5. Procedures and Functions

- Call a procedure or function (IDl or user-written):
  pro_name,p1,p2...., KEYWORD=pk1
  result=func_name(p1,p2,...KEYWORD=pk 1]

  p1,p2.... : positional parameters (optional), must appear in a particular order
  pk1.... : optional keyword variable,
  - appear in arbitrary order
  - shortcuts can be used

- IDL procedure (function) file template,

(filename with extension ".pro" ,e.g. pro_name.pro) :

```
PRO pro_name, p1,p2,.....,KEYWORD=pk1....
(FUNCTION func_name,p1,p2,... ,KEYWORD=pk1..... )
                ;p1,p2...     positional parameters
                ;pk1....      keyword parameter
.....
.....           ;IDL Code

(RETURN, result          ;functions only)
END
```

- Primitive example (file "quad.pro"):

```
FUNCTION quad,x
  r = x^2+x+400
  RETURN, r
END
```

  - ;call "quad" within an IDL PRINT- command

- Parameter passing:

  - variables are passed by reference, they can be modified in the calling routine: input and output parameters
  - constants, subscripted variables and structure tags are passed by value: input parameters

    - Change "quad.pro": overwrite the input argument

- **Compiling and Debugging of IDl routines**

  - A routine is compiled automatically before the first execution, when the file *"name.pro"* is in the IDL path or in the current working directory.

  - Compile a procedure or function:
    a) *IDlDEMenu: Run->Compile...*
    b) 'Executive' command on the command line
    .COMPILE name

- Setting the IDL Path:

    a) IDLDEMenu: file->Preferences
    b) IDL command:

!path = expand_path(' +/export/home/wg:')+ !path

• Debugging of procedures + functions:

a) see *IDLDE* Menu *Run* and toolbar buttons
Run->Step Info (execute 1 statement)
Run->Set Breakpoint (stop execution + ...)
.....

b) Use IDL executive commands, examples:

```
.step          ; execute 1 statement
.step 10       ; execute 10 statements
.skip 10       ; skip over 10 statements
```

.....

- Controll statements in IDL programs:
  - ***IF (a EO b) THEN ... ELSE***
  - ***FOR i=0,9 DO ...***
  - ***WHILE (NOT EOF (lun)) DO ...***
  - ***REPEAT ... UNTIL (b GT a)***
  - ***CASE test OF ...***
  - ***SWITCH test OF ... (new!)***

- Statement blocks with **BEGIN** and end with **END,ENDIF,ENDFOR... :**

IF (true) THEN BEGIN

. .

. .

ENDIF ELSE BEGIN

. .

. .

ENDELSE

- ***CASE*** *test* of ; 'test' : IDl variable

  0: ...

  1: ***BEGIN***

  .....

  END

  ELSE:

  ENDCASE

- ***GOTO, STOP*** ; 'stop' : IDL label

  ...

  STOP:

  ....

5.2 Using keywords and optional parameters

- Determine the number of parameters used in a call:

  number=N_PARAMS( )

- Determine if a keyword is defined:

a) ***defined=N_ELEMENTS(KEYWORD-VARIABLE)***

; returns the number of elements

b) ***defined=KEYWORD_SET(KEYWORD- $ VARIABLE)*** ; Used with toggle keywords:

; defined=1 (TRUE), =0 (FALSE)

- Simple example: function "multip"

```
FUNCTION multip, value, times, ADD=add
        if (N_ELEMENTS(add) GT 0) then begin
                value = value+add
        endif
        if N_PARAMS( ) eq 1 then begin
                RETURN, value*2
        endif
        RETURN, value*times
END
```

• Exercise:

Write an IDL-procedure, that draws a shaded surface with a wire mesh overplotted. IF desired, make a contour at the top of the plot and select a color index tor the surface and the contour plots:

PRO mysurface, data, CONT=cont, COLOR=color

- Keyword inheritance with the formal keyword parameter " *_EXTRA"*

(used e.g for wrapper function)

simple example:

```
pro myplot, data,_EXTRA=extra
        HELP, extra, /str          ; examine whats behind
                                   ; _EXTRA
        PLOT, data, _EXTRA=extra
end
```

- Call the procedure "myplot" with different keywords of IDL's *PLOT* command
- Exercise: Insert the *_EXTRA* -keyword into

  MYSURFACE. PRO

  - or into IDL's library routine

  SHOW3.PRO

  (copy show3.pro to your own directory !)

