Program For 1-D Missing Data Studies

Jon F. Claerbout

The following is a listing of the program as it was last used. The main program, listed first, is a system interface written in the C language [1]. You probably won't have much use for that, nor the beginning of the subroutine *miss* which just fetches parameters and annotates plots. The bulk of the code is in a language called *Ratfor* [2] which stands for *Rat*ional *for*tran. It is really a preprocesser to Fortran which I find pleasing to use. It should be readily comprehensible to Fortran programmers.

[1] Kernighan and Ritchie, *The C programming Language*, Prentice-Hall Publishing Company

[2] Kernighan and Plauger *Software Tools* Addison-Wesley Publishing Company

```
#include <stdio.h>
#include <math.h>
#define MIXED union {char *s; int *i; float *f}
int xargc; char **xargv;
int fdid;
FILE *fopen(), *fpar, *fdod;
float p[1026][10];
main(ac,av)
int ac; char **av;
      {
      static char par[35] = "header";
      static char id[35] = "stdin";
      static char od[35] = "fort09";
      xargc=ac; xargv=av;
      if((fdod=fopen(od,"w")) == NULL) {
            fprintf(stderr,"main.c can't plot at %s0,od);
            exit(-1);
            }
      miss_();
      exit(0);
      }


gechan_(nt,y)
int *nt;
float *y;
{
      int nread,it;
      if((nread=read(0,y,4**nt)) != 4**nt) {
            fprintf(stderr,"nread=%d nt=%d0,nread,nt);
            fprintf(stderr,"Out of data in gechan");
            return(0);
            }
}


pout_(m,ix,iy)
char *m;
int *ix,*iy;
      {
      short iix, iiy;
      iix= *ix;  iiy= *iy;
      putc(*m,fdod);
      puth(iix,fdod);
      puth(iiy,fdod);
      }


pentex_(ix,iy,size,string,a1,a2,a3)
char *string;
int *ix,*iy,*size;
MIXED *a1,*a2,*a3;
{
      short int key,n,isize,orient, iix, iiy;
```

```
    char line[128];
    char *sptr;
    orient=1;
    iix= *ix; iiy= *iy; isize= *size;
    sprintf(line,string,*a1,*a2,*a3);
    fprintf(stderr,"%s0,line);
    key= isize+32* orient;
    putc('m',fdod);
    puth(iix,fdod);
    puth(iiy,fdod);
    putc('t',fdod);
    puth(key,fdod);
    sptr= line;
    do { putc(*sptr,fdod); } while(*sptr++);
    }



subroutine miss
implicit undefined(a-z)
real    d(1026),g1(1026),g0(1026),buff(1026),mi(1026)
complex cd(513),cg1(513),cg0(513)
equivalence (d,cd),(g1,cg1),(g0,cg0)
complex g0g0,g0g1,g1g0,g1g1,g0d,g1d
real w(513),spn(513),spni(513)
real alpha,beta,gamma,gammap,logw,wgamma,det,lag,pow,noise,ws
integer it,nt,nth,iom,nthp,iter,iwt,nwt,trace,ntrace,igo
integer hicut,lowcut
nt = 512
nth = nt/2
nthp = nth+1
#                               get parameters, annotate plot
gamma=.2
call getpar('gamma','f',gamma)
gammap=gamma
lag=600.
call getpar('lag','f',lag)
pow=.5
call getpar('pow','f',pow)
noise=.1
call getpar('noise','f',noise)
nwt=1
call getpar('nwt','d',nwt)
call pentex(40,650,4,'Updating the Weights Within an Iteration',nt,nt,nt)
call pentex(20,20,2,'Claerbout',nt,nt,nt)
