------------------------------------------    .
484
485   subroutine directe(param,dishypo,Tps,Tpp)
486     ! _______                                                    ________    .mh
487     ! calcul du temps d'arrivée des ondes P et S directes (Pg et Sg)
488     ! pour 2 distances hypocentrales
489     ! modèle tabulaire
490     ! VpVs constant le long du profile
491     ! pas de séismes sous le moho
492     ! _______                                                    ________    .
493     use typetemps, only : parametre
494     ! _______                                                    ________    .
495     implicit none
496     type(parametre), intent(in) :: param                                    ! parametres du modèle
497     real(kind=wr), intent(in) :: dishypo                                    ! distance hypocentrale
498     real(kind=wr), intent(out) :: Tps,Tpp                                   ! temps des ondes Pg et Sg
499     ! ------------------------------------------------------------------    .
500     if(param%Zhypo.lt.(param%Zmoho-0.1_wr)) then                            ! séisme au dessus du moho !
501       Tps= dishypo/param%VC*param%VpVs
502       Tpp= dishypo/param%VC
503     else
504       Tps= 1.e9_wr
505       Tpp= 1.e9_wr
506     endif
507     ! ------------------------------------------------------------------    .
508   end subroutine directe
509
510     ! ------------------------------------------------------------------    .
511
512   subroutine refracte(param,dishypo,Tps,Tpp,alti,dcritique)
513     ! _______                                                    ________    .mh
514     ! calcul du temps d'arrivée des ondes P et S réfractées (Pn et Sn)
515     ! pour une distance hypocentrale
516     ! modèle tabulaire
517     ! VpVs constant le long du profile
518     ! _______                                                    ________    .
519     use typetemps, only : parametre
520     ! _______                                                    ________    .
521     implicit none
522     type(parametre), intent(in) :: param                                    ! parametres du modèle
523     real(kind=wr), intent(in) :: dishypo                                    ! distance hypocentrale
524     real(kind=wr), intent(out) :: Tps,Tpp                                   ! temps des ondes Pn et Sn
525     real(kind=wr), intent(in), optional :: alti                             ! altitude de la station (m)
526     real(kind=wr), intent(out), optional :: dcritique                       ! distance hypocentrale critique (à partir de laquelle les premières réfractions ont
```

```
        lieu)
    ! -------                                                    --------     .
    real(kind=wr) :: anglei,depi,dist_c,dist_m,altista,dc
    ! ------------------------------------------------------------------     .
    if(param%Zhypo.lt.(param%Zmoho-0.1_wr)) then                               ! séisme au dessus du moho !
      anglei = asin(param%VC/param%VM)
      if(IsNaN(anglei)) then
        write(*,*)'problème dans refracte : paramètre de rai = NaN car VC > VM '
        stop
      endif
      ! -------                                                  --------     .
      if (present(alti)) then
        altista = alti/1000.0_wr
      else
        altista = 0.0_wr
      endif
      depi = sqrt(dishypo**2.0_wr - (param%Zhypo+altista)**2.0_wr)
      ! -------                                                  --------     .
      dc = (param%Zmoho-param%Zhypo)*tan(anglei) + &
         (param%Zmoho+altista)*tan(anglei)
      dist_m = depi - dc
      dist_c = (param%Zmoho-param%Zhypo)/cos(anglei) + &
             (param%Zmoho+altista)/cos(anglei)
      Tpp = dist_c/param%VC + dist_m/param%Vm
      Tps = param%VpVs*Tpp
      dc = sqrt(dc**2.0_wr + (param%Zhypo+altista)**2.0_wr )
      ! -------                                                  --------     .
      if (present(dcritique)) dcritique = dc
    else
      Tps= 1.e9_wr
      Tpp= 1.e9_wr
      if (present(dcritique)) dcritique = 1.e9_wr
    endif
    ! ------------------------------------------------------------------     .
  end subroutine refracte

    ! ------------------------------------------------------------------     .

  subroutine refracte_mohovar(mohocentroid,param,sta,dishypo,Tps,Tpp,alti,dcritique,pfd)
    ! -------                                                    --------     .mh
    ! calcul du temps d'arrivée des ondes P et S réfractées (Pn et Sn)
    ! pour une distance hypocentrale
    ! -> modèle non tabulaire avec mohocentroid
    ! VpVs constant le long du profile
    ! -------                                                  --------     .
    use typetemps
    use distance_epi
    ! -------                                                  --------     .
    implicit none
    type (amoho_centroid), intent (in) :: mohocentroid
    type(parametre), intent(in) :: param                                   ! parametres du modèle
    type(stations), intent (in) :: sta
    real(kind=wr), intent(in) :: dishypo                                   ! distance hypocentrale
    real(kind=wr), intent(out) :: Tps,Tpp                                  ! temps des ondes Pn et Sn
    real(kind=wr), intent(in), optional :: alti                            ! altitude de la station (m)
    real(kind=wr), intent(out), optional :: dcritique                      ! distance hypocentrale critique (à partir de laquelle les premières réfractions ont
        lieu)
    real(kind=wr), intent(out), optional :: pfd                            ! profondeur du moho sous le séisme
    ! -------                                                  --------     .
    real(kind=wr) :: anglei,depi,dist_c,dist_m,altista,dc
    real(kind=wr) :: pfdmohoSTA,pfdmohoEvent,angleap,pfd1,pfd2
    real(kind=wr) :: dSTAlon,dSTAlat,dEVlon,dEVlat
    ! ------------------------------------------------------------------     .
    anglei = asin(param%VC/param%VM)
    if(IsNaN(anglei)) then
```

```
590       write(*,*)'problème dans refracte_mohovar : paramètre de rai = NaN car VC > VM '
591       stop
592     endif
593     ! -------                                                  --------     .
594     if (present(alti)) then
595       altista = alti/1000.0_wr
596     else
597       altista = 0.0_wr
598     endif
599     depi = sqrt(dishypo**2.0_wr - (param%Zhypo+altista)**2.0_wr)
600     ! -------                                                  --------     .
601     ! vecteur normal du moho (lon, lat, z) en km
602     call dellipsgc(mohocentroid%latC,mohocentroid%lonC,sta%lat,mohocentroid%lonC,dSTAlat)
603     if (sta%Lat.gt.mohocentroid%latC) dSTAlat=-dSTAlat
604     call dellipsgc(mohocentroid%latC,mohocentroid%lonC,mohocentroid%latC,sta%lon,dSTAlon)
605     if (mohocentroid%lonC.gt.sta%Lon) dSTAlon=-dSTAlon
606     ! prod scalaire : sta(dSTAlon,dSTAlat,pfdmohoSTA) . mohocentroid(alpha,beta,gamma) = 0
607     pfdmohoSTA=param%Zmoho-dSTAlon*mohocentroid%alph/mohocentroid%gamma-dSTAlat*mohocentroid%beta/mohocentroid%gamma
608     call dellipsgc(mohocentroid%latC,mohocentroid%lonC,param%Lat,mohocentroid%lonC,dEVlat)
609     if (param%Lat.gt.mohocentroid%latC) dEVlat=-dEVlat
610     call dellipsgc(mohocentroid%latC,mohocentroid%lonC,mohocentroid%latC,param%Lon,dEVlon)
611     if (mohocentroid%lonC.gt.param%Lon) dEVlon=-dEVlon
612     ! prod scalaire : event(dEVlon,dEVlat,0) . mohocentroid(alpha,beta,gamma) = 0
613     pfdmohoEvent=param%Zmoho-dEVlon*mohocentroid%alph/mohocentroid%gamma-dEVlat*mohocentroid%beta/mohocentroid%gamma
614     ! -------                                                  --------     .
615     angleap=atan(abs(pfdmohoEvent-pfdmohoSTA)/depi)                         ! pendage apparent du moho
616     ! -------                                                  --------     .
617     pfd1=(pfdmohoEvent-param%Zhypo)*cos(angleap)                            ! distance entre moho et séisme, perpendiculaire au moho
618     pfd2=(pfdmohoSTA+altista)*cos(angleap)                                  ! distance entre moho et station, perpendiculaire au moho
619     dc = pfd1*tan(anglei) + pfd2*tan(anglei)                                ! distance parcourue dans la croûte projetée sur le moho
620     dist_m = depi/cos(angleap) - dc                                         ! distance parcourue dans le manteau
621     dist_c = pfd1/cos(anglei) + pfd2/cos(anglei)                            ! distance parcourue dans la croûte
622     Tpp = dist_c/param%VC + dist_m/param%Vm
623     Tps = param%VpVs*Tpp
624     dc = sqrt(dc**2.0_wr + (param%Zhypo+altista)**2.0_wr) * cos(anglei)     ! distance dcritique pour les premieres réfractées
625     ! -------                                                  --------     .
626     if (present(dcritique)) dcritique = dc
627     if (present(pfd)) pfd = pfdmohoEvent
628     ! ------------------------------------------------------------------     .
629   end subroutine refracte_mohovar
630
631     ! ------------------------------------------------------------------     .
632
633   subroutine reflechie(param,dishypo,Tps,Tpp,alti)
634     ! -------                                                  --------     .mh
635     ! calcul du tmps d'arrivée des ondes P et S réflechies (PmP et SmS)
636     ! pour une distance hypocentrale
637     ! modèle tabulaire
638     ! VpVs constant le long du profile
639     ! -------                                                  --------     .
640     use typetemps, only : parametre
641     ! -------                                                  --------     .
642     implicit none
643     type(parametre), intent(in) :: param                                    ! parametres du modèle
644     real(kind=wr), intent(in) :: dishypo                                    ! distance hypocentrale
645     real(kind=wr), intent(out) :: Tps,Tpp                                   ! temps des ondes PmP et SmS
646     real(kind=wr), intent(in), optional :: alti                             ! altitude de la station (m)
647     ! -------                                                  --------     .
648     real(kind=wr) :: depi,dc1,dc2, altista
649     ! ------------------------------------------------------------------     .
650     if (present(alti)) then
651       altista = alti/1000.0_wr
652     else
653       altista = 0.0_wr
654     endif
```

```
    ! _______                                                    ________    .
    if((param%Zhypo+altista).lt.dishypo) then
      depi = sqrt(dishypo**2.0_wr-(param%Zhypo+altista)**2.0_wr)
      ! _______                                                  ________    .
      dc1 = depi * 1.0_wr / &
            ((param%Zmoho-param%Zhypo) / (param%Zmoho+altista) + 1.0_wr)
      dc2 = depi - dc1
      dc1 = sqrt(dc1**2.0_wr + (param%Zmoho-param%Zhypo)**2.0_wr)
      dc2 = sqrt(dc2**2.0_wr + (param%Zmoho+altista)**2.0_wr)
      Tpp = (dc1 + dc2) / param%VC
      Tps= Tpp*param%VpVs
    else
      write(*,*)'problème dans reflechie : réflexion impossible '
    endif
    ! ------------------------------------------------------------    .
  end subroutine reflechie

    ! ------------------------------------------------------------    .

  subroutine reflechie2(param,dishypo,Tps,Tpp,alti)
    ! _______                                                    ________    .mh
    ! calcul du tmps d'arrivée des ondes P et S réflechies (PmP2 et SmS2)
    ! pour une distance hypocentrale
    ! modèle tabulaire
    ! VpVs constant le long du profile
    ! _______                                                    ________    .
    use typetemps, only : parametre
    ! _______                                                    ________    .
    implicit none
    type(parametre), intent(in) :: param                                    ! parametres du modèle
    real(kind=wr), intent(in) :: dishypo                                    ! distance hypocentrale
    real(kind=wr), intent(out) :: Tps,Tpp                                   ! temps des ondes PmP2 et SmS2
    real(kind=wr), intent(in), optional :: alti                             ! altitude de la station (m)
      ! _______                                                  ________    .
    real(kind=wr) :: depi,dc1,dc2,altista
    ! ------------------------------------------------------------    .
    if (present(alti)) then
      altista = alti/1000.0_wr
    else
      altista = 0.0_wr
    endif
    ! _______                                                    ________    .
    if((param%Zhypo+altista).lt.dishypo) then
      depi = sqrt(dishypo**2.0_wr-(param%Zhypo+altista)**2.0_wr)
      ! _______                                                  ________    .
      dc1 = depi * 1.0_wr / ((param%Zmoho-param%Zhypo) / &
            (3.0_wr*(param%Zmoho+altista))+1.0_wr)
      dc2 = (depi - dc1)/3.0_wr
      dc1 = sqrt(dc1**2.0_wr+(param%Zmoho-param%Zhypo)**2.0_wr)
      dc2 = sqrt(dc2**2.0_wr+(param%Zmoho+altista)**2.0_wr)
      Tpp = (dc1 + 3.0_wr*dc2)/param%VC
      Tps= Tpp*param%VpVs
    else
      write(*,*)'problème dans reflechie2 : réflexion impossible '
    endif
    ! ------------------------------------------------------------    .
  end subroutine reflechie2

    ! ------------------------------------------------------------    .

  subroutine pPn_sSn(param,dishypo,Tps,Tpp,alti,dcritique)
    ! _______                                                    ________    .mh
    ! calcul du temps d'arrivée des ondes P et S réfractées (pPn et sSn)
    ! pour une distance hypocentrale
    ! modèle tabulaire
```

```
    ! VpVs constant le long du profile
    ! -------                                                    ---------    .
    use typetemps, only : parametre
    ! -------                                                    ---------    .
    implicit none
    type(parametre), intent(in) :: param                                    ! parametres du modèle
    real(kind=wr), intent(inout) :: dishypo                                 ! distance hypocentrale
    real(kind=wr), intent(out) :: Tps,Tpp                                   ! temps des ondes pPn et sSn
    real(kind=wr), intent(in), optional :: alti                             ! altitude de la station (m)
    real(kind=wr), intent(out), optional :: dcritique                       ! distance hypocentrale critique (à partir de laquelle les premières réfractions ont
        lieu)
    ! -------                                                    ---------    .
    real(kind=wr) :: anglei,depi,dist_c,dist_m,altista,dc
    ! ------------------------------------------------------------------    .
    if (present(alti)) then
      altista = alti/1000.0_wr
    else
      altista = 0.0_wr
    endif
    ! ------------------------------------------------------------------    .
    anglei = asin(param%VC/param%VM)
    if(IsNaN(anglei)) then
      write(*,*)'problème dans pPn_pSn : paramètre de rai = NaN car VC > VM '
      stop
    endif
      ! -------                                                  ---------    .
    if (dishypo==0.0_wr) then
      depi = tan(anglei)*param%Zhypo + (2.0_wr*param%Zmoho+1.5_wr*altista)*tan(anglei)
      dishypo = sqrt(depi**2.0_wr + param%Zhypo**2.0_wr)
    else
      depi = sqrt(dishypo**2.0_wr-(param%Zhypo)**2.0_wr)
    endif
      ! -------                                                  ---------    .
    dc = tan(anglei)*param%Zhypo + (2.0_wr*param%Zmoho+1.5_wr*altista)*tan(anglei)
    dist_m = depi  - dc
    dist_c = (param%Zhypo)/cos(anglei) + (2.0_wr*param%Zmoho+1.5_wr*altista)/cos(anglei)
    Tpp = dist_c/param%VC + dist_m/param%Vm
    Tps = param%VpVs*Tpp
    dc = sqrt(dc**2.0_wr + (param%Zhypo+altista)**2.0_wr )
    if (present(dcritique)) dcritique = dc
    ! ------------------------------------------------------------------    .
  end subroutine pPn_sSn

END MODULE pb_direct



! ********************************************************************    .
! ********************************************************************    .
```

2.26 MAC SRC/MOD/printmess.f90

```
! Librairie de subroutines permettant l'impression de messages à l'écran
! septembre 2013
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .

MODULE affiche

    use modparam

    implicit none

```

```
14      private
15
16      public  :: print_mess_1,print_mess_2,print_mess_2bis,print_mess_3, &
17                 print_mess_3bis,print_mess_4,print_mess_5
18      public  :: print_mess_fin
19      public  :: print_mess_finchainemin
20      public  :: print_mess_finchainemax
21      public  :: print_line
22      public  :: print_messchaine
23
24
25  CONTAINS
26
27  !-----------------------------------------------------------------------      .
28
29
30    subroutine print_mess_1
31      !-------                                                 --------      .mh
32      implicit none                                                         ! début du programme che_coldruns_init
33      write(*,*)
34      write(*,*)'-------------------------------------------------------'
35      write(*,*)'----------------- CHE2016 version 1.6 -----------------'
36      write(*,*)'-------------------------------------------------------'
37      write(*,*)'                         meric.haugmard@univ-nantes.fr'
38      write(*,*)'                                      méric Haugmard'
39      write(*,*)'                                           2013-2016'
40      write(*,*)
41      write(*,*)'-------------  initialisation coldruns  -------------' ;
42      write(*,*)
43      !-----------------------------------------------------------------------      .
44    end subroutine print_mess_1
45
46  !-----------------------------------------------------------------------      .
47
48    subroutine print_mess_fin
49      !-------                                                 --------      .mh
50      implicit none                                                         ! fin du programme che_plot
51      write(*,*)
52      write(*,*)'-------------------------------------------------------'
53      write(*,*)'---------------------  fin prog  ----------------------'
54      write(*,*)'-------------------------------------------------------'
55      write(*,*)
56      !-----------------------------------------------------------------------      .
57    end subroutine print_mess_fin
58
59  !-----------------------------------------------------------------------      .
60
61    subroutine print_mess_2
62      !-------                                                 --------      .mh
63      implicit none                                                         ! début McMC
64      write(*,*)
65      write(*,*)'-------------------  début coldruns  -------------------'
66      write(*,*)
67      !-----------------------------------------------------------------------      .
68    end subroutine print_mess_2
69
70  !-----------------------------------------------------------------------      .
71
72    subroutine print_mess_2bis
73      !-------                                                 --------      .mh
74      implicit none                                                         ! début McMC
75      write(*,*)
76      write(*,*)'-------------------  début hotruns  -------------------'
77      write(*,*)
78      !-----------------------------------------------------------------------      .
```

```
  end subroutine print_mess_2bis

! ------------------------------------------------------------------------     .

  subroutine  print_mess_finchainemin(a,b)
    ! -------                                                 ---------     .mh
    ! fin de chaque chaîne - première étape
    implicit none
    real(KIND=wr) :: a,b
    ! ------------------------------------------------------------------     .
    write(*,'(a46,f7.2)')' fonction coût minimale         :              ',a
    write(*,'(a43,f7.2,a2)')' acceptance                  :              ',b,' %'
    ! ------------------------------------------------------------------     .
  end subroutine print_mess_finchainemin

! ------------------------------------------------------------------------     .

  subroutine  print_mess_finchainemax(a,b,c)
    ! -------                                                 ---------     .mh
    ! fin de chaque chaîne - seconde étape
    implicit none
    real(KIND=wr) :: a,b
    integer(KIND=wi) :: c
    ! ------------------------------------------------------------------     .
    write(*,'(a40,i15)')' nombre de modèles sélectionnés :     ',c
    write(*,'(a46,f7.2)')' fonction coût minimale         :              ',a
    write(*,'(a43,f7.2,a2)')' acceptance                  :              ',b,' %'
    ! ------------------------------------------------------------------     .
  end subroutine print_mess_finchainemax

! ------------------------------------------------------------------------     .

  subroutine print_mess_3
    ! -------                                                 ---------     .mh
    implicit none                                                           ! fin McMC
    write(*,*)
    write(*,*)'--------------------  fin coldruns  --------------------'
    write(*,*)
    ! ------------------------------------------------------------------     .
  end subroutine print_mess_3

! ------------------------------------------------------------------------     .

  subroutine print_mess_3bis
    ! -------                                                 ---------     .mh
    implicit none                                                           ! fin McMC
    write(*,*)
    write(*,*)'--------------------  fin hotruns  --------------------'
    write(*,*)
    ! ------------------------------------------------------------------     .
  end subroutine print_mess_3bis

! ------------------------------------------------------------------------     .

  subroutine print_mess_4
    ! -------                                                 ---------     .mh
    implicit none                                                           ! calcul des moyennes
    write(*,*)
    write(*,*)'----------------- calcul a posteriori -----------------'
    write(*,*)
    ! ------------------------------------------------------------------     .
  end subroutine print_mess_4

! ------------------------------------------------------------------------     .

```

```
  subroutine print_mess_5
    ! -------                                                  ---------     .mh
    implicit none                                                           ! production des script pour les figures
    write(*,*)
    write(*,*)'------ production des script pour les figures -------'
    write(*,*)
    ! ------------------------------------------------------------------     .
  end subroutine print_mess_5

! ------------------------------------------------------------------------     .

  subroutine print_line
    ! -------                                                  ---------     .mh
    implicit none                                                           ! entre chaque chaîne
    write(*,*)
    write(*,*)'------------------------------------------------------------'
    write(*,*)
    ! ------------------------------------------------------------------     .
  end subroutine print_line

! ------------------------------------------------------------------------     .

  subroutine print_messchaine(i,n)
    ! -------                                                  ---------     .mh
    ! en début de chaîne
    implicit none
    integer(KIND=wi), intent(in) :: i,n
    ! ------------------------------------------------------------------     .
    write(*,*)
    write(*,*)'-- chaîne numéro :    ',i,'/',n,' --'
    ! ------------------------------------------------------------------     .
  end subroutine print_messchaine

END MODULE affiche


! ************************************************************************     .
! ************************************************************************     .
```

2.27 MAC SRC/MOD/rechercheinit.f90

```
! Subroutines permettant la réduction du prior pour lon/lat
! juillet 2014
! ************************************************************************     .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------     .
! ************************************************************************     .
! ------------------------------------------------------------------------     .

MODULE rechercheepi

    use modparam

    implicit none

    private

    public  :: zoneRecherche, initparam

CONTAINS

! ------------------------------------------------------------------------     .


  subroutine initparam(nbtps,D,param_init,pEpis,nb)
```

```
24      ! -------                                                    ---------     .mh
25      ! initialise les paramertres hypocentraux et le modèle de Terre
26      ! -------                                                    ---------     .
27      ! VC entre min et max
28      ! VM entre min et max, avec VM > VC + 0.1
29      ! Zmoho entre min et max
30      ! vPvS entre min et max
31      ! -------                                                    ---------     .
32      ! lon,lat -> méthodes des hémisphères, tiré dans un prior restreint par zoneRecherche
33      ! Zhypo entre min et max
34      ! Tzero correspond à lon,lat,VC et Zhypo
35      ! -------                                                    ---------     .
36      use typetemps
37      use time, only : basetime, difftime
38      use mt19937
39      use distance_epi
40      ! -------                                                    ---------     .
41      implicit none
42      ! -------                                                    ---------     .
43      integer(KIND=wi), intent (in) :: nbtps(nbseismes)                          ! nombre station et nombre de données de temps par séismes
44      type(dataall), intent (inout) :: D(nbseismes)                              ! données de temps
45      type(parametres), intent (out) :: param_init                               ! paramètres
46      type(priorEPI), intent (inout) :: pEpis(nbseismes)
47      integer(KIND=wi), intent (in) :: nb                                        ! au carré, nb de cases possible
48      ! -------                                                    ---------     .
49      integer(KIND=wi) :: i,j
50      integer(KIND=wi), save :: ok=999
51      real(KIND=wr), save :: mini_Vc,maxi_Vc,mini_Vm,maxi_Vm,mini_Zmoho
52      real(KIND=wr), save :: maxi_Zmoho,mini_VpVs,maxi_VpVs,mini_Zhypo,maxi_Zhypo
53      real(KIND=wr) :: ec,moy,val,depi,dhypo
54      type(date_sec) :: a_ref_temps,a_time
55      real(KIND=wr) :: sommecoef,a_coef
56      ! ------------------------------------------------------------------     . lus dans le prior (PARAM/priorIn.d)
57      if(ok==999)then                                                            ! lecture unique. -> save
58        ok=0
59        open(975, FILE = 'PARAM/priorIn_COLD.d',status='old',iostat = ok)
60        if (ok .ne. 0) then
61          write(*,*)'problème dans initparam : le fichier PARAM/priorIn_COLD.d n''existe pas '
62          stop
63        endif
64        ! -------                                                  ---------     .
65        read(975,*)mini_Vc,maxi_Vc,ec
66        read(975,*)mini_Vm,maxi_Vm,ec
67        read(975,*)mini_Zmoho,maxi_Zmoho,ec
68        read(975,*)mini_VpVs,maxi_VpVs,ec
69        read(975,*)mini_Zhypo,maxi_Zhypo,ec
70        close(975)
71      endif
72      ! ------------------------------------------------------------------     .
73      ! -------- parametres du modèle de terre                       ----     .
74      ! vitesse dans la croûte des ondes P
75      param_init%VC = (maxi_Vc-mini_Vc) * genrand_real3() + mini_Vc
76      ! vitesse dans le manteau des ondes P
77      val=mini_Vm
78      if(mini_Vm.lt.(param_init%VC+0.1_wr)) val = param_init%VC+0.1_wr           ! force VC < VM (réfracction)
79      param_init%VM = (maxi_Vm-val) * genrand_real3() + val
80      ! profondeur du moho
81      param_init%Zmoho = (maxi_Zmoho-mini_Zmoho) * genrand_real3() + mini_Zmoho
82      ! ratio de vitesse
83      param_init%VpVs = (maxi_VpVs-mini_VpVs) * genrand_real3() + mini_VpVs
84      do i=1,nbseismes
85        ! ------------------------------------------------------------------     .
86        ! -------- parametres hypocentraux                           ----     .
87        ! profondeur du séisme
88        val=maxi_Zhypo
```

```
89      if(maxi_Zhypo.gt.(param_init%Zmoho-0.1_wr)) val = param_init%Zmoho-0.1_wr  ! force Zhypo < Zmoho
90      param_init%Zhypo(i) = (val-mini_Zhypo) * genrand_real3() + mini_Zhypo
91      ! -------                                                 --------    .
92      ! épicentre
93      ! prend une maille parmis les pEpis(i)%nb
94
95      if (pEpis(i)%nb.lt.(nb*nb/6)) then                                  ! si réduction d'un moins un quart -> sinon gros gap azimutal
96        ok = int(genrand_real1()*real(pEpis(i)%nb,wr)+1.0_wr)             ! aléatoire de 1 à pEpis(i)%k
97        val = genrand_real1()*2.0_wr - 1.0_wr                             ! entre 1.0 et -1.0
98        val = val * pEpis(i)%pEpi(1)%distcarre * 360.0_wr / ( 2.0_wr * pi * rT)
99        param_init%lat(i) = pEpis(i)%pEpi(ok)%lat + val                   ! ajoute un delta + ou - pEpis(i)%pEpi(1)%distcarre au noeud de la maille choisie
100       val = genrand_real1()*2.0_wr - 1.0_wr                             ! entre 1.0 et -1.0
101       val = val * pEpis(i)%pEpi(1)%distcarre* 360.0_wr / ( 2.0_wr * pi * rT * cos(param_init%lat(i)*pi/180.0_wr))
102       param_init%lon(i) = pEpis(i)%pEpi(ok)%lon + val                   ! ajoute un delta + ou - pEpis(i)%pEpi(1)%distcarre au noeud de la maille choisie
103     else
104       if (nbtps(i).ge.2) then ! si au moins deux données (ce qui dervrait être le cas, sinon on va pas très loin !)
105         ! barycentre : longitudes et latitudes des stations pondérées par les temps d'arrivées des ondes P directes
106         ! geometrie simple et bien mieux que l'habitude prise de prendre lon et lat de la premiere station
107         do j=1,nbtps(i)
108
109           a_ref_temps%date = D(i)%datatps(nbtps(i))%tpsR%date           ! temps de référence, le plus vieux
110           a_ref_temps%sec = D(i)%datatps(nbtps(i))%tpsR%secP+30.0_wr
111           call basetime(a_ref_temps)                                    ! reste en base 60/12/365 ...
112           sommecoef = 0.0_wr
113           param_init%Lon(i) = 0.0_wr
114           param_init%Lat(i) = 0.0_wr
115           if (D(i)%datatps(j)%typeonde=="G") then ! pour simplifier, on ne travaille qu'avec les ondes directes
116             a_time%date=D(i)%datatps(j)%tpsR%date
117             a_time%sec=D(i)%datatps(j)%tpsR%secP
118             call difftime(a_coef,a_ref_temps,a_time)
119             if (a_coef.gt.0.0_wr) then ! au cas ou le plus vieux, n'est pas le plus vieux
120               sommecoef = sommecoef + a_coef*a_coef
121               param_init%Lon(i) = param_init%Lon(i) + D(i)%datatps(j)%sta%lon * a_coef*a_coef
122               param_init%Lat(i) = param_init%Lat(i) + D(i)%datatps(j)%sta%lat * a_coef*a_coef
123             endif
124           endif
125         enddo
126         moy=0.01_wr ; ec=0.1_wr
127         param_init%Lon(i) = param_init%Lon(i) / sommecoef + normal(moy,ec)
128         param_init%Lat(i) = param_init%Lat(i) / sommecoef + normal(moy,ec)
129       else
130         moy=0.01_wr ; ec=0.1_wr
131         param_init%Lon(i) = D(i)%datatps(1)%sta%lon + normal(moy,ec)
132         param_init%lat(i) = D(i)%datatps(1)%sta%lat + normal(moy,ec)
133       endif
134     endif
135     ! -------                                                 --------    .
136     ! temps initial
137     ok=-1
138     do j=1,nbtps(i)
139       if ((ok==-1).and.(D(i)%datatps(j)%typeonde=='G').and.(D(i)%datatps(j)%coefP.lt.3)) then ! première onde P directe
140         ok=j
141       endif
142     enddo
143     ! -------                                                 --------    .
144     if (ok.ne.-1) then                                                  ! il existe des ondes directes
145       ! distance épicentrale avec la premiere station
146       call dellipsgc(D(i)%datatps(ok)%sta%lat,D(i)%datatps(ok)%sta%lon,param_init%lat(i),param_init%lon(i),depi)
147       ! distance hyponcentrale avec la premiere station
148       dhypo=sqrt(depi*depi+param_init%Zhypo(i)*param_init%Zhypo(i))
149       val = dhypo / param_init%VC
150     else                                                                ! aléatoire ...
151       ok = 1
152       moy=1.0_wr
153       ec=10.0_wr
```

