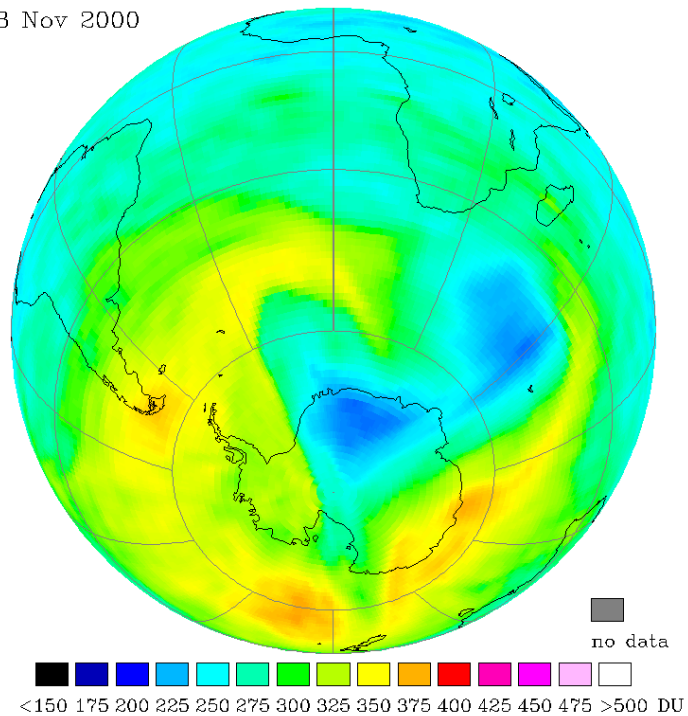


Assimilated GOME ozone fields HDF data file user manual

Henk Eskes

8 August 2002

Assimilated GOME ozone
18 Nov 2000



Contents

Content of the "o3col.hdf" files	3
Fortran 90 sample code to read the ozone fields	4
C sample code to read the ozone fields	7
A simple plot of the ozone fields using IDL	12

Content of the "o3col.hdf" files

The HDF files consist of a number of fields describing the content, followed by two arrays, one with the ozone field ("O3_column") and one with the forecast uncertainty field ("O3_std"). Notes:

- **Author, Affiliation, Email:** name and co-ordinates of the person responsible for the data.
- **Data_created_by:** the assimilation code that produced the ozone fields.
- **Ozone_field_date:** The UTC time of the ozone field (e.g. *not* the creation time of the data file).
- **Number_of_longitudes** etc: Description of the longitude-latitude grid. For instance the longitudes are given by (i runs from 1 to **Number_of_longitudes**):
$$\text{lon}(i) = \text{Longitude_range}(1) + (i-1) * \text{Longitude_step}$$
- **Field_O3_column** and **Field_O3_std:** a brief description of the fields
- **Units:** the units of the fields
- **Note:** A brief description how the arrays are stored in the file

SDS Global Attributes

Name	Value
Author	H.J. Eskes
Affiliation	KNMI (Royal Netherlands Meteorological Institute)
Email	eskes@knmi.nl
Data_created_by	TM3-DAM, version 3
Ozone_field_date	2001, 12, 5, 12, 0, 0
Date_format	year, month, day, hour, minute, second
Number_of_longitudes	240
Longitude_range	-178.5, 180.0
Longitude_step	1.5
Number_of_latitudes	181
Latitude_range	-90.0, 90.0
Latitude_step	1.0
Field_O3_column	Ozone column distribution
Field_O3_std	Ozone column forecast uncertainty
Units	Dobson units
Note	GZIP compressed 2-byte integer arrays

SDS Arrays

Name

O3_column
O3_std

Fortran 90 sample code to read the ozone fields

To read the files with Fortran 90/95, the following actions have to be taken:

- Download the HDF 4 libraries from <http://hdf.ncsa.uiuc.edu> (HDF 4.1r5)
- Include the HDF libraries in the program.
Example: add the following `HDFLIBS` line to the Makefile:

```
LIBP = -L /usr/local/lib  
HDFLIBS = -l mfhdf -l df -l z -l jpeg  
myprogram.exe: ${OBSJ} F90 ${OBSJ} ${HDFLIBS}
```