call pentex(40,20,2,'Algorithm: delta D = alpha*w**gamma',nt,nt,nt)
call pentex(60,20,2,'Notebook:  March 8,1981')
call pentex(80,20,2,'lag for smoothing noise, lag = %f',lag,lag,lag)
call pentex(100,20,2,'exponent for noise estimate, pow= %f',pow,pow,pow)
call pentex(120,20,2,'noise fraction at max,  noise= %f',noise,noise,noise)
call pentex(140,20,2,'weight restimations per iteration,  nwt= %d',nwt,nwt,nwt)
call pentex(170,400,3,'time domain',nt,nt,nt)
call pentex(170,1300,3,'frequency domain',nt,nt,nt)
call pentex(60+140,20,2,'Original Data',nt,nt,nt)
call pentex(60+2*140,20,2,'First Guess upon Original Data',nt,nt,nt)
call pentex(105+120,1730,2,'Inverse weighting function',nt,nt,nt)
call pentex(40+140,1830,2,'Weighting function',nt,nt,nt)
#                       bring in data.
call gechan(300,buff)
call gechan(nt,buff)
call taper(nt,buff)
#                       "true" data and weight
call fftr(nth,buff,+1.)
call weight(lag,pow,noise,nt,buff,w,spn)
call fftri(nth,buff,-1.)
trace = 1
iter=0
call peep(trace,iter,nt,buff,buff,w,spn,spni,gammap)
#                       set up piecewise linear weight
#hicut = nthp/3
#do it = 1,nthp
#    w(it)=1.e-30
#do it = hicut,nthp
#    w(it) = (it-hicut+.5)/(nthp-hicut+.5)
#lowcut = 20
#do it = 1,lowcut
#    w(it) = (lowcut-it+.01)/lowcut
#                       punch some holes in the data.
do it = 1,nt
     mi(it) = 1.
do it = 2,512,2
     mi(it) = 0.
do it = 170,174
     mi(it) = 0.
do it = 260,270
     mi(it) = 0.
do it = 320,340
     mi(it) = 0.
do it = 380,420
     mi(it) = 0.
do it = 1,nt
     d(it) = buff(it)*mi(it)
do it = 1,nt
     mi(it) = 1.-mi(it)
call fftr(nth,d,1.)
ntrace=6
iter=0
do trace = 2,ntrace {
     call fftri(nth,cd,-1.)
     call peep(trace,iter,nt,d,buff,w,spn,spni,gammap)
     call fftr(nth,cd,+1.)
     if(trace==ntrace)
          break
     do igo = 1, 2**(trace-2) {
          iter=iter+1
#                            make search direction g
          do iom=1,nthp {
               logw=alog(w(iom))
```

```
            wgamma=exp(gamma*logw)
            cg0(iom)=wgamma*cd(iom)
            cg1(iom)=wgamma*logw*cd(iom)
            }
#                       make g0
        call fftri(nth,g0,-1.)
        do it=1,nt
            g0(it)=g0(it)*mi(it)
        call fftr(nth,g0,+1.)
#                       make g1
        call fftri(nth,g1,-1.)
        do it=1,nt
            g1(it)=g1(it)*mi(it)
        call fftr(nth,g1,+1.)
#                       try several weights
        do iwt = 1,nwt {
#                           make alpha and beta
            g0g0=0.
            g0g1=0.
            g0d =0.
            g1g0=0.
            g1g1=0.
            g1d =0.
            do iom=1,nthp {
                ws=w(iom)
                g0g0 = g0g0 + ws*conjg(cg0(iom))*cg0(iom)
                g0g1 = g0g1 + ws*conjg(cg0(iom))*cg1(iom)
                g0d  = g0d  + ws*conjg(cg0(iom))*cd (iom)
                g1g0 = g1g0 + ws*conjg(cg1(iom))*cg0(iom)
                g1g1 = g1g1 + ws*conjg(cg1(iom))*cg1(iom)