```
154         val = abs(normal(moy,ec))
155       endif
156
157       param_init%Tzero(i)%date = D(i)%datatps(ok)%tpsR%date                ! secondes avant la première arrivée PG
158       param_init%Tzero(i)%sec = D(i)%datatps(ok)%tpsR%secP - val
159       call basetime(param_init%Tzero(i))                                   ! reste en base 60/12/365 ...
160       ! -------                                                 --------    .
161     enddo
162     ! ---------------------------------------------------------------      .
163   end subroutine initparam
164
165     ! ---------------------------------------------------------------      .
166
167   subroutine zoneRecherche(nbtps,D,pEpis,nb)
168     ! -------                                                 --------    .mh
169     ! recherche, sans a priori, la zone de l'épicentre par les données de temps d'arrivée
170     ! chaque couple de station défini deux hémisphères (délimité par leur médiatrice)
171     ! l'hémishère contenant la station ou l'arrivée de l'onde est la plus tardive est
172     ! -------                                                 --------    .
173     use typetemps, only : dataall, date_secPS, priorEPI
174     use time, only : difftime
175     use distance_epi
176     ! -------                                                 --------    .
177     implicit none
178     integer(KIND=wi), intent (in) :: nbtps(nbseismes)
179     type(dataall), intent (inout) :: D(nbseismes)
180     type(priorEPI), intent (inout) :: pEpis(nbseismes)
181     integer(KIND=wi), intent (out) :: nb
182     ! -------                                                 --------    .
183     type(date_secPS):: atime
184     ! -------                                                 --------    .
185     integer(KIND=wi) :: nbmax
186     integer(KIND=wi), dimension(:,:), allocatable :: grille
187     integer(KIND=wi) :: i, j, k, l, m
188     integer(KIND=wi) :: Amin, Amax
189     ! -------                                                 --------    .
190     real(KIND=wr) :: deltag, gsize, dfP, dfS, d1, d2
191     real(KIND=wr) :: alon, alat, alon1, alat1, alon0, alat0, alon2, alat2
192     character (LEN=5) :: numberchaine
193     logical :: done
194     ! ---------------------------------------------------------------      .
195     deltag=2.5_wr                                                        ! taille du maillage de la grille (km)
196     gsize=300.0_wr                                                       ! taille de la grille (km) ; distance de recherche depuis la plus proche station
197     nb=int(gsize/deltag)
198     ! -------                                                 --------    . définition de la grille de recherche
199     ! pour grille(i,j) -> lat = alat + j*deltag* 360.0_wr / ( 2.0_wr * pi * rT)
200     ! pour grille(i,j) -> lon = alon + i*deltag* 360.0_wr / sin((90.0_wr - lat)/180.0_wr*pi)
201     ! -------                                                 --------    .
202     ! si i,j = o -> ~ centré sur la plus proche station
203     ! -------                                                 --------    .
204     allocate(grille(-nb:nb,-nb:nb))
205     ! ---------------------------------------------------------------      .
206     open(511, FILE = 'OUTPUT/GMT/script0.sh',status='replace')
207     write(511,'(a)')"gmtset LABEL_FONT_SIZE 15"                          ! nouvelles options GMT
208     write(511,'(a)')'gmtset HEADER_FONT_SIZE 15'
209     write(511,'(a)')"gmtset ANNOT_FONT_PRIMARY Times-Roman"
210     write(511,'(a)')"gmtset ANNOT_FONT_SECONDARY Times-Roman"
211     write(511,'(a)')"gmtset PAPER_MEDIA A3"
212     write(511,'(a)')"gmtset TIME_LANGUAGE FR"
213     write(511,'(a)')"gmtset TRANSPARENCY 50"
214     write(511,'(a)')"gmtset PLOT_DEGREE_FORMAT dddmm"
215     write(511,'(a)')"gmtset BASEMAP_TYPE fancy"
216     write(511,'(a)')"gmtset CHAR_ENCODING ISOLatin1+"
217     ! ---------------------------------------------------------------      .
218     do i=1,nbseismes                                                     ! pour chaque séisme
```

```
219       nbmax=min(nbtps(i),50)                                          ! max. 50 stations ...
220       done=.false.
221       do while (.not.done)
222         write(numberchaine(1:5),'(i5)')i
223         open(508, FILE = 'OUTPUT/GMT/zoneRecherche-'//trim(adjustl(numberchaine))//'.d',status='replace')
224         open(509, FILE = 'OUTPUT/GMT/zRsta-'//trim(adjustl(numberchaine))//'.d',status='replace')
225         open(510, FILE = 'OUTPUT/GMT/zRepi-'//trim(adjustl(numberchaine))//'.d',status='replace')
226         ! -------                                                 --------    . initialisation
227         grille=0
228         ! -------                                                 --------    . la plus proche station
229         atime=D(i)%datatps(1)%tpsR
230         do j=1,nbmax
231           call difftime(dfP,dfS,atime,D(i)%datatps(j)%tpsR)
232           if (dfP.ge.0.0_wr) then
233             alon=D(i)%datatps(j)%sta%lon+0.1_wr                       ! plus ~ 100 m au Nord et à l'Est
234             alat=D(i)%datatps(j)%sta%lat+0.1_wr
235           endif
236         enddo
237         ! ---------------------------------------------------------    . on compare que les P, Pn et Pg séparément
238         do j=1,nbmax
239           do k=j+1,nbmax
240             ! -------                                               --------    . pour chaque couple (avec coef > 3)
241             if ((D(i)%datatps(k)%coefP.lt.3).and.(D(i)%datatps(j)%coefP.lt.3).and. &
242               (D(i)%datatps(k)%typeonde==D(i)%datatps(j)%typeonde)) then
243               ! -------                                             --------    . la station la plus proche : 1
244               call difftime(dfP,dfS,D(i)%datatps(k)%tpsR,D(i)%datatps(j)%tpsR)
245               if (abs(dfP).ge.Tminsec) then                         ! écart minimal de temps entre deux stations
246                 if (dfP.ge.0.0_wr) then
247                   alon1=D(i)%datatps(j)%sta%lon
248                   alat1=D(i)%datatps(j)%sta%lat
249                   alon2=D(i)%datatps(k)%sta%lon
250                   alat2=D(i)%datatps(k)%sta%lat
251                 else
252                   alon1=D(i)%datatps(k)%sta%lon
253                   alat1=D(i)%datatps(k)%sta%lat
254                   alon2=D(i)%datatps(j)%sta%lon
255                   alat2=D(i)%datatps(j)%sta%lat
256                 endif
257                 ! -------                                           --------    . données S
258                 if ((D(i)%datatps(k)%andS=='S').and.(D(i)%datatps(j)%andS=='S').and. &
259                   (D(i)%datatps(k)%coefS.lt.3).and.(D(i)%datatps(k)%coefS.lt.3)) then
260                   ! -------                                         --------    . données non cohérentes -> coef = 4
261                   if ((dfP*dfS).lt.0.0_wr) then
262                     write(*,*)'séisme ',i,' pondération +1 (onde S',D(i)%datatps(j)%typeonde,') pour ', &
263                       D(i)%datatps(j)%sta%staname,' et ',D(i)%datatps(k)%sta%staname
264                       D(i)%datatps(j)%coefS=min(D(i)%datatps(j)%coefS+1,4)
265                       D(i)%datatps(k)%coefS=min(D(i)%datatps(k)%coefS+1,4)
266                   endif
267                 endif
268                 ! -------                                           --------    . pour chaque point de la grille
269                 do l=-nb,nb
270                   do m=-nb,nb
271                     ! -------                                       --------    .
272                     alat0=alat + real(m,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT)
273                     alon0=alon + real(l,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT * cos(alat0*pi/180.0_wr))
274                     ! -------                                       --------    . distance entre point de la grille et une station
275                     if ((l==0).and.(m==0)) write(509,*)alon2,alat2
276                     if ((l==0).and.(m==0)) write(509,*)alon1,alat1
277                     if ((alon0.ne.alon1).and.(alat1.ne.alat0)) then
278                       call  dellipsgc(alat0,alon0,alat1,alon1,d1)
279                     else
280                       d1=0.0_wr
281                     endif
282                     if ((alon0.ne.alon2).and.(alat2.ne.alat0)) then
283                       call  dellipsgc(alat0,alon0,alat2,alon2,d2)
```

```
                    else
                      d2=0.0_wr
                    endif
                    ! _______                                    ________    . pour l'hemisphere le plus loin grille -= 1
                    if (d2.lt.d1) then
                      grille(l,m)=grille(l,m)-1
                    endif
                    ! _______                                    ________    .
                  enddo
                enddo

              endif
              ! _______                                          ________    .
            endif
            ! _______                                           ________    .
          enddo
        enddo
        ! ------------------------------------------------------------------    .
        do l=-nb,nb
          do m=-nb,nb
            alat0=alat + real(m,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT)
            alon0=alon + real(l,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT * cos(alat0*pi/180.0_wr))
            write(508,*)alon0,alat0,grille(l,m)
          enddo
        enddo
        ! ------------------------------------------------------------------    .
        ! _______                                                  ________    . reste 0 dans la grille de recherche ?
        Amin=10000
        Amax=-10000
        do l=-nb,nb
          do m=-nb,nb
            if(Amin.gt.grille(l,m)) Amin=grille(l,m)
            if(Amax.lt.grille(l,m)) Amax=grille(l,m)
          enddo
        enddo
        ! ------------------------------------------------------------------    .
        ! _______                                                  ________    . recherche des données P abérentes
        if (Amax.ne.0) then
          do l=-nb,nb
            do m=-nb,nb
              if(grille(l,m)==Amax) then
                alat0=alat + real(m,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT)
                alon0=alon + real(l,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT * cos(alat0*pi/180.0_wr))
                ! _______                                          ________    . quelles stations ?
                do j=1,nbmax
                  do k=j+1,nbmax
                    ! _______                                      ________    . pour chaque couple (avec coef > 3)
                    if ((D(i)%datatps(k)%coefP.lt.3).and.(D(i)%datatps(j)%coefP.lt.3).and. &
                       (D(i)%datatps(k)%typeonde==D(i)%datatps(j)%typeonde)) then
                      ! _______                                    ________    . la station la plus proche : 1
                      call difftime(dfP,dfS,D(i)%datatps(k)%tpsR,D(i)%datatps(j)%tpsR)
                      if (abs(dfP).ge.Tminsec) then                             ! écart minimal de temps entre deux stations
                        if (dfP.ge.0.0_wr) then
                          alon1=D(i)%datatps(j)%sta%lon
                          alat1=D(i)%datatps(j)%sta%lat
                          alon2=D(i)%datatps(k)%sta%lon
                          alat2=D(i)%datatps(k)%sta%lat
                        else
                          alon1=D(i)%datatps(k)%sta%lon
                          alat1=D(i)%datatps(k)%sta%lat
                          alon2=D(i)%datatps(j)%sta%lon
                          alat2=D(i)%datatps(j)%sta%lat
                        endif
                        if ((alon0.ne.alon1).and.(alat1.ne.alat0)) then
                          call  dellipsgc(alat0,alon0,alat1,alon1,d1)
```

```
                        else
                          d1=0.0_wr
                        endif
                        if ((alon0.ne.alon2).and.(alat2.ne.alat0)) then
                          call  dellipsgc(alat0,alon0,alat2,alon2,d2)
                        else
                          d2=0.0_wr
                        endif
                        ! _______                                    ________    . donnée abérente pour l'une des deux stations :
                        if (d2.lt.d1) then
                          write(*,*)'séisme ',i,' pondération +1 (onde P',D(i)%datatps(j)%typeonde,') pour ', &
                            D(i)%datatps(j)%sta%staname,' et ',D(i)%datatps(k)%sta%staname
                          D(i)%datatps(j)%coefP=min(D(i)%datatps(j)%coefP+1,4)
                          D(i)%datatps(k)%coefP=min(D(i)%datatps(k)%coefP+1,4)
                        endif
                      endif
                    endif
                  enddo
                enddo
              endif
            enddo
          enddo
          close(508)
          close(509)
          close(510)
          done=.false.
        else
          done=.true.
        endif
      enddo
      ! ____________________________________________________________    .
      ! _______                                    ________    . taille de la grille de recherche
      k=0
      do l=-nb,nb
        do m=-nb,nb
          if(grille(l,m)==0) then
            alat0=alat + real(m,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT)
            alon0=alon + real(l,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT * cos(alat0*pi/180.0_wr))
            k=k+1
            write(510,*)alon0,alat0
          endif
        enddo
      enddo
      ! _______                                    ________    . tirage aléatoire dans la grille de recherche
      allocate(pEpis(i)%pEpi(k))
      k=0
      do l=-nb,nb
        do m=-nb,nb
          if(grille(l,m)==0) then
            alat0=alat + real(m,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT)
            alon0=alon + real(l,wr)*deltag* 360.0_wr / ( 2.0_wr * pi * rT * cos(alat0*pi/180.0_wr))
            k=k+1
            pEpis(i)%nb=k
            pEpis(i)%pEpi(k)%lon=alon0
            pEpis(i)%pEpi(k)%lat=alat0
            pEpis(i)%pEpi(k)%distcarre=deltag/2.0_wr
          endif
        enddo
      enddo
      ! ____________________________________________________________    .
      ! _______                                    ________    . plot
      d1= deltag / ( 2.0_wr * pi * rT) * 360.0_wr
      d2= deltag / ( 2.0_wr * pi * rT * cos(alat0*pi/180.0_wr)) * 360.0_wr

      write(511,*)"# echo 'execution du script GMT rechercheepi - "//trim(adjustl(numberchaine))//"'"
```

```
      write(511,*)"BEFORE=$SECONDS"
      write(511,*)'file=OUTPUT/GMT/init-'//trim(adjustl(numberchaine))//'.ps'
      write(511,*)'geoproj=-JM5i'
      write(511,*)'minmax -I0.001 OUTPUT/GMT/zoneRecherche-'//trim(adjustl(numberchaine))//'.d > toto.txt'
      write(511,*)'read geozone < toto.txt'
      write(511,*)'rm -rf toto.txt'
      write(511,'(a,E13.7,a,E13.7,2a)')'makecpt -Chot -I -T',real(Amin-1,wr),'/',real(Amax,wr),'/1.0', &
      '> OUTPUT/GMT/colorpalinit.cpt'
      write(numberchaine(1:5),'(f5.1)')deltag
      write(511,'(a,i9,a)')'psbasemap $geozone $geoproj -Ba1:."Prior \072 ',int(real(k,wr)*deltag*deltag), &
        ' km \262 (maille '//trim(adjustl(numberchaine))//' km)":SnWe -K -Xc -Yc >  $file'
      write(numberchaine(1:5),'(i5)')i
      write(511,'(a,E13.7,a1,E13.7,2a)')'nearneighbor $geozone -I',d1/5.0_wr,'/',d2/5.0_wr,' OUTPUT/GMT/zoneRecherche', &
        '-'//trim(adjustl(numberchaine))//'.d -F -N4 -S6K -GOUTPUT/GMT/topo.grd'
      write(511,'(2a)')'grdimage $geozone $geoproj OUTPUT/GMT/topo.grd -Qnan ', &
        '-COUTPUT/GMT/colorpalinit.cpt -B0 -O -Sn -K -N >> $file'
      !write(511,*)'sort OUTPUT/GMT/zRsta-'//trim(adjustl(numberchaine))//'.d | uniq | sphtriangulate -Qv -T > OUTPUT/GMT/voronoi.d'
      !write(511,*)'psxy $geoproj $geozone -m -K -O OUTPUT/GMT/voronoi.d -W1,-- >>  $file'
      write(511,*)'pscoast $geozone $geoproj -Df+ -W1 -O -K >> $file'
      write(511,*)'sort OUTPUT/GMT/zRsta-'//trim(adjustl(numberchaine))//'.d | uniq | ', &
        'psxy $geozone $geoproj -St0.25 -W6 -Gred -K -O >> $file'
      !write(511,*)'psxy $geozone $geoproj OUTPUT/GMT/zRepi-'//trim(adjustl(numberchaine))//'.d -Sa0.1 -W1 -Ggreen -O -K >> $file'
      write(511,'(a,f5.1,a1,f5.1,2a)')"awk '{ print $1, $2,"" 0.0,",deltag,",",deltag,"""}' OUTPUT/GMT/zRepi", &
        "-"//trim(adjustl(numberchaine))//".d | psxy $geozone $geoproj -SJ -W1,white -O -K >> $file"
      write(511,'(2a)')'psxy $geozone $geoproj OUTPUT/GMT/ellipse-'//trim(adjustl(numberchaine))//'.txt ', &
        '-Sa0.25 -W1,gray -Gblue -O >> $file'
      write(511,*)"ps2raster OUTPUT/GMT/init-"//trim(adjustl(numberchaine))//".ps -Tf -A -P"
      write(511,'(2a)')"mv OUTPUT/GMT/init-"//trim(adjustl(numberchaine))//".pdf ", &
        "OUTPUT/figures/init-"//trim(adjustl(numberchaine))//".pdf"
      write(511,*)"ELAPSED=$(($SECONDS-$BEFORE))"
      write(511,*)"# echo $ELAPSED secondes"
      ! -------                                                    --------     .
    enddo
    close(511)
    deallocate(grille)
    ! ------------------------------------------------------------------     .
  end subroutine zoneRecherche

END MODULE rechercheepi



! ********************************************************************     .
! ********************************************************************     .
```

2.28 SRC/MOD/stat.f90

```
! Librairie de subroutines permettant des regressions linéaires
! octobre 2013
! ********************************************************************     .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
! ********************************************************************     .
! ------------------------------------------------------------------     .

MODULE statistiques

    use modparam

    implicit none

    private

    public  :: correlationaffpond, correlationpond
    public  :: inv_normal_cumulative_distrib_func
    public  :: Rpcalc
```

```
19      public  :: autovariance
20
21      ! -------                                                    --------     .
22      interface Rpcalc
23        module procedure Rpcalc_bis, Rpcalc_ter   ! différents arguments
24      end interface Rpcalc
25      ! -------                                                    --------     .
26
27  CONTAINS
28
29  ! ------------------------------------------------------------------------     .
30
31    subroutine correlationaffpond(a,R2,dph,XY)
32      ! -------- calcul correlation ponderé affine                 --------     .mh
33      ! calcul le coeficient directeur a pour XY(:,1) = X ; XY(:,2) = Y et XY(:,3) = pondération [0;1]
34      ! R2 correspond au chi2
35      ! ordonnée à l'origine nulle
36      ! -------                                                    --------     .
37      implicit none
38      integer (KIND=wi), intent (in) :: dph
39      real (KIND=wr), intent (in)  ::  XY(dph,3)
40      real (KIND=wr), intent (out) ::  R2,a
41      ! -------                                                    --------     .
42      real (KIND=wr) :: sxi,syi,b
43      integer (KIND=wi) :: i
44      ! ------------------------------------------------------------------     .
45      if (dph.gt.1) then
46        sxi=0.0_wr
47        syi=0.0_wr
48        do i=1,dph
49          sxi=sxi+XY(i,2)*XY(i,1)*XY(i,3)
50          syi=syi+XY(i,1)*XY(i,3)*XY(i,1)
51        enddo
52        a=(sxi)/(syi)
53        b=0.0_wr
54        call chi2calc(XY,a,b,dph,R2)
55        if(R2.gt.0.0_wr) then
56          if(IsNaN(a)) then
57            write(*,*)'problème dans correlationaffpond : IsNaN(a)',a
58            stop
59          endif
60          if(IsNaN(b)) then
61            write(*,*)'problème dans correlationaffpond : IsNaN(a)',b
62            stop
63          endif
64        else
65          R2=-1._wr
66          a=-1._wr
67        endif
68      else
69        R2=-1._wr
70        a=-1._wr
71      endif
72      ! ------------------------------------------------------------------     .
73    end subroutine correlationaffpond
74
75  ! ------------------------------------------------------------------------     .
76
77    subroutine correlationpond(a,b,R2,dph,XY)
78      ! -------- calcul correlation ponderé                        --------     .mh
79      ! calcul le coeficient directeur a et ordonnée à l'origine b
80      ! pour XY(:,1) = X ; XY(:,2) = Y et XY(:,3) = pondération [0;1]
81      ! R2 correspond au chi2
82      ! -------                                                    --------     .
83      implicit none
```

```
    integer (KIND=wi), intent (in) :: dph
    real (KIND=wr), intent (in)  ::  XY(dph,3)
    real (KIND=wr), intent (out) ::  R2,a,b
    real (KIND=wr) :: sxi,syi,sxiyi,sxi2,spond
    integer (KIND=wi) :: i
    !-----------------------------------------------------------------------.
    if (dph.gt.1) then
      a=0.0_wr
      b=0.0_wr
      sxi=0.0_wr
      syi=0.0_wr
      sxiyi=0.0_wr
      sxi2=0.0_wr
      spond=0.0_wr
      do i=1,dph
        spond=spond + XY(i,3)
        sxi=sxi+XY(i,1)*XY(i,3)
        syi=syi+XY(i,2)*XY(i,3)
        sxiyi=sxiyi+XY(i,1)*XY(i,2)*XY(i,3)
        sxi2=sxi2+XY(i,1)*XY(i,1)*XY(i,3)
      enddo
      a=(spond*sxiyi-sxi*syi)/(spond*sxi2-sxi*sxi)
      b=(syi*sxi2-sxi*sxiyi)/(spond*sxi2-sxi*sxi)
      call chi2calc(XY,a,b,dph,R2)
      if(R2.gt.0.0_wr) then
        if(IsNaN(a)) then
          write(*,*)'problème dans correlationpond : IsNaN(a)',a
          stop
        endif
        if(IsNaN(b)) then
          write(*,*)'problème dans correlationpond : IsNaN(b)',b
          stop
        endif
      else
        R2=-1._wr
        a=-1._wr
      endif
    else
      R2=-1._wr
      a=-1._wr
    endif
    !-----------------------------------------------------------------------.
  end subroutine correlationpond

!-----------------------------------------------------------------------.

  subroutine chi2calc(XY,a,b,dph,chi2)
    !-------                                                 ---------    .mh
    ! calcul du chi2
    !-------                                                 ---------    .
    implicit none
    integer (KIND=wi), intent (in) :: dph
    real (KIND=wr), intent (in)  :: XY(dph,3),a,b
    real (KIND=wr), intent (out) :: chi2
    !-------                                                 ---------    .
    real (KIND=wr) ::  yth(dph), test
    integer (KIND=wi) i, ik
    !-----------------------------------------------------------------------.
    chi2=0.0_wr
    do ik=1,dph
      yth(ik)=a*XY(ik,1)+b
      chi2=chi2+(XY(ik,2)-yth(ik))**(2.0_wr)
    enddo
    chi2=chi2/real(dph,wr)
    if(IsNaN(chi2)) then
```

```
149       write(*,*)'problème dans chi2calc : IsNaN(chi2)',a,b,dph,chi2
150       test=0.0_wr
151       do i=1,dph
152         ! write(*,*)i,XY(i,1),XY(i,2),XY(i,3)
153         test=test+XY(i,3)
154       enddo
155       if (test.ge.0.00000001_wr) then
156         stop
157       else
158         chi2=-999.99_wr
159       endif
160     endif
161     ! ------------------------------------------------------------------    .
162   end subroutine chi2calc
163 
164 ! ------------------------------------------------------------------    .
165 
166   subroutine Rpcalc_bis(XY,dph,nb,Rp)
167     ! _______                                                 ________    .mh
168     ! calcul du Rp (abs), Coefficient de corrélation linéaire de Bravais-Pearson
169     ! _______                                                 ________    .
170     implicit none
171     integer (KIND=wi), intent (in) :: dph,nb
172     real (KIND=wr), intent (in)  :: XY(nb,3)
173     real (KIND=wr), intent (out) :: Rp
174     ! _______                                                 ________    .
175     real (KIND=wr) ::  A, B, C, x, y
176     integer (KIND=wi) i
177     ! ------------------------------------------------------------------    .
178     if (dph.gt.2) then
179       ! ------------------------------------------------------------------    .
180       if (nb.lt.dph) then
181         write(*,*)'problème dans Rpcalc_bis : i < dph ',nb,' < ',dph
182         stop
183       endif
184       ! _______                                                 ________    .
185       A = 0.0_wr
186       B = 0.0_wr
187       C = 0.0_wr
188       x = 0.0_wr
189       y = 0.0_wr
190       ! _______                                                 ________    .
191       do i=1,dph
192         if (IsNaN(X)) then
193           write(*,*)'problème dans Rpcalc_bis : IsNaN(x)',x
194           stop
195         endif
196         if (IsNaN(y)) then
197           write(*,*)'problème dans Rpcalc_bis : IsNaN(y)',y
198           stop
199         endif
200         x = x + XY(i,1)
201         y = y + XY(i,2)
202       enddo
203       x = x/real(dph,wr)
204       y = y/real(dph,wr)
205       ! _______                                                 ________    .
206       do i=1,dph
207         A = A + (XY(i,1) - x) * (XY(i,2) - y)
208         B = B + (XY(i,1) - x)**2.0_wr
209         C = C + (XY(i,2) - y)**2.0_wr
210       enddo
211       ! _______                                                 ________    .
212       Rp = abs(A/sqrt(B*C))
213       ! _______                                                 ________    .
```

```
214       if (IsNaN(Rp)) then
215         write(*,*)A, B, C, x, y, dph,nb
216         write(*,*)'problème dans Rpcalc_bis : IsNaN(Rp) : ',Rp,dph,nb
217         write(*,*)XY(1,:)
218         write(*,*)XY(2,:)
219         write(*,*)
220         Rp=0.0_wr ! si que pg
221         !stop
222       endif
223       ! ------------------------------------------------------------------ .
224     else
225       ! ------------------------------------------------------------------ .
226       Rp=0.0_wr
227     endif
228     ! ------------------------------------------------------------------ .
229   end subroutine Rpcalc_bis
230
231 ! ---------------------------------------------------------------------- .
232
233   subroutine Rpcalc_ter(vX,vY,nb,Rp)
234     ! _______                                                  ________ .mh
235     ! calcul du Rp, Coefficient de corrélation linéaire de Bravais-Pearson
236     ! _______                                                  ________ .
237     implicit none
238     integer (KIND=wi), intent (in) :: nb
239     real (KIND=wr), intent (in)  :: vX(nb), vY(nb)
240     real (KIND=wr), intent (out) :: Rp
241     ! _______                                                  ________ .
242     real (KIND=wr) ::  A, B, C, x, y
243     integer (KIND=wi) i
244     ! ------------------------------------------------------------------ .
245     if (nb.gt.2) then
246       ! ------------------------------------------------------------------ .
247       A = 0.0_wr
248       B = 0.0_wr
249       C = 0.0_wr
250       x = 0.0_wr
251       y = 0.0_wr
252       ! _______                                                  ________ .
253       do i=1,nb
254         if (IsNaN(X)) then
255           write(*,*)'problème dans Rpcalc_ter : IsNaN(x)',x
256           stop
257         endif
258         if (IsNaN(y)) then
259           write(*,*)'problème dans Rpcalc_ter : IsNaN(y)',y
260           stop
261         endif
262         x = x + vX(i)
263         y = y + vY(i)
264       enddo
265       x = x/real(nb,wr)
266       y = y/real(nb,wr)
267       ! _______                                                  ________ .
268       do i=1,nb
269         A = A + (vX(i) - x) * (vY(i) - y)
270         B = B + (vX(i) - x)**2.0_wr
271         C = C + (vY(i) - y)**2.0_wr
272       enddo
273       ! _______                                                  ________ .
274       Rp = A/sqrt(B*C)
275       ! _______                                                  ________ .
276       if (IsNaN(Rp)) then
277         write(*,*)A, B, C, x, y, nb
278         write(*,*)'problème dans Rpcalc_ter : IsNaN(Rp)',Rp,nb
```

```
279         write(*,*)
280         stop
281       endif
282       ! ------------------------------------------------------------------     .
283     else
284       ! ------------------------------------------------------------------     .
285       Rp=0.0_wr
286     endif
287     ! ------------------------------------------------------------------     .
288   end subroutine Rpcalc_ter
289
290 ! ------------------------------------------------------------------     .
291
292   subroutine inv_normal_cumulative_distrib_func(p,dinvnorm)
293     ! -------                                                 ---------     .mh
294     ! free, fast, and accurate way of computing the inverse normal cumulative distribution function.
295     ! algorithm with a relative error less than 1.15 x 10-9 in the entire region
296     ! on sait : p(dinvnorm)=0.5_wr*erfc(-dinvnorm/sqrt(2.0_wr))
297     ! on cherche : dinvnorm(p)=?
298     ! Peter John Acklam (pjacklam@online.no)
299     ! http://home.online.no/~pjacklam/notes/invnorm
300     ! -------                                                 ---------     .
301     implicit none
302     real (KIND=wr), intent (in)  :: p
303     real (KIND=wr), intent (out) :: dinvnorm
304     ! -------                                                 ---------     .
305     real (KIND=wr) :: p_low,p_high
306     real (KIND=wr) :: a1,a2,a3,a4,a5,a6
307     real (KIND=wr) :: b1,b2,b3,b4,b5
308     real (KIND=wr) :: c1,c2,c3,c4,c5,c6
309     real (KIND=wr) :: d1,d2,d3,d4
310     real (KIND=wr) :: z,q,r
311     ! ------------------------------------------------------------------     .
312     a1=-39.6968302866538_wr
313     a2=220.946098424521_wr
314     a3=-275.928510446969_wr
315     a4=138.357751867269_wr
316     a5=-30.6647980661472_wr
317     a6=2.50662827745924_wr
318     b1=-54.4760987982241_wr
319     b2=161.585836858041_wr
320     b3=-155.698979859887_wr
321     b4=66.8013118877197_wr
322     b5=-13.2806815528857_wr
323     c1=-0.00778489400243029_wr
324     c2=-0.322396458041136_wr
325     c3=-2.40075827716184_wr
326     c4=-2.54973253934373_wr
327     c5=4.37466414146497_wr
328     c6=2.93816398269878_wr
329     d1=0.00778469570904146_wr
330     d2=0.32246712907004_wr
331     d3=2.445134137143_wr
332     d4=3.75440866190742_wr
333     p_low=0.02425_wr
334     p_high=1.0_wr-p_low
335     ! -------                                                 ---------     .
336     if ((p.le.0.0_wr).or.(p.ge.1.0_wr)) then
337       write(*,*)'problème dans inv_normal_cumulative_distrib_func : p= ',p
338     endif
339     ! -------                                                 ---------     .
340     if(p.lt.p_low) then
341       q=sqrt(-2.0_wr*log(p))
342       z=(((((c1*q+c2)*q+c3)*q+c4)*q+c5)*q+c6)/((((d1*q+d2)*q+d3)*q+d4)*q+1.0_wr)
343       dinvnorm=z
```