- Add the fortran header file `hdf.inc` to the directory with the program code
- Read the HDF ozone field file with the following subroutine `RdO3co1HDF`

```

subroutine RdO3colHDF(filnam,nx,ny,o3field,o3std, &
    mdtg,lonmin,lonmax,latmin,latmax)
!-----
! Read a total ozone field "o3field", dimension nx*ny,
!   from the file "filnam" in HDF 4 format
!
! "filnam" - filename of the HDF file (in)
! "nx","ny" - dimension of the fields (to be checked, input)
! "o3field" - The field to be read (out)
! "o3std" - The corresponding error estimate field to be read (out)
! "mdtg" - Time, e.g. 1999 12 3 21 0 0 (out)
! "lonmin,lonmax,latmin,latmax" - longitude/latitude min/max values (out)
!
!                                     Henk Eskes, November 2000
!-----
implicit none
! hdf parameter definition file for fortran 90:
include 'hdf.inc'
character(*),intent(in) :: filnam
integer,intent(in) :: nx,ny
real(4),dimension(nx,ny),intent(out) :: o3field,o3std
integer,dimension(6),intent(out) :: mdtg
real,intent(out) :: lonmin,lonmax,latmin,latmax
!
integer :: sds_id,att_id,io,istat,i,j
integer :: nx_file,ny_file,indx
real,dimension(2) :: lonlatrange
!
integer(2),dimension(:,:),allocatable :: io3field,io3std
!
integer :: sfstart,sfattr,sfrnatt,sfn2index
integer :: sfrdata,sfselect,sfendacc,sfend
!
allocate(io3field(nx,ny))
allocate(io3std(nx,ny))
!
io = sfstart(filnam,DFACC_READ)
att_id = sfattr(io,'Number_of_longitudes')
istat = sfrnatt(io,att_id,nx_file)
att_id = sfattr(io,'Number_of_latitudes')
istat = sfrnatt(io,att_id,ny_file)
if( (nx /= nx_file) .or. (ny /= ny_file) )then
    print*,'ERROR RdO3colHDF: wrong array dimensions on file'
    print*,' file : nx,ny = ',nx_file,ny_file
    print*,' arrays: nx,ny = ',nx,ny
    stop
end if
att_id = sfattr(io,'Ozone_field_date')
istat = sfrnatt(io,att_id,mdtg)
att_id = sfattr(io,'Longitude_range')
istat = sfrnatt(io,att_id, lonlatrange )

```

```

lonmin = lonlatrange(1)
lonmax = lonlatrange(2)
att_id = sfattr(io,'Latitude_range')
istat = sfrnatt(io,att_id, lonlatrange )
latmin = lonlatrange(1)
latmax = lonlatrange(2)
!
! read total ozone field
!
indx = sfn2index(io,'O3_column')
if (indx .eq. -1) then
  print*,'ERROR RdO3colHDF: ozone field array not found'
  stop
end if
sds_id = sfselect(io, indx)
istat = sfrdata(sds_id, (/0,0/),(/1,1/),(/nx,ny/),io3field)
if ( istat /= SUCCEED ) then
  print*,'ERROR RdO3colHDF: Failed to read o3field from file'
  stop
endif
istat = sfendacc(sds_id)
!
! read uncertainty field
!
indx = sfn2index(io,'O3_std')
if (indx .eq. -1) then
  print*,'ERROR RdO3colHDF: uncertainty field array not found'
  stop
end if
sds_id = sfselect(io, indx)
istat = sfrdata(sds_id, (/0,0/),(/1,1/),(/nx,ny/),io3std)
if ( istat /= SUCCEED ) then
  print*,'ERROR RdO3colHDF: Failed to read o3std from file'
  stop
endif
istat = sfendacc(sds_id)
!
istat = sfend(io)
!
do i=1,nx
  do j=1,ny
    o3field(i,j)=real(io3field(i,j))
    o3std(i,j)=real(io3std(i,j))
  enddo
enddo
!
deallocate(io3field)
deallocate(io3std)
!
end subroutine RdO3colHDF

```

C sample code to read the ozone fields

To read the files with C, the following actions have to be taken:

- Download the HDF 4 libraries from <http://hdf.ncsa.uiuc.edu> (HDF 4.1r5)
- Include the HDF libraries in the program.
Example: add the following `LFLAGS` line to the Makefile:

```
CC = gcc
CFLAGS = -I /include
LFLAGS = -L /lib -lmfhdf -ldf -lz -ljpeg
Read03field : Read03Field.o $(CC) ${COPTS} -o Read03Field Read03Field.o $(LFLAGS)
```

