```
#include "oct.h"
#include "field.h"
#include "inertial.h"
#include "lyapext.h"

//27-july-2005 added RK4_tau_v...
//all.cc 1.0.8 12/04/2006
//adding AVISO Eastern Mediterranean.....
//20/02/2007 adding the Kerguelen plateau.....

DEFUN_DLD (select, args, ,
  "Octave file select(sys_code).")
{

int sys_code;
sys_code=(int)args(0).scalar_value();

Matrix out(1,1);

out(0,0)=select_sys(sys_code);
pField->freezewarning();
pField->print_par();

return octave_value(out);
}

DEFUN_DLD (lon2posx, args, ,
  "Octave file lon2posx(lon).")
{

double lon;
lon=args(0).scalar_value();

Matrix out(1,1);

out(0,0)=lon2posx(lon);

return octave_value(out);
}

DEFUN_DLD (boxmuller, , ,
  "Octave file boxmuller().")
{
double y1,y2;
Matrix out(2,1);
boxmuller(&y1,&y2);
out(0,0)=y1;
out(1,0)=y2;
return octave_value(out);
}

DEFUN_DLD (Tpol2cart, args, ,
  "Octave file pol2cart(lon,lat).")
{

double lat,lon,x,y,z;
lon=args(0).scalar_value();
lat=args(1).scalar_value();
Matrix out(3,1);

pol2cart(lon,lat,&x,&y,&z);

out(0,0)=x;
out(1,0)=y;
out(2,0)=z;

return octave_value(out);
}

DEFUN_DLD (Tcart2pol, args, ,
  "Octave file cart2pol(x,y,z).")
{
```

```
double lat,lon,x,y,z;
x=args(0).scalar_value();
y=args(1).scalar_value();
z=args(2).scalar_value();

Matrix out(2,1);
cart2pol(x,y,z,&lon,&lat);

out(0,0)=lon;
out(1,0)=lat;

return octave_value(out);
}

DEFUN_DLD (Tdist, args, ,
  "Octave file d=Tdist(x1,y1,x2,y2). x1,y1 are lon and lat of point 1, same for point 2. Tdist gives
the either the Euclidean distance (default) or the distance over the sphere (the length of the arc, in
deg.) depending on pField->datatype.")
{

double x1,y1,x2,y2,d;
x1=args(0).scalar_value();
y1=args(1).scalar_value();
x2=args(2).scalar_value();
y2=args(3).scalar_value();

Matrix out(1,1);
Tdist(x1,y1,x2,y2,&d);

out(0,0)=d;

return octave_value(out);
}

DEFUN_DLD (scalecount_, args, ,
  "Octave file n=scalecount_(v,sc). Given a COLUMN vector: [x1 y1 x2 y2..xN yN] returns a N-long flag
vector (one flag for each point) to put a disc in order to cover the whole set.")
{

double sc,*v;
int N,quality,ndisk,ct,*flagct;
Matrix Mv=args(0).matrix_value();
sc=args(1).scalar_value();
N=(int)Mv.rows()/2;
v=new double[N*2];
flagct=new int[N];
Matrix out(N,1);
//printf("scalecount_:\nN:%d\n",N);

for(ct=0;ct<2*N;ct++) v[ct]=Mv(ct,0);
//printf("scalecount_:first cycle ok.\n");

scalecount(v,N,sc,flagct,&ndisk,&quality);
//printf("scalecount_:scalecount ok.\n");
for(ct=0;ct<N;ct++) out(ct,0)=flagct[ct];

return octave_value(out);
}




DEFUN_DLD (lat2posy, args, ,
  "Octave file lat2posy(lat).")
{

double lat;
lat=args(0).scalar_value();

Matrix out(1,1);
```

```
out(0,0)=lat2posy(lat);

return octave_value(out);
}









DEFUN_DLD (posy2lat, args, ,
  "Octave file posy2lat(posy).")
{

double posy;
posy=args(0).scalar_value();

Matrix out(1,1);

out(0,0)=posy2lat(posy);

return octave_value(out);
}

DEFUN_DLD (set_par, args, ,
  "Octave file set_par([par1 par2 ...]').")
{
double *par_list;
int num_par,j1;
Matrix out(1,1);
Matrix Mpar_list=args(0).matrix_value();
num_par=(int)Mpar_list.rows();

par_list=new double[num_par];

for (j1=0;j1<num_par;j1++) par_list[j1]=Mpar_list(j1,0);
pField->set_par(par_list);
pField->freezewarning();
pField->print_par();


out(0,0)=0;
delete par_list;
return octave_value(out);
}

DEFUN_DLD (set_I, args, ,
  "Octave file set_I(tau) or set_I().")
{
double par;


Matrix out(1,1);
if (args.length()==1) {
        par=args(0).scalar_value();
        Tau=par;
        Itau=1/Tau;
        }
printf("\n Inertial time:%lf (1/tau=%lf)\n",Tau,Itau);

out(0,0)=Tau;

return octave_value(out);
}

DEFUN_DLD (freezetime, args, ,
  "Octave file freezetime(fr_time). Freezes the velocity field at the state of time fr_time. Use
```

```
unfreezetime() to cancel.")
{
double fr_time;


Matrix out(1,1);
if (args.length()==1) {
        fr_time=args(0).scalar_value();
        pField->freeze=1;
        pField->frozentime=fr_time;
        out(0,0)=fr_time;
        }
pField->freezewarning();
return octave_value(out);
}

DEFUN_DLD (unfreezetime, args, ,
  "Octave file unfreezetime(fr_time). Reset the normal time flow for the velocity field.")
{
printf("Attempting to unfreeze the time..\n");
Matrix out(1,1);

pField->freeze=0;
out(0,0)=pField->frozentime;

pField->freezewarning();
return octave_value(out);
}




DEFUN_DLD (RK4, args, ,
  "Octave file out=RK4(tspan,x0,Nstep). tspan is a 2x1 column with initial and final time, x0 is a (1
+N)x1 column (see field.h).")
{
double tspan[2],*x0,*trj;
int numx0,Nstep,j;


Matrix Mtspan=args(0).matrix_value();
Matrix Mx0=args(1).matrix_value();
numx0=(int)Mx0.rows()/2;
Nstep=(int)args(2).scalar_value();
Matrix out(Nstep*numx0*2,1);

trj=new double[Nstep*numx0*2];

x0=new double[numx0*2];
if (x0==NULL||trj==NULL) {
        printf("Error in the number of initial points or allocating the trajectories.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
for(j=0;j<numx0*2;j++) x0[j]=Mx0(j,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d\n",tspan[0],tspan[1],numx0,Nstep);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);

RK4(tspan,x0,numx0,Nstep,trj);
for(j=0;j<numx0*2*Nstep;j++) out(j,0)=trj[j];

delete x0;
delete trj;
return octave_value(out);
}

DEFUN_DLD (RK4_I, args, ,
  "Octave file out=RK4_I(tspan,x0,Nstep). tspan is a 2x1 column with initial and final time, x0 is a (1
+N)x1 column (see field.h).")
```

```
{
double tspan[2],*x0,*trj;
int numx0,Nstep,j;


Matrix Mtspan=args(0).matrix_value();
Matrix Mx0=args(1).matrix_value();
numx0=(int)Mx0.rows()/4;
Nstep=(int)args(2).scalar_value();
Matrix out(Nstep*numx0*4,1);

trj=new double[Nstep*numx0*4];

x0=new double[numx0*4];
if (x0==NULL||trj==NULL) {
        printf("Error in the number of initial points or allocating the trajectories.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
for(j=0;j<numx0*4;j++) x0[j]=Mx0(j,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d\n",tspan[0],tspan[1],numx0,Nstep);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);

RK4_I(tspan,x0,numx0,Nstep,trj);
for(j=0;j<numx0*4*Nstep;j++) out(j,0)=trj[j];

delete x0,trj;
return octave_value(out);
}



DEFUN_DLD (RK4_tau, args, ,
  "Octave file out=RK4_tau(tspan,x0,Nstep,delta). tspan is a 2x1 column with initial and final time,
x0 is a (1+N)x1 column (see field.h).")