```
344       else
345         if(p.le.p_high) then
346           q=p-0.5_wr
347           r=q*q
348           z=(((((a1*r+a2)*r+a3)*r+a4)*r+a5)*r+a6)*q/(((((b1*r+b2)*r+b3)*r+b4)*r+b5)*r+1.0_wr)
349           dinvnorm=z
350         elseif(p.lt.1.0_wr) then
351           q=sqrt(-2.0_wr*log(1.0_wr-p))
352           z=-(((((c1*q+c2)*q+c3)*q+c4)*q+c5)*q+c6)/((((d1*q+d2)*q+d3)*q+d4)*q+1.0_wr)
353           dinvnorm=z
354         endif
355       endif
356       ! -------                                                    --------     .
357       if (IsNaN(dinvnorm)) then
358         write(*,*)'problème dans inv_normal_cumulative_distrib_func : IsNaN(dinvnorm)',dinvnorm
359         stop
360       endif
361       ! ------------------------------------------------------------------     .
362     end subroutine inv_normal_cumulative_distrib_func
363
364       ! ------------------------------------------------------------------     .
365
366     subroutine autovariance(vec,itermax,k,nom)
367       ! -------                                                    --------     .
368       ! Calcul de la fonction d'autocovariance, Ck (mesure la covariance entre une variable
369       ! et cette même variable à des dates différentes, pour un délai k)
370       ! La fonction d'autocovariance (Ck) est normalisée en fonction d'autocorrélation (rk)
371       !
372       ! La représentation des fonctions d'autocorrélation des paramètres permet de donner
373       ! des indications sur le nombre d'itérations requis pour que les valeurs des paramètres
374       ! échantillonnées par l'algorithme soient décorrélées, ainsi que sur la valeur de l'écarttype
375       ! de la gaussienne à employer (e.g. Drilleau, 2013).
376
377       ! -------                                                    --------     .
378       implicit none
379       integer(KIND=wi), intent(in) :: itermax,k
380       real(KIND=wr), intent(in) :: vec(itermax)
381       character(len=50), intent(in) :: nom
382       ! -------                                                    --------     .
383       integer(kind=wi) :: i, t, ok
384       real(KIND=wr) :: moy,C0,Ck,rk
385       integer(kind=wi) :: num=0
386       ! -------                                                    --------     .
387       num=num+1
388       ! -------                                                    --------     .
389       moy=0.0_wr
390       do i=1,itermax
391         moy=moy+vec(i)
392       enddo
393       moy=moy/real(itermax,wr)
394       ! -------                                                    --------     .
395       C0=0.0_wr
396       do t=1,itermax
397         C0=C0+(vec(t)-moy)**2.0_wr/real(itermax,wr)
398       enddo
399       ! -------                                                    --------     .
400       ok=0
401       open(unit=5000+num,file=nom,STATUS="replace",iostat=ok)
402       if (ok .ne. 0) then
403         write(*,*)"problème dans autovariance : le fichier "//nom//" n''existe pas "
404         stop
405       endif
406       ! -------                                                    --------     .
407       do i=0,k,1
408         Ck=0.0_wr
```

```
409         do t=1,itermax-i
410           Ck=Ck+(vec(t)-moy)*(vec(t+i)-moy)/real(itermax-i,wr)
411         enddo
412         rk=Ck/C0
413         write(5000+num,*)i,rk
414       enddo
415       close(5000+num)
416       ! -------                                                  --------     .
417     end subroutine autovariance
418
419 END MODULE statistiques
420
421
422
423 ! ********************************************************************     .
424 ! ********************************************************************     .
```

2.29 SRC/MOD/subparam.f90

```
1  ! Librairie de subroutines concernant les parametres d'inversion
2  ! septembre 2013
3  ! ********************************************************************     .
4  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr           --------     .
5  ! ********************************************************************     .
6  ! --------------------------------------------------------------------     .
7
8  MODULE sub_param
9
10    use modparam
11
12    implicit none
13
14    private
15
16    public  :: lect_prior
17    public  :: moycoldruns
18    public  :: init_div
19    public  :: calc_accept
20    public  :: lectparam
21    public  :: nb_mod_selec
22    public  :: lect_mod_select
23    public  :: moy_mod_select
24    public  :: moy_ec, mediane
25    public  :: dist_apriori
26    public  :: paramfixe
27    public  :: inR, outR
28
29
30 CONTAINS
31
32    ! --------------------------------------------------------------------     .
33
34   subroutine lect_prior(param_p,param_init,CorH)
35     ! -------                                                  --------     .mh
36     ! lecture du prior dans PARAM/priorIn.d
37     ! -------                                                  --------     .
38     use typetemps, only : parametresinv, parametres
39     use time
40     ! -------                                                  --------     .
41     implicit none
42     type(parametresinv), intent(out) :: param_p                              ! paramètres de l'inversion
43     type(parametres), intent(in) :: param_init                               ! un jeu de paramètres
44     character (LEN=1), intent (in) :: CorH                                   ! cold or Hot runs
45     ! -------                                                  --------     .
46     integer(KIND=wi) :: i, ok
```

```
47      real(KIND=wr) :: dist, dist_et                                          ! disance de recherche
48      real(KIND=wr), parameter :: dcherche = 350.0_wr                         ! à priori : épicentre est à moins de 350 km de la premiere station !!!
49      real(KIND=wr) :: val
50      ! ------------------------------------------------------------------    .
51      param_p%Rayon=dcherche
52      ! ------------------------------------------------------------------    .
53      ok=0
54      param_p%valNew=param_init
55      param_p%valOld=param_init
56      param_p%mini=param_init
57      param_p%maxi=param_init
58      param_p%ecartype=param_init
59      ! -------                                                 ---------     . min max pour Lon et Lat
60      do i=1, nbseismes
61        if ((param_p%valNew%Lat(i).gt.89._wr).or. &
62          (param_p%valNew%Lat(i).lt.-89._wr).or. &
63          (param_p%valNew%Lon(i).gt.179._wr).or. &
64          (param_p%valNew%Lon(i).lt.-179._wr)) then
65          write(*,*)'problème dans lect_prior 1 : longitude ou latitude actuelle trop près des coutures !'
66          write(*,*)'pas prévu',i,param_p%valNew%Lat(i),param_p%valNew%Lon(i)
67          stop
68        endif
69      enddo
70      ! -------                                                 ---------     .
71      ok=0
72      if (CorH.eq."C") then
73        open(980, FILE = 'PARAM/priorIn_COLD.d',status='old',iostat = ok)
74      elseif (CorH.eq."H") then
75        open(980, FILE = 'PARAM/priorIn_HOT.d',status='old',iostat = ok)
76      else
77        write(*,*)'problème dans lect_prior : ni Coldruns ni Hotruns'
78      endif
79      if (ok .ne. 0) then
80        write(*,*)'problème dans lect_prior : le fichier PARAM/priorIn.d n''existe pas '
81         stop
82      endif
83      ! -------                                                 ---------     . PRIOR pour le modèle de terre
84      read(980,*)param_p%mini%Vc,param_p%maxi%Vc,param_p%ecartype%Vc           ! vitesse croute (en km/s)
85      read(980,*)param_p%mini%Vm,param_p%maxi%Vm,param_p%ecartype%Vm           ! vitesse manteau (en km/s)
86      read(980,*)param_p%mini%Zmoho,param_p%maxi%Zmoho,param_p%ecartype%Zmoho ! moho (km), positif vers le bas, 0 = niveau de la mer
87      read(980,*)param_p%mini%VpVs,param_p%maxi%VpVs,param_p%ecartype%VpVs     ! Vp/Vs (sans unités), croute et manteau
88      ! profondeur de hypoventre (km), positif vers le bas, négatif si relief, 0 = niveau de la mer
89      read(980,*)param_p%mini%Zhypo(1),param_p%maxi%Zhypo(1),param_p%ecartype%Zhypo(1)
90      do i=2, nbseismes
91          param_p%mini%Zhypo(i)=param_p%mini%Zhypo(1)
92          param_p%maxi%Zhypo(i)= param_p%maxi%Zhypo(1)
93          param_p%ecartype%Zhypo(i)=param_p%ecartype%Zhypo(1)
94      enddo
95      ! -------                                                 ---------     . séisme dans la croute
96      if ((param_p%maxi%Zmoho+0.1_wr).le.param_p%maxi%Zhypo(1)) then
97        param_p%maxi%Zmoho=param_p%maxi%Zhypo(1)+0.1_wr
98      endif
99      ! -------                                                 ---------     .
100     read(980,*)dist_et                                                      ! distance (km) correspondant à l'écart type pour Lon et Lat
101     param_p%ec_horizontal=dist_et
102     ! -------                                                 ---------     .
103     call tempszero(param_p%ecartype%Tzero(1)%date)
104     read(980,*)param_p%ecartype%Tzero(1)%sec                                ! écart type (secondes) pour le temps initial
105     call basetime(param_p%ecartype%Tzero(1))                                ! reste en base 60/12/365 ...
106     close(980)
107     ! -------                                                 ---------     . on cherche la solution dans un rayon de 'dcherche' km (pas au-delà !!!!)
108     do i=1, nbseismes
109       dist = dcherche * 360.0_wr / ( 2.0_wr * pi * rT)
110       param_p%mini%Lat(i) = param_init%Lat(i) - dist
111       param_p%maxi%Lat(i) = param_init%Lat(i) + dist
```

```
      dist = dcherche * 360.0_wr / ( 2.0_wr * pi * rT) / &
        sin((90.0_wr-param_init%Lat(i))/180.0_wr*pi)
      param_p%mini%Lon(i) = param_init%Lon(i) - dist
      param_p%maxi%Lon(i) = param_init%Lon(i) + dist
      ! -------                                                 --------     . écart-types lon et lat homogènes (lu en km : dist_et)
      param_p%ecartype%Lat(i) = dist_et * 360.0_wr / ( 2.0_wr * pi * rT)
      param_p%ecartype%Lon(i) = param_p%ecartype%Lat(i) /  &
        sin((90.0_wr-param_init%Lat(i))/180.0_wr*pi)
      ! -------                                                 --------     .
      if ((param_p%maxi%Lat(i).gt.89.9_wr).or. &
            (param_p%mini%Lat(i).lt.-89.9_wr).or. &
            (param_p%maxi%Lon(i).gt.179.9_wr).or. &
            (param_p%mini%Lon(i).lt.-179.9_wr)) then
        write(*,*)'problème dans lect_prior 2 : longitude ou latitude min/max trop près des coutures !'
        write(*,*)'pas prévu',i,param_p%valNew%Lat(i),param_p%valNew%Lon(i)
        stop
      endif
    enddo
    ! -------                                                 --------     . min max pour le temps initial
    do i=1, nbseismes
      param_p%maxi%Tzero(i)%sec = param_init%Tzero(i)%sec + 1.0_wr * 60.0_wr ! plus ou moins 1 minutes
      call basetime(param_p%maxi%Tzero(i))                                  ! reste en base 60/12/365 ...
      param_p%mini%Tzero(i)%sec = param_init%Tzero(i)%sec - 1.0_wr * 60.0_wr ! plus ou moins 1 minutes
      call basetime(param_p%mini%Tzero(i))                                  ! reste en base 60/12/365 ...
      ! -------                                                 --------     . au cas où ...
      call basetime(param_p%valNew%Tzero(i))                                ! reste en base 60/12/365 ...
      call basetime(param_p%valOld%Tzero(i))                                ! reste en base 60/12/365 ...
      param_p%ecartype%Tzero(i)=param_p%ecartype%Tzero(1)
    enddo
    ! -------                                                 --------     . verification au cas ou ...
    ! coldruns appartient bien au nouveau prior ? :
    if (CorH.eq."H") then
      do i=1, nbseismes
        if (param_p%valNew%lon(i).le.param_p%mini%lon(i)) param_p%valNew%lon(i)=param_p%mini%lon(i)
        if (param_p%valNew%lon(i).ge.param_p%maxi%lon(i)) param_p%valNew%lon(i)=param_p%maxi%lon(i)
        if (param_p%valNew%lat(i).le.param_p%mini%lat(i)) param_p%valNew%lat(i)=param_p%mini%lat(i)
        if (param_p%valNew%lat(i).ge.param_p%maxi%lat(i)) param_p%valNew%lat(i)=param_p%maxi%lat(i)
        if (param_p%valNew%Zhypo(i).le.param_p%mini%Zhypo(i)) param_p%valNew%Zhypo(i)=param_p%mini%Zhypo(i)
        if (param_p%valNew%Zhypo(i).ge.param_p%maxi%Zhypo(i)) param_p%valNew%Zhypo(i)=param_p%maxi%Zhypo(i)
        call difftime(val,param_p%valNew%Tzero(i),param_p%maxi%Tzero(i))
        if (val.gt.0.0_wr) param_p%valNew%Tzero(i)=param_p%maxi%Tzero(i)
        call difftime(val,param_p%valNew%Tzero(i),param_p%mini%Tzero(i))
        if (val.lt.0.0_wr) param_p%valNew%Tzero(i)=param_p%mini%Tzero(i)
      enddo
      if (param_p%valNew%VC.le.param_p%mini%VC) param_p%valNew%VC=param_p%mini%VC
      if (param_p%valNew%VC.ge.param_p%maxi%VC) param_p%valNew%VC=param_p%maxi%VC
      if (param_p%valNew%VM.le.param_p%mini%VM) param_p%valNew%VM=param_p%mini%VM
      if (param_p%valNew%VM.ge.param_p%maxi%VM) param_p%valNew%VM=param_p%maxi%VM
      if (param_p%valNew%Zmoho.le.param_p%mini%Zmoho) param_p%valNew%Zmoho=param_p%mini%Zmoho
      if (param_p%valNew%Zmoho.ge.param_p%maxi%Zmoho) param_p%valNew%Zmoho=param_p%maxi%Zmoho
      if (param_p%valNew%VpVs.le.param_p%mini%VpVs) param_p%valNew%VpVs=param_p%mini%VpVs
      if (param_p%valNew%VpVs.ge.param_p%maxi%VpVs) param_p%valNew%VpVs=param_p%maxi%VpVs
      param_p%valOld=param_p%valNew
      ! -------                                                 --------     .
      ok=0
      if (param_init%VC.le.param_p%mini%VC) ok=-1
      if (param_init%VC.ge.param_p%maxi%VC) ok=-1
      if (param_init%VM.le.param_p%mini%VM) ok=-1
      if (param_init%VM.ge.param_p%maxi%VM) ok=-1
      if (param_init%Zmoho.le.param_p%mini%Zmoho) ok=-1
      if (param_init%Zmoho.ge.param_p%maxi%Zmoho) ok=-1
      if (param_init%VpVs.le.param_p%mini%VpVs) ok=-1
      if (param_init%VpVs.ge.param_p%maxi%VpVs) ok=-1
      do i=1,nbseismes
        if (param_init%Lon(i).le.param_p%mini%Lon(i)) ok=-1
```

```
        if (param_init%Lon(i).ge.param_p%maxi%Lon(i)) ok=-1
        if (param_init%Lat(i).le.param_p%mini%Lat(i)) ok=-1
        if (param_init%Lat(i).ge.param_p%maxi%Lat(i)) ok=-1
        if (param_init%Zhypo(i).le.param_p%mini%Zhypo(i)) ok=-1
        if (param_init%Zhypo(i).ge.param_p%maxi%Zhypo(i)) ok=-1
        call difftime(val,param_init%Tzero(i),param_p%maxi%Tzero(i))
        if (val.gt.0.0_wr) ok=-1
        call difftime(val,param_init%Tzero(i),param_p%mini%Tzero(i))
        if (val.lt.0.0_wr) ok=-1
      enddo
      ! -------                                                  --------     .
      if ((ok==-1).and.(.not.FLAGterrefixe)) then
        write(*,*)'problème dans lect_prior : rectifier les priors pour hotruns'
        write(*,*) '(trop resserré par rapport aux coldruns)'
        write(*,*)
        write(*,*)param_init
        write(*,*)param_p%mini
        write(*,*)param_p%maxi
        stop
      endif
      ! -------                                                  --------     .
    endif
    ! ---------------------------------------------------------------------     .
  end subroutine lect_prior

    ! ---------------------------------------------------------------------     .

  subroutine paramfixe(param_p)
    ! -------                                                  --------     .mh
    ! lecture des paramtres fixe si nécessaires
    ! -------                                                  --------     .
    use typetemps, only : parametresinv, date_sec
    use time
    ! -------                                                  --------     .
    implicit none
    type(parametresinv), intent(inout) :: param_p                         ! paramètres de l'inversion
    ! -------                                                  --------     .
    type(date_sec) :: tpsref,tps
    real(KIND=wr) :: moy, ec
    integer(KIND=wi) :: i
    integer(KIND=wi) :: ok
    ! ---------------------------------------------------------------     . lecture des paramètres de terre si fixes
    ok=0
    if(FLAGterre) then
      open(unit=50,file="PARAM/paramTerre.d",STATUS="old",iostat = ok)
      if (ok .ne. 0) then
        write(*,*)'problème dans paramfixe : le fichier PARAM/paramTerre.d n''existe pas '
        stop
      endif
      read(50,*)moy,ec
      param_p%valNew%VC=moy
      param_p%valOld%VC=moy
      param_p%mini%VC=moy-3.0_wr*ec
      param_p%maxi%VC=moy+3.0_wr*ec
      param_p%ecartype%VC=ec
      read(50,*)moy,ec
      param_p%valNew%VM=moy
      param_p%valOld%VM=moy
      param_p%mini%VM=moy-3.0_wr*ec
      param_p%maxi%VM=moy+3.0_wr*ec
      param_p%ecartype%VM=ec
      read(50,*)moy,ec
      param_p%valNew%Zmoho=moy
      param_p%valOld%Zmoho=moy
      param_p%mini%Zmoho=moy-3.0_wr*ec
```

```
242       param_p%maxi%Zmoho=moy+3.0_wr*ec
243       param_p%ecartype%Zmoho=ec
244       read(50,*)moy,ec
245       param_p%valNew%VpVs=moy
246       param_p%valOld%VpVs=moy
247       param_p%mini%VpVs=moy-3.0_wr*ec
248       param_p%maxi%VpVs=moy+3.0_wr*ec
249       param_p%ecartype%VpVs=ec
250       close(50)
251     endif
252     ! ------------------------------------------------------------------     . lecture des paramètres hypocentraux si fixes
253     ok=0
254     if(FLAGhypo) then
255       open(unit=51,file="PARAM/paramHypo.d",STATUS="old",iostat = ok)
256       if (ok .ne. 0) then
257         write(*,*)'problème dans paramfixe : le fichier PARAM/paramHypo.d n''existe pas '
258         stop
259       endif
260       do i=1,nbseismes
261         read(51,*)moy,ec
262         param_p%valNew%lon(i)=moy
263         param_p%valOld%lon(i)=moy
264         param_p%mini%lon(i)=moy-3.0_wr*ec
265         param_p%maxi%lon(i)=moy+3.0_wr*ec
266         param_p%ecartype%lon(i)=ec
267         read(51,*)moy,ec
268         param_p%valNew%lat(i)=moy
269         param_p%valOld%lat(i)=moy
270         param_p%mini%lat(i)=moy-3.0_wr*ec
271         param_p%maxi%lat(i)=moy+3.0_wr*ec
272         param_p%ecartype%lat(i)=ec
273         read(51,*)moy,ec
274         param_p%valNew%Zhypo(i)=moy
275         param_p%valOld%Zhypo(i)=moy
276         param_p%mini%Zhypo(i)=moy-3.0_wr*ec
277         param_p%maxi%Zhypo(i)=moy+3.0_wr*ec
278         param_p%ecartype%Zhypo(i)=ec
279         read(51,*)tpsref
280         read(51,*)moy,ec
281         tpsref%sec=moy
282         call basetime(tpsref)
283         param_p%valNew%Tzero(i)=tpsref
284         param_p%valOld%Tzero(i)=tpsref
285         tps=tpsref
286         tps%sec=tps%sec-3.0_wr*ec
287         param_p%mini%Tzero(i)=tps
288         tps=tpsref
289         tps%sec=tps%sec+3.0_wr*ec
290         param_p%maxi%Tzero(i)=tps
291         call tempszero(tps%date)
292         tps%sec=ec
293         param_p%ecartype%Tzero(i)=tps
294       enddo
295       close(51)
296     endif
297     ! ------------------------------------------------------------------     .
298   end subroutine paramfixe
299
300     ! ------------------------------------------------------------------     .
301
302   subroutine moycoldruns(nbChaineMV,param_best,misfit,nbChaineMVhot,dc)
303     ! -------                                                 ---------     .mh
304     ! fait la moyenne (et écart-type) des meilleurs modèles pour les coldruns
305     ! -------                                                 ---------     .
306     use typetemps, only : parametres, fcout, coldmoy, coldmoyval
```

```
    use time
    ! -------                                                    ---------     .
    implicit none
    integer(KIND=wi), intent (in) :: nbChaineMV                                ! nombre de coldruns
    integer(KIND=wi), intent (in) :: nbChaineMVhot                             ! nombre de hotrun
    type(parametres), intent (in) :: param_best(nbChaineMV)                    ! triés par misfit (triparam)
    type(fcout), intent (in) :: misfit(nbChaineMV)
    type(coldmoy), intent (out) :: dc                                          ! moyennes et écarts-types des modèles du coldrun
    ! -------                                                    ---------     .
    type(coldmoyval) :: cval                                                   ! modèles du coldrun
    integer(KIND=wi) :: i, j
    real(KIND=wr) :: vecT(nbChaineMV)
    real(KIND=wr) :: vecS(nbChaineMVhot)
    ! ------------------------------------------------------------------     . allocation dynamique
    allocate(cval%Tmis(nbChaineMV),cval%TVC(nbChaineMV),cval%TVM(nbChaineMV), &
    cval%TZmoho(nbChaineMV),cval%TVpVs(nbChaineMV),cval%TLat(nbChaineMV,nbseismes), &
    cval%TLon(nbChaineMV,nbseismes),cval%TZhypo(nbChaineMV,nbseismes), &
    cval%TTzero(nbChaineMV,nbseismes))
    allocate(cval%Smis(nbChaineMVhot),cval%SVC(nbChaineMVhot), &
    cval%SVM(nbChaineMVhot),cval%SZmoho(nbChaineMVhot), &
    cval%SVpVs(nbChaineMVhot),cval%SLat(nbChaineMVhot,nbseismes), &
    cval%SLon(nbChaineMVhot,nbseismes),cval%SZhypo(nbChaineMVhot,nbseismes), &
    cval%STzero(nbChaineMVhot,nbseismes))
    ! -------                                                    ---------     .
    do i=1, nbseismes
      dc%tempsrefcold(i) = param_best(1)%Tzero(i)
      dc%tempsrefcold(i)%sec = 0.0_wr
      call tempszero(dc%moytot%par%Tzero(i)%date)
      call tempszero(dc%ectot%par%Tzero(i)%date)
      call tempszero(dc%moyselect%par%Tzero(i)%date)
      call tempszero(dc%ecselect%par%Tzero(i)%date)
    enddo
    ! -------                                                    ---------     . toutes les chaînes
    do i=1, nbChaineMV
      cval%Tmis(i) = misfit(i)%best
      cval%TVC(i) = param_best(i)%VC
      cval%TVM(i) = param_best(i)%VM
      cval%TZmoho(i) = param_best(i)%Zmoho
      cval%TVpVs(i) = param_best(i)%VpVs
      do j=1, nbseismes
        cval%TLat(i,j) = param_best(i)%Lat(j)
        cval%TLon(i,j) = param_best(i)%Lon(j)
        cval%TZhypo(i,j) = param_best(i)%Zhypo(j)
        call difftime(cval%TTzero(i,j),param_best(i)%Tzero(j),dc%tempsrefcold(j))
      enddo
    enddo
    ! -------                                                    ---------     . les chaînes sélectionnées
    do i =1,nbChaineMVhot
      cval%Smis(i) = misfit(i)%best
      cval%SVC(i) = param_best(i)%VC
      cval%SVM(i) = param_best(i)%VM
      cval%SZmoho(i) = param_best(i)%Zmoho
      cval%SVpVs(i) = param_best(i)%VpVs
      do j=1, nbseismes
        cval%SLat(i,j) = param_best(i)%Lat(j)
        cval%SLon(i,j) = param_best(i)%Lon(j)
        cval%SZhypo(i,j) = param_best(i)%Zhypo(j)
        call difftime(cval%STzero(i,j),param_best(i)%Tzero(j),dc%tempsrefcold(j))
      enddo
    enddo
    ! -------                                                    ---------     . calcul moyenne pour tous les meilleurs modeles des coldruns
    call moy_ec(cval%Tmis,nbChaineMV,nbChaineMV,dc%moytot%mis,dc%ectot%mis)
    call moy_ec(cval%TVC,nbChaineMV,nbChaineMV,dc%moytot%par%VC,dc%ectot%par%VC)
    call moy_ec(cval%TVM,nbChaineMV,nbChaineMV,dc%moytot%par%VM,dc%ectot%par%VM)
    call moy_ec(cval%TZmoho,nbChaineMV,nbChaineMV,dc%moytot%par%Zmoho,dc%ectot%par%Zmoho)
```

```
372     call moy_ec(cval%TVpVs,nbChaineMV,nbChaineMV,dc%moytot%par%VpVs,dc%ectot%par%VpVs)
373     do j=1, nbseismes
374       vecT=cval%TLon(:,j)
375       call moy_ec(vecT,nbChaineMV,nbChaineMV,dc%moytot%par%Lon(j),dc%ectot%par%Lon(j))
376       vecT=cval%TLat(:,j)
377       call moy_ec(vecT,nbChaineMV,nbChaineMV,dc%moytot%par%Lat(j),dc%ectot%par%Lat(j))
378       vecT=cval%TZhypo(:,j)
379       call moy_ec(vecT,nbChaineMV,nbChaineMV,dc%moytot%par%Zhypo(j),dc%ectot%par%Zhypo(j))
380       vecT=cval%TTzero(:,j)
381       call moy_ec(vecT,nbChaineMV,nbChaineMV,dc%moytot%par%Tzero(j)%sec,dc%ectot%par%Tzero(j)%sec)
382     enddo
383     ! _______                                                    ________     . calcul moyenne pour tous les meilleurs modeles des coldruns sélectionnés
384     call moy_ec(cval%Smis,nbChaineMVhot,nbChaineMVhot,dc%moyselect%mis,dc%ecselect%mis)
385     call moy_ec(cval%SVC,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%VC,dc%ecselect%par%VC)
386     call moy_ec(cval%SVM,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%VM,dc%ecselect%par%VM)
387     call moy_ec(cval%SZmoho,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%Zmoho,dc%ecselect%par%Zmoho)
388     call moy_ec(cval%SVpVs,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%VpVs,dc%ecselect%par%VpVs)
389     do j=1, nbseismes
390       vecS(:)=cval%SLon(:,j)
391       call moy_ec(vecS,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%Lon(j),dc%ecselect%par%Lon(j))
392       vecS(:)=cval%SLat(:,j)
393       call moy_ec(vecS,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%Lat(j),dc%ecselect%par%Lat(j))
394       vecS(:)=cval%SZhypo(:,j)
395       call moy_ec(vecS,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%Zhypo(j),dc%ecselect%par%Zhypo(j))
396       vecS(:)=cval%STzero(:,j)
397       call moy_ec(vecS,nbChaineMVhot,nbChaineMVhot,dc%moyselect%par%Tzero(j)%sec,dc%ecselect%par%Tzero(j)%sec)
398     enddo
399     ! _______                                                    ________     .
400     deallocate(cval%Tmis,cval%TVC,cval%TVM,cval%TZmoho,cval%TVpVs,cval%TLat,cval%TLon, &
401     cval%TZhypo,cval%TTzero,cval%Smis,cval%SVC,cval%SVM,cval%SZmoho,cval%SVpVs,cval%SLat, &
402     cval%SLon,cval%SZhypo,cval%STzero)
403     ! ------------------------------------------------------------------     . vérif
404     do i=1,nbseismes
405       if (dc%moytot%par%lon(i).gt.179._wr) dc%moytot%par%lon(i)=0.0_wr
406       if (dc%moytot%par%lon(i).lt.-179._wr) dc%moytot%par%lon(i)=0.0_wr
407       if (dc%ectot%par%lon(i).gt.179._wr) dc%ectot%par%lon(i)=0.0_wr
408       if (dc%ectot%par%lon(i).lt.-179._wr) dc%ectot%par%lon(i)=0.0_wr
409       if (dc%moytot%par%lat(i).gt.89._wr) dc%moytot%par%lat(i)=0.0_wr
410       if (dc%moytot%par%lat(i).lt.-89._wr) dc%moytot%par%lat(i)=0.0_wr
411       if (dc%ectot%par%lat(i).gt.89._wr) dc%ectot%par%lat(i)=0.0_wr
412       if (dc%ectot%par%lat(i).lt.-89._wr) dc%ectot%par%lat(i)=0.0_wr
413       if (dc%moyselect%par%lon(i).gt.179._wr) dc%moyselect%par%lon(i)=0.0_wr
414       if (dc%moyselect%par%lon(i).lt.-179._wr) dc%moyselect%par%lon(i)=0.0_wr
415       if (dc%ecselect%par%lon(i).gt.179._wr) dc%moyselect%par%lon(i)=0.0_wr
416       if (dc%ecselect%par%lon(i).lt.-179._wr) dc%moyselect%par%lon(i)=0.0_wr
417       if (dc%moyselect%par%lat(i).gt.89._wr) dc%ecselect%par%lat(i)=0.0_wr
418       if (dc%moyselect%par%lat(i).lt.-89._wr) dc%ecselect%par%lat(i)=0.0_wr
419       if (dc%ecselect%par%lat(i).gt.89._wr) dc%ecselect%par%lat(i)=0.0_wr
420       if (dc%ecselect%par%lat(i).lt.-89._wr) dc%ecselect%par%lat(i)=0.0_wr
421     enddo
422     ! ------------------------------------------------------------------     .
423   end subroutine moycoldruns
424
425     ! ------------------------------------------------------------------     .
426
427   subroutine init_div(misfit,acceptance)
428     ! _______                                                    ________     .mh
429     ! initialise quelques variables
430     ! _______                                                    ________     .
431     use typetemps, only : fcout, accept
432     ! _______                                                    ________     .
433     implicit none
434     type(fcout),intent(out) :: misfit                                       !    fonction coût
435     type(accept),intent(out) :: acceptance                                  !       acceptance
436     ! ------------------------------------------------------------------     .
```