- Add the c header files (`CFLAGS`) to the directory with the program code
- Read the HDF ozone field file with the subroutine `Rd03colHDF` (as in the sample code).

```

/* Include HDF header */
#include "mfhdf.h"

#define DIMLON 240
#define DIMLAT 181

/* interface */

int RdO3colHDF( char *filnam,
float o3field[] [], float o3std[] [],
int *mdtg,
float *lonmin, float *lonmax, float *latmin, float *latmax );

/* MAIN */

main()
{
float o3field[DIMLAT][DIMLON];
float o3std[DIMLAT][DIMLON];
float lonmin, lonmax, latmin, latmax;
int  mdtg[6],ilon,ilat,rerror;
float lon,lat;
char  NameOfFile[120] = "o3col.hdf";

/*
Returns ozone value at longitude and latitude = (lon,lat)
*/

rerror = RdO3colHDF( NameOfFile, o3field, o3std,
mdtg, &lonmin, &lonmax, &latmin, &latmax );

ilon = 73 ;
ilat = 14 ;
lon = lonmin + (ilon-1)*( lonmax - lonmin ) / DIMLON ;
lat = latmin + (ilat-1)*( latmax - latmin ) / DIMLAT ;

printf ("Main: lon, lat = %f %f, O3 = %f , dO3 = %f \n",
lon,lat,o3field[ilat-1][ilon-1],o3std[ilat-1][ilon-1]);
}

/*****
RdO3colHDF:
Read a total ozone field "o3fld",
from the file "filnam" in HDF 4 format

"filnam" - filename of the HDF file (in)
"o3fld" - The field to be read (out)
"o3s" - The corresponding error estimate field to be read (out)
"mdtg" - Time, e.g. 1999 12 3 21 0 0 (out)
"lonmin,lonmax,latmin,latmax" - longitude/latitude min/max values (out)

```

```
*****/
```

```
int RdO3colHDF( char *filnam,
float o3fld[DIMLAT][DIMLON], float o3s[DIMLAT][DIMLON],
int *mdtg,
float *lonmin, float *lonmax, float *latmin, float *latmax )
{

    /* Variable declaration */

    int32 sd_id, att_id, sds_id, sds_index;
    intn status;
    int32 start[2], stride[2], edges[2];
    /* int16 data[][] */
    int i, j;
    int32 nx_file, ny_file;
    float lonlatrange[2];
    int16 O3storage[DIMLAT][DIMLON];

    /*
    * Open the file for reading and initialize the SD interface.
    */
    sd_id = SDstart (filnam, DFACC_READ);
    if (sd_id == FAIL){
        printf ("RdO3colHDF: ERROR unable to open file with read access\n");
        return 1;
    }

    /*
    read dimensions of arrays
    */
    att_id = SDfindattr(sd_id, "Number_of_longitudes");
    status = SDreadattr(sd_id, att_id, &nx_file);
    att_id = SDfindattr(sd_id, "Number_of_latitudes");
    status = SDreadattr(sd_id, att_id, &ny_file);
    if ( (DIMLON != nx_file) || (DIMLAT != ny_file) ) {
        printf ( "ERROR RdO3colHDF: wrong array dimensions on file \n" );
        printf ( " in file : nx,ny = %5d%5d \n",nx_file,ny_file );
        printf ( " arrays : nx,ny = %5d%5d \n",DIMLON,DIMLAT );
        return 2; }
    else
        printf ( "RdO3colHDF: nx,ny = %5d%5d \n",nx_file,ny_file );

    /*
    read date
    */
    att_id = SDfindattr(sd_id, "Ozone_field_date");
    status = SDreadattr(sd_id, att_id, mdtg);
    printf ( "RdO3colHDF: mdtg = %4d%4d%4d%4d%4d%4d \n",
        mdtg[0],mdtg[1],mdtg[2],mdtg[3],mdtg[4],mdtg[5] );
}
```

```

/*
  read longitude and latitude range
*/
att_id = SDfindattr(sd_id, "Longitude_range");
status = SDreadattr(sd_id, att_id, lonlatrange );
*lonmin = lonlatrange[0];
*lonmax = lonlatrange[1];
printf ( "RdO3colHDF: lon range = %f %f \n",
  *lonmin,*lonmax );

att_id = SDfindattr(sd_id, "Latitude_range");
status = SDreadattr(sd_id, att_id, lonlatrange );
*latmin = lonlatrange[0];
*latmax = lonlatrange[1];
printf ( "RdO3colHDF: lat range = %f %f \n",
  *latmin,*latmax );

start[0] = 0;
start[1] = 0;
stride[0] = 1;
stride[1] = 1;
edges[0] = DIMLAT;
edges[1] = DIMLON;

/*
* Read ozone field
*/
printf ( "RdO3colHDF: reading ozone field \n" );
sds_index = SDnametoindex(sd_id,"O3_column");
if ( sds_index == -1 ) {
  printf ( "RdO3colHDF: ERROR ozone field array not found \n" );
  return 3;
}
sds_id = SDselect (sd_id, sds_index);
status = SDreaddata (sds_id, start, stride, edges, (VOIDP) O3storage);
if ( status == FAIL ) {
  printf ( "RdO3colHDF: ERROR reading ozone field array \n" );
  return 4;
}
status = SDendaccess (sds_id);

for ( i=0; i < DIMLON; i++ )
  for ( j=0; j < DIMLAT; j++ ){
    o3fld[j][i] = (float)O3storage[j][i];
  }

/*
* Read ozone forecast uncertainty field
*/
printf ( "RdO3colHDF: reading ozone forecast uncertainty \n" );