{
double tspan[2],*x0,delta,tau;
int numx0,Nstep,elmax,j;

Matrix out(2,1);
Matrix Mtspan=args(0).matrix_value();
Matrix Mx0=args(1).matrix_value();
numx0=(int)Mx0.rows()/2;
Nstep=(int)args(2).scalar_value();
delta=args(3).scalar_value();

x0=new double[numx0*2];
if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
for(j=0;j<numx0*2;j++) x0[j]=Mx0(j,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);

RK4_tau(tspan,x0,numx0,Nstep,delta,&elmax,&tau);

out(0,0)=(double)tau;
out(1,0)=(double)elmax;
delete x0;
return octave_value(out);
}

DEFUN_DLD (RK4_I_tau, args, ,
  "Octave file out=RK4_I_tau(tspan,x0,Nstep,delta). tspan is a 2x1 column with initial and final time,
```

```
x0 is a (1+N)x1 column (see field.h).")
{
double tspan[2],*x0,delta,tau;
int numx0,Nstep,elmax,j;

Matrix out(2,1);
Matrix Mtspan=args(0).matrix_value();
Matrix Mx0=args(1).matrix_value();
numx0=(int)Mx0.rows()/4;
Nstep=(int)args(2).scalar_value();
delta=args(3).scalar_value();

x0=new double[numx0*4];
if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
for(j=0;j<numx0*4;j++) x0[j]=Mx0(j,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);

RK4_I_tau(tspan,x0,numx0,Nstep,delta,&elmax,&tau);

out(0,0)=(double)tau;
out(1,0)=(double)elmax;
delete x0;
return octave_value(out);
}

DEFUN_DLD (FSLEext, args, ,
  "Octave file out=FSLEext(tspan,x00,delta0,Nstep,delta). tspan is a 2x1 column with initial and final
time, x00 is a column vector with x and y component, out is a matrix giving:
[lambda1,lambda2;theta1,theta2] (see lyapext.h).")
{
double tspan[2],x00[2],delta0,delta,l1,l2,theta1,theta2;
int Nstep,elmax,j;

Matrix out(2,2);
Matrix Mtspan=args(0).matrix_value();
Matrix Mx00=args(1).matrix_value();
delta0=args(2).scalar_value();
Nstep=(int)args(3).scalar_value();
delta=args(4).scalar_value();
x00[0]=Mx00(0,0);
x00[1]=Mx00(1,0);


tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
FSLEext_(tspan,x00,delta0,Nstep,delta,&l1,&l2,&theta1,&theta2);

out(0,0)=(double)l1;
out(1,0)=(double)l2;
out(0,1)=(double)theta1;
out(1,1)=(double)theta2;


return octave_value(out);
}

DEFUN_DLD (FTLEext, args, ,
  "Octave file out=FTLEext(tspan,x00,delta0,Nstep). tspan is a 2x1 column with initial and final time,
x00 is a column vector with x and y component, out is a matrix giving: [lambda1,lambda2;theta1,theta2]
(see lyapext.h).")
{
```

```
double tspan[2],x00[2],delta0,l1,l2,theta1,theta2;
int Nstep,elmax,j;

Matrix out(2,2);
Matrix Mtspan=args(0).matrix_value();
Matrix Mx00=args(1).matrix_value();
delta0=args(2).scalar_value();
Nstep=(int)args(3).scalar_value();

x00[0]=Mx00(0,0);
x00[1]=Mx00(1,0);


tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
FTLEext_(tspan,x00,delta0,Nstep,&l1,&l2,&theta1,&theta2);

out(0,0)=(double)l1;
out(1,0)=(double)l2;
out(0,1)=(double)theta1;
out(1,1)=(double)theta2;


return octave_value(out);
}

DEFUN_DLD (FSLEn, args, ,
  "Octave file out=FSLEn(tspan,x00,numpt,delta0,Nstep,delta). tspan is a 2x1 column with initial and
final time, x00 is a column vector with x and y component, out is a vector giving: [lambda1;theta1]
(see lyapext.h).")
{
double tspan[2],x00[2],delta0,delta,l1,theta1;
int Nstep,numpt;

Matrix out(2,1);
Matrix Mtspan=args(0).matrix_value();
Matrix Mx00=args(1).matrix_value();
numpt=(int)args(2).scalar_value();
delta0=args(3).scalar_value();
Nstep=(int)args(4).scalar_value();
delta=args(5).scalar_value();
x00[0]=Mx00(0,0);
x00[1]=Mx00(1,0);


tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
FSLEn_(tspan,x00,numpt,delta0,Nstep,delta,&l1,&theta1);

out(0,0)=(double)l1;
out(1,0)=(double)theta1;


return octave_value(out);
}


DEFUN_DLD (RK4_delta, args, ,
  "Octave file out=RK4_delta(tspan,x0,Nstep). tspan is a 2x1 column with initial and final time, x0 is
a (1+N)x1 column (see field.h).")
{
double tspan[2],*x0,delta,tau,direxp,dirx0;
int numx0,Nstep,elmax,j;

Matrix out(3,1);
Matrix Mtspan=args(0).matrix_value();
```

```
Matrix Mx0=args(1).matrix_value();
numx0=(int)Mx0.rows()/2;
Nstep=(int)args(2).scalar_value();

x0=new double[numx0*2];
if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
for(j=0;j<numx0*2;j++) x0[j]=Mx0(j,0);

//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);

RK4_delta(tspan,x0,numx0,Nstep,&direxp,&delta,&dirx0);

out(0,0)=(double)delta;
out(1,0)=(double)direxp;
out(2,0)=(double)dirx0;
delete x0;
return octave_value(out);
}


DEFUN_DLD (rawdata_out, args, ,
  "Octave file vel=rawdata_out(indx).")
{

long indx;
double U,V;


indx=long(args(0).scalar_value());

Matrix out(2,1);
pField->rawdata_out(indx,&U,&V);

out(0,0)=U;
out(1,0)=V;

return octave_value(out);
}

DEFUN_DLD (xyp, args, ,
  "Octave file xyp(t,x,y).")
{

double t,x,y,xyp[2];


t=args(0).scalar_value();
x=args(1).scalar_value();
y=args(2).scalar_value();

Matrix out(2,1);
pField->xyp(t,x,y,xyp);

out(0,0)=xyp[0];
out(1,0)=xyp[1];

return octave_value(out);
}
DEFUN_DLD (field_geometry, args, ,
  "Octave file field_geometry(disttype,datatype,gridtype).\ndisttype=1 (Euclidean), 2 (sphere).
\ndatatype=1 (deg./sec.), 2 (cm/sec.)\ngridtype=0 (flat), 1 (sphere, regular), 2 (sphere Mercator).")
{

int disttype,datatype,gridtype;
Matrix out(3,1);
```

```
disttype=(int)args(0).scalar_value();
datatype=(int)args(1).scalar_value();
gridtype=(int)args(2).scalar_value();

pField->datatype=datatype;
pField->disttype=disttype;
pField->gridtype=gridtype;

out(0,0)=disttype;
out(1,0)=datatype;
out(2,0)=gridtype;

return octave_value(out);
}

DEFUN_DLD (field_noise, args, ,
  "Octave file field_noise(noise_on_off,noiselevel).\n noise_on_off=0 for no noise.")
{

int noise_on_off;
double noiselevel;
Matrix out(2,1);

noise_on_off=(int)args(0).scalar_value();
noiselevel=(double)args(1).scalar_value();

pField->noise=noise_on_off;
pField->noiselevel=noiselevel;

out(0,0)=noise_on_off;
out(1,0)=noiselevel;

return octave_value(out);
}




DEFUN_DLD (field_geometry_read, args, ,
  "Octave file out=field_geometry_read().\nout(1)=disttype, out(2)=datatype out(3)=gridtype (see
field_geometry for help).")
{

Matrix out(3,1);


out(0,0)=(double)pField->disttype;
out(1,0)=(double)pField->datatype;
out(2,0)=(double)pField->gridtype;

return octave_value(out);
}

DEFUN_DLD (print_par, args, ,
  "Octave file print_par(). Print the parameters of the active field.")