```
437     misfit%old=100000.0_wr
438     misfit%best=100000.0_wr
439     acceptance%N=int(0,wl)                                          ! modèle non accepté
440     acceptance%O=int(0,wl)                                          ! modèle accepté car meilleur que précendant
441     acceptance%NO=int(0,wl)                                         ! modèle accepté car repêché par le Metropolis
442     acceptance%val=0.0_wr
443     ! ------------------------------------------------------------  .
444   end subroutine init_div
445
446     ! ------------------------------------------------------------
447
448   subroutine calc_accept(acceptance)
449     ! -------                                           ---------   .mh
450     ! calcul de l'acceptance
451     ! -------                                           ---------   .
452     use typetemps, only : accept
453     ! -------                                           ---------   .
454     implicit none
455     type(accept),intent(inout) :: acceptance                        !       acceptance (%)
456     ! ------------------------------------------------------------  .
457     ! acceptance = ( accepté du premier coup + repêchés ) / total
458     acceptance%val = real(acceptance%O+acceptance%NO,wr)/ &
459                 real(acceptance%NO+acceptance%N+acceptance%O,wr)*100.0_wr
460     ! ------------------------------------------------------------  .
461   end subroutine calc_accept
462
463     ! ------------------------------------------------------------  .
464
465   subroutine lectparam(nbChaineMV1,nbChaineMV2,maxiter2,maxiter1,chut)
466     ! -------                                           ---------   .mh
467     ! lecture du nombre de chaîne de Markox (nbChaineMV)
468     ! lecture du nombre d'itération par chaîne de Markox (maxiter)
469     ! -------                                           ---------   .
470     implicit none
471     integer(KIND=wi),intent(out) :: nbChaineMV1,nbChaineMV2         ! cold puis hot runs
472     integer(KIND=wi),intent(out) :: maxiter1,maxiter2               ! cold puis hot runs
473     integer(KIND=wi) :: ok
474     logical, intent(in), optional :: chut
475     ! -------                                           ---------   .
476     logical :: printtest
477     ! ------------------------------------------------------------  .
478     ok = 0
479     open(unit=981,file="PARAM/iteration.d",STATUS="old",iostat = ok)
480     if (ok .ne. 0) then
481       write(*,*)'problème dans lectparam : le fichier PARAM/iteration.d n''existe pas '
482       stop
483     endif
484     read(981,*)nbChaineMV1,maxiter1
485     read(981,*)nbChaineMV2,maxiter2
486     close(981)
487     ! -------                                           ---------   . verif
488     if (nbChaineMV2 .gt. nbChaineMV1) then
489         write(*,*)'problème dans lectparam : nbChaineMV hot > nbChaineMV cold '
490         stop
491     endif
492     ! -------                                           ---------   .
493     printtest=.true.
494     if (present(chut)) then
495       if (chut) printtest=.false.
496     endif
497     ! -------                                           ---------   .
498     if(printtest) then
499       write(*,*)'nombre de séismes :                    ',nbseismes
500       write(*,*)'nombre de chaînes de Markov (cold) :   ',nbChaineMV1
501       write(*,*)'nombre de chaînes de Markov (hot) :    ',nbChaineMV2
```

```
502       write(*,*)'nombre d''itérations par chaîne (cold) :',maxiter1
503       write(*,*)'nombre d''itérations par chaîne (hot) : ',maxiter2
504       write(*,*)'nombre modèles testés          :       ',maxiter1*nbChaineMV1+maxiter2*nbChaineMV2
505     endif
506     ! ------------------------------------------------------------------     .
507   end subroutine lectparam
508
509     ! ------------------------------------------------------------------     .
510
511   subroutine nb_mod_selec(nbparam,nbChaineMV)
512     ! -------                                                 ---------     .mh
513     ! relecture des modeles sélectionnés (distrib. a posteriori), définition du nombre de modèles sélectionnés après hotruns
514     ! -------                                                 ---------     .
515     use typetemps, only : paramisfit
516     ! -------                                                 ---------     .
517     implicit none
518     integer(KIND=wi), intent(out) :: nbparam
519     integer(KIND=wi), intent(in) :: nbChaineMV
520     ! -------                                                 ---------     .
521     integer(KIND=wi) :: i,k,ok,noctet
522     type(paramisfit) :: pm
523     character (LEN=5) :: numberchaine
524     ! ------------------------------------------------------------------     .
525     nbparam = 0
526     ! -------                                                 ---------     .
527     do i=1,nbChaineMV
528       ok = 0
529       write(numberchaine(1:5),'(i5)')i
530       inquire ( iolength = noctet ) pm
531       open(unit=1000+i,file="OUTPUT/files/"//trim(adjustl(numberchaine))//".bin",STATUS="old",access='direct',RECL=noctet,iostat=ok)
532       if (ok .ne. 0) then
533         write(*,*)"problème dans nb_mod_selec : le fichier OUTPUT/files/"//trim(adjustl(numberchaine))//".bin n''existe pas "
534         stop
535       endif
536       k=0
537       do while(ok .eq. 0)                                           ! boucle pour compter le nombre de lignes du fichier
538         k=k+1
539         read(1000+i,REC=k,iostat = ok)pm%mis,pm%par
540         if (ok .eq. 0) nbparam = nbparam + 1
541       enddo
542       close(1000+i)
543     enddo
544     write(*,*)'nombre de modèles retenus      :       ',nbparam
545     ! ------------------------------------------------------------------     .
546   end subroutine nb_mod_selec
547
548     ! ------------------------------------------------------------------     .
549
550   subroutine lect_mod_select(p,dp,nbChaineMV,mis,p_best)
551     ! -------                                                 ---------     .mh
552     ! relecture des modeles sélectionnés (distrib. a posteriori) après hotruns
553     ! -------                                                 ---------     .
554     use statistiques
555     use typetemps
556     use time
557     ! -------                                                 ---------     .
558     implicit none
559     type(parametresinv), intent(in) ::  p                         ! paramètres
560     integer(KIND=wi), intent(in) :: nbChaineMV                    ! nombre de chaînes
561     type(parametres), intent(in):: p_best(nbChaineMV)             ! meilleur jeu de parametre pour chaque chaîne (en liens avec "mis")
562     type(fcout), intent(in) :: mis(nbChaineMV)                    ! meilleur fonction coût pour chaque chaîne
563     type(densityplot), intent(inout) :: dp                        ! vecteur des modèles sélectionnés
564     ! -------                                                 ---------     .
565     type(paramisfit) :: pm
566     integer(KIND=wi) :: ok,i,j,k,l,m,n
```

```
567     integer(KIND=wi) :: noct
568     character (LEN=5) :: numberchaine
569     character (LEN=5) :: numberseisme
570     type(date_sec) :: temps_mod
571     real(KIND=wr) :: minmis,size,gap
572     real(KIND=wr) :: minmax_5,minmax_1
573     real(KIND=wr) :: minmax_2(nbseismes),minmax_3(nbseismes),minmax_4(nbseismes)
574     real(KIND=wr) :: minmax_6(nbseismes),minmax_7(nbseismes),minmax_8(nbseismes)
575     character(len=50) :: namefile
576     ! ------------------------------------------------------------------      .
577     call nb_mod_selec(dp%nbparam,nbChaineMV)                                  ! cherche le nombre de modèles sélectionnés
578     ! -------                                                 ---------       .
579     allocate(dp%mis%vec(dp%nbparam))                                          ! allocation dynamique du 'type' dp
580     allocate(dp%VC%vec(dp%nbparam))
581     allocate(dp%VM%vec(dp%nbparam))
582     allocate(dp%Zmoho%vec(dp%nbparam))
583     allocate(dp%VpVs%vec(dp%nbparam))
584     do i=1,nbseismes
585       allocate(dp%Lat(i)%vec(dp%nbparam))
586       allocate(dp%Lon(i)%vec(dp%nbparam))
587       allocate(dp%Zhypo(i)%vec(dp%nbparam))
588       allocate(dp%Tzero(i)%vec(dp%nbparam))
589     enddo
590     ! -------                                                 ---------       .
591     ! sélectionne le modèle dont le misfit est le plus bas, afin de définir une réference de temps en Année, Mois, Jour, heure et min
592     ! pour la suite du programme, le parametre Temps_zero correspondra à ce temps_ref (min) + dp%Tzero%vec(n) (secondes relatives)
593     ! -------                                                 ---------       .
594     do j=1,nbseismes
595       minmis=1.e9_wr
596       do i=1,nbChaineMV
597         if(mis(i)%best.lt.minmis) then
598             minmis=mis(i)%best
599             dp%temps_ref(j)=p_best(i)%Tzero(j)
600           endif
601       enddo
602       dp%temps_ref(j)%sec=0.0_wr
603       call basetime(dp%temps_ref(j))
604     enddo
605     ! -------                                                 ---------       . lecture des modèles sélectionnés
606     n=0
607     l=1
608     do i=1,nbChaineMV
609       ok = 0
610       write(numberchaine(1:5),'(i5)')i
611       inquire ( iolength = noct ) pm
612       open(unit=1250+i,file="OUTPUT/files/"//trim(adjustl(numberchaine))//".bin",STATUS="old",access='direct',RECL=noct,iostat=ok)
613       ! -------                                               ---------       .
614       if (ok .ne. 0) then
615         write(*,*)"problème dans lect_mod_select : le fichier OUTPUT/files/"//trim(adjustl(numberchaine))//".bin n''existe pas "
616         stop
617       endif
618       ! -------                                               ---------       .
619       k=0
620       do while(ok.eq.0)
621         k=k+1
622         read(1250+i,REC=k,iostat = ok)pm%mis,pm%par
623         if (ok .eq. 0) then
624           n=n+1
625           dp%mis%vec(n)=pm%mis
626           dp%VC%vec(n)=pm%par%VC
627           dp%VM%vec(n)=pm%par%VM
628           dp%Zmoho%vec(n)=pm%par%Zmoho
629           dp%VpVs%vec(n)=pm%par%VpVs
630           do j=1, nbseismes
631             dp%Lat(j)%vec(n)=pm%par%Lat(j)
```

```
            dp%Lon(j)%vec(n)=pm%par%Lon(j)
            dp%Zhypo(j)%vec(n)=pm%par%Zhypo(j)
            temps_mod=pm%par%Tzero(j)
            call difftime(dp%Tzero(j)%vec(n),temps_mod,dp%temps_ref(j))    ! dp%Tzero : temps relatif à temps_ref
          enddo
        endif
      enddo
      ! ------------------------------------------------------------------    .
      ! calcul des fonction d'autovariance pour chaque parametre et chaque chaîne
      ! ------------------------------------------------------------------    .
      m=min(autocorr,k/3)
      ! -------                                                 ---------    .
      namefile="OUTPUT/GMT/autovar_VC"//trim(adjustl(numberchaine))//".txt"
      call autovariance(dp%VC%vec(l:n),k,m,namefile)
      namefile="OUTPUT/GMT/autovar_VM"//trim(adjustl(numberchaine))//".txt"
      call autovariance(dp%VM%vec(l:n),k,m,namefile)
      namefile="OUTPUT/GMT/autovar_Zmoho"//trim(adjustl(numberchaine))//".txt"
      call autovariance(dp%Zmoho%vec(l:n),k,m,namefile)
      namefile="OUTPUT/GMT/autovar_VpVs"//trim(adjustl(numberchaine))//".txt"
      call autovariance(dp%VpVs%vec(l:n),k,m,namefile)
      do j=1, nbseismes
        write(numberseisme(1:5),'(i5)')j
        namefile="OUTPUT/GMT/autovar_lon_"//trim(adjustl(numberseisme))//"_"//trim(adjustl(numberchaine))//".txt"
        call autovariance(dp%Lon(j)%vec(l:n),k,m,namefile)
        namefile="OUTPUT/GMT/autovar_lat_"//trim(adjustl(numberseisme))//"_"//trim(adjustl(numberchaine))//".txt"
        call autovariance(dp%Lat(j)%vec(l:n),k,m,namefile)
        namefile="OUTPUT/GMT/autovar_Zhypo_"//trim(adjustl(numberseisme))//"_"//trim(adjustl(numberchaine))//".txt"
        call autovariance(dp%Zhypo(j)%vec(l:n),k,m,namefile)
        namefile="OUTPUT/GMT/autovar_Tzero_"//trim(adjustl(numberseisme))//"_"//trim(adjustl(numberchaine))//".txt"
        call autovariance(dp%Tzero(j)%vec(l:n),k,m,namefile)
      enddo
      ! ------------------------------------------------------------------    .
      l=n+1
      ! -------                                                 ---------    .
      close(1250+i)
    enddo
    ! -------                                                 ---------    .
    if (n.ne.dp%nbparam) then
      write(*,*)"problème dans lect_mod_select : mauvaise lecture des modeles sélectionnés"
    endif
    ! ------- nom des légendes GMT                            ---------    . format GMT
    dp%mis%char      ='        fonction co\373t        '
    dp%VC%char       =' V@-C @-\050km\056s@+-1@+\051  '
    dp%VM%char       =' V@-M @-\050km\056s@+-1@+\051  '
    dp%Zmoho%char    ='        moho \050km\051         '
    dp%VpVs%char     ='        V@-P @- / V@-S @-       '
    do i=1,nbseismes
      dp%Lat(i)%char     ='      latitude \050\260\051     '
      dp%Lon(i)%char     ='     longitude \050\260\051     '
      dp%Zhypo(i)%char   ='     Hypocentre \050km\051      '
      dp%Tzero(i)%char   =' temps initial \050s\051   '
    enddo
    ! ------- nom des fichiers d'entrée GMT                   ---------    . format GMT
    dp%mis%name="mis"
    dp%VC%name="_vc"
    dp%VM%name="_vm"
    dp%Zmoho%name="_zm"
    dp%VpVs%name="vps"
    do i=1,nbseismes
      dp%Lat(i)%name="lat"
      dp%Lon(i)%name="lon"
      dp%Zhypo(i)%name="_zh"
      dp%Tzero(i)%name="_to"
    enddo
    ! -------                                                 ---------    .
```

```
    ! cherche les bornes max et min du prior pour les paramètres de structures + pour la profondeur du séisme (Zhypo)
    ! _______                                                    ________    .
    dp%VC%themin=p%mini%VC
    dp%VM%themin=p%mini%VM
    dp%Zmoho%themin=p%mini%Zmoho
    dp%VpVs%themin=p%mini%VpVs
    do i=1,nbseismes
       dp%Zhypo(i)%themin=p%mini%Zhypo(i)
    enddo
    dp%VC%themax=p%maxi%VC
    dp%VM%themax=p%maxi%VM
    dp%Zmoho%themax=p%maxi%Zmoho
    dp%VpVs%themax=p%maxi%VpVs
    do i=1,nbseismes
       dp%Zhypo(i)%themax=p%maxi%Zhypo(i)
    enddo
    ! _______                                                    ________    .
    ! cherche les bornes max et min des modéles sélectionnés pour les paramètres de hypocenraux (lon,lat,temps initial) et la fonction coût
    ! _______                                                    ________    .
    minmax_1=100000.0_wr
    minmax_2=100000.0_wr
    minmax_3=100000.0_wr
    minmax_4=100000.0_wr
    minmax_5=-100000.0_wr
    minmax_6=-100000.0_wr
    minmax_7=-100000.0_wr
    minmax_8=-100000.0_wr
    do i=1,dp%nbparam
       if(dp%mis%vec(i).lt.minmax_1) then
          minmax_1=dp%mis%vec(i)
          dp%mis%themin=minmax_1
       endif
       if(dp%mis%vec(i).gt.minmax_5) then
          minmax_5=dp%mis%vec(i)
          dp%mis%themax=minmax_5
       endif
       do j=1,nbseismes
          if(dp%Lon(j)%vec(i).lt.minmax_2(j)) then
             minmax_2(j)=dp%Lon(j)%vec(i)
             dp%Lon(j)%themin=minmax_2(j)
          endif
          if(dp%Lat(j)%vec(i).lt.minmax_3(j)) then
             minmax_3(j)=dp%Lat(j)%vec(i)
             dp%Lat(j)%themin=minmax_3(j)
          endif
          if(dp%Tzero(j)%vec(i).lt.minmax_4(j)) then
             minmax_4(j)=dp%Tzero(j)%vec(i)
             dp%Tzero(j)%themin=minmax_4(j)
          endif
          if(dp%Lon(j)%vec(i).gt.minmax_6(j)) then
             minmax_6(j)=dp%Lon(j)%vec(i)
             dp%Lon(j)%themax=minmax_6(j)
          endif
          if(dp%Lat(j)%vec(i).gt.minmax_7(j)) then
             minmax_7(j)=dp%Lat(j)%vec(i)
             dp%Lat(j)%themax=minmax_7(j)
          endif
          if(dp%Tzero(j)%vec(i).gt.minmax_8(j)) then
             minmax_8(j)=dp%Tzero(j)%vec(i)
             dp%Tzero(j)%themax=minmax_8(j)
          endif
       enddo
    enddo
    ! _______                                                    ________    .
    gap=0.05_wr                                                             ! ajoute un gap de 5% autour du min et du max pour affichage
```

```
762     do j=1,nbseismes
763       size=dp%Lon(j)%themax-dp%Lon(j)%themin
764       dp%Lon(j)%themax=dp%Lon(j)%themax+gap*size
765       dp%Lon(j)%themin=dp%Lon(j)%themin-gap*size
766       size=dp%Lat(j)%themax-dp%Lat(j)%themin
767       dp%Lat(j)%themax=dp%Lat(j)%themax+gap*size
768       dp%Lat(j)%themin=dp%Lat(j)%themin-gap*size
769     enddo
770     gap=0.01_wr                                                    ! ajoute un gap de 1% autour du min et du max pour affichage
771     do j=1,nbseismes
772       size=dp%Tzero(j)%themax-dp%Tzero(j)%themin
773       dp%Tzero(j)%themax=dp%Tzero(j)%themax+gap*size
774       dp%Tzero(j)%themin=dp%Tzero(j)%themin-gap*size
775     enddo
776     size=dp%mis%themax-dp%mis%themin
777     dp%mis%themax=dp%mis%themax+gap*size
778     dp%mis%themin=dp%mis%themin-gap*size
779     ! ------------------------------------------------------------------     .
780   end subroutine lect_mod_select
781
782     ! ------------------------------------------------------------------     .
783
784   subroutine moy_mod_select(dp,nbChaineMV,param_best,misfit)
785     ! -------                                                ---------     .mh
786     ! calcul moy(s), ecart-type(s) et mode des modèles sélectionnés, sur tous les paramètres
787     ! -------                                                ---------     .
788     use typetemps, only : densityplot, parametres, fcout
789     use time, only : difftime
790     ! -------                                                ---------     .
791     implicit none
792     ! -------                                                ---------     .
793     type(densityplot), intent(inout) :: dp
794     type(parametres), intent(in) :: param_best(nbChaineMV)
795     type(fcout), intent(in) :: misfit(nbChaineMV)
796     integer(KIND=wi), intent(in) :: nbChaineMV
797     ! -------                                                ---------     .
798     real(KIND=wr) :: vec(nbChaineMV)
799     integer :: i,j
800     ! ------------------------------------------------------------------     .
801     if(dp%nbparam.gt.11000) then                                    ! si au moins 11000 modèles (on travail sur des grands nombres, tout de même)
802     ! calcul des moyennes :
803       call moy_mod_select_one (dp%mis,dp%mis,dp%nbparam,dp%deltaxy,dp%mis%vec10000)
804       call moy_mod_select_one (dp%VC,dp%mis,dp%nbparam,dp%deltaxy,dp%VC%vec10000)
805       call moy_mod_select_one (dp%VM,dp%mis,dp%nbparam,dp%deltaxy,dp%VM%vec10000)
806       call moy_mod_select_one (dp%Zmoho,dp%mis,dp%nbparam,dp%deltaxy,dp%Zmoho%vec10000)
807       call moy_mod_select_one (dp%VpVs,dp%mis,dp%nbparam,dp%deltaxy,dp%VpVs%vec10000)
808       do i=1,nbseismes
809         call moy_mod_select_one (dp%Lat(i),dp%mis,dp%nbparam,dp%deltaxy,dp%Lat(i)%vec10000)
810         call moy_mod_select_one (dp%Lon(i),dp%mis,dp%nbparam,dp%deltaxy,dp%Lon(i)%vec10000)
811         call moy_mod_select_one (dp%Zhypo(i),dp%mis,dp%nbparam,dp%deltaxy,dp%Zhypo(i)%vec10000)
812         call moy_mod_select_one (dp%Tzero(i),dp%mis,dp%nbparam,dp%deltaxy,dp%Tzero(i)%vec10000)
813       enddo
814     else
815       write(*,*)'problème dans moy_mod_select : pas assez de modèles, d''un point de vue statistique'
816       stop
817     endif
818
819     ! ------------------------------------------------------------------     .
820     ! ecriture des moyenne et écartyes dans paramHypo.d et paramTerre.d
821     ! -> fichiers ustilisables comme INPUT dans une autre execution ...
822     ! ------------------------------------------------------------------     .
823     open(unit=50,file="OUTPUT/input/paramTerre_new.d",STATUS="replace")
824       write(50,*)dp%VC%moy_1000,dp%VC%ec_1000
825       write(50,*)dp%VM%moy_1000,dp%VM%ec_1000
826       write(50,*)dp%Zmoho%moy_1000,dp%Zmoho%ec_1000
```

```
827         write(50,*)dp%VpVs%moy_1000,dp%VpVs%ec_1000
828       close(50)
829       open(unit=51,file="OUTPUT/input/paramHypo_new.d",STATUS="replace")
830         do i=1,nbseismes
831           write(51,*)dp%lon(i)%moy_1000,dp%lon(i)%ec_1000
832           write(51,*)dp%lat(i)%moy_1000,dp%lat(i)%ec_1000
833           write(51,*)dp%Zhypo(i)%moy_1000,dp%Zhypo(i)%ec_1000
834           write(51,*)dp%temps_ref(i)
835           write(51,*)dp%Tzero(i)%moy_1000,dp%Tzero(i)%ec_1000
836         enddo
837       close(51)
838       ! ------------------------------------------------------------------      .
839       ! -------                                                         --------      .
840       ! calcul des moyennes sur l'ensemble du meilleur modèle de chaque chaîne
841       do i=1,nbChaineMV
842         vec(i)=misfit(i)%best
843       enddo
844       call moy_ec(vec,nbChaineMV,nbChaineMV,dp%mis%moy_bestchaine,dp%mis%ec_bestchaine)
845       ! -------                                                         --------      .
846       do i=1,nbChaineMV
847         vec(i)=param_best(i)%VC
848       enddo
849       call moy_ec(vec,nbChaineMV,nbChaineMV,dp%VC%moy_bestchaine,dp%VC%ec_bestchaine)
850       ! -------                                                         --------      .
851       do i=1,nbChaineMV
852         vec(i)=param_best(i)%VM
853       enddo
854       call moy_ec(vec,nbChaineMV,nbChaineMV,dp%VM%moy_bestchaine,dp%VM%ec_bestchaine)
855       ! -------                                                         --------      .
856       do i=1,nbChaineMV
857         vec(i)=param_best(i)%Zmoho
858       enddo
859       call moy_ec(vec,nbChaineMV,nbChaineMV,dp%Zmoho%moy_bestchaine,dp%Zmoho%ec_bestchaine)
860       ! -------                                                         --------      .
861       do i=1,nbChaineMV
862         vec(i)=param_best(i)%VpVs
863       enddo
864       call moy_ec(vec,nbChaineMV,nbChaineMV,dp%VpVs%moy_bestchaine,dp%VpVs%ec_bestchaine)
865       ! -------                                                         --------      .
866       do j=1,nbseismes
867         do i=1,nbChaineMV
868           vec(i)=param_best(i)%Lat(j)
869         enddo
870         call moy_ec(vec,nbChaineMV,nbChaineMV,dp%Lat(j)%moy_bestchaine,dp%Lat(j)%ec_bestchaine)
871         ! -------                                                         --------      .
872         do i=1,nbChaineMV
873           vec(i)=param_best(i)%Lon(j)
874         enddo
875         call moy_ec(vec,nbChaineMV,nbChaineMV,dp%Lon(j)%moy_bestchaine,dp%Lon(j)%ec_bestchaine)
876         ! -------                                                         --------      .
877         do i=1,nbChaineMV
878           vec(i)=param_best(i)%Zhypo(j)
879         enddo
880         call moy_ec(vec,nbChaineMV,nbChaineMV,dp%Zhypo(j)%moy_bestchaine,dp%Zhypo(j)%ec_bestchaine)
881         ! -------                                                         --------      .
882         do i=1,nbChaineMV
883           call difftime(vec(i),param_best(i)%Tzero(j),dp%temps_ref(j))
884         enddo
885         call moy_ec(vec,nbChaineMV,nbChaineMV,dp%Tzero(j)%moy_bestchaine,dp%Tzero(j)%ec_bestchaine)
886       enddo
887       ! ------------------------------------------------------------------      .
888     end subroutine moy_mod_select
889
890       ! ------------------------------------------------------------------      .
891
```

```
subroutine moy_mod_select_one (a_param,mis_param,nbparam,deltaxy,VEC_10000_tri)
  ! _______                                                    ________    .mh
  ! calcul des moyennes, ecart-types et modes des modèles sélectionnés, pour un unique paramètre
  ! là, j'aurais pu être plus élégant au niveau syntaxique,
  ! en utilisant les fonctions de la norme f90
  ! au lieu de boucler, comme un cochon, sur tout.
  ! mais pour ~500 000 modèles, ca reste rapide ...
  ! _______                                                    ________    .
  use typetemps , only : densityplot_one
  use cpt_temps
  use tri , only : tri_bulle
  ! _______                                                    ________    .
  implicit none
  ! _______                                                    ________    .
  type(densityplot_one), intent(inout) :: a_param
  type(densityplot_one), intent(in) :: mis_param
  integer(KIND=wi), intent(in) :: nbparam
  integer(KIND=wi), intent(in) :: deltaxy
  real(KIND=wr), intent(out)   :: VEC_10000_tri(10000,2)
  ! _______                                                    ________    .
  integer(KIND=wi) :: i,j,n,size
  integer(KIND=wi) :: vmode(deltaxy)
  real(KIND=wr) :: VEC_10000(10000,2), max, med
  integer(KIND=wi), save :: count = 0
  character (LEN=30) :: chaine
  logical :: ok
  ! ------------------------------------------------------------------    .
  a_param%delta=(a_param%themax-a_param%themin)/real(deltaxy,wr)          ! calcul le pas (discretisation pour le mode et diagramme de densité)
  ! ------------------------------------------------------------------    .
  ! vecteur avec les 10 000 meilleurs modèles (les modèles sont tous différents ; c.-à.-d. sans doublons)
  ! ------------------------------------------------------------------    .
  do i=1,10000                                                           ! initialisation
    VEC_10000(i,1)=-1.0_wr
    VEC_10000(i,2)=10000.0_wr+real(i*100,wr)
  enddo
  ! _______                                                    ________    .
  bestmod : do i=1,nbparam
    write(chaine(1:30),*)"paramètre : ",a_param%name                     ! chaîne pour la barre d'avancement
    count=count+1
    call progress(count,nbparam*(5+4*nbseismes),chaine)                  ! barre d'avancement : il existe (4 + 4*nbseismes) paramètres + (1) la fonction coût
    ! _______                                                  ________    . cherche les 10 000 meilleurs modèles (non uniques)
    max=-1.0_wr
    piremod : do j=1,10000                                               ! cherche le pire modèles (possédant le plus grand misfit)
      ok=.true.
      if ((VEC_10000(j,2)==mis_param%vec(i)).and.(VEC_10000(j,1)==a_param%vec(i))) then
        ok=.false.                                                       ! modèle existe déjà, pas de doublons
        exit piremod
      else
        if(VEC_10000(j,2).gt.max) then
          max=VEC_10000(j,2)
          n=j
        endif
      endif
    enddo piremod
    if((ok).and.(VEC_10000(n,2).gt.mis_param%vec(i))) then               ! remplace le pire modèle
      VEC_10000(n,1)=a_param%vec(i)
      VEC_10000(n,2)=mis_param%vec(i)
    endif
  enddo bestmod
  ! ------------------------------------------------------------------    .
  n=10000
  call tri_bulle(VEC_10000,n,VEC_10000_tri)                              ! tri croissant des meilleurs modèles (sans doublons)
  if (VEC_10000_tri(10000,1)==-1.0_wr) then
    write(*,*)'problème dans moy_mod_select_one : pas assez de modèles différents (< 10 000)'
    stop
```

