A Program For Downward Continuing Slant Stacks

Robert W. Clayton

The following is a listing of the program for downward continuing slant stacks. It is the program that was used in the refraction inversion examples given in this report, and in SEP-24 and SEP-25.

The main program is written in 'C', and its purpose is to parse the input parameters, and transfer the data in and out of the Array Processor (FPS 120b). The computations are done in the routine 'image' which is written in vector-chaining language of the Array Processor.

```
/*  main program for downward continuation of slant stacks
 *  The real work is done by the AP routine image().
 *  Robert W. Clayton, Stanford University, Stanford CA 94305
 */
#include  <stdio.h>
#define SIZE   2000
int         nt;         /* # of time points */
int         ntuse;          /* # of time points to use */
int         ntpad;          /* # of time points to pad to for power of 2 (FFT) */
float       dt;         /* delta t */
float       p0;         /* starting p value */
float       dp;         /* delta p */
int         np;         /* # of p's to do */
int         npskip;         /* skip this many p's before starting */
int         nz;         /* # of depth points */
int         nzpad;          /* # of depth points to pad to for FFT */
float       dz;         /* delta z */
float       ang  =0.0;      /* static phase shift in p-z plane */
float       wfac      =1.0;       /* freq. factor:  =1 for primaries, =2 for 1st-multiples */
float       xdata[SIZE];
float vm[512], dep[512], vel[512];
float       *f    =xdata;
float       *wramp    =xdata;
struct const          /* constants to loaded into the AP */
  {
      float pstart;         /* starting p */
      float pinc;           /* delta p */
      float wstart;         /* starting freq. (always zero) */
      float winc;           /* delta freq. */
      float cosang;         /* cos of phase shift angle */
      float sinang;         /* sin of phase shift angle */
  }  *const;
char cinput[40];
char coutput[40];
char cvin[40];
int xargc; char **xargv;
main(ac,av)
int ac; char **av;
  {
      int iz, ip, npts, base, index, input, output;
      float dw,  *ptr;
      double sin(), cos();
      FILE *vin, *fopen();
      xargc=ac; xargv=av;

/* get parameters */
      getpar_("nt","d",&nt);
      getpar_("ntuse","d",&ntuse);
      getpar_("ntpad","d",&ntpad);
      getpar_("np","d",&np);
      getpar_("npskip","d",&npskip);
      getpar_("nz","d",&nz);
      getpar_("nzpad","d",&nzpad);
      getpar_("dt","f",&dt);
```

```
    getpar_("dz","f",&dz);
    getpar_("dp","f",&dp);
    getpar_("p0","f",&p0);
    ang= 0.0;
    getpar_("ang","f",&ang);
    wfac= 1.0;
    getpar_("wfac","f",&wfac);

    if(getpar_("input","s",cinput)==0)
        err("must specify input0);
    if( (input=open(cinput,0))== -1)
        err("cannot open %s0,cinput);

    if(getpar_("output","s",coutput)==0)
        err("must specify output0);
    if( (output=creat(coutput,0664))== -1)
        err("cannot creat %s0,coutput);

    if(getpar_("vin","s",cvin)==0)
        err("must specify vin0);
    if( (vin=fopen(cvin,"r"))== NULL)
        err("cannot open %s0,cvin);

/* print parameters */
    printf("input field= %s0,cinput);
    printf("input velocity model= %s0,cvin);
    printf("output field= %s0,coutput);