{
Matrix out(1,1);

pField->print_par();

out(0,0)=0;
return octave_value(out);
}

DEFUN_DLD (LUT_fill, args, ,
  "Octave file LUT_fill(). Fill the LUT field with values from the active field.")
{
unsigned long sz,jx,jy,jt,indx;
double x,y,t,stx,sty,stt,xyp[2];
Matrix out(1,1);
out(0,0)=0;
```

```
sz=(unsigned long)Lut_field.numx*(unsigned long)Lut_field.numy*(unsigned long)Lut_field.numt;

if(Lut_field.memoryisallocated) {
        delete Lut_field.datax;
        delete Lut_field.datay;
        }


if (sz<=0) {
        printf("Lut_field not initialised. Use set_par.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Lut_field.datax=new float[sz];
Lut_field.datay=new float[sz];
Lut_field.memoryisallocated=1;

if ((Lut_field.datax==NULL)||(Lut_field.datay==NULL)) {
        printf("Not enough memory! Try with a smaller size.\n");
        return octave_value(out);
         }

if (pField==&Lut_field) {
        printf("The LUT cannot be filled from itself. Choose another active field.\n");
        return octave_value(out);
         }

printf("Done.\n\n");
pField->print_par();
printf("Filling the LUT with data from the active field...\n",sz);

stx=(Lut_field.xf-Lut_field.xi)/(double)(Lut_field.numx-1);
sty=(Lut_field.yf-Lut_field.yi)/(double)(Lut_field.numy-1);
stt=(Lut_field.tf-Lut_field.ti)/(double)(Lut_field.numt-1);


for(jt=0;jt<Lut_field.numt;jt++) for(jy=0;jy<Lut_field.numy;jy++) for(jx=0;jx<Lut_field.numx;jx++) {

x=Lut_field.xi+stx*(double)jx;
y=Lut_field.yi+sty*(double)jy;
t=Lut_field.ti+stt*(double)jt;

pField->xyp(t,x,y,xyp);
indx=jx+jy*Lut_field.numx+jt*Lut_field.numx*Lut_field.numy;
//printf("x:%lf y:%lf t:%lf u:%lf v:%lf indx:%ld\n",x,y,t,xyp[0],xyp[1],indx);
Lut_field.datax[indx]=(float)xyp[0];
Lut_field.datay[indx]=(float)xyp[1];
}


printf("Done.\n\n");



out(0,0)=0;
return octave_value(out);
}

DEFUN_DLD (LUT_DVD_fill, args, ,
  "Octave file LUT_DVD_fill(t). Fill the LUT from the DVD.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y,t,par[8];
float fx,fy;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
t=(double)(unsigned)args(0).scalar_value();

par[0]=0.0;
```

```
par[3]=0.0;
par[1]=331*1e6-1.0/331+1e6; //in cm! The speed is cm/sec
par[4]=156*1e6-1.0/1e6;
par[2]=331.01;
par[5]=156.01;
par[6]=0;
par[7]=t*24*60*60-1.0/(t*24*60*60);//in sec!
par[8]=t+1;

Lut_field.set_par(par);
sz=331*156*(unsigned long)Lut_field.numt;

if(Lut_field.memoryisallocated) {
        delete Lut_field.datax;
        delete Lut_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Lut_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Lut_field.datax=new float[sz];
Lut_field.datay=new float[sz];
Lut_field.memoryisallocated=1;

if ((Lut_field.datax==NULL)||(Lut_field.datay==NULL)) {
        printf("Not enough memory! Try with a smaller time.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the LUT with data from IMAGEN files (second layer)...\n");
fidx=fopen("/home/dovidio/data/IMAGEN/u_02","rb");
fidy=fopen("/home/dovidio/data/IMAGEN/v_02","rb");
printf("Day:");
for(jt=0;jt<Lut_field.numt;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Lut_field.numy;jy++) for(jx=0;jx<Lut_field.numx;jx++) {

                fread(&fx,4,1,fidx);
                fread(&fy,4,1,fidy);

                indx=jx+((Lut_field.numy-1)-jy)*Lut_field.numx+jt*Lut_field.numx*Lut_field.numy;
                Lut_field.datax[indx]=(float)fx;
                Lut_field.datay[indx]=-(float)fy;
                }
        }

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}

DEFUN_DLD (LUT_IMAGEN_fill, args, ,
  "Octave file LUT_IMAGEN_fill(t). Fill the LUT from the DVD.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y,t,par[8];
float fx,fy,RTerra=6370e5,lambda;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
```

```
t=(double)(unsigned)args(0).scalar_value();

par[0]=354.7500;
par[3]=30.2415;
par[1]=396.0000; //in cm! The speed is cm/sec
par[4]=45.4424;
par[2]=331.01;
par[5]=156.01;
par[6]=0;
par[7]=t*24*60*60-1.0/(t*24*60*60);//in sec!
par[8]=t+1;

Lut_field.set_par(par);
sz=331*156*(unsigned long)Lut_field.numt;

if(Lut_field.memoryisallocated) {
        delete Lut_field.datax;
        delete Lut_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Lut_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Lut_field.datax=new float[sz];
Lut_field.datay=new float[sz];
Lut_field.memoryisallocated=1;

if ((Lut_field.datax==NULL)||(Lut_field.datay==NULL)) {
        printf("Not enough memory! Try with a smaller time.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the LUT with data from IMAGEN files (second layer)...\n");
fidx=fopen("/home/dovidio/data/IMAGEN/u_02","rb");
fidy=fopen("/home/dovidio/data/IMAGEN/v_02","rb");
printf("Day:");
for(jt=0;jt<Lut_field.numt;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Lut_field.numy;jy++) for(jx=0;jx<Lut_field.numx;jx++) {

                fread(&fx,4,1,fidx);
                fread(&fy,4,1,fidy);

                lambda=par[3]+(par[4]-par[3])*(float)jy/(float)Lut_field.numy;

                indx=jx+jy*Lut_field.numx+jt*Lut_field.numx*Lut_field.numy;
                Lut_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Lut_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}

DEFUN_DLD (LUT_frame_fill, args, ,
  "Octave file LUT_frame_fill(U,V,framenumber). Fill the LUT frame framenumber with matrix M.")
{
unsigned long sz,jx,jy,jt,indx,framenumber;
```

```
Matrix out(1,1);
Matrix U=args(0).matrix_value();
Matrix V=args(1).matrix_value();

framenumber=(unsigned long)args(2).scalar_value();
out(0,0)=0;

sz=(unsigned long)Lut_field.numx*(unsigned long)Lut_field.numy*(unsigned long)Lut_field.numt;

if (sz<=0) {
        printf("Lut_field not initialised. Use set_par.\n");
         return octave_value(out);
         }

if((framenumber<0)||(framenumber>Lut_field.numt)) {
printf("Invalid frame number.\n");
        return octave_value(out);
         }

if((U.rows()!=Lut_field.numx)||(U.columns()!=Lut_field.numy)) {
printf("Frame size mismatch.\n");
        return octave_value(out);
         }

if((V.rows()!=Lut_field.numx)||(V.columns()!=Lut_field.numy)) {
printf("Frame size mismatch.\n");
        return octave_value(out);
         }

jt=framenumber;

if((jt>(Lut_field.numt-1))||(jt<0)) {
        printf("Wrong framenumber (use 0<=framenumber<=%d, or reset the parameters with set_par).
