NUMERICAL RECIPES

Webnote No. 18, Rev. 1

Code for External or Memory-Local Fourier Transform

```
void fourfs(NRvector<fstream*> &file, VecInt_I &nn, const Int isign) {
```
fourfs.h

One- or multi-dimensional Fourier transform of a large data set stored on external media. On input, `nn[0..ndim-1]` contains the lengths of each dimension (number of real and imaginary value pairs), which must be powers of 2. Here `ndim` is the number of dimensions. `file[0..3]` contains the stream pointers to 4 temporary files, each large enough to hold half of the data. The four streams must be opened in the system's "binary" (as opposed to "text") mode. The input data must be in normal C++ array order (rightmost index changing most quickly), with its first half stored in file `file[0]`, its second half in `file[1]`, in native floating point form. `KBF` real numbers are processed per buffered read or write. `isign` should be set to 1 for the Fourier transform, to -1 for its inverse. On output, values in the array `file` may have been permuted; the first half of the result is stored in `file[2]`, the second half in `file[3]`. N.B.: For `ndim` > 1, the output is *not* stored in normal C++ array order. Instead it is the leftmost array index that cycles most quickly, then the second leftmost, and so on. If `ndim` $= 2$, this means that the output is stored by columns rather than rows; it is the transpose of the output that would have been produced by `fourn`.

```
    const Int KBF=128;
    static Int mate[4]={1,0,3,2};
    Int cc,cc0,j,j12,jk,k,kk,n=1,mm,kc=0,kd,ks,kr,na,nb,nc,nd,nr,ns,nv;
    Doub tempr,tempi,wr,wi,wpr,wpi,wtemp,theta;
    VecDoub afa(KBF),afb(KBF),afc(KBF);
    Int ndim=nn.size();
    for (j=0;j<ndim;j++) {
        n *= nn[j];
        if (nn[j] <= 1) throw("invalid Doub or wrong ndim in fourfs");
    }
    nv=0;
    jk=nn[nv];
    mm=n;
    ns=n/KBF;
    nr=ns >> 1;
    kd=KBF >> 1;
    ks=n;
    fourew(file,na,nb,nc,nd);
```