```
957     endif
958     ! ------------------------------------------------------------------      .
959     a_param%best=VEC_10000_tri(1,1)                                          ! modèle possédant la fonction coût la plus basse
960     ! ------------------------------------------------------------------      .
961     do i=1,deltaxy                                                           ! initialise, calcul du mode
962       vmode(i)=0
963     enddo
964     ! -------                                                 ---------      . calcul de la médiane
965     med=2.0_wr
966     call mediane(med,a_param%vec,nbparam,a_param%mediane)
967     ! -------                                                 ---------      . calcul du mode
968     do i=1,nbparam
969      if ((a_param%vec(i).lt.a_param%themin).or.(a_param%vec(i).gt.a_param%themax)) then
970        write(*,*)'problème dans moy_mod_select_one 1 : calcul du mode impossible',i,a_param%name, &
971        a_param%vec(i),a_param%themin,a_param%themax,a_param%delta,j
972        ! stop
973      else
974        j = int((a_param%vec(i)-a_param%themin)/a_param%delta)+1
975        if ((j.gt.0).and.(j.le.deltaxy)) then
976          vmode(j) = vmode(j) + 1
977        else
978          write(*,*)'problème dans moy_mod_select_one 2 : calcul du mode impossible',i,a_param%name, &
979          a_param%vec(i),a_param%themin,a_param%themax,a_param%delta,j
980          stop
981        endif
982
983      endif
984     enddo
985     ! -------                                                 ---------      .
986     j=0
987     max=-1.0_wr
988     do i=1,deltaxy                                                           ! mode : maximum
989       if(real(vmode(i),wr).gt.max) then
990         max = real(vmode(i),wr)
991         j=i
992       endif
993     enddo
994     a_param%mode = a_param%themin + (real(j,wr)-0.5_wr) * a_param%delta
995     ! -------                                                 ---------      . calcul des moyennes et des écart-types
996     call moy_ec(a_param%vec,nbparam,nbparam,a_param%moy_tot,a_param%ec_tot)! pour les nbparam modèles (tous les sélectionnés)
997     n=10000
998     size=n
999     call moy_ec(VEC_10000_tri(:,1),size,n,a_param%moy_10000,a_param%ec_10000) ! pour les 10000 meilleurs modèles
1000    n=1000
1001    call moy_ec(VEC_10000_tri(:,1),size,n,a_param%moy_1000,a_param%ec_1000) ! pour les 1000 meilleurs modèles
1002    n=100
1003    call moy_ec(VEC_10000_tri(:,1),size,n,a_param%moy_100,a_param%ec_100) ! pour les 100 meilleurs modèles
1004    ! ------------------------------------------------------------------      .
1005  end subroutine moy_mod_select_one
1006
1007    ! ------------------------------------------------------------------      .
1008
1009  subroutine mediane(val,tab,bufLength,med)
1010    ! -------                                                 ---------      .mh
1011    ! calcul de la médiane (si val=2.0)
1012    ! calcul du premier quatrile (si val=4.0)
1013    ! calcul du second quatrile (si val=4.0/3.0)
1014    ! -------                                                 ---------      .
1015    ! Modified algorithm according to http://www.geocities.com/zabrodskyvlada/3alg.html
1016    ! Contributed by Heinz Klar
1017    ! -------                                                 ---------      .
1018    implicit none
1019    integer (KIND=wi), intent(in) :: bufLength
1020    real (KIND=wr), intent(in) :: val
1021    real (KIND=wr), intent(in) :: tab(bufLength)
```

```
    real (KIND=wr), intent(out) :: med
    ! ------                                                    --------     .
    real (KIND=wr) :: buf(bufLength)
    integer (KIND=wi):: i,j,n,l,m
    real (KIND=wr) :: dum
    ! ------------------------------------------------------------------     .
    if(bufLength.lt.750000) then
      buf=tab
      m=bufLength-1
      n=int(real(bufLength,8)/val)
      med=buf(n)
      l=1
      ! ------                                                    --------     .
      do while (l.lt.m)
        i=l
        j=m
        do while ((j.ge.n).and.(i.le.n))
          do while (buf(i).lt.med)
            i=i+1
          enddo
          do while (med.lt.buf(j))
            j=j-1
          enddo
          dum=buf(j)
          buf(j)=buf(i)
          buf(i)=dum
          i=i+1
          j=j-1
        enddo
        if (j.lt.n) l=i
        if (n.lt.i) m=j
        med=buf(n)
      enddo
    else
      write(*,*)'problème dans mediane : trop de modèles ',bufLength
      med=0.0_wr
    endif
    ! ------------------------------------------------------------------     .
  end subroutine mediane

    ! ------------------------------------------------------------------     .

  subroutine inR(D,R,nbtps,nbstaR,mis)
    ! ------                                                    --------     .mh
    ! lecture des résidus dans D, à chaque itération  et stock dans R
    ! dans la suite R est écrit dans un fichier (cf outR) | (si FLAGresSTA=.true.)
    ! ------                                                    --------     .
    use typetemps, only : dataall, residus
    ! ------                                                    --------     .
    implicit none
    ! ------                                                    --------     .
    integer(KIND=wi), intent(in) :: nbtps(nbseismes), nbstaR
    real(KIND=wr), intent(in) :: mis
    type(dataall), intent(in) :: D(nbseismes)
    type(residus), intent(inout) :: R(nbstaR)                               ! résidus
    ! ------                                                    --------     .
    integer(KIND=wi) :: i,j,k
    ! ------------------------------------------------------------------     .
    do i=1,nbseismes
      do j=1,nbtps(i)
        do k=1,nbstaR
          if (D(i)%datatps(j)%sta%staname==R(k)%staname) then
            if (D(i)%datatps(j)%typeonde=='G') then
              R(k)%nbPg = R(k)%nbPg + 1
              R(k)%resPg(R(k)%nbPg,1) = D(i)%datatps(j)%dTP
```

```
              R(k)%resPg(R(k)%nbPg,2) = mis
              if (D(i)%datatps(j)%andS=='S') then
                R(k)%nbSg = R(k)%nbSg + 1
                R(k)%resSg(R(k)%nbSg,1) = D(i)%datatps(j)%dTS
                R(k)%resSg(R(k)%nbSg,2) = mis
              endif
            elseif (D(i)%datatps(j)%typeonde=='N') then
              R(k)%nbPn = R(k)%nbPn + 1
              R(k)%resPn(R(k)%nbPn,1) = D(i)%datatps(j)%dTP
              R(k)%resPn(R(k)%nbPn,2) = mis
              if (D(i)%datatps(j)%andS=='S') then
                R(k)%nbSn = R(k)%nbSn + 1
                R(k)%resSn(R(k)%nbSn,1) = D(i)%datatps(j)%dTS
                R(k)%resSn(R(k)%nbSn,2) = mis
              endif
            else
              write(*,*)'problème dans inR : onde ni directe ni réfractée ... ? '
              stop
            endif
          endif
        enddo
      enddo
    enddo
    ! ------------------------------------------------------------------     .
end subroutine inR

 ! ------------------------------------------------------------------     .

subroutine outR(R,nbstaR)
    ! -------                                                 --------     .mh
    ! écriture des résidus dans des fichiers | (si FLAGresSTA=.true.)
    ! -------                                                 --------     .
    use typetemps, only : residus
    ! -------                                                 --------     .
    implicit none
    ! -------                                                 --------     .
    integer(KIND=wi), intent(in) :: nbstaR
    type(residus), intent(inout) :: R(nbstaR)                              ! résidus (si FLAGresSTA=.true.)
    ! -------                                                 --------     .
    integer(KIND=wi) :: i,j,noctet,ok
    real(KIND=wr) :: min,max,val
    ! ------------------------------------------------------------------     .
    ok = 0
    i = 0
    do while (ok == 0)
      i=i+1
      if (R(i)%nbPg.gt.1) then                                             ! il existe des ondes directes
        inquire (iolength = noctet)R(i)%resPg(1,1)
        ok=1
      elseif (R(i)%nbPn.gt.1) then
        inquire (iolength = noctet)R(i)%resPn(1,1)                         ! ou sinon, au moins des réfractées
        ok=1
      endif
      if (i.gt.nbstaR) then
        write(*,*)'problème dans outR', i
        stop
      endif
    enddo
    ! -------                                                 --------     .
    do i=1,nbstaR
      ! ------------------------------------------------------------------     . Pg
      if (R(i)%nbPg.gt.0) then
        open(unit=1000,file="OUTPUT/files/STA/"//R(i)%staname//"-PG.bin",access='direct',RECL=(noctet))
        min=1.e9_wr
        max=-1.e9_wr
```

```
      do j=1,R(i)%nbPg
        if(min.gt.R(i)%resPg(j,2)) min=R(i)%resPg(j,2)
        if(max.lt.R(i)%resPg(j,2)) max=R(i)%resPg(j,2)
      enddo
      ! --------                                                  ---------     .
      ! on garde dans les 10 meilleurs %
      ! c'est pas très propre, mais ca a le mérite d'être rapide !
      val=min+(max-min)/10.0_wr
      ! --------                                                  ---------     .
      do j=1,R(i)%nbPg
        if(R(i)%resPg(j,1).le.val) write(1000,rec=R(i)%nbPgT+j)R(i)%resPg(j,1)
      enddo
      close(1000)
      R(i)%nbPgT=R(i)%nbPgT+R(i)%nbPg
      R(i)%nbPg=0
    endif
    ! ------------------------------------------------------------------     . Pn
    if (R(i)%nbPn.gt.0) then
      open(unit=1000,file="OUTPUT/files/STA/"//R(i)%staname//"-PN.bin",access='direct',RECL=(noctet))
      min=1.e9_wr
      max=-1.e9_wr
      do j=1,R(i)%nbPn
        if(min.gt.R(i)%resPn(j,2)) min=R(i)%resPn(j,2)
        if(max.lt.R(i)%resPn(j,2)) max=R(i)%resPn(j,2)
      enddo
      ! --------                                                  ---------     .
      ! on garde dans les 10 meilleurs %
      ! c'est pas très propre, mais ca a le mérite d'être rapide !
      val=min+(max-min)/10.0_wr
      ! --------                                                  ---------     .
      do j=1,R(i)%nbPn
        if(R(i)%resPn(j,1).le.val) write(1000,rec=R(i)%nbPnT+j)R(i)%resPn(j,1)
      enddo
      close(1000)
      R(i)%nbPnT=R(i)%nbPnT+R(i)%nbPn
      R(i)%nbPn=0
    endif
    ! ------------------------------------------------------------------     . Sg
    if (R(i)%nbSg.gt.0) then
      open(unit=1000,file="OUTPUT/files/STA/"//R(i)%staname//"-SG.bin",access='direct',RECL=(noctet))
      min=1.e9_wr
      max=-1.e9_wr
      do j=1,R(i)%nbSg
        if(min.gt.R(i)%resSg(j,2)) min=R(i)%resSg(j,2)
        if(max.lt.R(i)%resSg(j,2)) max=R(i)%resSg(j,2)
      enddo
      ! --------                                                  ---------     .
      ! on garde dans les 10 meilleurs %
      ! c'est pas très propre, mais ca a le mérite d'être rapide !
      val=min+(max-min)/10.0_wr
      ! --------                                                  ---------     .
      do j=1,R(i)%nbSg
        if(R(i)%resSg(j,1).le.val) write(1000,rec=R(i)%nbSgT+j)R(i)%resSg(j,1)
      enddo
      close(1000)
      R(i)%nbSgT=R(i)%nbSgT+R(i)%nbSg
      R(i)%nbSg=0
    endif
    ! ------------------------------------------------------------------     . Sn
    if (R(i)%nbSn.gt.0) then
      open(unit=1000,file="OUTPUT/files/STA/"//R(i)%staname//"-SN.bin",access='direct',RECL=(noctet))
      min=1.e9_wr
      max=-1.e9_wr
      do j=1,R(i)%nbSn
        if(min.gt.R(i)%resSn(j,2)) min=R(i)%resSn(j,2)
```

```
          if(max.lt.R(i)%resSn(j,2)) max=R(i)%resSn(j,2)
        enddo
        ! --------                                                    ---------    .
        ! on garde dans les 10 meilleurs %
        ! c'est pas très propre, mais ca a le mérite d'être rapide !
        val=min+(max-min)/10.0_wr
        ! --------                                                    ---------    .
        do j=1,R(i)%nbSn
          if(R(i)%resSn(j,1).le.val) write(1000,rec=R(i)%nbSnT+j)R(i)%resSn(j,1)
        enddo
        close(1000)
        R(i)%nbSnT=R(i)%nbSnT+R(i)%nbSn
        R(i)%nbSn=0
      endif
      ! ------------------------------------------------------------------    .
    enddo
    ! ------------------------------------------------------------------    .
  end subroutine outR

    ! ------------------------------------------------------------------    .

  subroutine moy_ec(param,size,n,moy,ec)
    ! --------                                                    ---------    .mh
    ! calcul moy et ecart-type des n meilleurs modèles
    ! --------                                                    ---------    .
    implicit none
    ! --------                                                    ---------    .
    integer(KIND=wi), intent(in) :: size, n
    real(KIND=wr), intent(in) :: param(size)
    real(KIND=wr), intent(out) :: moy,ec
    ! --------                                                    ---------    .
    integer(KIND=wi) :: i
    ! ------------------------------------------------------------------    .
    if (n.le.size) then
      moy = 0.0_wr
      ec = 0.0_wr
      do i = 1,n
        moy =  moy + param(i)
      enddo
      moy =  moy / real(n,wr)
      do i = 1,n
        ec = ec + (param(i)-moy)**2.0_wr
      enddo
      ec = sqrt(ec/real(n,wr))
    else
      write(*,*)'problème dans moy_ec : calcul de moyenne impossible'
      stop
    endif
    ! ------------------------------------------------------------------    .
  end subroutine moy_ec

    ! ------------------------------------------------------------------    .

  subroutine dist_apriori(j,rang,nbtps,D,nbm_ap,temps_ref,pEpis,nb,acentroid)
    ! --------                                                    ---------    .mh
    !  calcul une densité de probabilité a priori pour chaque paramètre
    !  c'est à dire un point de départ de la Chaine de Markov
    !  de plus, la méthode Geiger est appliqué sur chaque séismes avec des modèles aléatoirs de terre
    ! ------------------------------------------------------------------    .
    use typetemps, only : dataall, date_sec, parametre, parametres, priorEPI, amoho_centroid, mvPall_2_P1
    use time, only : basetime,difftime
    use tri, only : triparam
    use mt19937
    use cpt_temps
    use invGEIGER
```

```
1282    use rechercheepi
1283    ! -------                                                  --------     .
1284    implicit none
1285    integer(KIND=wi), intent (in) :: j
1286    integer(KIND=wi), intent (in) :: rang
1287    integer(KIND=wi), intent(in) :: nbtps(nbseismes)
1288    type(dataall), intent(inout) :: D(nbseismes)
1289    integer(KIND=wi), intent(in) :: nbm_ap
1290    type(date_sec), intent(in) :: temps_ref(nbseismes)
1291    type(priorEPI), intent (inout) :: pEpis(nbseismes)
1292    integer(KIND=wi), intent (in) :: nb                                ! au carré, nb de cases possible
1293    type (amoho_centroid), intent (in) :: acentroid
1294    ! -------                                                  --------     .
1295    type(parametres), allocatable :: papriori(:)
1296    type(parametre), allocatable :: one_param(:)
1297    real(KIND=wr), allocatable :: valmis(:)
1298    type(parametre) :: pgeiger,pgeigertest,pgeigermoy,pgeigerec
1299    integer(KIND=wi) :: i,n,l,k
1300    real(KIND=wr) :: val,moy,ec,moyMIS,ecMIS
1301    character(len=5) :: x
1302    character(LEN=30) :: one_string,two_string
1303    real(KIND=wr) :: onemisfit
1304    logical :: conv                                                ! "true" si geiger convergent
1305    ! ------------------------------------------------------------     .
1306    character (LEN=1) :: mod
1307    ! ------------------------------------------------------------     .
1308    allocate(papriori(nbm_ap))
1309    ! ------------------------------------------------------------     .
1310    ! on ne sauve les modèles de terre que pour un rang :
1311    ! ------------------------------------------------------------     .
1312    ! ------------------------------------------------------------     .
1313    write(one_string(1:30),'(a30)')"                                                "
1314    if (rang==0) then
1315      open(unit=992,file="OUTPUT/GMT/aprio_vc-1.bin",STATUS="replace",access='direct',RECL=8)
1316      open(unit=993,file="OUTPUT/GMT/aprio_vm-1.bin",STATUS="replace",access='direct',RECL=8)
1317      open(unit=994,file="OUTPUT/GMT/aprio_zm-1.bin",STATUS="replace",access='direct',RECL=8)
1318      open(unit=995,file="OUTPUT/GMT/apriovps-1.bin",STATUS="replace",access='direct',RECL=8)
1319    endif
1320    ! -------                                                  --------     .
1321    do i=1,nbm_ap
1322      if (rang==0) write(one_string(1:30),'(a20,1x,i4,a1,i4)')" densité apriori:   ",j,"/",nbseismes
1323      if (rang==0) call progress(i,nbm_ap,one_string)
1324      ! -------                                                --------     .
1325      call initparam(nbtps,D,papriori(i),pEpis,nb)
1326      ! -------                                                --------     . apriori
1327      if (rang==0) then
1328        write(992,REC=i)real(papriori(i)%VC,8)
1329        write(993,REC=i)real(papriori(i)%VM,8)
1330        write(994,REC=i)real(papriori(i)%Zmoho,8)
1331        write(995,REC=i)real(papriori(i)%VpVs,8)
1332      endif
1333    enddo
1334    ! -------                                                  --------     . fin fichiers pour le modele de terre
1335    if (rang==0) then
1336      close(992)
1337      close(993)
1338      close(994)
1339      close(995)
1340    endif
1341    ! ------------------------------------------------------------     . apriori
1342    write(x(1:5),'(i5)')j
1343    open(unit=996,file="OUTPUT/GMT/aprio_zh-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
1344    open(unit=997,file="OUTPUT/GMT/apriolon-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
1345    open(unit=998,file="OUTPUT/GMT/apriolat-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
1346    open(unit=999,file="OUTPUT/GMT/aprio_to-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
```

```
    do i=1,nbm_ap
      write(996,REC=i)real(papriori(i)%Zhypo(j),8)
      write(997,REC=i)real(papriori(i)%Lon(j),8)
      write(998,REC=i)real(papriori(i)%Lat(j),8)
      call difftime(val,papriori(i)%Tzero(j),temps_ref(j))
      write(999,REC=i)real(val,8)
    enddo
    close(996)
    close(997)
    close(998)
    close(999)
    write(two_string(1:30),'(a30)')"                                                    "
    ! ------------------------------------------------------------------    .
    ! ------------------------------------------------------------------    .
    write(x(1:5),'(i5)')j
    ! -------                                                 --------    .
    open(unit=1996,file="OUTPUT/GMT/geiger_zh-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
    open(unit=1997,file="OUTPUT/GMT/geigerlon-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
    open(unit=1998,file="OUTPUT/GMT/geigerlat-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
    open(unit=1999,file="OUTPUT/GMT/geiger_to-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=8)
    do i=1,nbm_ap/50
      ! -------                                                 --------    .
      if (rang==0) then
        write(two_string(1:30),'(a20,1x,i4,a1,i4)')" méthode Geiger : 1 ",j,"/",nbseismes
        call progress(i,nbm_ap/50,two_string)
      endif
      ! -------                                                 --------    . geiger
      mod='I'
      call mvPall_2_P1(pgeiger,papriori(i),j)
      call dogeigerone(j,nbtps(j),D(j)%datatps,pgeiger,acentroid,mod,onemisfit,chut=.true.,con=conv)
      ! -------                                                 --------    .
      write(1996,REC=i)real(pgeiger%Zhypo,8)
      write(1997,REC=i)real(pgeiger%Lon,8)
      write(1998,REC=i)real(pgeiger%Lat,8)

      call difftime(val,pgeiger%Tzero,temps_ref(j))
      write(1999,REC=i)real(val,8)
      ! -------                                                 --------    .
    enddo
    close(1996)
    close(1997)
    close(1998)
    close(1999)
    ! ------------------------------------------------------------------    .
    ! ------------------------------------------------------------------    .
    n=5                                                                     ! nb de modèles * toutes pfd ...
    ! ------------------------------------------------------------------    .
    ! avec les modèles d'Arroucau                                           !
    ! ------------------------------------------------------------------    .
    open(unit=2999,file="OUTPUT/GMT/Arroucau_geiger-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=40)
    l=0
    do i=1,n
      ! -------                                                 --------    .
      if (rang==0) then
        write(two_string(1:30),'(a20,1x,i4,a1,i4)')" méthode Geiger : 2 ",i,"/",n
        call progress(i,n,two_string)
      endif
      ! -------                                                 --------    . geiger
      call mvPall_2_P1(pgeiger,papriori(i),j)
      mod='A'
      pgeiger%Zhypo=genrand_real3()
      do while(pgeiger%Zhypo.lt.(32.0_wr-0.1_wr))
        pgeigertest=pgeiger
        call dogeigerone(j,nbtps(j),D(j)%datatps,pgeigertest,acentroid,mod,onemisfit,chut=.true.,con=conv)
        if (conv) then
```

```
1412            l=l+1
1413            call difftime(val,pgeigertest%Tzero,temps_ref(j))
1414            write(2999,REC=l)real(onemisfit,8),real(pgeigertest%Zhypo,8),real(pgeigertest%Lon,8),real(pgeigertest%Lat,8),real(val,8)
1415          endif
1416          pgeiger%Zhypo=pgeiger%Zhypo+5.0_wr*genrand_real3()
1417        enddo
1418
1419        ! -------                                                 --------     .
1420      enddo
1421      ! -------                                                 --------     . relecture
1422      if (l.ge.3) then
1423        ! -------                                                 --------     .
1424        allocate(valmis(l),one_param(l))
1425        ! -------                                                 --------     .
1426        do i=1,l
1427          read(2999,REC=i)onemisfit,one_param(i)%Zhypo,one_param(i)%Lon,one_param(i)%Lat,val
1428          valmis(i)=onemisfit
1429          one_param(i)%Tzero=temps_ref(j)
1430          one_param(i)%Tzero%sec = one_param(i)%Tzero%sec + val
1431          call basetime(one_param(i)%Tzero)
1432          one_param(i)%VC=0.0_wr
1433          one_param(i)%VM=0.0_wr
1434          one_param(i)%Zmoho=0.0_wr
1435          one_param(i)%VpVs=0.0_wr
1436        enddo
1437        ! -------                                                 --------     . tri croissant
1438        call triparam(l,valmis,one_param)
1439        ! -------                                                 --------     . calcul moy et ecart-type des i meilleurs modèles
1440        i=max(2,3*l/4)
1441        call moy_ec(valmis,l,i,moyMIS,ecMIS)
1442        call moy_ec(one_param(:)%Zhypo,l,i,moy,ec)
1443        pgeigermoy%Zhypo=moy
1444        pgeigerec%Zhypo=ec
1445        call moy_ec(one_param(:)%Lat,l,i,moy,ec)
1446        pgeigermoy%Lat=moy
1447        pgeigerec%Lat=ec
1448        call moy_ec(one_param(:)%Lon,l,i,moy,ec)
1449        pgeigermoy%Lon=moy
1450        pgeigerec%Lon=ec
1451        valmis=0.0_wr
1452        open(unit=299,file="OUTPUT/GMT/ArrALL-"//trim(adjustl(x))//".txt",STATUS="replace")
1453        open(unit=298,file="OUTPUT/GMT/ArrALLt-"//trim(adjustl(x))//".txt",STATUS="replace")
1454        do k=1,i
1455          call difftime(valmis(k),one_param(k)%Tzero,temps_ref(j))
1456          write(299,*)real(one_param(k)%Lon,8),real(one_param(k)%Lat,8)
1457          write(298,*)real(one_param(k)%Lon,8),real(one_param(k)%Lat,8),"10 0 5 LM Arroucau"
1458        enddo
1459        close(298)
1460        close(299)
1461        call moy_ec(valmis,l,i,moy,ec)
1462        deallocate(valmis,one_param)
1463        ! -------                                                 --------     .
1464        open(unit=2998,file="OUTPUT/GMT/Arroucau_f-"//trim(adjustl(x))//".d",STATUS="replace")
1465        write(2998,*)i,moyMIS,ecMIS
1466        write(2998,*)pgeigermoy%Zhypo,pgeigermoy%Lon,pgeigermoy%Lat,moy
1467        write(2998,*)pgeigerec%Zhypo,pgeigerec%Lon,pgeigerec%Lat,ec
1468        close(2998)
1469        open(unit=2997,file="OUTPUT/GMT/Arroucau_all-"//trim(adjustl(x))//".tex",STATUS="replace")
1470        write(2997,*)"\begin{table}[!ht] \scriptsize \centering  \renewcommand{\arraystretch}{1.5}"
1471        write(2997,*)"\begin{tabular}{|>{\centering}m{2.5cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
1472        write(2997,*)"|>{\centering}m{2.2cm}|>{\centering}m{2.2cm}|m{2.1cm}<{\centering}|}"
1473        write(2997,*)"\hline "
1474        write(2997,*)"{\bf \large mod\'ele} & {fonction co\^ut} & {Z$_{hypo}$ (km)} & {longitude (\degree)} & ", &
1475          " {latitude (\degree)} & {T$_{z\acute{e}ro}$ (s)} \\"
1476        write(2997,*)"\hline "
```

```
      write(2997,*)"\hline "
      write(2997,5007)"mod\'eles de terre de Arroucau & \np{",real(moyMIS,wr), &
        "} $\pm$ \np{",2.0_wr*real(ecMIS,wr),"} & \np{",pgeigermoy%Zhypo,"} $\pm$ \np{",2.0_wr*pgeigerec%Zhypo, &
        "} & \np{",pgeigermoy%Lon,"} $\pm$ \np{",2.0_wr*pgeigerec%Lon,"} & \np{",pgeigermoy%Lat, &
        "} $\pm$ \np{",2.0_wr*pgeigerec%Lat,"} & \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} \\ "
      write(2997,*)"\hline "
      write(2997,*)"\end{tabular}"
      write(2997,*)"\end{table}"
      close(2997)
    else
      open(unit=2998,file="OUTPUT/GMT/Arroucau_f-"//trim(adjustl(x))//".d",STATUS="replace")
      write(2998,*)"0 0.0 0.0"
      write(2998,*)"0.0 0.0 0.0 0.0"
      write(2998,*)"0.0 0.0 0.0 0.0"
      close(2998)
      open(unit=2997,file="OUTPUT/GMT/Arroucau_all-"//trim(adjustl(x))//".tex",STATUS="replace")
      write(2997,*)" "
      close(2997)
      open(unit=299,file="OUTPUT/GMT/ArrALL-"//trim(adjustl(x))//".txt",STATUS="replace")
      write(299,*)"0 0.0 0.0"
      close(299)
      open(unit=298,file="OUTPUT/GMT/ArrALLt-"//trim(adjustl(x))//".txt",STATUS="replace")
      write(298,*)"0 0.0 0.0 10 0 5 LM Arroucau"
      close(298)
    endif
    close(2999)

    ! ------------------------------------------------------------------     .
    ! avec le modèle Si-HEx                                                  !
    ! ------------------------------------------------------------------     .

    open(unit=3999,file="OUTPUT/GMT/SiHex_geiger-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=40)
    l=0
    do i=1,n
      ! -------                                                 ---------     .
      if (rang==0) then
        write(two_string(1:30),'(a20,1x,i4,a1,i4)')" méthode Geiger : 3 ",i,"/",n
        call progress(i,n,two_string)
      endif
      ! -------                                                 ---------     . geiger
      call mvPall_2_P1(pgeiger,papriori(i),j)
      mod='S'
      pgeiger%Zhypo=genrand_real3()
      do while(pgeiger%Zhypo.le.(30.0_wr-0.1_wr))
        pgeigertest=pgeiger
        call dogeigerone(j,nbtps(j),D(j)%datatps,pgeigertest,acentroid,mod,onemisfit,chut=.true.,con=conv)
        if (conv) then
          l=l+1
          call difftime(val,pgeigertest%Tzero,temps_ref(j))
        write(3999,REC=l)real(onemisfit,8),real(pgeigertest%Zhypo,8),real(pgeigertest%Lon,8),real(pgeigertest%Lat,8),real(val,8)
        endif
        pgeiger%Zhypo=pgeiger%Zhypo+5.0_wr*genrand_real3()
      enddo
      ! -------                                                 ---------     .
    enddo
    ! -------                                                 ---------     . relecture
    if (l.ge.3) then
      ! -------                                                 ---------     .
      allocate(valmis(l),one_param(l))
      ! -------                                                 ---------     .
      do i=1,l
        read(3999,REC=i)onemisfit,one_param(i)%Zhypo,one_param(i)%Lon,one_param(i)%Lat,val
        valmis(i)=onemisfit
        one_param(i)%Tzero=temps_ref(j)
        one_param(i)%Tzero%sec = one_param(i)%Tzero%sec + val
```

```
1542          call basetime(one_param(i)%Tzero)
1543          one_param(i)%VC=0.0_wr
1544          one_param(i)%VM=0.0_wr
1545          one_param(i)%Zmoho=0.0_wr
1546          one_param(i)%VpVs=0.0_wr
1547        enddo
1548        ! -------                                                ---------    . tri croissant
1549        call triparam(l,valmis,one_param)
1550        ! -------                                                ---------    . calcul moy et ecart-type des i meilleurs modèles
1551        i=max(2,3*l/4)
1552        call moy_ec(valmis,l,i,moyMIS,ecMIS)
1553        call moy_ec(one_param(:)%Zhypo,l,i,moy,ec)
1554        pgeigermoy%Zhypo=moy
1555        pgeigerec%Zhypo=ec
1556        call moy_ec(one_param(:)%Lat,l,i,moy,ec)
1557        pgeigermoy%Lat=moy
1558        pgeigerec%Lat=ec
1559        call moy_ec(one_param(:)%Lon,l,i,moy,ec)
1560        pgeigermoy%Lon=moy
1561        pgeigerec%Lon=ec
1562        valmis=0.0_wr
1563        open(unit=399,file="OUTPUT/GMT/SiHexALL-"//trim(adjustl(x))//".txt",STATUS="replace")
1564        open(unit=398,file="OUTPUT/GMT/SiHexALLt-"//trim(adjustl(x))//".txt",STATUS="replace")
1565        do k=1,i
1566          call difftime(valmis(k),one_param(k)%Tzero,temps_ref(j))
1567          write(399,*)real(one_param(k)%Lon,8),real(one_param(k)%Lat,8)
1568          write(398,*)real(one_param(k)%Lon,8),real(one_param(k)%Lat,8),"10 0 5 LM SiHex"
1569        enddo
1570        close(399)
1571        close(398)
1572        call moy_ec(valmis,l,i,moy,ec)
1573        deallocate(valmis,one_param)
1574        ! -------                                                ---------    .
1575        open(unit=3998,file="OUTPUT/GMT/SiHex_f-"//trim(adjustl(x))//".d",STATUS="replace")
1576        write(3998,*)i,moyMIS,ecMIS
1577        write(3998,*)pgeigermoy%Zhypo,pgeigermoy%Lon,pgeigermoy%Lat,moy
1578        write(3998,*)pgeigerec%Zhypo,pgeigerec%Lon,pgeigerec%Lat,ec
1579        close(3998)
1580        open(unit=3996,file="OUTPUT/GMT/SiHex_all-"//trim(adjustl(x))//".tex",STATUS="replace")
1581        write(3996,*)"\begin{table}[!ht] \scriptsize \centering  \renewcommand{\arraystretch}{1.5}"
1582        write(3996,*)"\begin{tabular}{|>{\centering}m{2.5cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
1583        write(3996,*)"|>{\centering}m{2.2cm}|>{\centering}m{2.2cm}|m{2.1cm}<{\centering}|}"
1584        write(3996,*)"\hline "
1585        write(3996,*)"\hline "
1586        write(3996,5007)"mod\'eles de terre \textsc{Si--Hex} & \np{",real(moyMIS,wr), &
1587          "} $\pm$ \np{",2.0_wr*real(ecMIS,wr),"} & \np{",pgeigermoy%Zhypo,"} $\pm$ \np{",2.0_wr*pgeigerec%Zhypo, &
1588          "} & \np{",pgeigermoy%Lon,"} $\pm$ \np{",2.0_wr*pgeigerec%Lon,"} & \np{",pgeigermoy%Lat, &
1589          "} $\pm$ \np{",2.0_wr*pgeigerec%Lat,"} & \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} \\ "
1590        write(3996,*)"\hline "
1591        write(3996,*)"\end{tabular}"
1592        write(3996,*)"\end{table}"
1593        close(3996)
1594      else
1595        open(unit=3998,file="OUTPUT/GMT/SiHex_f-"//trim(adjustl(x))//".d",STATUS="replace")
1596        write(3998,*)"0 0.0 0.0"
1597        write(3998,*)"0.0 0.0 0.0 0.0"
1598        write(3998,*)"0.0 0.0 0.0 0.0"
1599        close(3998)
1600        open(unit=3996,file="OUTPUT/GMT/SiHex_all-"//trim(adjustl(x))//".tex",STATUS="replace")
1601        write(3996,*)" "
1602        close(3996)
1603        open(unit=399,file="OUTPUT/GMT/SiHexALL-"//trim(adjustl(x))//".txt",STATUS="replace")
1604        write(399,*)"0 0.0 0.0"
1605        close(399)
1606        open(unit=398,file="OUTPUT/GMT/SiHexALLt-"//trim(adjustl(x))//".txt",STATUS="replace")
```

