January, 2007
Mozart_plot.x: code written by: Philippe Fulsack
IDL: scripts written by Sergei Medvedev
****

Post-processing SOPALE Output

Geodynamics Group
Department of Oceanography
Dalhousie University
Halifax, NS, B3H 4J1 Canada

http://geodynamics.oceanography.dal.ca

January, 2007

1 TABLE OF CONTENTS

2 INTRODUCTION

SOPALE produces various binary files as output (Fig. 1). These SOPALE output files need to be post-processed before the viewing, analyzing/ interpreting of model results. “P690” is the computer system where members of the Dalhousie GeoDynamics Group run SOPALE models. See http://geodynamics.oceanography.dal.ca/bonny/docs/sopale_overview.txt for the explanation of SOPALE output.

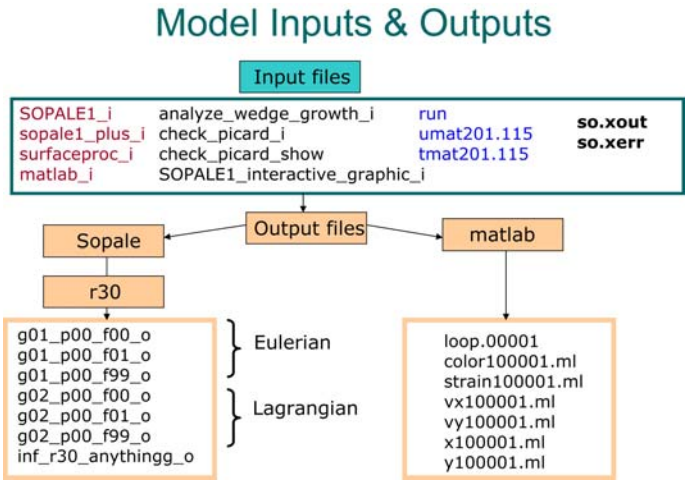


Figure 1: Directory structure of the Sopale input and output (binary format). The SOPALE outputs are stored in the subdirectory “Sopale/r30/”. The restart files are g01_p00_f00_o and g02_p00_f00_o. The file g01_p00_f01_o is the Eulerian output of frame 1 and g02_p00_f01_0 is the Lagrangian output of frame 1. The input file “matlab_i” specifies the parameters which will be saved in the MatLab outputs and the numbers of saved outputs. MatLab outputs are stored in the subdirectory “matlab”.

Users can analyze model results by using the numerical values or by the post processing procedures. Bonny Lee wrote the program “out2asc” to obtain the values of selected parameters from the SOPALE outputs. The model results can also be post-processed by using various softwares such as MatLab, Generic Mapping Tool (GMT), Interactive Data Language (IDL) (Fig. 2). This document will discuss a post-processing method, which the Dalhousie Geodynamics Group is currently using the most - the Interactive Data Language (IDL) scripts.

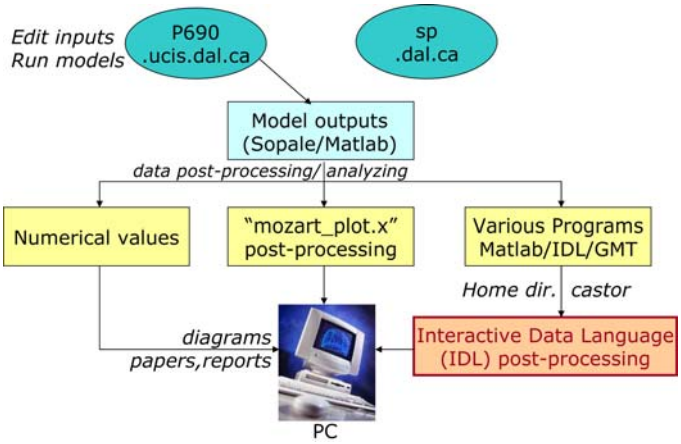


Figure 2: Various methods of analyzing SOPALE outputs: interpretation of the numerical values, SOPALE “Mozart_plot.x” post-processing, IDL post-processing. The IDL scripts are run on Users’ home directories while the “Mozart_plot.x” can be run on the p690.

The postprocessing program "mozart_plot.x" was written by Philippe Fullsack. As explained in the user's guide (http://geodynamics.oceanography.dal.ca/bonny/docs/sopale_postproc.txt), the program mozart_plot.x is controlled by a combination of text input files and interactive graphic interface. The text input files specify parameters such as window size, the output frames, contour lines, etc. You can select options such as field values and which grids to plot in the graphic interface. Mozart_plot.x displays the plots on screen, using X Window graphics, and can save outputs of the screen plot as the postscript format (Fig. 3).