The first phase of the transform starts here.

```
    for (;;) {
```
Start of the computing pass.

```
        theta=isign*3.141592653589793/(n/mm);
        wtemp=sin(0.5*theta);
        wpr = -2.0*wtemp*wtemp;
        wpi=sin(theta);
        wr=1.0;
        wi=0.0;
        mm >>= 1;
        for (j12=0;j12<2;j12++) {
            kr=0;
            do {
                cc0=(*file[na]).tellg()/sizeof(Doub);
```

```
            (*file[na]).read((char *) &afa[0],KBF*sizeof(Doub));
            cc=(*file[na]).tellg()/sizeof(Doub);
            if ((cc-cc0) != KBF) throw("read error 1 in fourfs");
            cc0=(*file[nb]).tellg()/sizeof(Doub);
            (*file[nb]).read((char *) &afb[0],KBF*sizeof(Doub));
            cc=(*file[nb]).tellg()/sizeof(Doub);
            if ((cc-cc0) != KBF) throw("read error 2 in fourfs");
            for (j=0;j<KBF;j+=2) {
                tempr=wr*afb[j]-wi*afb[j+1];
                tempi=wi*afb[j]+wr*afb[j+1];
                afb[j]=afa[j]-tempr;
                afa[j] += tempr;
                afb[j+1]=afa[j+1]-tempi;
                afa[j+1] += tempi;
            }
            kc += kd;
            if (kc == mm) {
                kc=0;
                wr=(wtemp=wr)*wpr-wi*wpi+wr;
                wi=wi*wpr+wtemp*wpi+wi;
            }
            cc0=(*file[nc]).tellp()/sizeof(Doub);
            (*file[nc]).write((char *) &afa[0],KBF*sizeof(Doub));
            cc=(*file[nc]).tellp()/sizeof(Doub);
            if ((cc-cc0) != KBF) throw("write error 1 in fourfs");
            cc0=(*file[nd]).tellp()/sizeof(Doub);
            (*file[nd]).write((char *) &afb[0],KBF*sizeof(Doub));
            cc=(*file[nd]).tellp()/sizeof(Doub);
            if ((cc-cc0) != KBF) throw("write error 2 in fourfs");
        } while (++kr < nr);
        if (j12 == 0 && ks != n && ks == KBF) {
            na=mate[na];
            nb=na;
        }
        if (nr == 0) break;
    }
    fourew(file,na,nb,nc,nd);          Start of the permutation pass.
    jk >>= 1;
    while (jk == 1) {
        mm=n;
        jk=nn[++nv];
    }
    ks >>= 1;
    if (ks > KBF) {
        for (j12=0;j12<2;j12++) {
            for (kr=0;kr<ns;kr+=ks/KBF) {
                for (k=0;k<ks;k+=KBF) {
                    cc0=(*file[na]).tellg()/sizeof(Doub);
                    (*file[na]).read((char *) &afa[0],KBF*sizeof(Doub));
                    cc=(*file[na]).tellg()/sizeof(Doub);
                    if ((cc-cc0) != KBF) throw("read error 3 in fourfs");
                    cc0=(*file[nc]).tellp()/sizeof(Doub);
                    (*file[nc]).write((char *) &afa[0],KBF*sizeof(Doub));
                    cc=(*file[nc]).tellp()/sizeof(Doub);
                    if ((cc-cc0) != KBF) throw("write error 3 in fourfs");
                }
                nc=mate[nc];
            }
            na=mate[na];
        }
        fourew(file,na,nb,nc,nd);
    } else if (ks == KBF) nb=na;
    else break;
}
```

```
    j=0;
```
The second phase of the transform starts here. Now, the remaining permutations are sufficiently local to be done in place.
```
    for (;;) {
        theta=isign*3.141592653589793/(n/mm);
        wtemp=sin(0.5*theta);
        wpr = -2.0*wtemp*wtemp;
        wpi=sin(theta);
        wr=1.0;
        wi=0.0;
        mm >>= 1;
        ks=kd;
        kd >>= 1;
        for (j12=0;j12<2;j12++) {
            for (kr=0;kr<ns;kr++) {
                cc0=(*file[na]).tellg()/sizeof(Doub);
                (*file[na]).read((char *) &afc[0],KBF*sizeof(Doub));
                cc=(*file[na]).tellg()/sizeof(Doub);
                if ((cc-cc0) != KBF) throw("read error 4 in fourfs");
                kk=0;
                k=ks;
                for (;;) {
                    tempr=wr*afc[kk+ks]-wi*afc[kk+ks+1];
                    tempi=wi*afc[kk+ks]+wr*afc[kk+ks+1];
                    afa[j]=afc[kk]+tempr;
                    afb[j]=afc[kk]-tempr;
                    afa[++j]=afc[++kk]+tempi;
                    afb[j++]=afc[kk++]-tempi;
                    if (kk < k) continue;
                    kc += kd;
                    if (kc == mm) {
                        kc=0;
                        wr=(wtemp=wr)*wpr-wi*wpi+wr;
                        wi=wi*wpr+wtemp*wpi+wi;
                    }
                    kk += ks;
                    if (kk > KBF-1) break;
                    else k=kk+ks;
                }
                if (j > KBF-1) {
                    cc0=(*file[nc]).tellp()/sizeof(Doub);
                    (*file[nc]).write((char *) &afa[0],KBF*sizeof(Doub));
                    cc=(*file[nc]).tellp()/sizeof(Doub);
                    if ((cc-cc0) != KBF) throw("write error 4 in fourfs");
                    cc0=(*file[nd]).tellp()/sizeof(Doub);
                    (*file[nd]).write((char *) &afb[0],KBF*sizeof(Doub));
                    cc=(*file[nd]).tellp()/sizeof(Doub);
                    if ((cc-cc0) != KBF) throw("write error 5 in fourfs");
                    j=0;
                }
            }
            na=mate[na];
        }
        fourew(file,na,nb,nc,nd);
        jk >>= 1;
        if (jk > 1) continue;
        mm=n;
        do {
            if (nv < ndim-1) jk=nn[++nv];
            else return;
        } while (jk == 1);
    }
}
```

fourfs.h

```
void fourew(NRvector<fstream*> &file, Int &na, Int &nb, Int &nc, Int &nd) {
```

Utility used by `fourfs`. Rewinds and renumbers the four files.

```
    Int i;
    for (i=0;i<4;i++) (*file[i]).seekp(0);
    for (i=0;i<4;i++) (*file[i]).seekg(0);
    SWAP(file[1],file[3]);
    SWAP(file[0],file[2]);
    na=2;
    nb=3;
    nc=0;
    nd=1;
}
```