\n",Lut_field.numt-1);
        return octave_value(out);
        }

if(Lut_field.memoryisallocated==0) {
        printf("Trying to reserve two %ld float arrays...\n",sz);

        Lut_field.datax=new float[sz];
        Lut_field.datay=new float[sz];
        Lut_field.memoryisallocated=1;

        if ((Lut_field.datax==NULL)||(Lut_field.datay==NULL)) {
                printf("Not enough memory! Try with a smaller size.\n");
                Lut_field.memoryisallocated=0;

                return octave_value(out);
                }
        for(indx=0;indx<sz;indx++) {Lut_field.datax[indx]=0;Lut_field.datay[indx]=0;}
        }

for(jy=0;jy<Lut_field.numy;jy++) for(jx=0;jx<Lut_field.numx;jx++) {

indx=jx+jy*Lut_field.numx+jt*Lut_field.numx*Lut_field.numy;
//printf("jx:%d jy:%d u:%lf v:%lf indx:%ld\n",jx,jy,U(jx,jy),V(jx,jy),indx);
Lut_field.datax[indx]=(float)U(jx,jy);
Lut_field.datay[indx]=(float)V(jx,jy);
}


printf("Loaded frame %ld.\n",jt);



out(0,0)=0;
return octave_value(out);
}
```

```
DEFUN_DLD (IMAGEN_fill, args, ,
  "Octave file IMAGEN_fill(t). Fill the IMAGEN field from the IMAGEN files.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays;
float fx,fy,RTerra=6370e5,lambda;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();

//Imagen_field.set_par(par);
sz=331*156*numdays;

if(Imagen_field.memoryisallocated) {
        delete Imagen_field.datax;
        delete Imagen_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Imagen_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Imagen_field.datax=new float[sz];
Imagen_field.datay=new float[sz];
Imagen_field.memoryisallocated=1;

if ((Imagen_field.datax==NULL)||(Imagen_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }
Imagen_field.numt=numdays;
printf("Done.\n\n");

printf("Filling the Imagen field with data from IMAGEN files (second layer)...\n");
fidx=fopen("/home/dovidio/data/IMAGEN/u_02","rb");
fidy=fopen("/home/dovidio/data/IMAGEN/v_02","rb");
printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<156;jy++) for(jx=0;jx<331;jx++) {

                fread(&fx,4,1,fidx);
                fread(&fy,4,1,fidy);

                lambda=posy2lat(jy);

                indx=jx+jy*331+jt*156*331;
                Imagen_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Imagen_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}
DEFUN_DLD (IMAGEN_fill_nomean, args, ,
  "Octave file IMAGEN_fill_nomean(t). Fill the IMAGEN field from the IMAGEN files.")
{
long jx,jy,jt,indx,sz,indxt;
double x,y;
```

```
int numdays;
float fx,fy,fUt,fVt,RTerra=6370e5,lambda,Ut[51636],Vt[51636];
FILE *fidx,*fidy,*fidUt,*fidVt;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();

printf("\n**Velocity data from IMAGEN second layer SUBTRACTING THE AVERAGE**");

//Imagen_field.set_par(par);
sz=331*156*numdays;

if(Imagen_field.memoryisallocated) {
        delete Imagen_field.datax;
        delete Imagen_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Imagen_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Imagen_field.datax=new float[sz];
Imagen_field.datay=new float[sz];
Imagen_field.memoryisallocated=1;

if ((Imagen_field.datax==NULL)||(Imagen_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }
Imagen_field.numt=numdays;
printf("Done.\n\n");

printf("Filling the Imagen field with data from IMAGEN files (second layer)...\n");

fidx=fopen("/home/dovidio/data/IMAGEN/u_02","rb");
fidy=fopen("/home/dovidio/data/IMAGEN/v_02","rb");
fidUt=fopen("/home/dovidio/data/IMAGEN/mean_u_02","rb");
fidVt=fopen("/home/dovidio/data/IMAGEN/mean_v_02","rb");

for(jy=0;jy<156;jy++) for(jx=0;jx<331;jx++) {
        indxt=jx+jy*331;

        fread(&fUt,4,1,fidUt);
        fread(&fVt,4,1,fidVt);

        Ut[indxt]=fUt;
        Vt[indxt]=fVt;

        }


printf("Day:");

for(jt=0;jt<numdays;jt++) {

        printf("%ld... ",jt+1);
        for(jy=0;jy<156;jy++) for(jx=0;jx<331;jx++) {

                fread(&fx,4,1,fidx);
                fread(&fy,4,1,fidy);

                lambda=posy2lat(jy);

                indx=jx+jy*331+jt*156*331;
                indxt=jx+jy*331;
                fx=fx-Ut[indxt];
                fy=fy-Vt[indxt];
                Imagen_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Imagen_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
```

```
                }
        }

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
fclose(fidUt);
fclose(fidVt);
return octave_value(out);
}

DEFUN_DLD (altimetry_fill, args, ,
  "Octave file altimetry_fill(t). Fill the altimetry field from the altimetry files.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num;
float fx,fy,RTerra=6370e5,lambda;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();

//Imagen_field.set_par(par);
sz=205*80*numdays;

if(Altimetry_field.memoryisallocated) {
        delete Altimetry_field.datax;
        delete Altimetry_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Altimetry_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Altimetry_field.datax=new float[sz];
Altimetry_field.datay=new float[sz];

if ((Altimetry_field.datax==NULL)||(Altimetry_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the altimetry field with data from Altimetry files (29 March '95-30 September '99)...
\n");
fidx=fopen("/home/dovidio/data/altimetry/u99bin.dat","rb");
fidy=fopen("/home/dovidio/data/altimetry/v99bin.dat","rb");
if ((fidx==NULL)||(fidy==NULL)) {
        printf("Altimetry files not found.\n");
        return octave_value(out);
         }

printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jx=0;jx<205;jx++) for(jy=0;jy<80;jy++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in altimetry files, reading aborted. Try with less days.
\n");
```

```
                        delete Altimetry_field.datax;
                        delete Altimetry_field.datay;
                        Altimetry_field.default_par();

                        return octave_value(out);
                        }
                lambda=30+jy*.2;

                indx=jx+jy*205+jt*80*205;
                Altimetry_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Altimetry_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }

Altimetry_field.memoryisallocated=1;
Altimetry_field.numt=numdays;

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}

DEFUN_DLD (altimetry_MDT_fill, args, ,
  "Octave file altimetry_MDT_fill(t,flag). Fill the altimetry field from the altimetry files. flag: -1
altimetry only; 0 altimetry+MDT; +1 MDT only.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag;
float fx,fy,fxvMDT[205*80],fyvMDT[205*80],RTerra=6370e5,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy,*fidxMDT,*fidyMDT;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();
flag=(int)args(1).scalar_value();

//Imagen_field.set_par(par);
sz=205*80*numdays;

if(Altimetry_field.memoryisallocated) {
        delete Altimetry_field.datax;
        delete Altimetry_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Altimetry_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Altimetry_field.datax=new float[sz];
Altimetry_field.datay=new float[sz];

if ((Altimetry_field.datax==NULL)||(Altimetry_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the altimetry field with data from Altimetry files (29 March '95-30 September '99)
with MDT field...\n");
fidx=fopen("/home/dovidio/data/altimetry/u99bin.dat","rb");
fidy=fopen("/home/dovidio/data/altimetry/v99bin.dat","rb");
fidxMDT=fopen("/home/dovidio/data/altimetry/uMDTbin.dat","rb");
```

```
fidyMDT=fopen("/home/dovidio/data/altimetry/vMDTbin.dat","rb");


if ((fidx==NULL)||(fidy==NULL)) {
        printf("Altimetry files not found.\n");
        return octave_value(out);
         }

if ((fidxMDT==NULL)||(fidyMDT==NULL)) {
        printf("MDT files not found.\n");
        return octave_value(out);
         }

for(jt=0;jt<205*80;jt++) {
        fx_numMDT=fread(&fxMDT,4,1,fidxMDT);
        fy_numMDT=fread(&fyMDT,4,1,fidyMDT);
        fxvMDT[jt]=fxMDT;
        fyvMDT[jt]=fyMDT;
        if ((fx_numMDT==0)||(fy_numMDT==0)) {
                printf("MDT files corrupted, reading aborted.\n");

                delete Altimetry_field.datax;
                delete Altimetry_field.datay;
                Altimetry_field.default_par();

                return octave_value(out);
                }
        }

printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jx=0;jx<205;jx++) for(jy=0;jy<80;jy++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in altimetry files, reading aborted. Try with less days.