                g1d  = g1d  + ws*conjg(cg1(iom))*cd (iom)
                }
            det  =  g0g0*g1g1 - g1g0*g0g1
            alpha = (-g1g1*g0d +g0g1*g1d )/det
            beta  = ( g1g0*g0d -g0g0*g1d )/det
            do iom=1,nthp
                cd(iom) = cd(iom)+alpha*cg0(iom)+beta*cg1(iom)
                gammap=gamma+beta/alpha
            call weight(lag,pow,noise,nt,cd,w,spn)
            }
        ############gamma=gammap
        }
    }
return
end


subroutine weight(lag,pow,noise,nt,cd,w,spn)
implicit undefined(a-z)
real w(nt),spn(513)
complex cd(nt/2+1),cdd(513),cds(513)
real lag,pow,noise,top,bigest
integer nt,nth,iom,nthp
```

```
nth = nt/2
nthp = nth+1
call move(nt+2,cd,cdd)
do iom = 1,nthp
     cdd(iom) = conjg(cdd(iom))*cdd(iom)
#                         make weight
call move(nt+2,cdd,cds)
call csmo(nt,lag,cds)
top=bigest(nt+2,cdd)
do iom=1,nthp
     spn(iom)=cdd(iom)/top+noise*noise*(cds(iom)/top)**pow
do iom=1,nthp
     w(iom)=1./spn(iom)
return
end


subroutine csmo(nt,auto,v)
# externally v is complex cross-spectrum
# internally we taper the crosscorrelation
implicit undefined(a-z)
real v(nt),tap(1026),auto,rho
complex ctap(513),cv
equivalence (tap,ctap)
integer it,nt,nth,iom,nthp
nth=nt/2
nthp=nth+1
rho=1.-1./auto
do iom=1,nthp {
     cv=cexp(cmplx(0.,3.14159265*(iom-1.)/nthp))
     cv=(1.+rho*cv)/(1.-rho*cv)
     cv=(.5*(1.-rho)/(1.+rho))*(cv+conjg(cv))
     ctap(iom)=cv
     }
call fftri(nth,tap,-1.)
call fftri(nth,v,-1.)
do it=1,nt
     v(it)=v(it)*tap(it)
call fftr(nth,v,+1.)
return
end


subroutine peep(trace,iter,nt,y,buff,w,spn,spni,gamma)
implicit undefined(a-z)
real y(nt),buff(nt),w(nt+1),spn(nt+1),spni(nt+1),b,bigest,gamma
integer nt,idy,nth,iom,nthp,trace,iter
idy = 2
nth = nt/2
nthp=nth+1
call pentex(70+140*trace,600,2,'iteration = %d  gamma''= %f',iter,gamma,iter)
b = amax1(bigest(nt,buff),bigest(nt,y))
call plot(trace,idy,0,nt,buff,b)
call plot(trace,idy,0,nt,y,b)
call fftr(nth,y,+1.)
b = bigest(nt+2,y)
call plot(trace,idy,idy*nt+10,nt+2,y,b)
do iom=1,nthp
      spn(iom)=1./sqrt(w(iom))
do iom=1,nthp
      spni(iom)=sqrt(w(iom))
b=bigest(nt/2+1,spn)
call plot(trace,idy*2,idy*nt+10,nt/2+1,spn,b)
b=bigest(nt/2+1,spni)
call plot(trace,idy*2,idy*nt+10,nt/2+1,spni,b)
call fftri(nth,y,-1.)
return
end


subroutine plot(trace,idy,ishift,n,p,b)
dimension p(n)
integer trace
character*1 m,d,e
data m,d,e/"m","d","e"/
do i = 1,n {
      iy = ishift+idy*i
      ix = 100+trace*140-p(i)*90./b+.5
      if (i==1)
            call pout(m,ix,iy)
      call pout(d,ix,iy)
      iy = iy+idy
      }
return
end


subroutine taper(n,x)
dimension x(n)
do i = 1,30 {
      x(i) = x(i)*(i-1)/30.
      l = n-i+1
      x(l) = x(l)*(i-1)/30.
      }
return
end



real function bigest(n,x)
dimension x(n)
b = 0.
do i = 1,n
      if (abs(x(i))>b)
            b = abs(x(i))
bigest = b
return
```

```
end



subroutine move(n,x,y)
dimension x(n),y(n)
do i = 1,n
    y(i) = x(i)
return
end



subroutine fft(lx,cx,signi,scale)