```
      write(398,*)"0  0.0  0.0  10  0  5 LM SiHex"
      close(398)
    endif
    close(3999)

    ! ------------------------------------------------------------------     .
    ! avec le modèle du CEA                                                  !
    ! ------------------------------------------------------------------     .

    open(unit=4999,file="OUTPUT/GMT/CEA_geiger-"//trim(adjustl(x))//".bin",STATUS="replace",access='direct',RECL=40)
    l=0
    do i=1,n
      ! -------                                                  ---------     .
      if (rang==0) then
        write(two_string(1:30),'(a20,1x,i4,a1,i4)')" méthode Geiger : 4 ",i,"/",n
        call progress(i,n,two_string)
      endif
      ! -------                                                  ---------     . geiger
      call mvPall_2_P1(pgeiger,papriori(i),j)
      mod='C'
      pgeiger%Zhypo=genrand_real3()
      do while(pgeiger%Zhypo.le.(25.0_wr-0.1_wr))
        pgeigertest=pgeiger
        call dogeigerone(j,nbtps(j),D(j)%datatps,pgeigertest,acentroid,mod,onemisfit,chut=.true.,con=conv)
        if (conv) then
          l=l+1
          call difftime(val,pgeigertest%Tzero,temps_ref(j))
          write(4999,REC=l)real(onemisfit,8),real(pgeigertest%Zhypo,8),real(pgeigertest%Lon,8),real(pgeigertest%Lat,8),real(val,8)
        endif
        pgeiger%Zhypo=pgeiger%Zhypo+5.0_wr*genrand_real3()
      enddo
      ! -------                                                  ---------     .
    enddo
    ! -------                                                  ---------     . relecture
    if (l.ge.3) then
      ! -------                                                  ---------     .
      allocate(valmis(l),one_param(l))
      ! -------                                                  ---------     .
      do i=1,l
        read(4999,REC=i)onemisfit,one_param(i)%Zhypo,one_param(i)%Lon,one_param(i)%Lat,val
        valmis(i)=onemisfit
        one_param(i)%Tzero=temps_ref(j)
        one_param(i)%Tzero%sec = one_param(i)%Tzero%sec + val
        call basetime(one_param(i)%Tzero)
        one_param(i)%VC=0.0_wr
        one_param(i)%VM=0.0_wr
        one_param(i)%Zmoho=0.0_wr
        one_param(i)%VpVs=0.0_wr
      enddo
      ! -------                                                  ---------     . tri croissant
      call triparam(l,valmis,one_param)
      ! -------                                                  ---------     . calcul moy et ecart-type des i meilleurs modèles
      i=max(2,3*l/4)
      call moy_ec(valmis,l,i,moyMIS,ecMIS)
      call moy_ec(one_param(:)%Zhypo,l,i,moy,ec)
      pgeigermoy%Zhypo=moy
      pgeigerec%Zhypo=ec
      call moy_ec(one_param(:)%Lat,l,i,moy,ec)
      pgeigermoy%Lat=moy
      pgeigerec%Lat=ec
      call moy_ec(one_param(:)%Lon,l,i,moy,ec)
      pgeigermoy%Lon=moy
      pgeigerec%Lon=ec
      valmis=0.0_wr
      open(unit=499,file="OUTPUT/GMT/CEAALL-"//trim(adjustl(x))//".txt",STATUS="replace")
```

```
1672        open(unit=498,file="OUTPUT/GMT/CEAALLt-"//trim(adjustl(x))//".txt",STATUS="replace")
1673        do k=1,i
1674          call difftime(valmis(k),one_param(k)%Tzero,temps_ref(j))
1675          write(499,*)real(one_param(k)%Lon,8),real(one_param(k)%Lat,8)
1676          write(498,*)real(one_param(k)%Lon,8),real(one_param(k)%Lat,8),"10 0 5 LM CEA"
1677        enddo
1678        close(499)
1679        close(498)
1680        call moy_ec(valmis,l,i,moy,ec)
1681        deallocate(valmis,one_param)
1682        ! -------                                                 --------     .
1683        open(unit=4998,file="OUTPUT/GMT/CEA_f-"//trim(adjustl(x))//".d",STATUS="replace")
1684        write(4998,*)i,moyMIS,ecMIS
1685        write(4998,*)pgeigermoy%Zhypo,pgeigermoy%Lon,pgeigermoy%Lat,moy
1686        write(4998,*)pgeigerec%Zhypo,pgeigerec%Lon,pgeigerec%Lat,ec
1687        close(4998)
1688        open(unit=4996,file="OUTPUT/GMT/CEA_all-"//trim(adjustl(x))//".tex",STATUS="replace")
1689        write(4996,*)"\begin{table}[!ht] \scriptsize \centering  \renewcommand{\arraystretch}{1.5}"
1690        write(4996,*)"\begin{tabular}{|>{\centering}m{2.5cm}|>{\centering}m{2.1cm}|>{\centering}m{2.1cm}"
1691        write(4996,*)"|>{\centering}m{2.2cm}|>{\centering}m{2.2cm}|m{2.1cm}<{\centering}|}"
1692        write(4996,*)"\hline "
1693        write(4996,*)"\hline "
1694        write(4996,5007)"mod\'eles de terre \textsc{CEA} & \np{",real(moyMIS,wr), &
1695        "} $\pm$ \np{",2.0_wr*real(ecMIS,wr),"} & \np{",pgeigermoy%Zhypo,"} $\pm$ \np{",2.0_wr*pgeigerec%Zhypo, &
1696        "} & \np{",pgeigermoy%Lon,"} $\pm$ \np{",2.0_wr*pgeigerec%Lon,"} & \np{",pgeigermoy%Lat, &
1697        "} $\pm$ \np{",2.0_wr*pgeigerec%Lat,"} & \np{",moy,"} $\pm$ \np{",2.0_wr*ec,"} \\ "
1698        write(4996,*)"\hline "
1699        write(4996,*)"\end{tabular}"
1700        write(4996,*)"\end{table}"
1701        close(4996)
1702      else
1703        open(unit=4998,file="OUTPUT/GMT/CEA_f-"//trim(adjustl(x))//".d",STATUS="replace")
1704        write(4998,*)"0 0.0 0.0"
1705        write(4998,*)"0.0 0.0 0.0 0.0"
1706        write(4998,*)"0.0 0.0 0.0 0.0"
1707        close(4998)
1708        open(unit=4996,file="OUTPUT/GMT/CEA_all-"//trim(adjustl(x))//".tex",STATUS="replace")
1709        write(4996,*)" "
1710        close(4996)
1711        open(unit=499,file="OUTPUT/GMT/CEAALL-"//trim(adjustl(x))//".txt",STATUS="replace")
1712        write(499,*)"0 0.0 0.0"
1713        close(499)
1714        open(unit=498,file="OUTPUT/GMT/CEAALLt-"//trim(adjustl(x))//".txt",STATUS="replace")
1715        write(498,*)"0 0.0 0.0 10 0 5 LM CEA"
1716        close(498)
1717      endif
1718      close(4999)
1719      ! ---------------------------------------------------------------     .
1720
1721      ! ---------------------------------------------------------------     .
1722  5007 format (a,f15.2,a,f15.2,a,f9.2,a,f9.1,a,f9.4,a,f9.3,a,f9.4,a,f9.3,a,f9.2,a,f9.2,a)
1723      ! ---------------------------------------------------------------     .
1724      deallocate(papriori)
1725      ! ---------------------------------------------------------------     .
1726    end subroutine dist_apriori
1727
1728  END MODULE sub_param
1729
1730
1731
1732  ! ******************************************************************     .
1733  ! ******************************************************************     .
```

2.30 SRC/MOD/time.f90

```
! novembre 2013
! gère les problèmes liés au temps :
! calendrier Julien, base 60 ...
! ************************************************************      .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------     .
! ************************************************************      .
! ------------------------------------------------------------      .

MODULE time

    use modparam
    use typetemps, only : date_min, date_sec, date_secPS

    implicit none

    private

    public  :: tempszero
    public  :: basetime
    public  :: difftime
    public  :: JDATE,GDATE


    ! le temps est défini ici, soit en :
    ! - annee, mois, jour, heure, minutes et secondes
    ! - annee, mois, jour, heure, minutes, secondes pour ondes P, secondes pour ondes S
    ! les ondes S sont donc TOJOURS associées à des ondes P

    ! -------                                                 --------     .
    interface basetime
      module procedure basetimeP, basetimePS   ! basetime pour differents types
    end interface basetime
    ! -------                                                 --------     .
    interface difftime
      module procedure difftimeP, difftimePS  ! difftime pour differents types
    end interface difftime
    ! -------                                                 --------     .


CONTAINS

! ------------------------------------------------------------------      .

  subroutine JDATE (JD,YEAR,MONTH,DAY)
    ! -------                                                 --------     .mh
    ! COMPUTES THE JULIAN DATE (JD) GIVEN A GREGORIAN CALENDAR DATE (YEAR,MONTH,DAY)
    ! Reference: Fliegel, H. F. & van Flandern, T. C. 1968, Communications of the ACM, 11, 657.
    ! source : http://aa.usno.navy.mil/faq/docs/JD_Formula.php
    ! for years AD 1801-2099 :
    ! -------                                                 --------     .
    implicit none
    integer(KIND=wi), intent (out) :: JD
    integer(KIND=wi), intent (in) :: YEAR,MONTH,DAY
    integer(KIND=wi) :: I,J,K
    ! ------------------------------------------------------------------     .
    I= YEAR
    J= MONTH
    K= DAY
    JD= K-32075+1461*(I+4800+(J-14)/12)/4+ &
        367*(J-2-(J-14)/12*12)/12-3*((I+4900+(J-14)/12)/100)/4
    ! ------------------------------------------------------------------     .
  end subroutine

! ------------------------------------------------------------------      .

```

```
66  subroutine GDATE (JD,YEAR,MONTH,DAY)
67    ! _______                                                    ________     .mh
68    !COMPUTES THE GREGORIAN CALENDAR DATE (YEAR,MONTH,DAY) GIVEN THE JULIAN DATE (JD)
69    ! Reference: Fliegel, H. F. & van Flandern, T. C. 1968, Communications of the ACM, 11, 657.
70    ! source : http://aa.usno.navy.mil/faq/docs/JD_Formula.php
71    ! for years AD 1801-2099 :
72    ! _______                                                    ________     .
73    implicit none
74    integer(KIND=wi), intent (in) :: JD
75    integer(KIND=wi), intent (out) :: YEAR,MONTH,DAY
76    integer(KIND=wi) :: I,J,K,L,N
77    ! ---------------------------------------------------------------------     .
78    L= JD+68569
79    N= 4*L/146097
80    L= L-(146097*N+3)/4
81    I= 4000*(L+1)/1461001
82    L= L-1461*I/4+31
83    J= 80*L/2447
84    K= L-2447*J/80
85    L= J/11
86    J= J+2-12*L
87    I= 100*(N-49)+I+L
88    YEAR= I
89    MONTH= J
90    DAY= K
91    ! ---------------------------------------------------------------------     .
92  end subroutine
93
94 ! ---------------------------------------------------------------------     .
95
96  subroutine basetimePS(thedate)
97    ! _______                                                    ________     .mh
98    ! respect des bases 60 pour les sec et min, et 24 pour les heures
99    ! ansi que du découpage des années en mois et jours avec prise en compte des années bisextiles
100   ! les ondes P et S : type(date) identique -> secS peut-être >60 !!!
101   ! _______                                                    ________     .
102   implicit none
103   type(date_secPS), intent (inout) :: thedate
104   ! ---------------------------------------------------------------------     .
105   do while(thedate%secP.ge.60.0_wr)                                          ! pas plus de 60 sec dans une minute
106     thedate%secP=thedate%secP-60.0_wr
107     thedate%secS=thedate%secS-60.0_wr
108     thedate%date%min=thedate%date%min+1
109   enddo
110   ! _______                                                    ________     .
111   do while(thedate%secP.lt.0.0_wr)                                           ! pas moins de 0 sec dans une minute
112     thedate%secP=thedate%secP+60.0_wr
113     thedate%secS=thedate%secS+60.0_wr
114     thedate%date%min=thedate%date%min-1
115   enddo
116   ! _______                                                    ________     .
117   do while(thedate%date%min.ge.60)                                           ! pas plus de 60 min dans une heure
118     thedate%date%min=thedate%date%min-60
119     thedate%date%hour=thedate%date%hour+1
120   enddo
121   ! _______                                                    ________     .
122   do while(thedate%date%min.lt.0)                                            ! pas moins de 0 min dans une heure
123     thedate%date%min=thedate%date%min+60
124   thedate%date%hour=thedate%date%hour-1
125   enddo
126   ! _______                                                    ________     .
127   do while(thedate%date%hour.ge.24)                                          ! pas plus de 24 heures dans une journée
128     thedate%date%hour=thedate%date%hour-24
129     thedate%date%day=thedate%date%day+1
130     call JDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
```

```
131       call GDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
132     enddo
133     ! -------                                                  --------     .
134     do while(thedate%date%hour.lt.0)                                        ! pas moins de 0 heure dans une journée
135       thedate%date%hour=thedate%date%hour+24
136       thedate%date%day=thedate%date%day-1
137       call JDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
138       call GDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
139     enddo
140     ! ------------------------------------------------------------------     .
141   end subroutine basetimePS
142 
143 ! ------------------------------------------------------------------     .
144 
145   subroutine basetimeP(thedate)
146     ! -------                                                  --------     .mh
147     ! respect des bases 60 pour les sec et min, et 24 pour les heures
148     ! ansi que du decoupage des années en mois et jours avec prise en compte des années bisextiles
149     ! -------                                                  --------     .
150     implicit none
151     type(date_sec), intent (inout) :: thedate
152     ! ------------------------------------------------------------------     .
153     do while(thedate%sec.ge.60.0_wr)                                        ! pas plus de 60 sec dans une minute
154       thedate%sec=thedate%sec-60.0_wr
155       thedate%date%min=thedate%date%min+1
156     enddo
157     ! -------                                                  --------     .
158     do while(thedate%sec.lt.0.0_wr)                                         ! pas moins de 0 sec dans une minute
159       thedate%sec=thedate%sec+60.0_wr
160       thedate%date%min=thedate%date%min-1
161     enddo
162     ! -------                                                  --------     .
163     do while(thedate%date%min.ge.60)                                        ! pas plus de 60 min dans une heure
164       thedate%date%min=thedate%date%min-60
165       thedate%date%hour=thedate%date%hour+1
166     enddo
167     ! -------                                                  --------     .
168     do while(thedate%date%min.lt.0)                                         ! pas moins de 0 min dans une heure
169       thedate%date%min=thedate%date%min+60
170       thedate%date%hour=thedate%date%hour-1
171     enddo
172     ! -------                                                  --------     .
173     do while(thedate%date%hour.ge.24)                                       ! pas plus de 24 heure dans une journée
174       thedate%date%hour=thedate%date%hour-24
175       thedate%date%day=thedate%date%day+1
176       call JDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
177       call GDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
178     enddo
179     ! -------                                                  --------     .
180     do while(thedate%date%hour.lt.0)                                        ! pas moins de 0 heure dans une journée
181       thedate%date%hour=thedate%date%hour+24
182       thedate%date%day=thedate%date%day-1
183       call JDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
184       call GDATE (thedate%date%Jday,thedate%date%year,thedate%date%month,thedate%date%day)
185     enddo
186     ! ------------------------------------------------------------------     .
187   end subroutine basetimeP
188 
189 ! ------------------------------------------------------------------     .
190 
191   subroutine difftimePS(deltaP,deltaS,thedate1,thedate2)
192     ! -------                                                  --------     .mh
193     ! difference relative entre deux temps absolus, résultat en sec
194     ! ondes P et S
195     ! -------                                                  --------     .
```

```
196     implicit none
197     type(date_secPS), intent (in) :: thedate1,thedate2
198     real(KIND=wr), intent (out) :: deltaP,deltaS
199     real(KIND=wr) :: delta
200     ! ------------------------------------------------------------------    .
201     delta = real(thedate1%date%Jday-thedate2%date%Jday,wr)*24.0_wr*60.0_wr*60.0_wr + &
202     real(thedate1%date%hour-thedate2%date%hour,wr)*60.0_wr*60.0_wr + &
203     real(thedate1%date%min-thedate2%date%min,wr)*60.0_wr
204     ! -------                                                  --------    .
205     deltaP = delta + thedate1%secP - thedate2%secP
206     deltaS = delta + thedate1%secS - thedate2%secS
207     ! ------------------------------------------------------------------    .
208   end subroutine difftimePS
209
210 ! ------------------------------------------------------------------    .
211
212   subroutine difftimeP(deltatps,thedate1,thedate2)
213     ! -------                                                  --------    .mh
214     ! difference relative entre deux temps absolus, résultat en sec
215     ! -------                                                  --------    .
216     implicit none
217     type(date_sec), intent (in) :: thedate1,thedate2
218     real(KIND=wr), intent (out) :: deltatps
219     real(KIND=wr) :: delta
220     ! ------------------------------------------------------------------    .
221     delta = real(thedate1%date%Jday-thedate2%date%Jday,wr)*24.0_wr*60.0_wr*60.0_wr + &
222     real(thedate1%date%hour-thedate2%date%hour,wr)*60.0_wr*60.0_wr + &
223     real(thedate1%date%min-thedate2%date%min,wr)*60.0_wr
224     ! -------                                                  --------    .
225     deltatps = delta + thedate1%sec - thedate2%sec
226     ! ------------------------------------------------------------------    .
227   end subroutine difftimeP
228
229 ! ------------------------------------------------------------------    .
230
231    subroutine tempszero(atime)
232     ! -------                                                  --------    .mh
233     ! initialise le temps à zéro, pratique parfois ...
234     ! -------                                                  --------    .
235     implicit none
236     type(date_min), intent(out) :: atime
237     ! ------------------------------------------------------------------    .
238     atime%Jday=0
239     atime%year=0
240     atime%month=0
241     atime%day=0
242     atime%hour=0
243     atime%min=0
244     ! ------------------------------------------------------------------    .
245   end subroutine tempszero
246
247 END MODULE time
248
249
250
251 ! ********************************************************************    .
252 ! ********************************************************************    .
```

2.31 SRC/MOD/tirage.f90

```
1  ! Librairie de subroutines permettant le tirage aléatoire des paramètres
2  ! septembre 2013
3  ! ********************************************************************    .
4  ! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
5  ! ********************************************************************    .
```

```
! ------------------------------------------------------------------      .

MODULE tirage

    use modparam
    use typetemps, only : parametresinv, date_sec
    use mt19937

    implicit none

    private

    public  :: reflexion
    public  :: tirage_T, tirage_H, tirageUN


CONTAINS

    ! ------------------------------------------------------------------      .

  subroutine reflexion(valin,min,max,valout)
    ! -------                                                 ---------      .mh
    ! verifie si la valeur est dans le prior
    ! sinon : effectue une réflexion
    ! -------                                                 ---------      .
    implicit none
    real(KIND=wr),intent(in) :: valin,min,max
    real(KIND=wr),intent(out) :: valout
    ! -------                                                 ---------      .
    integer(KIND=wi) :: i
    ! ------------------------------------------------------------------      .
    ! -------                                                 ---------      .
    if (max.lt.min) write(*,*)'problème dans reflexion : ',max,'<',min
    i=0
    valout=valin
    do while ((valout.ge.max).or.(valout.le.min))                            ! tant que hors des bornes
      i=i+1
      ! -------                                               ---------      . réflexion supérieure
      if (valout.ge.max) then
        valout=valout-2.0_wr*abs(valout-max)
      end if
      ! -------                                               ---------      . réflexion inférieure
      if (valout.le.min) then
        valout=valout+2.0_wr*abs(min-valout)
      end if
      ! -------                                               ---------      .
      if(i.gt.250) then
        write(*,*)'problème dans reflexion : problème tirage hors bornes', valin,min,max,valout
        stop
      endif
      ! -------                                               ---------      . fin tant que
    enddo
    if (IsNaN(valout)) write(*,*)'problème dans reflexion : IsNaN(valout)', valin,min,max,valout
    ! ------------------------------------------------------------------      .
  end subroutine reflexion

! ------------------------------------------------------------------      .

  subroutine tirage_norm(vNew,vOld,vmini,vmaxi,vecartype)
    ! -------                                                 ---------      .mh
    ! tirage d'un paramètre selon une loi normale bornée
    ! modif : non !
    ! c'est plutot un changement de variable ! pour tiage normal
    ! -------                                                 ---------      .
    implicit none
```

```
71      real(KIND=wr),intent(in) :: vOld,vmini,vmaxi,vecartype
72      real(KIND=wr),intent(out) :: vNew
73      ! -------                                                  ---------     .
74      integer(KIND=wi) :: i
75      ! ---------------------------------------------------------------------     .
76      vNew = normal(vOld,vecartype)                                              ! tirage aléatoire
77      ! -------                                                  ---------     .
78      i=0
79      do while ((vNew.ge.vmaxi).or.(vNew.le.vmini))                              ! tant que hors des bornes
80        i=i+1
81        ! -------                                                ---------     . réflexion supérieure
82        if (vNew.ge.vmaxi) then
83          vNew=vNew-2.0_wr*abs(vNew-vmaxi)
84        end if
85        ! -------                                                ---------     . réflexion inférieure
86        if (vNew.le.vmini) then
87          vNew=vNew+2.0_wr*abs(vmini-vNew)
88        end if
89        ! -------                                                ---------     .
90        if(i.eq.200) then
91          write(*,*)'avertissement dans tirage_norm : tirage(param) -> iter > 100',vNew,vOld,vmini,vmaxi,vecartype
92          vNew = normal(vOld,vecartype)
93        endif
94        ! -------                                                ---------     .
95        if(i.gt.250) then
96          write(*,*)'problème dans tirage_norm : problème tirage_norm',vNew,vOld,vmini,vmaxi,vecartype
97          write(*,*)'souvent : si pas de contraintes sur moho ; il remonte et Zhypo est bien au delà !!!!! '
98          vNew=vmini+0.5_wr*abs(vmini-vmaxi)
99          write(*,*)'RECENTRE !',vNew
100         exit
101       endif
102       ! -------                                                ---------     . fin tant que
103     enddo
104     ! ---------------------------------------------------------------------     .
105   end subroutine tirage_norm
106
107 ! ---------------------------------------------------------------------     .
108
109   subroutine tirage_norm_lon(vNew,vOld,vmini,vmaxi,vecartype,lat,X,Y,R,dist_et)
110     ! -------                                                ---------     .mh
111     ! tirage de la longitude selon une loi normale bornée -> le prior de l'épicentre est ici un cercle !
112     ! -------                                                ---------     .
113     use dist_cercle
114     ! -------                                                ---------     .
115     implicit none
116     real(KIND=wr),intent(inout) :: vOld,vmini,vmaxi,vecartype
117     real(KIND=wr),intent(in) :: X,Y,R,lat,dist_et
118     real(KIND=wr),intent(out) :: vNew
119     integer(KIND=wi), save :: x_0=0
120     ! -------                                                ---------     .
121     integer(KIND=wi) :: i
122     real(KIND=wr) :: nR,Lo1,Lo2,La1,La2,alpha,eclat
123     ! ---------------------------------------------------------------------     . calcul de l'écart-type
124     eclat  = dist_et * 360.0_wr / ( 2.0_wr * pi * rT)
125     vecartype = eclat / sin((90.0_wr-lat)/180.0_wr*pi)
126     ! ---------------------------------------------------------------------     . calcul des bornes min et max
127     ! petit cercle : centre(X,Y) et rayon (R, en km)
128     nR=180.0_wr*R/pi/rT
129     ! petit cercle : de latitude (lat) : 2pts {x,lat}, pour tous x
130     alpha=90._wr-lat
131     ! -------                                                ---------     .
132     call dist2c(Y,90.0_wr,X,0.0_wr,nR,alpha,La1,Lo1,La2,Lo2)
133     ! ---------------------------------------------------------------------     .
134     if ((abs(Lo1)+abs(La1)+abs(Lo2)+abs(La2)).lt.1000._wr) then                ! une ou deux intersections
135       vmini=min(Lo1,Lo2,vOld)
```

```
136       vmaxi=max(Lo1,Lo2,vOld)
137       ! ------------------------------------------------------------------     .
138       if(isnan(vmini))stop
139       if(isnan(vmaxi))stop
140       ! ------------------------------------------------------------------     .
141       vNew = normal(vOld,vecartype)                                            ! tirage aléatoire
142       ! -------                                                 ---------      .
143       i=0
144       do while ((vNew.ge.vmaxi).or.(vNew.le.vmini))                            ! tant que hors des bornes
145         i=i+1
146         ! -------                                               ---------      . réflexion supérieure
147         if (vNew.ge.vmaxi) then
148           vNew=vNew-2.0_wr*abs(vNew-vmaxi)
149         end if
150         ! -------                                               ---------      . réflexion inférieure
151         if (vNew.le.vmini) then
152           vNew=vNew+2.0_wr*abs(vmini-vNew)
153         end if
154         ! -------                                               ---------      .
155         if(i.eq.200) then
156           write(*,*)'avertissement dans tirage_norm_lon : tirage(param) -> iter > 100', &
157           vNew,vOld,vmini,vmaxi,vecartype
158           vNew = normal(vOld,vecartype)
159         endif
160         ! -------                                               ---------      .
161         if(i.gt.250) then
162           if (abs(vmini-vmaxi).lt.0.0007_wr) then                              ! quelques mètres de différence  (~50m)
163             write(*,*)'avertissement dans tirage_norm_lon : quelques mètres de différence (~50m)',vmini,vmaxi
164             vNew = vOld
165           else
166             write(*,*)'problème dans tirage_norm_lon : problème tirage_norm', &
167               vNew,vOld,vmini,vmaxi,vecartype
168             stop
169           endif
170         endif
171         ! -------                                               ---------      . fin tant que
172       enddo
173     ! ------------------------------------------------------------------     .
174     else                                                                       ! pas d'intersections
175       vNew=vOld
176       x_0=x_0+1
177       if(x_0.gt.10*nbseismes) then
178         write(*,*)'problème dans tirage_norm_lon : pas d''intersection'
179         write(*,*)vNew,vOld,vmini,vmaxi
180         write(*,*)Y,90.0_wr,X,0.0_wr,nR,alpha,La1,Lo1,La2,Lo2
181       elseif(x_0.gt.250*nbseismes) then
182         write(*,*)'problème dans tirage_norm_lon : pas d''intersection > 250 fois'
183         stop
184       endif
185     endif
186     ! ------------------------------------------------------------------     .
187   end subroutine tirage_norm_lon
188
189 ! ------------------------------------------------------------------     .
190
191   subroutine tirage_norm_lat(vNew,vOld,vmini,vmaxi,vecartype,lon,X,Y,R)
192     ! -------                                                 ---------      .mh
193     ! tirage de la latitude selon une loi normale bornée -> le prior de l'épicentre est ici un cercle !
194     ! -------                                                 ---------      .
195     use dist_cercle
196     ! -------                                                 ---------      .
197     implicit none
198     real(KIND=wr),intent(inout) :: vOld,vmini,vmaxi,vecartype
199     real(KIND=wr),intent(in) :: X,Y,R,lon
200     real(KIND=wr),intent(out) :: vNew
```