\n");

                        delete Altimetry_field.datax;
                        delete Altimetry_field.datay;
                        Altimetry_field.default_par();

                        return octave_value(out);
                        }
                lambda=30+jy*.2;

                indx=jx+jy*205+jt*80*205;

                if(flag==0) {
                        fx=fx+fxvMDT[jx*80+jy];
                        fy=fy+fyvMDT[jx*80+jy];
                        }
                if(flag==1) {
                        fx=fxvMDT[jx*80+jy];
                        fy=fyvMDT[jx*80+jy];
                        }

                Altimetry_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Altimetry_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }

Altimetry_field.memoryisallocated=1;
Altimetry_field.numt=numdays;

printf("Done.\n\n");
```

```
out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}

DEFUN_DLD (Aviso_Med_fill, args, ,
  "Octave file Aviso_Med_fill(t). Fill the AVISO ( Med.) field from the altimetry files. t=number of
frames (1 for each week).")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag;
float fx,fy,RTerra=6370e5,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();

//Imagen_field.set_par(par);
sz=Aviso_Med_field.numx*Aviso_Med_field.numy*numdays;

printf("Memory is allocated: %d\n",Aviso_Med_field.memoryisallocated);

if(Aviso_Med_field.memoryisallocated) {
        printf("Deleting previously allocated memory...\n");
        delete Altimetry_field.datax;
        delete Altimetry_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Aviso_Med_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Aviso_Med_field.datax=new float[sz];
Aviso_Med_field.datay=new float[sz];

if ((Aviso_Med_field.datax==NULL)||(Aviso_Med_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the altimetry field with data from AVISO files (1 January 2003 - 31 December 2004)...
\n");
fidx=fopen("/user/dovidio/data/Aviso_Med/altim_2003_2004_u.bin","rb");
fidy=fopen("/user/dovidio/data/Aviso_Med/altim_2003_2004_v.bin","rb");

if ((fidx==NULL)||(fidy==NULL)) {
        printf("AVISO ( Med.) velocity files not found.\n");
        return octave_value(out);
         }


printf("Day (1 day for each week):");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Aviso_Med_field.numy;jy++) for(jx=0;jx<Aviso_Med_field.numx;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in AVISO files, reading aborted. Try with less days.\n");

                        delete Aviso_Med_field.datax;
                        delete Aviso_Med_field.datay;
                        Aviso_Med_field.default_par();
```

```
                        return octave_value(out);
                        }
                lambda=Aviso_Med_field.lat_i+jy*.125;

                indx=jx+jy*Aviso_Med_field.numx+jt*Aviso_Med_field.numy*Aviso_Med_field.numx;

                Aviso_Med_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Aviso_Med_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }

Aviso_Med_field.memoryisallocated=1;
Aviso_Med_field.numt=numdays;

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}


DEFUN_DLD (Aviso_Med_fill_long, args, ,
  "Octave file Aviso_Med_fill_long(numframes,framestoskip). Fill the AVISO ( Med.) field from the
altimetry files. numframes=number of frames (1 for each week), framestoskip=frames to skip from 14
Oct. 1992 (>=0).")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag,framestoskip;
float fx,fy,RTerra=6370e5,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();
framestoskip=(int)args(1).scalar_value();


//Imagen_field.set_par(par);
sz=Aviso_Med_field.numx*Aviso_Med_field.numy*numdays;
printf("Memory is allocated: %d\n",Aviso_Med_field.memoryisallocated);

Aviso_Med_field.allocate_mem(sz);

/*
if(Aviso_Med_field.memoryisallocated) {
        printf("Deleting previously allocated memory...\n");
        delete Altimetry_field.datax;
        delete Altimetry_field.datay;
        Aviso_Med_field.memoryisallocated=0;
        printf("Done.\n");
        }


if (sz<=0) {
        printf("Error in initialising Aviso_Med_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

//Aviso_Med_field.datax=new float[sz];
//Aviso_Med_field.datay=new float[sz];

Aviso_Med_field.datax=new float[2470608];
Aviso_Med_field.datay=new float[2470608];
Aviso_Med_field.memoryisallocated=1;

if ((Aviso_Med_field.datax==NULL)||(Aviso_Med_field.datay==NULL)) {
```

```
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }


printf("Done.\n\n");
*/
//printf("Filling the altimetry field with data from AVISO files (14 October 1992 - 18 May 2005)...
\n");
//fidx=fopen("/user/dovidio/data/Aviso_Med/altim_2003_2004_u.bin","rb");
//fidy=fopen("/user/dovidio/data/Aviso_Med/altim_2003_2004_v.bin","rb");


printf("Filling the altimetry field with data from AVISO files (14 October 1992 - 18 May 2005)...\n");
fidx=fopen("/rdsk/dovidio/aviso_vels/altim_19921014_20050518_u.bin","rb");
fidy=fopen("/rdsk/dovidio/aviso_vels/altim_19921014_20050518_v.bin","rb");

if ((fidx==NULL)||(fidy==NULL)) {
        printf("AVISO ( Med.) velocity files not found.\n");
        return octave_value(out);
         }

fseek(fidx,(long)framestoskip*4*Aviso_Med_field.numx*Aviso_Med_field.numy,SEEK_SET);
fseek(fidy,(long)framestoskip*4*Aviso_Med_field.numx*Aviso_Med_field.numy,SEEK_SET);


printf("Day (1 day for each week):");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Aviso_Med_field.numy;jy++) for(jx=0;jx<Aviso_Med_field.numx;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in AVISO files, reading aborted. Try with less days.\n");

                        delete Aviso_Med_field.datax;
                        delete Aviso_Med_field.datay;
                        Aviso_Med_field.default_par();

                        return octave_value(out);
                        }
                lambda=Aviso_Med_field.lat_i+jy*.125;

                indx=jx+jy*Aviso_Med_field.numx+jt*Aviso_Med_field.numy*Aviso_Med_field.numx;

                Aviso_Med_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Aviso_Med_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }
fclose(fidx);
fclose(fidy);


Aviso_Med_field.numt=numdays;
Aviso_Med_field.framesskipped=framestoskip;
printf("Done.\n\n");



out(0,0)=0;

return octave_value(out);
}

DEFUN_DLD (Kerguelen2004_2005_fill, args, ,
  "Octave file Kerguelen2004_2005_fill(numframes,framestoskip). Fill the Kerguelen field from the
altimetry files. numframes=number of frames (1 for each day), framestoskip=frames to skip from 1 July
(>=0).")
{
unsigned long jx,jy,jt,indx,sz;
```

```
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag,framestoskip;
float fx,fy,RTerra=6370e5,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();
framestoskip=(int)args(1).scalar_value();
Kerguelen_field.lat_i=-59.7500;
Kerguelen_field.lat_f=-40.2500;
Kerguelen_field.lon_i=60.25;
Kerguelen_field.lon_f=89.75;
Kerguelen_field.numx=119;
Kerguelen_field.numy=79;
//Imagen_field.set_par(par);
sz=Kerguelen_field.numx*Kerguelen_field.numy*numdays;
printf("Memory is allocated: %d\n",Kerguelen_field.memoryisallocated);

Kerguelen_field.allocate_mem(sz);


printf("Filling the Kerguelen field with data from Kerguelen files (1 July 2004-30 June 2005)...\n");
fidx=fopen("/home/dovidio/octave/kerguelen/Kerguelen_U_20040701_20050630.bin","rb");
fidy=fopen("/home/dovidio/octave/kerguelen/Kerguelen_V_20040701_20050630.bin","rb");

if ((fidx==NULL)||(fidy==NULL)) {
        printf("Kerguelen velocity files not found.\n");
        return octave_value(out);
         }

fseek(fidx,(long)framestoskip*4*Kerguelen_field.numx*Kerguelen_field.numy,SEEK_SET);
fseek(fidy,(long)framestoskip*4*Kerguelen_field.numx*Kerguelen_field.numy,SEEK_SET);


printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Kerguelen_field.numy;jy++) for(jx=0;jx<Kerguelen_field.numx;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in Kerguelen files, reading aborted. Try with less days.