    printf("phase angle rotation= %6.1f degrees0,ang);
    printf("wfac= %3.1f0,wfac);
    printf("nt= %d  ntuse= %d ntpad= %d dt= %7.4f0,nt,ntuse,ntpad,dt);
    printf("np= %d  npskip= %d  p0= %7.4f dp= %7.4f0,np,npskip,p0,dp);
    printf("nz= %d   nzpad= %d  dz= %7.4f0,nz,nzpad,dz);
    ang= ang* 3.14159/180.0;

/* set constant parameters into xdata */
    base= 6 + (np-npskip) + 2*nz + 3*ntpad;
    dw= 2.0*3.14159/(ntpad*dt);
    dw *= wfac;
    const= (struct const *)xdata;
    const->pstart= dp*npskip + p0;
    const->pinc  = dp;
    const->wstart= 0.0;
    const->winc  = 2.0*dz*dw;
    const->cosang= cos(ang);
    const->sinang= sin(ang);

/* read in velocity function specified as vertices of (depth,velocity) */
    for(npts=0; fscanf(vin,"%f %f",&dep[npts],&vel[npts])>0; npts++);
/* interpolate to v(iz*dz), i=0....(nz-1) */
    vdm(vm,nz,dz,dep,vel,npts);
/* put 1/(v(z)**2) in the AP */
    ptr= &xdata[6];
    for(iz=0; iz< nz; iz++) *ptr++ = 1.0/( vm[iz]*vm[iz]);
    apclr();
```

```
      apput(xdata,0,nz+6,2);
/* skip first npskip vectors */
      lseek(input,npskip*nt*4,0);
/* read input data and transfer to the AP */
      for(ip=npskip, index=base; ip< np; ip++ )
        {
          if(read(input,xdata,nt*4) != nt*4)
               fprintf(stderr,"read error0);
          apput(xdata,index,ntuse,2);
          index += ntuse;
        }

/* let the AP do the work */
      timer(0);
      image(np-npskip,base,ntuse,ntpad,base,nz,nzpad);
      apwait();
      timer(1,"OP time=");
/* get the data out of the AP and write to the output file */
      for(ip=npskip, index=base; ip< np; ip++)
        {
          apget(xdata,index,nz,2);
          index += nz;
          write(output,xdata,nz*4);
        }
  }

err(fmt,a1,a2,a3)
char *fmt;
double a1, a2, a3;
  {
      /* report error and die */
      fprintf(stderr,fmt,a1,a2,a3);
      exit(-1);
  }
timer(icode,str)
int icode;
char *str;
  {
      /* time between calls with icode=0 and icode=1 */
      static long int start;
      long int last;
      int elapse;
      if(icode == 0)
        {
          time(&start);
          return;
        }
      time(&last);
      elapse= last- start;
      printf("%s  %d0,str,elapse);
  }

vdm(vm,nz,dz,dep,vel,npts)
float *vm, *dep, *vel, dz;
```

```
int nz, npts;
  {
     /* interpolate velocity model */
     int iz, ind;
     double z, slope, v1, v2, d1, d2, fabs();
     for(iz=0, ind=0; iz<nz; iz++)
       {
          z= iz*dz;
          if( dep[ind+1] < z && ind < npts-1 ) ind++;
          v1= vel[ind];
          v2= vel[ind+1];
          d1= dep[ind];
          d2= dep[ind+1];
          slope= 0.0;
          if(fabs(d2-d1) > 1.e-2) slope= (v2-v1)/(d2-d1);
          vm[iz]= v1+ slope*(z-d1);
       }
  }

double fabs(x)
double x;
  {
     if(x<0.0) return(-x);
     return(x);
  }


" downward continuation program for slant stacks
" language: Floating Point Systems Vector Chainer Language.
" Robert W. Clayton, Stanford University, Stanford CA 94305
"
     define image(np,data,nt,ntpad,out,nz,nzpad)
     local sm2, wramp, f, rot, cvec, cvec1, tdata
     local fptr, zcnt, pramp, rot1, nzpad2, nt2
     nt2   = ntpad / 2
     nzpad2 = nzpad * 2
"
" constants in low memory are:
"      loc 0:    p0   -starting p
"      loc 1:    dp   -p increment
"      loc 2:    w0   -starting freq. (always zero)