```
    integer(KIND=wi), save :: x_0=0
    ! -------                                                    ---------    .
    integer(KIND=wi) :: i
    real(KIND=wr) :: nR,Lo1,Lo2,La1,La2,phi
    ! ------------------------------------------------------------------    . calcul des bornes min et max
    ! petit cercle : centre(X,Y) et rayon (R, en km)
    nR=180.0_wr*R/pi/rT ! deg
    ! grand cercle : de longitude (lon) : 2pts {lon,x}, pour tous x
    ! -------                                                    ---------    .
    phi=lon+90.0_wr
    call dist2c(0.0_wr,Y,phi,X,90.0_wr,nr,La1,Lo1,La2,Lo2)
    ! ------------------------------------------------------------------    .
    if ((abs(Lo1)+abs(La1)+abs(Lo2)+abs(La2)).lt.1000._wr) then            ! une ou deux intersections
      vmini=min(La1,La2,vOld)
      vmaxi=max(La1,La2,vOld)
      ! ----------------------------------------------------------------    .
      if(isnan(vmini))stop
      if(isnan(vmaxi))stop
      ! ----------------------------------------------------------------    .
      vNew = normal(vOld,vecartype)                                        ! tirage aléatoire
      ! -------                                                  ---------    .
      i=0
      do while ((vNew.ge.vmaxi).or.(vNew.le.vmini))                        ! tant que hors des bornes
        i=i+1
        ! -------                                                ---------    . réflexion supérieure
        if (vNew.ge.vmaxi) then
          vNew=vNew-2.0_wr*abs(vNew-vmaxi)
        end if
        ! -------                                                ---------    . réflexion inférieure
        if (vNew.le.vmini) then
          vNew=vNew+2.0_wr*abs(vmini-vNew)
        end if
        ! -------                                                ---------    .
        if(i.eq.200) then
          write(*,*)'avertissement dans tirage_norm_lat : tirage(param) -> iter > 250',vNew,vOld,vmini,vmaxi,vecartype
          vNew = normal(vOld,vecartype)
        endif
        ! -------                                                ---------    .
        if(i.gt.250) then
          if (abs(vmini-vmaxi).lt.0.0005_wr) then                          ! quelques mètres de différence (~50m)
            write(*,*)'avertissement dans tirage_norm_lat : quelques mètres de différence (~50m)',vmini,vmaxi
            vNew = vOld
          else
            write(*,*)'problème dans tirage_norm_lat : problème tirage_norm',vNew,vOld,vmini,vmaxi,vecartype
            stop
          endif
        endif
        ! -------                                                ---------    . fin tant que
      enddo
    ! ------------------------------------------------------------------    .
    else                                                                   ! pas d'intersections
      vNew=vOld
      x_0=x_0+1
      if(x_0.gt.10*nbseismes) then
        write(*,*)'problème dans tirage_norm_lat : pas d''intersection'
        write(*,*)vNew,vOld,vmini,vmaxi
        write(*,*)0.0_wr,Y,phi,X,90.0_wr,nr,La1,Lo1,La2,Lo2
      elseif(x_0.gt.250*nbseismes) then
        write(*,*)'problème dans tirage_norm_lat : pas d''intersection > 100 fois'
        stop
      endif
    endif
    ! ------------------------------------------------------------------    .
  end subroutine tirage_norm_lat
```

```
266 ! ------------------------------------------------------------------     .
267
268   subroutine tirage_inv_norm(vNew,vOld,vmini,vmaxi,vecartype)
269     ! _______                                                 ________    .mh
270     ! tirage de l'inverse d'un paramètre selon une densité de probabilité de type loi normale
271     ! _______                                                 ________    .
272     implicit none
273     real(KIND=wr),intent(in) :: vOld,vmini,vmaxi,vecartype
274     real(KIND=wr),intent(out) :: vNew
275     ! _______                                                 ________    .
276     real(KIND=wr) :: ivOld,ivmini,ivmaxi,ivecartype
277     ! _______                                                 ________    .
278     integer(KIND=wi) :: i
279     ! ------------------------------------------------------------------    .
280     ivOld=1.0_wr/vOld
281     ivmini=1.0_wr/vmaxi
282     ivmaxi=1.0_wr/vmini
283     ivecartype=vecartype/vOld/vOld
284     vNew = 1.0_wr/normal(ivOld,ivecartype)
285     ! _______                                                 ________    .
286     i=0
287     do while ((vNew.ge.vmaxi).or.(vNew.le.vmini))                         ! tant que hors des bornes
288       i=i+1
289       ! _______                                               ________    . réflexion supérieure
290       if (vNew.ge.vmaxi) then
291         vNew=vNew-2.0_wr*abs(vNew-vmaxi)
292       end if
293       ! _______                                               ________    . réflexion inférieure
294       if (vNew.le.vmini) then
295         vNew=vNew+2.0_wr*abs(vmini-vNew)
296       end if
297       ! _______                                               ________    .
298       if(i.eq.200) then
299         write(*,*)'avertissement dans tirage_inv_norm  : tirage_log(param) -> iter > 100',vNew
300         vNew = 1.0_wr/normal(ivOld,ivecartype)
301       endif
302       ! _______                                               ________    .
303       if(i.gt.250) then
304         write(*,*)'problème dans tirage_inv_norm : tirage_log_norm',vNew
305         stop
306       endif
307       ! _______                                               ________    . fin tant que
308     enddo
309     ! ------------------------------------------------------------------    .
310   end subroutine tirage_inv_norm
311
312 ! ------------------------------------------------------------------     .
313
314   subroutine tirage_log_norm(vNew,vOld,vmini,vmaxi,vecartype)
315     ! _______                                                 ________    .mh
316     ! tirage d'un paramètre selon une densité de probabilité identique pour un paramètre x et son inverse 1/x
317     ! attention : ce n'est ni un tirage type loi normale, ni un tirage type loi log-normale
318     ! log -> ln -> logarithme naturel (à base e = 2,71828... )
319     ! _______                                                 ________    .
320     implicit none
321     real(KIND=wr),intent(in) :: vOld,vmini,vmaxi,vecartype
322     real(KIND=wr),intent(out) :: vNew
323     ! _______                                                 ________    .
324     integer(KIND=wi) :: i
325     real(KIND=wr) :: sigma
326     ! ------------------------------------------------------------------    .
327     sigma=log((vOld+vecartype)/vOld)
328     vNew = exp(normal(log(vOld),sigma))                                   ! pour une distribution normale
329     ! _______                                                 ________    .
330     i=0
```

```
331     do while ((vNew.ge.vmaxi).or.(vNew.le.vmini))                    ! tant que hors des bornes
332       i=i+1
333       ! -------                                          ---------     . réflexion supérieure
334       if (vNew.ge.vmaxi) then
335         vNew=vNew-2.0_wr*abs(vNew-vmaxi)
336       end if
337       ! -------                                          ---------     . réflexion inférieure
338       if (vNew.le.vmini) then
339         vNew=vNew+2.0_wr*abs(vmini-vNew)
340       end if
341       ! -------                                          ---------     .
342       if(i.eq.200) then
343         write(*,*)'avertissement dans tirage_log_norm  : tirage_log(param) -> iter > 100',vNew
344         vNew = exp(normal(log(vOld),sigma))
345       endif
346       ! -------                                          ---------     .
347       if(i.gt.250) then
348         write(*,*)'problème dans tirage_log_norm : tirage_log_norm',vNew
349         stop
350       endif
351       ! -------                                          ---------     . fin tant que
352     enddo
353     ! ------------------------------------------------------------     .
354   end subroutine tirage_log_norm
355
356 ! ------------------------------------------------------------------     .
357
358   subroutine tirage_norm_time(vNew,vOld,vmini,vmaxi,vecartype)
359     ! -------                                          ---------     .mh
360     ! tirage d'un paramètre spécial (date) selon une loi normale
361     ! -------                                          ---------     .
362     use time
363     implicit none
364     type(date_sec),intent(in) :: vOld,vmini,vmaxi,vecartype
365     type(date_sec),intent(out) :: vNew
366     ! -------                                          ---------     .
367     type(date_sec) :: vnull
368     integer(KIND=wi) :: i
369     real(KIND=wr) :: delta,deltamin,deltamax
370     ! ------------------------------------------------------------     .
371     vNew=vOld                                                        ! on conserve la même date
372     ! -------                                          ---------     .
373     call tempszero(vnull%date)
374     vnull%sec=0.0_wr
375     call difftime(delta,vnull,vecartype)                             ! écartype en sec
376     ! -------                                          ---------     .
377     vNew%sec = normal(vOld%sec,abs(delta))                           ! tirage aléatoire
378     ! -------                                          ---------     .
379     call basetime(vNew)                                              ! reste en base 60/12/365 ...
380     ! -------                                          ---------     .
381     i=0
382     call difftime(deltamin,vNew,vmini)
383     call difftime(deltamax,vNew,vmaxi)
384     do while ((deltamax.ge.0.0_wr).or.(deltamin.le.0.0_wr))          ! tant que hors des bornes
385       i=i+1
386       ! -------                                          ---------     . réflexion supérieure
387       if (deltamax.ge.0.0_wr) then
388         vNew%sec=vNew%sec-2.0_wr*abs(deltamax)
389         call basetime(vNew)                                          ! reste en base 60/12/365 ...
390         call difftime(deltamax,vNew,vmaxi)
391       end if
392         ! -------                                          ---------     . réflexion inférieure
393       if (deltamin.le.0.0_wr) then
394         vNew%sec=vNew%sec+2.0_wr*abs(deltamin)
395         call basetime(vNew)                                          ! reste en base 60/12/365 ...
```

```
396         call difftime(deltamin,vNew,vmini)
397       end if
398       ! -------                                                  --------     .
399       if(i.eq.10) then
400         write(*,*)'avertissement dans tirage_norm_time : tirage(date) -> iter > 10',vNew%sec
401         vNew=vOld
402         vNew%sec = normal(vOld%sec,vecartype%sec)
403         call basetime(vNew)                                                 ! reste en base 60/12/365 ...
404       endif
405       ! -------                                                  --------     .
406       if(i.gt.50) then
407         write(*,*)'problème dans tirage_norm_time : tirage_norm_time'
408         stop
409       endif
410       ! -------                                                  --------     . fin tant que
411     enddo
412     ! ------------------------------------------------------------------     .
413   end subroutine tirage_norm_time
414
415 ! ------------------------------------------------------------------     .
416
417   subroutine tirage_H(p,i,all)
418     ! -------                                                  --------     .mh
419     ! tirage des tous les paramètres Hypocentraux simultanément ou un seul pour un séisme
420     ! les paramètres sont libres ou fixes (FLAGhypofixe)
421     ! -------                                                  --------     .
422     ! all = true : tire tous les parametres
423     ! all = false : tire au hasard, un des parametres
424     ! -------                                                  --------     . (FLAGhypofixe : MOD/modparam.f90)
425     ! FLAGhypofixe = true : tirage aléatoire selon une loi normale dont la moyenne est fixe
426     ! FLAGhypofixe = false : tirage aléatoire selon une loi normale
427     !   dont la moyenne correspond à l'iteration précédente (vrai McMC)
428     ! -------                                                  --------     .
429     use time
430     ! -------                                                  --------     .
431     implicit none
432     type(parametresinv), intent(inout) :: p
433     integer(KIND=wi), intent(in) :: i
434     logical, intent(in) :: all
435     ! -------                                                  --------     .
436     type(date_sec) :: tpsref
437     ! ------                                                   --------     .
438     real(KIND=wr) :: val, val2
439     integer(KIND=wi) :: pourcentage
440     logical :: P1,P2,P3,P4
441     ! ------------------------------------------------------------------     . tous les parametres ou un seul ?
442     if (all) then
443       P1=.true.
444       P2=.true.
445       P3=.true.
446       P4=.true.
447     else
448       pourcentage=int(genrand_real1()*100._wr)                          ! aléatoire de 0 à 99
449       P1=.false.
450       P2=.false.
451       P3=.false.
452       P4=.false.
453       select case (pourcentage)
454         case (0:24)
455           P1=.true.
456         case (25:49)
457           P2=.true.
458         case (50:74)
459           P3=.true.
460         case (75:99)
```

```
          P4=.true.
        end select
      endif
      ! ------------------------------------------------------------------      .
      if (FLAGhypofixe) then
        ! ------------------------------------------------------------------      . paramètres hypocentraux fixes
        val=p%mini%Lat(i)+(p%maxi%Lat(i)-p%mini%Lat(i))/2.0_wr
        if (P1) call tirage_norm(p%valNew%Lat(i),val,p%mini%Lat(i),p%maxi%Lat(i),p%ecartype%Lat(i))
        ! -------                                                  ---------      .
        val=p%mini%lon(i)+(p%maxi%lon(i)-p%mini%lon(i))/2.0_wr
        if (P2) call tirage_norm(p%valNew%lon(i),val,p%mini%lon(i),p%maxi%lon(i),p%ecartype%lon(i))
        ! -------                                                  ---------      .
        tpsref=p%mini%Tzero(i)
        tpsref%sec=tpsref%sec+3.0_wr*p%ecartype%Tzero(i)%sec
        if (P3) call tirage_norm_time(p%valNew%Tzero(i),tpsref,p%mini%Tzero(i),p%maxi%Tzero(i),p%ecartype%Tzero(i))
        ! -------                                                  ---------      .
        val2=p%maxi%Zhypo(i)
        if(p%maxi%Zhypo(i).gt.(p%valNew%Zmoho-0.1_wr)) then                     ! hypocentre dans la croûte
          val2=p%valNew%Zmoho-0.1_wr
        endif
        val=p%mini%Zhypo(i)+(p%maxi%Zhypo(i)-p%mini%Zhypo(i))/2.0_wr
        if (P4) call tirage_norm(p%valNew%Zhypo(i),val,p%mini%Zhypo(i),val2,p%ecartype%Zhypo(i))
        ! ------------------------------------------------------------------      .
      else
        ! ------------------------------------------------------------------      . paramètres hypocentraux libres
        if (P1) call tirage_norm_time(p%valNew%Tzero(i),p%valOld%Tzero(i),p%mini%Tzero(i),p%maxi%Tzero(i),p%ecartype%Tzero(i))
        ! -------                                                  ---------      .
        if (P2) call tirage_norm_lat(p%valNew%Lat(i),p%valOld%Lat(i),p%mini%Lat(i),p%maxi%Lat(i), &
          p%ecartype%Lat(i),p%valNew%Lon(i),p%centreX(i),p%centreY(i),p%Rayon(i))
        ! -------                                                  ---------      .
        if (P3) call tirage_norm_lon(p%valNew%Lon(i),p%valOld%Lon(i),p%mini%Lon(i),p%maxi%Lon(i), &
          p%ecartype%Lon(i),p%valNew%Lat(i),p%centreX(i),p%centreY(i),p%Rayon(i),p%ec_horizontal)
        ! -------                                                  ---------      .
        val2=p%maxi%Zhypo(i)
        if(p%maxi%Zhypo(i).gt.(p%valNew%Zmoho-0.1_wr)) then                     ! hypocentre dans la croûte
          val2=p%valNew%Zmoho-0.1_wr
        endif
        if (P4) call tirage_norm(p%valNew%Zhypo(i),p%valOld%Zhypo(i),p%mini%Zhypo(i),val2,p%ecartype%Zhypo(i))
        ! ------------------------------------------------------------------      .
      endif
      ! ------------------------------------------------------------------      .
    end subroutine tirage_H

! ----------------------------------------------------------------------      .

    subroutine tirage_T(p,all,vpvs)
      ! -------                                                  ---------      .mh
      ! tirage des tous les paramètres de Terre simultanément ou un seul
      ! les paramètres sont libres ou fixes (FLAGterrefixe)
      ! -------                                                  ---------      .
      ! all = true : tire tous les parametres
      ! all = false : tire au hasard, un des parametres
      ! vpvs = true : tire ou peut tirer VpVs
      ! vpvs = false : ne peut pas tirer VpVs
      ! -------                                                  ---------      . (FLAGterrefixe : MOD/modparam.f90)
      ! FLAGterrefixe = true : tirage aléatoire selon une loi normale dont la moyenne est fixe
      ! FLAGterrefixe = false : tirage aléatoire selon une loi normale
      !   dont la moyenne correspond à l'iteration précédente (vrai McMC)
      ! -------                                                  ---------      .
      implicit none
      type(parametresinv), intent(inout) :: p
      logical, intent(in) :: all, vpvs
      ! -------                                                  ---------      .
      real(KIND=wr) :: val,val2
      integer(KIND=wi) :: i,pourcentage
```

```
526     logical :: P1,P2,P3,P4
527     !-------------------------------------------------------------------     . tous les parametres ou un seul ?
528     if (all) then
529       P1=.true.
530       P2=.true.
531       P3=.true.
532       if (vpvs) then
533         P4=.true.
534       else
535         P4=.false.
536       endif
537     else
538       pourcentage=int(genrand_real1()*100._wr)                              ! aléatoire de 0 à 99
539       P1=.false.
540       P2=.false.
541       P3=.false.
542       P4=.false.
543       select case (pourcentage)
544         !case (0:24)
545         case (0:44)
546           P1=.true.
547         !case (25:49)
548         case (45:89)
549           P2=.true.
550         !case (50:74)
551         case (90:94)
552           P3=.true.
553         !case (75:99)
554         case (95:99)
555           if (vpvs) then
556             P4=.true.
557           else
558             P4=.false.
559             pourcentage=int(genrand_real1()*100._wr)                        ! aléatoire de 0 à 99
560             select case (pourcentage)
561               case (0:32)
562                 P1=.true.
563               case (33:65)
564                 P2=.true.
565               case (66:99)
566                 P3=.true.
567             end select
568           endif
569       end select
570     endif
571     !-------------------------------------------------------------------     .
572     if (FLAGterrefixe) then
573       !-----------------------------------------------------------------     . paramètres de terres fixes
574       if (P1) then
575         val=p%mini%Zmoho
576         do i=1,nbseismes
577           if(val.lt.(p%valNew%Zhypo(i)+0.1_wr)) then                        ! hypocentre dans la croûte
578             val=p%valNew%Zhypo(i)+0.1_wr
579           endif
580         enddo
581         val2=val+(p%maxi%Zmoho-val)/2.0_wr
582         call tirage_norm(p%valNew%Zmoho,val2,val,p%maxi%Zmoho,p%ecartype%Zmoho)
583       endif
584       !--------                                                 ---------     .
585       val2=p%maxi%VC
586       if(p%maxi%VC.gt.(p%valNew%VM-0.1_wr)) then                            ! respecte VM > VC
587         val2=p%valNew%VM-0.1_wr
588       endif
589       val=p%mini%VC+(p%maxi%VC-p%mini%VC)/2.0_wr
590       if (P2) call tirage_log_norm(p%valNew%VC,val,p%mini%VC,val2,p%ecartype%VC)
```

```
591       ! -------                                                  ---------     .
592       if (P3) then
593         val2=p%mini%VM
594         if(p%mini%VM.lt.(p%valNew%VC+0.1_wr)) then                             ! respecte VM > VC
595           val2=p%valNew%VC+0.1_wr
596         endif
597         val=p%mini%VM+(p%maxi%VM-p%mini%VM)/2.0_wr
598         call tirage_log_norm(p%valNew%VM,val,val2,p%maxi%VM,p%ecartype%VM)
599       endif
600       ! -------                                                  ---------     .
601       val=p%mini%VpVs+(p%maxi%VpVs-p%mini%VpVs)/2.0_wr
602       if (P4) call tirage_norm(p%valNew%VpVs,val,p%mini%VpVs,p%maxi%VpVs,p%ecartype%VpVs)
603       ! ---------------------------------------------------------------     .
604     else
605       ! ---------------------------------------------------------------     . paramètres de terres libres
606       if (P1) then
607         val=p%mini%Zmoho
608         do i=1,nbseismes
609           if(val.lt.(p%valNew%Zhypo(i)+0.1_wr)) then                         ! hypocentre dans la croûte
610             val=p%valNew%Zhypo(i)+0.1_wr
611           endif
612         enddo
613         call tirage_norm(p%valNew%Zmoho,p%valOld%Zmoho,val,p%maxi%Zmoho,p%ecartype%Zmoho)
614       endif
615       ! -------                                                  ---------     .
616       if (P2) call tirage_log_norm(p%valNew%VC,p%valOld%VC,p%mini%VC,p%maxi%VC,p%ecartype%VC)
617       ! -------                                                  ---------     .
618       if (P3) then
619         val=p%mini%VM
620         if(p%mini%VM.lt.(p%valNew%VC+0.1_wr)) then                             ! respecte VM > VC
621           val=p%valNew%VC+0.1_wr
622         endif
623         call tirage_log_norm(p%valNew%VM,p%valOld%VM,val,p%maxi%VM,p%ecartype%VM)
624       endif
625       ! -------                                                  ---------     .
626       if (P4) call tirage_norm(p%valNew%VpVs,p%valOld%VpVs,p%mini%VpVs,p%maxi%VpVs,p%ecartype%VpVs)
627     endif
628     ! ---------------------------------------------------------------     .
629   end subroutine tirage_T
630
631 ! ---------------------------------------------------------------     .
632
633   subroutine tirageUN(p,ap1,i)
634     ! -------                                                  ---------     .mh
635     ! tirage d'un seul paramètres : ap(1-8) pour le ième séisme
636     ! avec ap = 1->VC,2->VM,3->Zmoho,4->VpVs,5->Lat,6->Lon,7->Zhypo,8->Tzero
637     ! -------                                                  ---------     .
638     implicit none
639     type(parametresinv), intent(inout) :: p
640     integer(KIND=wi), intent(in) :: ap1                                    ! quel paramètre ?
641     integer(KIND=wi), intent(in) :: i                                      ! quel séisme ?
642     ! -------                                                  ---------     .
643     real(KIND=wr) :: val,val2
644     integer(KIND=wi) :: j,ap
645     ! ---------------------------------------------------------------     .
646     if (ap1.gt.8) then
647       ap=4+mod(ap1-5,4)+1
648     else
649       ap=ap1
650     endif
651     ! ---------------------------------------------------------------     . VC -> 1
652     if (ap==1) then
653       val=p%maxi%VC
654       if(p%maxi%VC.gt.(p%valNew%VM-0.1_wr)) then                           ! respecte VM > VC
655         val=p%valNew%VM-0.1_wr
```

```
      endif
      call tirage_log_norm(p%valNew%VC,p%valOld%VC,p%mini%VC,val,p%ecartype%VC)
    endif
    ! ------------------------------------------------------------------    . VM -> 2
    if (ap==2) then
      val=p%mini%VM
      if(p%mini%VM.lt.(p%valNew%VC+0.1_wr)) then                            ! respecte VM > VC
        val=p%valNew%VC+0.1_wr
      endif
      call tirage_log_norm(p%valNew%VM,p%valOld%VM,val,p%maxi%VM,p%ecartype%VM)
    endif
    ! ------------------------------------------------------------------    . Zmoho -> 3
    if (ap==3) then
      val=p%mini%Zmoho
      do j=1,nbseismes
        if(val.lt.(p%valNew%Zhypo(j)+0.1_wr)) then                          ! hypocentre dans la croûte
          val=p%valNew%Zhypo(j)+0.1_wr
        endif
      enddo
      call tirage_norm(p%valNew%Zmoho,p%valOld%Zmoho,val,p%maxi%Zmoho,p%ecartype%Zmoho)
    endif
    ! ------------------------------------------------------------------    . VpVs -> 4
    if (ap==4) call tirage_norm(p%valNew%VpVs,p%valOld%VpVs,p%mini%VpVs,p%maxi%VpVs,p%ecartype%VpVs)
    ! ------------------------------------------------------------------    . Lat(i) -> 5
    if (ap==5) call tirage_norm_lat(p%valNew%Lat(i),p%valOld%Lat(i),p%mini%Lat(i),p%maxi%Lat(i), &
      p%ecartype%Lat(i),p%valNew%Lon(i),p%centreX(i),p%centreY(i),p%Rayon(i))
    ! ------------------------------------------------------------------    . Lon(i) -> 6
    if (ap==6) call tirage_norm_lon(p%valNew%Lon(i),p%valOld%Lon(i),p%mini%Lon(i),p%maxi%Lon(i), &
      p%ecartype%Lon(i),p%valNew%Lat(i),p%centreX(i),p%centreY(i),p%Rayon(i),p%ec_horizontal)
    ! ------------------------------------------------------------------    . Zhypo(i) -> 7
    if (ap==7) then
      val2=p%maxi%Zhypo(i)
      if(p%maxi%Zhypo(i).gt.(p%valNew%Zmoho-0.1_wr)) then                   ! hypocentre dans la croûte
        val2=p%valNew%Zmoho-0.1_wr
      endif
      call tirage_norm(p%valNew%Zhypo(i),p%valOld%Zhypo(i),p%mini%Zhypo(i), &
        val2,p%ecartype%Zhypo(i))
    endif
    ! ------------------------------------------------------------------    . Tzero(i) -> 8
    if (ap==8) call tirage_norm_time(p%valNew%Tzero(i),p%valOld%Tzero(i),p%mini%Tzero(i), &
      p%maxi%Tzero(i),p%ecartype%Tzero(i))
    ! ------------------------------------------------------------------    .
  end subroutine tirageUN



END MODULE tirage



! *****************************************************************    .
! *****************************************************************    .
```

2.32 SRC/MOD/tri.f90

```
! Librairie de subroutines avec Algorithmes_de_tri/Tri_rapide (ou pas ...)
! septembre 2013
! *****************************************************************    .
! ------- Méric Haugmard meric.haugmard@univ-nantes.fr        --------    .
! *****************************************************************    .
! ------------------------------------------------------------------    .

MODULE tri

    use modparam
```

```
11
12      implicit none
13
14      private
15
16      public  :: tri_bulle
17      public  :: tridata
18      public  :: triparam
19      public  :: melangetab
20
21
22      ! ------                                                    --------    .
23      interface tri_bulle
24        module procedure tri_bulle_reel, tri_bulleRstring
25      end interface tri_bulle
26      ! ------                                                    --------    .
27      interface triparam
28        module procedure triparams, triparam1
29      end interface triparam
30      ! ------                                                    --------    .
31      interface melangetab
32        module procedure melangetab1, melangetab3
33      end interface melangetab
34      ! ------                                                    --------    .
35
36
37  CONTAINS
38
39      ! ------------------------------------------------------------------    .
40
41
42      subroutine tri_bulle_reel(InList,n,OutList)
43      ! ------                                                    --------    .
44      implicit none
45      ! ------                                                    --------    .
46      integer(KIND=wi), intent(in) :: n
47      real(KIND=wr), dimension(:,:), intent(in) :: InList
48      real(KIND=wr), dimension(:,:), intent(out) :: OutList
49      ! ------------------------------------------------------------------    .
50      if (size(InList,2)==2) then
51        call tri_bulle2col(InList,n,OutList)
52      else if (size(InList,2)==3) then
53        call tri_bulle3col(InList,n,OutList)
54      else
55        write(*,*)'problème dans tri_bulle, tableau non conforme ',size(InList,2)
56        stop
57      endif
58      ! ------------------------------------------------------------------    .
59
60      contains
61
62        subroutine tri_bulle2col(InList,n,OutList)
63          ! ------                                                --------    .
64          ! permet le tri croissant d'un vecteur de 2 dimensions
65          ! en fonction de la seconde colonne
66          ! ------                                                --------    .
67          implicit none
68          ! ------                                                --------    .
69          integer(KIND=wi), intent(in) :: n
70          real(KIND=wr), dimension(n,2), intent(in) :: InList
71          real(KIND=wr), dimension(n,2), intent(out) :: OutList
72          ! ------                                                --------    .
73          real(KIND=wr) :: pass(2)
74          integer(KIND=wi) :: i,j
75          real(KIND=wr) :: delta
```

```
76          ! ------------------------------------------------------------------
77          OutList=InList
78          do j=2,n
79            do i=j,2,-1
80              delta=OutList(i,2)-OutList(i-1,2)
81              if (delta.lt.0.0_wr) then
82                pass(:) = OutList(i-1,:)
83                OutList(i-1,:) = OutList(i,:)
84                OutList(i,:) = pass(:)
85              endif
86            enddo
87          enddo
88          ! ------------------------------------------------------------------
89        end subroutine tri_bulle2col
90
91          ! ------------------------------------------------------------------
92
93        subroutine tri_bulle3col(InList,n,OutList)
94          ! --------                                          ---------
95          ! permet le tri croissant d'un vecteur de 3 dimensions
96          ! en fonction de la troisième colonne
97          ! --------                                          ---------
98          implicit none
99          ! --------                                          ---------
100         integer(KIND=wi), intent(in) :: n
101         real(KIND=wr), dimension(n,3), intent(in) :: InList
102         real(KIND=wr), dimension(n,3), intent(out) :: OutList
103         ! --------                                          ---------
104         real(KIND=wr) :: pass(3)
105         integer(KIND=wi) :: i,j
106         real(KIND=wr) :: delta
107         ! ------------------------------------------------------------------
108         OutList=InList
109         do j=2,n
110           do i=j,2,-1
111             delta=OutList(i,3)-OutList(i-1,3)
112             if (delta.lt.0.0_wr) then
113               pass(:) = OutList(i-1,:)
114               OutList(i-1,:) = OutList(i,:)
115               OutList(i,:) = pass(:)
116             endif
117           enddo
118         enddo
119         ! ------------------------------------------------------------------
120       end subroutine tri_bulle3col
121
122   end subroutine tri_bulle_reel
123
124 ! ------------------------------------------------------------------
125
126   subroutine tri_bulleRstring(InOut_reel,InOut_string,n)
127     ! --------                                          ---------
128     ! permet le tri Dé-croissant d'un vecteur de 2 dimensions
129     ! en fonction de la seconde colonne
130     ! --------                                          ---------
131     implicit none
132     ! --------                                          ---------
133     integer(KIND=wi), intent(in) :: n
134     real(KIND=wr), dimension(n), intent(inout) :: InOut_reel
135     character(LEN=4), dimension(n), intent(inout) :: InOut_string
136     ! --------                                          ---------
137     character(LEN=4) :: passS
138     real(KIND=wr) :: passR
139     integer(KIND=wi) :: i,j
140     real(KIND=wr) :: delta
```