\n");

                        delete Kerguelen_field.datax;
                        delete Kerguelen_field.datay;
                        Kerguelen_field.default_par();

                        return octave_value(out);
                        }
                lambda=Kerguelen_field.lat_i+jy*.125;

                indx=jx+jy*Kerguelen_field.numx+jt*Kerguelen_field.numy*Kerguelen_field.numx;

                Kerguelen_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Kerguelen_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }
fclose(fidx);
fclose(fidy);


Kerguelen_field.numt=numdays;
Kerguelen_field.framesskipped=framestoskip;
printf("Done.\n\n");



out(0,0)=0;
```

```
return octave_value(out);
}

DEFUN_DLD (Kerguelen2003_2006_fill, args, ,
  "Octave file Kerguelen2003_2006_fill(numframes,framestoskip). Fill the Kerguelen field from the
altimetry files. numframes=number of frames (1 for each day), framestoskip=frames to skip from 1 July
(>=0).")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag,framestoskip;
float fx,fy,RTerra=6370e5,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();
framestoskip=(int)args(1).scalar_value();
Kerguelen_field.lat_i=-59.7500;
Kerguelen_field.lat_f=-35.2500;
Kerguelen_field.lon_i=57.25;
Kerguelen_field.lon_f=109.75;
Kerguelen_field.numx=211;
Kerguelen_field.numy=99;

//Imagen_field.set_par(par);
sz=Kerguelen_field.numx*Kerguelen_field.numy*numdays;
printf("Memory is allocated: %d\n",Kerguelen_field.memoryisallocated);

Kerguelen_field.allocate_mem(sz);


printf("Filling the Kerguelen field with data from Kerguelen files (1 July 2003-30 June 2006)...\n");
fidx=fopen("/home/dovidio/octave/kerguelen/Kerguelen_U_20030701_20060630.bin","rb");
fidy=fopen("/home/dovidio/octave/kerguelen/Kerguelen_V_20030701_20060630.bin","rb");

if ((fidx==NULL)||(fidy==NULL)) {
        printf("Kerguelen velocity files not found.\n");
        return octave_value(out);
         }

fseek(fidx,(long)framestoskip*4*Kerguelen_field.numx*Kerguelen_field.numy,SEEK_SET);
fseek(fidy,(long)framestoskip*4*Kerguelen_field.numx*Kerguelen_field.numy,SEEK_SET);


printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Kerguelen_field.numy;jy++) for(jx=0;jx<Kerguelen_field.numx;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in Kerguelen files, reading aborted. Try with less days.
\n");

                        delete Kerguelen_field.datax;
                        delete Kerguelen_field.datay;
                        Kerguelen_field.default_par();

                        return octave_value(out);
                        }
                lambda=Kerguelen_field.lat_i+jy*.125;

                indx=jx+jy*Kerguelen_field.numx+jt*Kerguelen_field.numy*Kerguelen_field.numx;

                Kerguelen_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Kerguelen_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }
fclose(fidx);
fclose(fidy);
```

```
Kerguelen_field.numt=numdays;
Kerguelen_field.framesskipped=framestoskip;
printf("Done.\n\n");



out(0,0)=0;

return octave_value(out);
}
DEFUN_DLD (Pco2_fill, args, ,
  "Octave file Pco2_fill(). Fill the Pco2 field.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag,framestoskip;
float fx,fy,RTerra=637122900.0,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=89;
sz=Pco2_field.numx*Pco2_field.numy*numdays;
printf("Memory is allocated: %d\n",Pco2_field.memoryisallocated);

Pco2_field.allocate_mem(sz);


printf("Filling the Pco2 field with data from Pco2 files ...\n");
fidx=fopen("/home/dovidio/data/pco2/POM.08.ext.surf.grid.U.bin","rb");
fidy=fopen("/home/dovidio/data/pco2/POM.08.ext.surf.grid.V.bin","rb");

if ((fidx==NULL)||(fidy==NULL)) {
        printf("Pco2 files not found.\n");
        return octave_value(out);
         }

printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Pco2_field.numy;jy++) for(jx=0;jx<Pco2_field.numx;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in Pco2 files, reading aborted.\n");

                        delete Pco2_field.datax;
                        delete Pco2_field.datay;
                        Pco2_field.default_par();

                        return octave_value(out);
                        }
                lambda=Pco2_field.lat_i+jy*.05;

                indx=jx+jy*Pco2_field.numx+jt*Pco2_field.numy*Pco2_field.numx;

                Pco2_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                //Pco2_field.datax[indx]=(float)fx/RTerra*360.0/(2.0*pi);
                Pco2_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }
fclose(fidx);
fclose(fidy);


Pco2_field.numt=numdays;
printf("Done.\n\n");
```

```
out(0,0)=0;

return octave_value(out);
}
DEFUN_DLD (Pco2_fill_na, args, ,
  "Octave file Pco2_fill_na(). Fill the Pco2 field with data with no assimilation.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag,framestoskip;
float fx,fy,RTerra=637122900.0,lambda,fxMDT,fyMDT;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
numdays=89;
sz=Pco2_field.numx*Pco2_field.numy*numdays;
printf("Memory is allocated: %d\n",Pco2_field.memoryisallocated);

Pco2_field.allocate_mem(sz);


printf("Filling the Pco2 field with data from Pco2 files ...\n");
fidx=fopen("/home/dovidio/data/pco2/POM.ext.54.grid.U.bin","rb");
fidy=fopen("/home/dovidio/data/pco2/POM.ext.54.grid.V.bin","rb");

if ((fidx==NULL)||(fidy==NULL)) {
        printf("Pco2 files not found.\n");
        return octave_value(out);
         }

printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<Pco2_field.numy;jy++) for(jx=0;jx<Pco2_field.numx;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in Pco2 files, reading aborted.\n");

                        delete Pco2_field.datax;
                        delete Pco2_field.datay;
                        Pco2_field.default_par();

                        return octave_value(out);
                        }
                lambda=Pco2_field.lat_i+jy*.05;

                indx=jx+jy*Pco2_field.numx+jt*Pco2_field.numy*Pco2_field.numx;

                Pco2_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                //Pco2_field.datax[indx]=(float)fx/RTerra*360.0/(2.0*pi);
                Pco2_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }
fclose(fidx);
fclose(fidy);


Pco2_field.numt=numdays;
printf("Done.\n\n");



out(0,0)=0;

return octave_value(out);
}

DEFUN_DLD (windERA40_fill, args, ,
```

```
  "Octave file windERA40_fill(fr_num,fr_offset). Fill the wind datsets from the wind datafiles. It
assumes data every 6 hours, 181x360 at each deg.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int fr_num,fx_num,fy_num,fr_offset;
float fx,fy,RTerra_m=RTerra/100,lambda;
FILE *fidx,*fidy;
Matrix out(1,1);
out(0,0)=0;
fr_num=(int)args(0).scalar_value(); //number of frames 360x181 (one every six hours).
fr_offset=(int)args(1).scalar_value(); //number of frames 360x181 (one every six hours) to jump.

sz=181*360*fr_num;

if(WindERA40_field.memoryisallocated) {
        delete WindERA40_field.datax;
        delete WindERA40_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising WindERA40_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

WindERA40_field.datax=new float[sz];
WindERA40_field.datay=new float[sz];

if ((WindERA40_field.datax==NULL)||(WindERA40_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the ERA40 field with data from ERA40 files (U96_09_11.bin, V96_09_11.bin, m/sec) ...
\n");
fidx=fopen("/home/dovidio/data/demetra/vel96field/U96_09_11.bin","rb");
fidy=fopen("/home/dovidio/data/demetra/vel96field/V96_09_11.bin","rb");


if ((fidx==NULL)||(fidy==NULL)) {
        printf("ERA40 files not found.\n");
        return octave_value(out);
         }

if(WindERA40_field.datatype==1) printf("Velocity converted in deg/sec\n");
else printf("Velocity converted in cm/sec\n");

printf("Frame number:");
for(jt=0;jt<fr_num;jt++) {

        printf("%lu... ",jt);
        for(jy=0;jy<181;jy++) for(jx=0;jx<360;jx++) {

                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in ERA40 files, reading aborted. Try with less frames.