6. Displaying Image Data

- Read an image, i.e. 'galaxy.dat' or 'ctscan.dat'

```
OPENR,lun,DIALOG_PICKFILE( ),/GET_LUN
im=BYTARR(256,256)
READU, LUN, IM
TV, im              ; default position: lower left
                      corner
TV, im,/order       ; reverse the row order
```

- Position the image in the window

TV,im,300,200 ; image offset in pixels

Displayacolor bar on the lett of the image:

```
bar=BYTARR(20,256)
FOR i=0,19 DO bar[i,*]=BINDGEN(256)
TV, bar
```

- Process image with basic filters and display them in one window :

TV,im,0 ; position 0, upper left corner
TV,smooth(im,5),1 ; boxcar average 5 by 5, pos. 1
TVSCI,MEDIAN(im,3),2 ; median smoothing
(suppress noise tips, does not blur large edges...)
TVSCI, SOBEL(im),3 ;edge enhancement operator

- Define you own filter
  - arbitrary filter kernels with the use of IDI's CONVOL-function
  - Frequency Domain Filtering, example:

imf=FFT(im,-1) ; image to frequency domain
filter= DIST(256) LT 50 ; define a low pass
TVSCI,SHIFT(filter ,128,128) ; display filter
imf=filter*imf ; apply the filter
TVSCL, FFT(imf,1),4 ; back to the spatial domain

Working with images

- Using '<' '>' operators tor contrast enhancement:

```
TV,im>40,0      ; '>' operator returns a value
                ;equal to larger of its operants
TVSCI,im>40,1 ;TVSCI uses the full color range
TVSCL,im > 40 < 200,3 ; set a range of value
i=WHERE ( (im GE 200) or (im le 100) )
im[i]=0
TVSCL,im
```

. ●Exercise 9.1, processi ng the file 'cereb.dat', steps:

- Read two X-Ray images into one array:
data=BYTARR(512,512,2)
- Display the images side by side in one window
- Substract the images to display the differences: (Tip: cast 1. image to integer with 'FIX': DIFF=FIX(images2)-images1

- Plot the histogram of the difference:
PLOT,HISTOGRAM (DIFF)
- Contrast enhancement by:
DIFF=HIST_EQUAL(DIFF)
- Plot the histogram after histogram equalization and compare the images

7. Gridding Random or Irregular Data

- Dataset with 200 random points in the xv plane:

  x=RANDOMU(SEED,200)
  y=RANDOMU(SEED,200)
  data = (x+y)^2

- Plot the location of the points in the xy plane:

  PLOT,....., PSYM=...

- Triangulating the x-y points:
  TRIANGULATE,x,y,tr ;Delauny triangulation
  Help,tr ;Number of triangles

- Create a regular grid:

  grid=TRIGRID(x,y,data,tr)

  HELP,grid ; by default 51 intervals:

  ; (MAX-MIN)/50

  SURFACE, grid

- Regular grid with more intervals:

  spacing=[0.01,0.01] ;

  grid= TRIGRID(x,y,data,tr ,spacing)

  ;optional argument to trigrid !

  SHADE_SURF,grid

8. Volume Visualization

- Create a simple dataset:
  vol_data=BYTARR(50,50,50)
  vol_data[10:30,5:25,10:30]=10
  vol_data [25:45,30:45,25:40] =200

- SHADE_VOLUME generates a list of vertices ('v') and a list of polygons ('p'). They define a 3D surface at a constant density level:

SHADE_VOLUME,vol_data,5,v,p
; density level (isosurface)=5

- set up a user defined 3D-transformation:
  SCALE3.XRANGE=[0,50],YRANGE=[0,50], $
  ZRANGE=[0,50]

- create a shaded representation of the surface defined by the polygons 'po and the vertices 'v' :
  TV ,POL YSHADE(v,p,/t3d)

- set the density level to 100 :

: :

: :

- Voxel rendering in IDL's Object Graphics
  oVol = obj_new('idlgrvolume', vol_data)
  xobjview, oVol

- Volume Visualisation with IDL's SLICER3 -Tool:
  vol_ptr = PTR_NEW(vol_data) ;put data to apointer
  SLICER3, vol_ptr ;SLICER3 with a pointer arg.

- reading the volumetric data file '…/ data/head.dat' with 57 images with 80 x 100 pixels:

```
data=bytarr(80, 1 00,57)
OPENR,unit,DIALOG_PICKFILE( ),/GET_LUN
READU,unit,data          ;Read the data
FREE_LUN, unit
SLICER3, PTR_NEW(data,/no_copy)
```