```
141     ! ------------------------------------------------------------------     .
142     do j=2,n
143       do i=j,2,-1
144         delta=InOut_reel(i)-InOut_reel(i-1)
145         if (delta.gt.0.0_wr) then
146           passR = InOut_reel(i-1)
147           passS = InOut_string(i-1)
148           InOut_reel(i-1) = InOut_reel(i)
149           InOut_string(i-1) = InOut_string(i)
150           InOut_reel(i) = passR
151           InOut_string(i) = passS
152         endif
153       enddo
154     enddo
155     ! ------------------------------------------------------------------     .
156   end subroutine tri_bulleRstring
157
158 ! ------------------------------------------------------------------------     .
159
160   subroutine tridata(nbtps,datatps)
161     ! -------                                                  ---------     .
162     ! permet le tri croissant d'un vecteur de type dataone en fonction des temps d'arrivés des ondes P
163     ! -------                                                  ---------     .
164     use typetemps, only : dataone
165     use time, only : difftime
166     ! -------                                                  ---------     .
167     implicit none
168     integer(KIND=wi),intent (in) :: nbtps
169     type(dataone),intent (inout) :: datatps(nbtps)
170     ! -------                                                  ---------     .
171     integer(KIND=wi) :: i,j
172     type(dataone) :: datapass
173     real(KIND=wr) :: deltaP,deltaS
174     ! ------------------------------------------------------------------     .
175     do j=2,nbtps
176       do i=j,2,-1
177         call difftime(deltaP,deltaS,datatps(i)%tpsR,datatps(i-1)%tpsR)
178         if (deltaP.lt.0.0_wr) then
179           datapass = datatps(i-1)
180           datatps(i-1) = datatps(i)
181           datatps(i) = datapass
182         endif
183       enddo
184     enddo
185     ! ------------------------------------------------------------------     .
186   end subroutine tridata
187
188 ! ------------------------------------------------------------------------     .
189
190   subroutine melangetab1(n,tab)
191     ! -------                                                  ---------     .
192     ! permet de mélanger un tableau par le mélange de Fisher-Yates (ou de Knuth) :
193     ! un algorithme générant une permutation aléatoire d'un ensemble fini
194     ! -------                                                  ---------     .
195     use mt19937
196     ! -------                                                  ---------     .
197     implicit none
198     integer(KIND=wi),intent (in) :: n
199     real(KIND=wr),intent (inout) :: tab(n)
200     ! -------                                                  ---------     .
201     integer(KIND=wi) :: i,j,pif
202     real(KIND=wr) :: pass
203     ! ------------------------------------------------------------------     .
204     do j=1,3                                                   ! 3 passages
205       do i=1,n
```

```
206         pif=1+int(genrand_real1()*real(n,wr),wi)                    ! equi-aléatoire de 1 à n
207         pass=tab(i)
208         tab(i)=tab(pif)
209         tab(pif)=pass
210       enddo
211     enddo
212     ! ------------------------------------------------------------------ .
213   end subroutine melangetab1
214
215 ! ------------------------------------------------------------------ .
216
217   subroutine melangetab3(n,tab)
218     ! -------                                          --------- .
219     ! permet de mélanger un tableau par le mélange de Fisher-Yates (ou de Knuth) :
220     ! un algorithme générant une permutation aléatoire d'un ensemble fini
221     ! -------                                          --------- .
222     use mt19937
223     ! -------                                          --------- .
224     implicit none
225     integer(KIND=wi),intent (in) :: n
226     real(KIND=wr),intent (inout) :: tab(3,n)
227     ! -------                                          --------- .
228     integer(KIND=wi) :: i,j,pif
229     real(KIND=wr) :: pass(3)
230     ! ------------------------------------------------------------------ .
231     do j=1,3                                                    ! 3 passages
232       do i=1,n
233         pif=1+int(genrand_real1()*real(n,wr),wi)                ! equi-aléatoire de 1 à n
234         pass(:)=tab(:,i)
235         tab(:,i)=tab(:,pif)
236         tab(:,pif)=pass(:)
237       enddo
238     enddo
239     ! ------------------------------------------------------------------ .
240   end subroutine melangetab3
241
242 ! ------------------------------------------------------------------ .
243
244   subroutine triparams(nb,misfit,param_best)
245     ! -------                                          --------- .
246     ! permet le tri croissant d'un jeu de paramètres en fonction de la fonction coût
247     ! pour tous les séismes
248     ! ------------------------------------------------------------------ .
249     use typetemps, only : parametres, fcout
250     ! -------                                          --------- .
251     implicit none
252     integer(KIND=wi), intent (in) :: nb
253     type(parametres), intent (inout) :: param_best(nb)
254     type(fcout), intent (inout) :: misfit(nb)
255     ! -------                                          --------- .
256     integer(KIND=wi) :: i,j
257     type(parametres) :: par
258     type(fcout) :: mis
259     ! ------------------------------------------------------------------ .
260     do j=2,nb
261       do i=j,2,-1
262         if (misfit(i)%best.lt.misfit(i-1)%best) then
263           mis = misfit(i-1)
264           misfit(i-1) = misfit(i)
265           misfit(i) = mis
266           par = param_best(i-1)
267           param_best(i-1) = param_best(i)
268           param_best(i) = par
269         endif
270       enddo
```

```
    enddo
    ! ------------------------------------------------------------------    .
  end subroutine triparams

! ------------------------------------------------------------------    .

  subroutine triparam1(nb,misfit,param_best)
    ! -------                                                 --------    .
    ! permet le tri croissant d'un jeu de paramètre en fonction de la fonction coût
    ! pour 1 séisme
    ! ------------------------------------------------------------------    .
    use typetemps, only : parametre
    ! -------                                                 --------    .
    implicit none
    integer(KIND=wi), intent (in) :: nb
    type(parametre), intent (inout) :: param_best(nb)
    real(KIND=wr), intent (inout)  :: misfit(nb)
    ! -------                                                 --------    .
    integer(KIND=wi) :: i,j
    type(parametre) :: par
    real(KIND=wr) :: mis
    ! ------------------------------------------------------------------    .
    do j=2,nb
      do i=j,2,-1
        if (misfit(i).lt.misfit(i-1)) then
          mis = misfit(i-1)
          misfit(i-1) = misfit(i)
          misfit(i) = mis
          par = param_best(i-1)
          param_best(i-1) = param_best(i)
          param_best(i) = par
        endif
      enddo
    enddo
    ! ------------------------------------------------------------------    .
  end subroutine triparam1

! ------------------------------------------------------------------    .

END MODULE tri



! ********************************************************************    .
! ********************************************************************    .
```

2.33 SRC/MOD/types.f90

```
! déclaration de l'ensemble des structures dérivées utilisées dans le programme CHE
! septembre 2013
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          --------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .


MODULE typetemps

    use modparam

    implicit none

    private

    ! type :
```

```
18      public  :: date_min, date_sec, date_secPS
19      public  :: stations
20      public  :: dataone, dataall
21      public  :: pond
22      public  :: parametre, parametres
23      public  :: parametresinv, paramisfit
24      public  :: densityplot_one, densityplot
25      public  :: fcout
26      public  :: accept
27      public  :: seismes
28      public  :: coldmoyval,coldmoy
29      public  :: ellip
30      public  :: residus
31      public  :: priorEPI
32      public  :: amoho_centroid
33
34      ! subroutines :
35      public  :: mvP1_2_Pall, mvPall_2_P1
36      public  :: vect2alph, alph2vect
37
38  ! ------------------------------------------------------------------ .
39
40    ! --------                                                 -------- ! date
41      TYPE date_min
42           integer(KIND=wi) :: Jday,year,month,day,hour,min              ! jours Julien, années, mois, jours du mois, heures et minutes en base 60, 24, 365,
                 12 ...
43      END TYPE date_min
44    ! --------                                                 -------- .
45    ! --------                                                 -------- ! date en secondes
46      TYPE date_sec
47           type(date_min) :: date                                        ! en minute
48           real(KIND=wr) :: sec                                          ! en secondes
49      END TYPE date_sec
50    ! --------                                                 -------- .
51    ! --------                                                 -------- ! date en secondes (avec arrivées des ondes S et P)
52      TYPE date_secPS
53           type(date_min) :: date                                        ! en minute
54           real(KIND=wr) :: secP,secS                                    ! en secondes
55      END TYPE date_secPS
56    ! --------                                                 -------- .
57    ! --------                                                 -------- ! caractéristiques du réseau sismologique utilisé
58      TYPE stations
59           character(LEN=4) :: staname                                   ! nom station
60           real(KIND=wr) :: lat,lon,alti                                 ! coordonnées station (degrée et m.a.s.l.)
61           real(KIND=wr) :: res_Pg,res_Pn,res_Sg,res_Sn                  ! résidus à la station, onde P et S
62      END TYPE stations
63    ! --------                                                 -------- .
64    ! --------                                                 -------- ! les DONNÉES, phase list (temps d'arrivées des ondes) pour 1 station, 1 séisme
65      TYPE dataone
66           type(stations) :: sta                                         ! données sur la station
67           type(date_secPS) :: tpsR, tpsTh                               ! temps d'arrivées des ondes R (réels) et Th (théorique)
68           character(LEN=1) :: typeonde                                  ! "G" pour onde directe, "N" pour réfractée au moho
69           integer(KIND=wi) :: coefP,coefS                               ! coef qualité de lecture (0(best)-4(worst))
70           character(LEN=1) :: andS                                      ! "S" si ondes S lues
71           real(KIND=wr) :: dTP,dTS                                      ! différence de temps d'arrivées des ondes R (réels) et Th (théorique), ondes P et S
72           real(KIND=wr) :: ws, wp                                       ! pondération des ondes P et S [0;1] (qualité et distance)
73           real(KIND=wr) :: tpsparcP,tpsparcS                            ! temps de parcours théorique des ondes P et S
74           real(KIND=wr) :: depi,dhypo                                   ! distance épi- et hypocentrale en km
75           real(KIND=wr) :: dcritiqueH                                   ! distance hypocentrale à partir de laquelle la réfraction commence, en km
76           real(KIND=wr) :: baz                                          ! back-azimuth en degrée
77           real(KIND=wr) :: sigP,sigS                                    ! écart-type sur les données en secondes
78      END TYPE dataone
79    ! --------                                                 -------- .
80    ! --------                                                 -------- ! les DONNÉES (temps d'arrivées des ondes) pour tous les séismes
81      TYPE dataall
```

```
82          type(dataone), dimension(:), allocatable :: datatps                 ! les DONNÉES (temps d'arrivées des ondes) pour 1 séisme
83      END TYPE dataall
84    ! --------                                                    --------      .
85    ! --------                                                    --------      ! coeficients de pondération pour chaque données
86      TYPE pond
87          real(KIND=wr) :: S_Pg, S_Sg, S_Pn, S_Sn                             ! somme des coef.
88          integer(KIND=wi) :: nPg, nPn ,nSg, nSn                              ! nombre de données pour chaque
89      END TYPE pond
90    ! --------
91    ! --------                                                    --------      ! ensemble des PARAMETRES de l'inversion pour plusieurs séismes
92      TYPE parametres
93          real(KIND=wr) :: VC,VM,Zmoho,VpVs                                   ! paramètres de terre
94          real(KIND=wr) :: Lat(nbseismes),Lon(nbseismes),Zhypo(nbseismes)     ! paramètres des hypocentres
95          type(date_sec) :: Tzero(nbseismes)                                  ! temps initial
96      END TYPE parametres
97    ! --------                                                    --------      .
98    ! --------                                                    --------      ! ensemble des parametres de l'inversion pour 1 seul séisme
99      TYPE parametre
100         real(KIND=wr) :: VC,VM,Zmoho,VpVs,Lat,Lon,Zhypo                     !
101         type(date_sec) :: Tzero                                             ! temps initial
102     END TYPE parametre
103   ! --------                                                    --------      .
104   ! --------                                                    --------      . valeurs des parametres au cours de l'inversion
105     TYPE parametresinv
106         type(parametres) :: valNew,valOld,mini,maxi,ecartype                ! à l'itération actuelle, précedente, valeur minimale, maximale et écart type de la
                gaussienne d'échantillonage
107         real(KIND=wr) :: centreX(nbseismes),centreY(nbseismes),Rayon(nbseismes),ec_horizontal
108     END TYPE parametresinv
109   ! --------                                                    --------      .
110   ! --------                                                    --------      . un modèle (i.e. misfit + jeu de paramètres)
111     TYPE paramisfit
112         type(parametres) :: par                                             ! un jeu de paramètre
113         real(KIND=wr) :: mis                                                ! valeur de la fonction coût
114     END TYPE paramisfit
115   ! --------                                                    --------      .
116   ! --------                                                    --------      ! relecture modèles sélectionnés et calculs a posteriori
117     TYPE densityplot_one
118         real(KIND=wr), dimension(:), allocatable :: vec                     ! ensemble de valeurs prise lors de l'inverssion pour ce parametre
119         character (LEN=30) :: char                                          ! chaîne de caractères pour les légendes GMT
120         character (LEN=3) :: name                                           ! nom en trois lettre pour les nom des fichiers
121         real(KIND=wr)themax,themin                                          ! min et max ou prior
122         real(KIND=wr) :: delta                                              ! valeur de la disrcétisation pour le diagramme de densité, les histogrammes et le
                calcul du mode
123         real(KIND=wr) :: best,mode,mediane                                  ! meilleur modèle, mode et médiane
124         real(KIND=wr) :: moy_tot,moy_100,moy_1000,moy_10000                 ! moyenne totale puis des 100, 1000, et 10000 meilleurs modèles (plus petite fonction
                coût)
125         real(KIND=wr) :: ec_tot,ec_100,ec_1000,ec_10000                     ! 1 écart type
126         real(KIND=wr) :: moy_bestchaine,ec_bestchaine                       ! moyenne et écart type de l'ensemble du meilleur modèle de chaque chaîne
127         real(KIND=wr) :: vec10000(10000,2)                                  ! stocke les 10000 meilleurs modèles (valeur du paramètre, valeur de la fonction coût
                )
128     END TYPE densityplot_one
129   ! --------                                                    --------      .
130   ! --------                                                    --------      ! relecture modèles sélectionnés et calculs a posteriori
131     TYPE densityplot
132         type(densityplot_one) :: mis,VC,VM,Zmoho,VpVs                       ! ensemble des paramètres
133         type(densityplot_one) :: Lat(nbseismes),Lon(nbseismes)
134         type(densityplot_one) :: Zhypo(nbseismes),Tzero(nbseismes)
135         type(date_sec) :: temps_ref(nbseismes)                              ! temps zéro en minutes du séisme
136         integer(KIND=wi) :: nbparam                                         ! nombre de modéles sélectionnées au cours de l'inversion McMC
137         integer(KIND=wi) :: deltaxy                                         ! nombre de disrcétisation pour le diagramme de densité, les histogrammes et le
                calcul du mode
138     END TYPE densityplot
139   ! --------                                                    --------      .
140   ! --------                                                    --------      ! fonction coût au cours de l'inversion
141     TYPE fcout
```

```
142         real(KIND=wr) :: old,new,best                                   ! à l'itération précedente, actuelle, meilleur pour la chaîne
143     END TYPE fcout
144   ! --------                                                  ---------    .
145   ! --------                                                  ---------    ! acceptance coût au cours de l'inversion
146     TYPE accept
147         integer(KIND=wl) :: N,O,NO                                      ! nombre de modèles non acceptés, acceptés et repêchés
148         real(KIND=wr) :: val                                            ! valeurs en pourcentage
149     END TYPE accept
150   ! --------                                                  ---------    .
151   ! --------                                                  ---------    ! événements sismiques du catalogue
152     TYPE seismes
153         type(date_sec) :: tps_init                                      ! date
154         real(KIND=wr) :: mag                                            ! mL
155         real(KIND=wr) :: lon, lat, pfd                                  ! coordonnées (degrés), profondeur hypocentre (km)
156         real(KIND=wr) :: d_t, d_epi, d_p                                ! différences en temps (s), distance épicentrale (km) et profondeur (km) entre le
                catalogue et le meilleur modèle rencontré lors de l'inversion
157         character(LEN=20) :: name                                       ! nom du bulletin
158     END TYPE seismes
159   ! --------                                                  ---------    !
160   ! --------                                                  ---------    ! pour des statistiques sur les coldruns (T : toutes les chaines ; S : les chaines
        sélectionnées)
161     TYPE coldmoyval
162         real(KIND=wr), dimension(:), allocatable :: Tmis,TVC,TVM,TZmoho,TVpVs
163         real(KIND=wr), dimension(:,:), allocatable :: TLat,TLon,TZhypo,TTzero
164         real(KIND=wr), dimension(:), allocatable :: Smis,SVC,SVM,SZmoho,SVpVs
165         real(KIND=wr), dimension(:,:), allocatable :: SLat,SLon,SZhypo,STzero
166     END TYPE coldmoyval
167   ! --------                                                  ---------    !
168   ! --------                                                  ---------    !
169     TYPE coldmoy
170         type(date_sec) :: tempsrefcold(nbseismes)
171         type(paramisfit) :: moytot,ectot,moyselect,ecselect             ! statistiques sur les coldruns
172     END TYPE coldmoy
173   ! --------                                                  ---------    !
174     TYPE ellip
175         real(KIND=wr) :: ang, axeA, axeB                                ! ellipse (autour de l'épicentre) +- 1 sigma des 1000 meilleurs modeles
176     END TYPE ellip
177   ! --------                                                  ---------    !
178     TYPE residus
179         character(LEN=4) :: staname
180         real(KIND=wr), dimension(:,:), allocatable :: resPg,resSg,resPn,resSn ! résidus à la station, onde P et S
181         integer(KIND=wi) :: nbPg,nbSg,nbPn,nbSn,nbPgT,nbSgT,nbPnT,nbSnT
182     END TYPE residus
183   ! --------                                                  ---------    !
184     TYPE apriorEPI
185         real(KIND=wr) lat,lon,distcarre                                 ! rechercher epicentre initial
186     END TYPE apriorEPI
187   ! --------                                                  ---------    !
188     TYPE priorEPI
189         integer(KIND=wi) nb                                             !  nb de cellules (mailles)
190         type(apriorEPI), dimension(:), allocatable :: pEpi
191     END TYPE priorEPI
192     ! --------                                                  ---------    ! amoho_centroid
193     TYPE amoho_centroid
194         real(KIND=wr) :: lonC,latC                                      ! centroïde
195         real(KIND=wr) :: alph,beta,gamma                                ! vecteur normal du moho (lon, lat, z)
196         real(KIND=wr) :: NS,EO                                          ! angle apparant du moho (lon,lat) au centroïde (0 degrés : horizontal ; ~ 90 degrés
              : vertical)
197     END TYPE amoho_centroid
198   ! ------------------------------------------------------------------    .
199
200
201 CONTAINS
202
203   ! ------------------------------------------------------------------    .
```

```
204
205   subroutine mvPall_2_P1(p1,p2,j)
206     ! --------                                                  ---------     .mh
207     ! parametres pour tous les séismes -> pour le jieme séisme
208     ! --------                                                  ---------     .
209     implicit none
210     ! --------                                                  ---------     .
211     integer(KIND=wi), intent (in) :: j
212     type(parametres), intent (in) :: p2
213     type(parametre), intent (inout) :: p1
214     ! ------------------------------------------------------------------     .
215     p1%VC=p2%VC
216     p1%VM=p2%VM
217     p1%Zmoho=p2%Zmoho
218     p1%VpVs=p2%VpVs
219     p1%Lat=p2%Lat(j)
220     p1%Lon=p2%Lon(j)
221     p1%Zhypo=p2%Zhypo(j)
222     p1%Tzero=p2%Tzero(j)
223     ! ------------------------------------------------------------------     .
224   end subroutine mvPall_2_P1
225
226   ! ------------------------------------------------------------------     .
227
228   subroutine mvP1_2_Pall(p2,p1,j)
229     ! --------                                                  ---------     .mh
230     ! parametre pour le jieme séisme -> pour tous les séismes
231     ! --------                                                  ---------     .
232     implicit none
233     ! --------                                                  ---------     .
234     integer(KIND=wi), intent (in) :: j
235     type(parametres), intent (inout) :: p2
236     type(parametre), intent (in) :: p1
237     ! ------------------------------------------------------------------     .
238     p2%VC=p1%VC
239     p2%VM=p1%VM
240     p2%Zmoho=p1%Zmoho
241     p2%VpVs=p1%VpVs
242     p2%Lat(j)=p1%Lat
243     p2%Lon(j)=p1%Lon
244     p2%Zhypo(j)=p1%Zhypo
245     p2%Tzero(j)=p1%Tzero
246     ! ------------------------------------------------------------------     .
247   end subroutine mvP1_2_Pall
248
249   ! ------------------------------------------------------------------     .
250
251   subroutine alph2vect(acentroid)
252     ! --------                                                  ---------     .mh
253     ! angle apparant du moho -> vecteur normal, moho non tabulaire
254     ! NS et OE : angle entre 0 (horizontal) et ~90 (vertical), penche positivement vers l'Est et le Sud
255     ! --------                                                  ---------     .
256     implicit none
257     type (amoho_centroid), intent (inout) :: acentroid
258     ! ------------------------------------------------------------------     .
259     acentroid%alph=tan(acentroid%EO*pi/180._wr)
260     acentroid%beta=tan(acentroid%NS*pi/180._wr)
261     acentroid%gamma=-1.0_wr
262     ! ------------------------------------------------------------------     .
263   end subroutine alph2vect
264
265   ! ------------------------------------------------------------------     .
266
267   subroutine vect2alph(acentroid)
268     ! --------                                                  ---------     .mh
```

```
    ! vecteur normal -> angle apparant du moho, moho non tabulaire
    ! NS et OE : angle entre 0 (horizontal) et ~90 (vertical), penche positivement vers l'Est et le Sud
    ! -------                                                    ---------    .
    implicit none
    type (amoho_centroid), intent (inout) :: acentroid
    ! ------------------------------------------------------------------    .
    acentroid%NS=atan(acentroid%beta/acentroid%gamma)/pi*180._wr
    acentroid%EO=atan(acentroid%alph/acentroid%gamma)/pi*180._wr
    ! ------------------------------------------------------------------    .
  end subroutine vect2alph

END MODULE typetemps



! ********************************************************************    .
! ********************************************************************    .
```

2.34 SRC/MOD/MOD_GMT/mkmoho_inc.f90

```
! extention programme avec un moho incliné
! FLAG_non_tabulaire=.true. dans SRC/MOD/modparam.f90
! fevrier 2015
! ********************************************************************    .
! -------- Méric Haugmard meric.haugmard@univ-nantes.fr          ---------    .
! ********************************************************************    .
! ------------------------------------------------------------------    .

MODULE figure_GMTmoho_inc

    use modparam

    implicit none

    private

    public  :: GMT_moho


CONTAINS

! ------------------------------------------------------------------    .

  subroutine GMT_moho(acentroid,l,nbtps,xmax,datatps,param)
    ! -------                                                  ---------    .mh
    ! Calcul les regressions sur les hodochrones et affiche l'hodochrone
    ! -------                                                  ---------    .
    use typetemps
    use pb_direct
    ! -------                                                  ---------    .
    implicit none
    integer(KIND=wi), intent (in) :: l
    integer(KIND=wi), intent(in) :: nbtps
    real(KIND=wr), intent(in) :: xmax
    type(dataone), intent(in) :: datatps(nbtps)
    type(parametre), intent(in)  :: param
    type (amoho_centroid), intent(in) :: acentroid
    ! -------                                                  ---------    .
    integer (KIND=wi) :: Noldtime, Nnewtime, ratetime
    integer (KIND=wi) :: j,k,ok,nmax
    real (KIND=wr) :: tl
    real(kind=wr) :: dishypo,Tps,Tpp,pfdsousseisme
    real(KIND=wr) :: v1, v2, lon1,lon2,lat1,lat2
    type(parametre) :: param_best
    character (LEN=5) :: numberfile
```

```
    ! ------------------------------------------------------------------     .
    param_best=param
    v1 = 2.0_wr * pi * rT / 360.0_wr                                        ! km / degree en lon
    v2 = 2.0_wr * pi * rT * sin((90.0_wr-param_best%lat)/180.0_wr*pi) /360.0_wr ! km / degree en lat
    ! -------                                                   ---------     .
    lon1 = param_best%lon - (xmax / v2 * 1.125_wr) / 2.0_wr
    lon2 = param_best%lon + (xmax / v2 * 1.125_wr) / 2.0_wr
    lat1 = param_best%lat - (xmax / v1 * 1.125_wr) / 2.0_wr
    lat2 = param_best%lat + (xmax / v1 * 1.125_wr) / 2.0_wr
    ! -------                                                   ---------     .
    ok=0
    write(numberfile(1:5),'(i5)')l
    open(101, FILE = "OUTPUT/GMT/moho-"//trim(adjustl(numberfile))//".txt",status='replace',iostat = ok)
    nmax=50
    do j=1,nmax
       do k=1,nmax
          param_best%Lon = lon1 - 1.5_wr + abs(lon2-lon1+3.0_wr)/real(nmax,wr)*real(j,wr)
          param_best%Lat = lat1 - 1.5_wr + abs(lat2-lat1+3.0_wr)/real(nmax,wr)*real(k,wr)
          ! --------                                                ---------     .
          ! datatps(1)%sta%Lon = lon1 - lon1*0.05_wr + abs(lon1-lon2)/real(nmax,wr)*1.1_wr*real(j,wr)
          ! datatps(1)%sta%Lat = lat1 - lat1*0.05_wr + abs(lat1-lat2)/real(nmax,wr)*1.1_wr*real(k,wr)
          ! --------                                                ---------     .
          call refracte_mohovar(acentroid,param_best,datatps(1)%sta,dishypo,Tps,Tpp,pfd=pfdsousseisme)
          write(101,*)param_best%Lon,param_best%Lat,abs(pfdsousseisme)
      enddo
    enddo
    close(101)
    ! -------                                                   ---------     .
    !open(106, FILE = "OUTPUT/GMT/sta-"//trim(adjustl(numberfile))//".txt",status='replace',iostat = ok)
    !do i=1,nbtps
    !   write(106,*)datatps(i)%sta%Lon,datatps(i)%sta%Lat,datatps(i)%dTP,datatps(i)%dTS
    !enddo
    !close(106)
    ! ------------------------------------------------------------------     .
    ! ------------------------------------------------------------------     .
    write(*,*)"ecriture du script GMT moho"
    write(600,*)"BEFORE=$SECONDS"
    call system_clock(Noldtime)
    write(600,*)"#####################################"
    write(600,*)"#####################################"
    write(600,*)
    write(600,*)"#########################################"
    write(600,*)"################## moho #################"
    write(600,*)"#########################################"
    write(600,*)"echo 'execution du script GMT moho'"
    write(600,*)"file=OUTPUT/GMT/topo_moho-"//trim(adjustl(numberfile))//".ps"
    write(600,*)"gmtset BASEMAP_TYPE plain"
    write(600,'(a10,E13.7,a1,E13.7,a1,E13.7,a1,E13.7)')"geozone=-R",lon1,"/",lon2,"/",lat1,"/",lat2
    write(600,'(a11,E13.7,a1,E13.7,a3)')"geoproj=-JC",param_best%lon,"/",param_best%lat,"/7i"
    write(600,*)"labasemap=-Bpa2.f.5/a1.f.25/a0WeSnz+"
    ! -------                                                   ---------     .
    write(600,*)"nearneighbor $geozone -GOUTPUT/GMT/geoid.grd OUTPUT/GMT/moho-"//trim(adjustl(numberfile))//".txt -F -I50k -S50K"
    write(600,*)"grdsample OUTPUT/GMT/geoid.grd -N500 -GOUTPUT/GMT/geoid2.grd"
    write(600,*)"grd2cpt OUTPUT/GMT/geoid2.grd -Csealand > OUTPUT/GMT/colorpal7.cpt "
    write(600,'(2a)')"grdimage $geozone $geoproj OUTPUT/GMT/geoid2.grd -COUTPUT/GMT/colorpal7.cpt ", &
    "$labasemap -Qs -Xc -Yc -K -N0 -S -P > $file"
    ! -------                                                   ---------     .
    !write(600,'(2a)')"psxy $geozone $geoproj OUTPUT/GMT/sta-"//trim(adjustl(numberfile))//".txt ", &
    !  "-Sc0.1i -COUTPUT/GMT/colorpal7.cpt -Wthinnest -O -K -P >> $file"
    ! -------                                                   ---------     .
    write(600,*)"pscoast -Z0 $geozone $geoproj -Df+ -Ia/blue -W2 -O -K -P >> $file"
    ! -------                                                   ---------     .
    write(600,*)"######### cercles de pondération ########"
    write(600,*)"echo ",param%Lon,param%Lat,"0",xmax,xmax, &
      " | psxy $geozone $geoproj -SE -W13,white -O -K >> $file"
```

```
111     write(600,*)"echo ",param%Lon,param%Lat,"0",xmax,xmax, &
112       " | psxy $geozone $geoproj -SE -W10  -O -K >> $file"
113     write(600,*)"################ Epicentre ################"
114     write(600,*)"echo",param%Lon,param%Lat," \"
115     write(600,*)" | psxy $geozone $geoproj -L -K -O -Wthinnest -Ggreen -Sa0.20i >> $file"
116     ! -------                                                    --------     .
117     write(600,*)"###### Limites du massif Armoricain #####"
118     write(600,*)"psxy $geozone $geoproj -A -W4,gray -O SRC/FILES/limitesMA -K -M >> $file"
119     write(600,*)"grdcontour $geozone $geoproj OUTPUT/GMT/geoid2.grd -C2.0 -A -W1 -O -K -P >> $file"
120     write(600,'(2a)')"psscale -D-1.5i/4i/5i/.35i -COUTPUT/GMT/colorpal7.cpt -I -O ", &
121         "-K -B1:""profondeur du moho \050km\051"": -P -Aa >> $file"
122     ! -------                                                    --------     .
123     write(600,*)"################ centroide ################"
124     write(600,*)"echo",acentroid%LonC,acentroid%LatC," \"
125     write(600,*)" | psxy $geozone $geoproj -L -K -O -Wthinnest -Ggreen -S+0.5i >> $file"
126     ! -------                                                    --------     .
127     write(600,*)"psbasemap $geozone $geoproj -JZ-2.5i -Ba0 -O >> $file"
128     write(600,*)"ps2raster -Tf -A $file"
129     write(600,'(2a)')"mv OUTPUT/GMT/topo_moho-"//trim(adjustl(numberfile))//".pdf ", &
130       "OUTPUT/figures/topo_moho-"//trim(adjustl(numberfile))//".pdf"
131     write(600,*)"#########################################"
132     write(600,*)"ELAPSED=$(($SECONDS-$BEFORE))"
133     write(600,*)" echo $ELAPSED secondes"
134     call system_clock(Nnewtime,ratetime)
135     tl=(real(Nnewtime,wr)-real(Noldtime,wr))/real(ratetime,wr)
136     write(*,'(a9,i2.2,'':'',i2.2,'':'',f9.2)')' temps : ',int(tl/3600.0_wr,wi), &
137     int((tl-real(int(tl/3600.0_wr,wi),wr)*3600.0_wr)/60.0_wr,wi),(tl-real(int(tl/60.0_wr,wi),wr)*60.0_wr)
138     ! ------------------------------------------------------------     .
139   end subroutine GMT_moho
140
141 END MODULE figure_GMTmoho_inc
142
143
144
145 ! *****************************************************************     .
146 ! *****************************************************************     .
```