(Frames=%d)\n",jt);

                        delete WindERA40_field.datax;
                        delete WindERA40_field.datay;
                        WindERA40_field.default_par();

                        return octave_value(out);
                        }
                lambda=jy+90;

                indx=jx+jy*360+jt*181*360;
```

```
                //WindERA40_field.datax[indx]=(float)fx;
                //WindERA40_field.datay[indx]=(float)fy;

                if(WindERA40_field.datatype==1) {
                        WindERA40_field.datax[indx]=(float)fx*100/fabs(RTerra*cos(lambda/360.0*2.0*pi))
*360.0/(2.0*pi);
                        WindERA40_field.datay[indx]=(float)fy*100/RTerra*360.0/(2.0*pi);
                        }
                else {
                        WindERA40_field.datax[indx]=(float)fx*100;
                        WindERA40_field.datay[indx]=(float)fy*100;
                        }
                }
        }

WindERA40_field.memoryisallocated=1;
WindERA40_field.numt=fr_num;

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}




DEFUN_DLD (IMAGEN_get_frame_u, args, ,
  "Octave file IMAGEN_get_frame_u(t). It gives the u velocity field at day t.")
{

long day,jx,jy,indx;
Matrix out(331,156);
day=(long)args(0).scalar_value();
if (day<1) {
        printf("\nIMAGEN_get_frame_u(t), t must be a positive integer.\n");
        return octave_value(out);
        }

if((!Imagen_field.memoryisallocated)||(day>Imagen_field.numt)) {
        printf("\nIMAGEN field not allocated or t too large!\n");
        return octave_value(out);
        }



for(jy=0;jy<156;jy++) for(jx=0;jx<331;jx++) {
        indx=jx+jy*331+(day-1)*156*331;
        out(jx,jy)=Imagen_field.datax[indx];
        }

return octave_value(out);
}
DEFUN_DLD (IMAGEN_get_frame_v, args, ,
  "Octave file IMAGEN_get_frame_v(t). It gives the v velocity field at day t.")
{

long day,jx,jy,indx;
Matrix out(331,156);
day=(long)args(0).scalar_value();
if (day<1) {
        printf("\nIMAGEN_get_frame_v(t), t must be a positive integer.\n");
        return octave_value(out);
        }

if((!Imagen_field.memoryisallocated)||(day>Imagen_field.numt)) {
```

```
        printf("\nIMAGEN field not allocated or t too large!\n");
        return octave_value(out);
        }



for(jy=0;jy<156;jy++) for(jx=0;jx<331;jx++) {
        indx=jx+jy*331+(day-1)*156*331;
        out(jx,jy)=Imagen_field.datay[indx];
        }

return octave_value(out);
}


DEFUN_DLD (POMME_MDT_fill, args, ,
  "Octave file POMME_MDT_fill(t,flag). Fill the POMME field from the altimetry files. flag: -1
altimetry only; 0 altimetry+MDT; +1 MDT only.")
{
unsigned long jx,jy,jt,indx,sz;
double x,y;
int
numdays,fx_num,fy_num,fx_numMDT,fy_numMDT,flag,latg_num=Pomme_field.numy,long_num=Pomme_field.numx;
float fx,fy,fxvMDT[long_num*latg_num],fyvMDT[long_num*latg_num],RTerra=6370e5,lambda,fxMDT,fyMDT;
double delta_lambda=(Pomme_field.lat_f-Pomme_field.lat_i)/(Pomme_field.numy-1);
FILE *fidx,*fidy,*fidxMDT,*fidyMDT;
Matrix out(1,1);
out(0,0)=0;
numdays=(int)args(0).scalar_value();
flag=(int)args(1).scalar_value();

//Imagen_field.set_par(par);
sz=latg_num*long_num*numdays;
printf("sz:%ld\n",sz);
if(Pomme_field.memoryisallocated) {
        delete Pomme_field.datax;
        delete Pomme_field.datay;
        }


if (sz<=0) {
        printf("Error in initialising Pomme_field.\n");
         return octave_value(out);
         }
printf("Trying to reserve two %ld float arrays...\n",sz);

Pomme_field.datax=new float[sz];
Pomme_field.datay=new float[sz];

if ((Pomme_field.datax==NULL)||(Pomme_field.datay==NULL)) {
        printf("Not enough memory! Please try with less days.\n");
        return octave_value(out);
         }

printf("Done.\n\n");

printf("Filling the POMME field with data from Altimetry files () with MDT field...\n");
fidx=fopen("/home/dovidio/data/POMME/POMME_ugmerged.bin","rb");
fidy=fopen("/home/dovidio/data/POMME/POMME_vgmerged.bin","rb");
fidxMDT=fopen("/home/dovidio/data/POMME/POMME_ugMDT.bin","rb");
fidyMDT=fopen("/home/dovidio/data/POMME/POMME_vgMDT.bin","rb");


if ((fidx==NULL)||(fidy==NULL)) {
        printf("POMME velocity field not found.\n");
        return octave_value(out);
         }

if ((flag!=-1)&&((fidxMDT==NULL)||(fidyMDT==NULL))) {
        printf("MDT files not found.\n");
        return octave_value(out);
         }
```

```
for(jx=0;jx<long_num;jx++) for(jy=0;jy<latg_num;jy++) {
        fx_numMDT=fread(&fxMDT,4,1,fidxMDT);
        fy_numMDT=fread(&fyMDT,4,1,fidyMDT);
        jt=jx+jy*long_num;
        fxvMDT[jt]=fxMDT;
        fyvMDT[jt]=fyMDT;
        if ((fx_numMDT==0)||(fy_numMDT==0)) {
                printf("MDT files corrupted, reading aborted.\n");

                delete Pomme_field.datax;
                delete Pomme_field.datay;
                Pomme_field.default_par();

                return octave_value(out);
                }
        }

printf("Day:");
for(jt=0;jt<numdays;jt++) {

        printf("%lu... ",jt);
        for(jx=0;jx<long_num;jx++) for(jy=0;jy<latg_num;jy++) {
                fx_num=fread(&fx,4,1,fidx);
                fy_num=fread(&fy,4,1,fidy);
                if ((fx_num==0)||(fy_num==0)) {
                        printf("EOF reached in altimetry files, reading aborted. Try with less days.
\n");

                        delete Pomme_field.datax;
                        delete Pomme_field.datay;
                        Pomme_field.default_par();

                        return octave_value(out);
                        }
                lambda=Pomme_field.lat_i+jy*delta_lambda;

                indx=jx+jy*long_num+jt*long_num*latg_num;
                //printf("indx=%d/n",indx);

                if(flag==0) {
                        fx=fx+fxvMDT[jy*long_num+jx];
                        fy=fy+fyvMDT[jy*long_num+jx];
                        }
                if(flag==1) {
                        fx=fxvMDT[jy*long_num+jx];
                        fy=fyvMDT[jy*long_num+jx];
                        }

                Pomme_field.datax[indx]=(float)fx/(RTerra*cos(lambda/360.0*2.0*pi))*360.0/(2.0*pi);
                Pomme_field.datay[indx]=(float)fy/RTerra*360.0/(2.0*pi);
                }
        }

Pomme_field.memoryisallocated=1;
Pomme_field.numt=numdays;

printf("Done.\n\n");



out(0,0)=0;
fclose(fidx);
fclose(fidy);
return octave_value(out);
}

DEFUN_DLD (RK4_tau_v, args, ,
  "Octave file out=RK4_tau_v(tspan,x_v,x0,Nstep,delta). tspan is a 2x1 column with initial and final
time, x0 is a (1+N)x1 column (see field.h).")
{
/*
```

```
x_v is a vector with all the points where to compute the FSLEs. x0 is a vector with N points around
the origin.