#  complex fourier transform.           (jfc 9/76)
#
#              lx      signi*2*pi*i*(j-1)*(k-1)/lx
#  cx(k)  =  scale * sum cx(j) * e
#             j=1          for k=1,2,...,lx=2**integer
#
complex cx(lx),cmplx,cw,cdel,ctemp
do i = 1,lx
    cx(i) = cx(i)*scale
j = 1
do i = 1,lx {
    if (i<=j) {
        ctemp = cx(j)
        cx(j) = cx(i)
        cx(i) = ctemp
        }
    m = lx/2
    while (j>m) {
        j = j-m
        m = m/2
        if (m<1)
            break 1
        }
    j = j+m
    }
l = 1
repeat {
    istep = 2*l
    cw = 1.
    arg = signi*3.14159265/l
    cdel = cmplx(cos(arg),sin(arg))
    do m = 1,l {
        do i = m,lx,istep {
            ctemp = cw*cx(i+l)
            cx(i+l) = cx(i)-ctemp
            cx(i) = cx(i)+ctemp
            }
        cw = cw*cdel
        }
    l = istep
```

```
     }
     until(l>=lx)
return
end



subroutine fftr(lx,cx,signi)
# fourier transform of a real time function.
# inputs-
#    lx=2**integer
#    cx=x(1)...x(2*lx) ,  dimensioned as x(2*lx+2)
#    signi = +1. or -1.
# output-
#    cx(1)...cx(lx+1) the spectrum on  0.le.omega.le.pi
complex cx(lx),conjg,cmplx,cw,cdel,ca,cb
call fft(lx,cx,-signi,.5)
cx(lx+1) = cx(1)
lxh = lx/2+1
cw = (0.,1.)
arg = signi*3.14159265/lx
cdel = cmplx(cos(arg),sin(arg))
do j = 1,lxh {
     jr = lx-j+2
     ca = conjg(cx(j))+cx(jr)
     cb = (conjg(cx(j))-cx(jr))*cw
     cx(j) = ca+cb
     cx(jr) = conjg(ca-cb)
     cw = cw*cdel
     }
return
end



subroutine fftri(lx,cx,signi)
# inverse fourier transform to a real time function.
complex cx(lx),cmplx,cw,cdel,conjg,ca,cb
nh = lx/2+1
cw = (0.,-1.)
arg = signi*3.14159265/lx
cdel = cmplx(cos(arg),sin(arg))
do j = 1,nh {
     jr = lx-j+2
     ca = cx(j)+conjg(cx(jr))
     cb = cx(j)-conjg(cx(jr))
     cb = cb*cw
     cx(j) = conjg(ca+cb)
     cx(jr) = ca-cb
     cw = cw*cdel
     }
call fft(lx,cx,-signi,1./(2.*lx))
return
end
```

Pig: An animal (Porcus omnivorous) closely allied to the human race by the splendor and vivacity of its appetite, which, however, is inferior in scope, for it balks at pig.

Day of inquiry. You will be subpoenaed.

Pro is to con as progress is to Congress.

Half Moon tonight. (At least its better than no Moon at all.)

If a President doesn't do it to his wife, he'll do it to his country.

Good news. Ten weeks from Friday will be a pretty good day.

Jesus Saves,
Moses Invests,
But only Buddha pays Dividends.

Happiness: An agreeable sensation arising from contemplating the misery of another.

This fortune intentionally not included.

Wasting time is an important part of living.

When Marriage is Outlawed,
Only Outlaws will have Inlaws.

Don't believe everything you hear or anything you say.

Glib's Fourth Law of Unreliability:

> Investment in reliability will increase until it exceeds the probable cost of errors, or until someone insists on getting some useful work done.

Stay away from hurricanes for a while.

Weinberg's First Law:

> Progress is made on alternate Fridays.