```

```

sds_index = SDnametoindex(sd_id,"O3_std");
if ( sds_index == -1 ) {
    printf ( "RdO3colHDF: ERROR ozone std field array not found \n" );
    return 5;
}
sds_id = SDselect (sd_id, sds_index);
status = SDreaddata (sds_id, start, stride, edges, (VOIDP) O3storage);
if ( status == FAIL ) {
    printf ( "RdO3colHDF: ERROR reading ozone std field array \n" );
    return 6;
}
status = SDendaccess (sds_id);

for ( i=0; i < DIMLON; i++ )
    for ( j=0; j < DIMLAT; j++ ){
        o3s[j][i] = (float)O3storage[j][i];
    }

/*
* Terminate access to the SD interface and close the file.
*/
status = SDend (sd_id);

return 0;
}

```

A simple plot of the ozone fields using IDL

The following IDL script reads an HDF data file and makes a simple postscript plot of the global ozone field.

```
; Read and plot "o3col" files in HDF format

; -----
; HDF file name

file = 'o3col.hdf'

; -----
; read general HDF file and return attributes/datasets

id = hdf_sd_start(file,/read)
hdf_sd_fileinfo,id,datasets,attributes

help,datasets
help,attributes

iret = 0

for i=0,attributes-1 do begin
  hdf_sd_attrinfo,id,i,name = name, data = d
  command = name+'=d'
  iret = execute(command)
  print, 'Retrieved Attribute:',name
endfor
for i=0,datasets-1 do begin
  sds = hdf_sd_select(id,i)
  hdf_sd_getinfo,sds,name=name
  print,'retrieving dataset',i,name
  iret = execute('hdf_sd_getdata,sds,'+name)
  if iret ne 1 then print,'error dataset',i
endfor
hdf_sd_end,id

; -----
; define grid

lats=-90.+findgen(181)/180.*180.
lons=-178.5+findgen(240)/240.*360.

; -----
; output to postscript file

set_plot,'PS'
filen='o3col.ps'
if !d.name eq 'PS' then device,file=filen,/color
```

```

; if !d.name eq 'PS' then device,file=filen,/color, $
;   xsize=33,ysize=20, xoffset=2

; -----
; define color table
; rainbow

loadct,25
; loadct,1 ; blue/white

nlevs=15
levs=findgen(nlevs)*20.+200.
cols=findgen(nlevs-1)*17+16
cols2=findgen(nlevs-1)*0

; -----
; create plot

!P.charsize=1.0
!x.margin=[7,7]
!y.margin=[8,4]
!p.thick=1.0
!x.thick=1.0
!y.thick=1.0

contour,o3_column,lons,lats, $
  xrange=[-180.,180.],xstyle=1,yrange=[-90.,90.],ystyle=1, $
  xticks=6,yticks=6, $
  nlevels=nlevs,levels=levs, $
  c_colors=cols,/fill
;contour,o3_column,lons,lats, $
;  nlevels=nlevs,levels=levs, $
;  c_colors=cols2,/overplot

map_set,xmargin=[7,7],ymargin=[8,4], $
  /continent,/cyl,limit=[-90,-180,90,180],/noborder,/noerase

o3col_kleurbalk,levs,cols,charsize=1.0,charthick=2.0,$
  lowleft=[0.1,0.1],xsize=0.8,$
  format='(i3)',unit=' [DU] '

if !d.name eq 'PS' then device,/close
uni='ghostview -a4 -center -magstep -2 '+filen+' &'
spawn,uni

end

```