*/
double tspan[2],*x0,delta,tau,xc[2];
int numx0,Nstep,elmax,j,jv,numx_v;

Matrix Mtspan=args(0).matrix_value();
Matrix Mx_v=args(1).matrix_value();
Matrix Mx0=args(2).matrix_value();
numx0=(int)Mx0.rows()/2;
numx_v=(int)Mx_v.rows()/2;
Matrix out(2,numx_v);
Nstep=(int)args(3).scalar_value();
delta=args(4).scalar_value();

x0=new double[numx0*2+2];
if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
//printf("tspan:%lf %lf numx0:%d numx_v:%d Nstep:%d delta:%lf\n",tspan[0],tspan
[1],numx0,numx_v,Nstep,delta);
//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
for(jv=0;jv<numx_v;jv++) {
        x0[0]=Mx_v(2*jv,0);
        x0[1]=Mx_v(2*jv+1,0);
//printf("jv:%d x0:%lf %lf\n",jv,Mx_v(2*jv,0),Mx_v(2*jv+1,0));
        for(j=0;j<numx0;j++) {
                x0[2+2*j]=Mx0(2*j,0)+Mx_v(2*jv,0);
                x0[2+2*j+1]=Mx0(2*j+1,0)+Mx_v(2*jv+1,0);
                }
        //for(j=0;j<(2*numx0+2);j++) printf("x0[%d]:%lf\n",j,x0[j]);
        RK4_tau(tspan,x0,numx0+1,Nstep,delta,&elmax,&tau);
//printf("elmax:%lf tau:%lf\n",(double)elmax,(double)tau);
        out(0,jv)=(double)tau;
        out(1,jv)=(double)elmax;
        }


delete x0;
return octave_value(out);
}

DEFUN_DLD (FSLEext_v, args, ,
  "Octave file out=FSLEext_v(tspan,x_v,delta0,Nstep,delta). tspan is a 2x1 column with initial and
final time, x0 is a (1+N)x1 column (see field.h).")
{
/*
x_v is a vector with all the points where to compute the FSLEs. x0 is a vector with N points around
the origin.
*/
double tspan[2],x0[2],delta,delta0,theta1,theta2,l1,l2;
int numx0,Nstep,j,jv,numx_v,dpct,mpct,Mpct;

Matrix Mtspan=args(0).matrix_value();
Matrix Mx_v=args(1).matrix_value();
numx_v=(int)Mx_v.rows()/2;
Matrix out(4,numx_v);
delta0=args(2).scalar_value();
Nstep=(int)args(3).scalar_value();
delta=args(4).scalar_value();

if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
```

```
tspan[1]=Mtspan(1,0);
//printf("tspan:%lf %lf numx0:%d numx_v:%d Nstep:%d delta:%lf\n",tspan[0],tspan
[1],numx0,numx_v,Nstep,delta);
//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
dpct=5;
mpct=5;
Mpct=mpct+dpct;
printf("\n========== START Computing FSLE ==========\n");
for(jv=0;jv<numx_v;jv++) {
        x0[0]=Mx_v(2*jv,0);
        x0[1]=Mx_v(2*jv+1,0);
        FSLEext_(tspan,x0,delta0,Nstep,delta,&l1,&l2,&theta1,&theta2);

//printf("jv:%d\n",jv);
//printf("jv/numx_v*100 = %f%%.\n",(double)jv/numx_v*100);
        if ((double)jv/numx_v*100>mpct && (double)jv/numx_v*100<Mpct){
                printf("Computing FSLE: %d%% done.\n",mpct);
                mpct=Mpct;
                Mpct=mpct+dpct;
        }
//      if (jv%100==0){
//              printf("Computed FSLE for point %d of %d \n",jv,numx_v);
//      }
        out(0,jv)=(double)l1;
        out(1,jv)=(double)l2;
        out(2,jv)=(double)theta1;
        out(3,jv)=(double)theta2;

        }

return octave_value(out);
}



DEFUN_DLD (FTLEext_v, args, ,
  "Octave file out=FTLEext_v(tspan,x_v,delta0,Nstep). tspan is a 2x1 column with initial and final
time, x0 is a (1+N)x1 column (see field.h).")
{
/*
x_v is a vector with all the points where to compute the FSLEs. x0 is a vector with N points around
the origin.
*/
double tspan[2],x0[2],delta0,theta1,theta2,l1,l2;
int numx0,Nstep,j,jv,numx_v;

Matrix Mtspan=args(0).matrix_value();
Matrix Mx_v=args(1).matrix_value();
numx_v=(int)Mx_v.rows()/2;
Matrix out(4,numx_v);
delta0=args(2).scalar_value();
Nstep=(int)args(3).scalar_value();

if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
//printf("tspan:%lf %lf numx0:%d numx_v:%d Nstep:%d delta:%lf\n",tspan[0],tspan
[1],numx0,numx_v,Nstep,delta);
//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
for(jv=0;jv<numx_v;jv++) {
        x0[0]=Mx_v(2*jv,0);
        x0[1]=Mx_v(2*jv+1,0);
        FTLEext_(tspan,x0,delta0,Nstep,&l1,&l2,&theta1,&theta2);

//printf("jv:%d\n",jv);
        out(0,jv)=(double)l1;
```

```
        out(1,jv)=(double)l2;
        out(2,jv)=(double)theta1;
        out(3,jv)=(double)theta2;

        }

return octave_value(out);
}
DEFUN_DLD (FSLEn_v, args, ,
  "Octave file out=FSLEn_v(tspan,x_v,numpt,delta0,Nstep,delta). tspan is a 2x1 column with initial and
final time, x0 is a (1+N)x1 column (see field.h).")
{
/*
x_v is a vector with all the points where to compute the FSLEs. x0 is a vector with N points around
the origin.
*/
double tspan[2],x0[2],delta,delta0,theta1,theta2,l1,l2;
int numx0,Nstep,j,jv,numx_v,numpt;

Matrix Mtspan=args(0).matrix_value();
Matrix Mx_v=args(1).matrix_value();
numx_v=(int)Mx_v.rows()/2;
Matrix out(2,numx_v);
numpt=(int)args(2).scalar_value();
delta0=args(3).scalar_value();
Nstep=(int)args(4).scalar_value();
delta=args(5).scalar_value();

if (x0==NULL) {
        printf("Error in the number of initial points.\n");
        return octave_value(out);
        }

tspan[0]=Mtspan(0,0);
tspan[1]=Mtspan(1,0);
//printf("tspan:%lf %lf numx0:%d numx_v:%d Nstep:%d delta:%lf\n",tspan[0],tspan
[1],numx0,numx_v,Nstep,delta);
//printf("tspan:%lf %lf numx0:%d Nstep:%d delta:%lf\n",tspan[0],tspan[1],numx0,Nstep,delta);
//for(j=0;j<numx0;j++) printf("x0_x:%lf x0_y:%lf\n",x0[2*j],x0[2*j+1]);
for(jv=0;jv<numx_v;jv++) {
        x0[0]=Mx_v(2*jv,0);
        x0[1]=Mx_v(2*jv+1,0);
        FSLEn_(tspan,x0,numpt,delta0,Nstep,delta,&l1,&theta1);

//printf("jv:%d\n",jv);
        out(0,jv)=(double)l1;
        out(1,jv)=(double)theta1;

        }

return octave_value(out);
}


DEFUN_DLD (UVext_v, args, ,
  "Octave file out=UVext_v(t,x_v). tspan is time, x0 is a (1+N)x1 column (see field.h).")
{

double t,xyp[2],x,y;
int jv,numx_v;

t=args(0).scalar_value();
Matrix Mx_v=args(1).matrix_value();
numx_v=(int)Mx_v.rows()/2;
Matrix out(2,numx_v);

for(jv=0;jv<numx_v;jv++) {
        x=Mx_v(2*jv,0);
        y=Mx_v(2*jv+1,0);
        pField->xyp(t,x,y,xyp);

        out(0,jv)=xyp[0];
```

```
        out(1,jv)=xyp[1];
        }

return octave_value(out);
}



DEFUN_DLD (LE2d, args, ,
  "Octave file out=LE2d(x0,deltat,delta0).x0 contains the evoluted of three points, forming at time 0
a L-shaped reference frame, deltat is the time interval, delta0 the initial separation.")
{
double deltat,l1,l2,theta1,theta2,x0[6],delta0;
int j;

Matrix out(4,1);
Matrix Mx0=args(0).matrix_value();
deltat=args(1).scalar_value();
delta0=args(2).scalar_value();

for(j=0;j<6;j++) x0[j]=Mx0(j,0);

LE2d_(x0,deltat,delta0,&l1,&l2,&theta1,&theta2);

out(0,0)=(double)l1;
out(1,0)=(double)l2;
out(2,0)=(double)theta1;
out(3,0)=(double)theta2;

return octave_value(out);
}
```