"      loc 3:    dw   -freq.  increment
"      loc 4:    cosang    -cos of phase shift angle
"      loc 5:    sinang    -sin of phase shift angle
"
" scratch memory assignments
"
     sm2=   6
     wramp= sm2   + nz
     pramp= wramp + nt2
     f=     pramp + np
     rot=   f     + nz
     rot1=  rot   + nzpad
     cvec=  rot   + nt2
```

```
      cvec1= cvec  + 1
      tdata= cvec  + ntpad
"     end of scratch= tdata + ntpad
"               = 6 + np + 2*nz + 3*ntpad
"     it is assumed that 2*nzpad <= 5*(ntpad/2)
"
"     data and out may be the same if nz <= nt
"     otherwise data must be >= out + (nz-nt)*np
"
"  construct p ramp    pramp[ip]= -(p0 + ip*dp)**2, ip=0 .... np-1
"
      call vramp(0,1,pramp,1,np)
      call vsq(pramp,1,pramp,1,np)
      call vneg(pramp,1,pramp,1,np)
"
"  construct frequency ramp   wramp[iw]= (w0 + iw*dw), iw=0 ... nt2-1
"
      call vramp(2,3,wramp,1,nt2)
"
"  construct f function   f[iz]= sqrt( abs( sm2[iz] - p**2 ) ), iz=0 ... nz-1
"
"               ***** loop over p's *****
"
nextp:    call vmov(sm2,1,f,1,nz)
      call vsadd(f,1,pramp,f,1,nz)
      call vabs(f,1,f,1,nz)
      call vsqrt(f,1,f,1,nz)
"
      call vclr(tdata,1,ntpad)
      call vmov(data,1,tdata,1,nt)
      call rfft(tdata,ntpad,1)
"
"               ***** loop over z's *****
"
      fptr= f
      zcnt= nz
nextz:    call vmov(wramp,1,rot,1,nt2)
      call vsmul(rot,1,fptr,rot,1,nt2)
      call vcos(rot,1,cvec,2,nt2)
      call vsin(rot,1,cvec1,2,nt2)
      call sve(tdata,2,fptr,nt2)
      call cvmul(tdata,2,cvec,2,tdata,2,nt2)
      fptr = fptr + 1
      zcnt = zcnt - 1
      if zcnt > 0 goto nextz
"
"               ***** end of loop over z's *****
"
"     apply frequency independent phase shift
"
      call vclr(rot,1,nzpad2)
      call vmov(f,1,rot,2,nz)
      call cfft(rot,nzpad,-1)
      call cvmul(rot,2,4,0,rot,2,nzpad)
```

```
      call vclr(rot1,1,nzpad)
      call cfft(rot,nzpad,1)
      call vmov(rot,2,out,1,nz)
"
"     call vmov(f,1,out,1,nz)
      data = data + nt
      out = out + nz
      pramp = pramp + 1
      np = np - 1
      if np > 0 goto nextp
      end
```

The goal of science is to build better mousetraps.
The goal of nature is to build better mice.

"Might as well be frank, monsieur. It would take a miracle to get you out of Casablanca and the Germans have outlawed miracles."

An elephant is a mouse with an operating system.

There was a young poet named Dan,
Whose poetry never would scan.
When told this was so,
He said,"yes, I know,
It's because I try to put every possible syllable into that last line that I can."

Mistakes are often the stepping stones to utter failure.

"Some scholars are like donkeys, they merely carry a lot of books."
-- Folk saying

Alexander Graham Bell is alive and well in New York, and still waiting for a dial tone.

Who made the world I cannot tell;
My hand, though now my knuckles bleed,
I never soiled with such a deed.

-- A. E. Housman

Hire the morally handicapped.

Hand: A singular instrument worn at the end of a human arm and commonly thrust into somebody's pocket.

George Orwell was an optimist.

If there is no God, who pops up the next Kleenex?
-- Art Hoppe

If God had wanted you to go around nude, He would have given you bigger hands.

You will be Told about it Tomorrow. Go Home and Prepare Thyself.

Down with categeorical imperative!