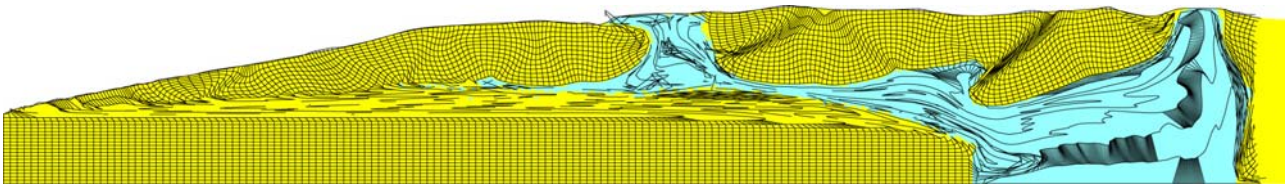


Figure 3: A postscript file created from "Mozart_plot.x" for a SOPALE salt model.

A set of the IDL scripts was developed by Sergei Medvedev during his postdoctoral work with the Geodynamics Group at the Department of Oceanography, Dalhousie University. These IDL scripts have been used to post-process Microfem and Sopale model results, which were presented in various journal papers and conference presentations. These IDL scripts were written for specific studies and for the code at the time Serge was working at the Dalhousie; therefore, users might need to modify the IDL scripts for the new codes with additional features.

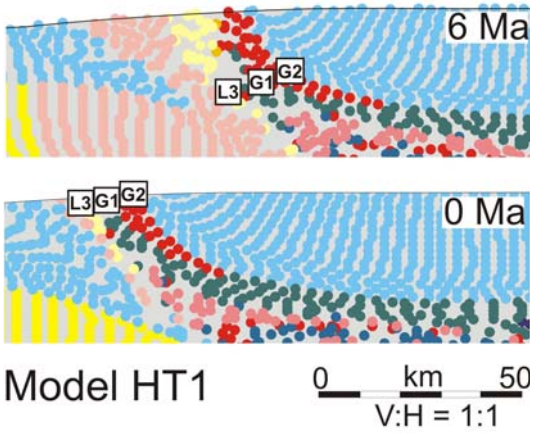


Figure 4: Plotting Lagrangian particles using IDL scripts. Source: Jamieson et al. GSL, in press.

The purpose of this User's Guide is to have an interface between IDL documentation and the specific IDL scripts which Sergei has written. Users can also find extra information from the online version of IDL User guide.

Here are the links to some IDL related sites.
http://www.dfanning.com/
http://www.informatik.fh-mannheim.de/idl/onlguide.pdf.
You can search for the IDL Reference Guide (IDL version 6.1, July 2004) from this site
http://www.informatik.fh-mannheim.de/idl/refguide.pdf
http://www.informatik.fh-mannheim.de/idl/quickref.pdf
Bonny has also put the IDL Reference Guide (IDL version 5.6, Research System Inc., October 2002) on the site http://geodynamics.oceanography.dal.ca/bonny/docs/idl_scripts/idlrefguide.pdf.

3 POST-PROCESSING USING "Mozart_plot.x" (code written by Philippe Fulsack, User Guide written by Bonny Lee)

http://geodynamics.oceanography.dal.ca/bonny/docs/sopale_postproc.txt

4 POST-PROCESSINGUSING INTERACTIVE LANGUAGE DATA (IDL) SCRIPTS

4.1 Hardware and software requirements

Hardware
IDL are currently installed on workstations Firedrake and Castor, Geodynamics Group at Dalhousie University.

Printers are specified in the IDL script ‘print.pro’.
Printers of the Dalhousie Geodynamics Group are as follows:
Color Printer: ‘Beaumontcolour’
Black and white printer: ‘`geophys-printer.ocean.dal.ca`’

Remote Users should save the postscript files after the IDL post-processing session and send those postscript files directly to their printers.
Hardware requirements:
RAM: should fit User's data.
Color: 256 colors or more. The number of colors used by IDL_M-S is limited to less than 256.

Software
1. The IDL_M-S code has been tested on IDL versions 5.02, 5.4 and 5.5. More information related to the most recent version of IDL can be found at the website http://www.ittvis.com/idl/

2. Animation can be compiled by using *gifcicle*.
3. GIF creator.
The easiest way to create gif files is to have license of IDL version 5.5 (or later) and the fee paid to UNISYS for the GIF compression (LZW) patent.

4.2 IDL_M-S PROGRAM STRUCTURE

4.2.1 Directory Structures

You will need to create two types of directories in the same machine for post-processing using IDL_M-S scripts – 'Data Directory' and 'IDL Scripts Directories'.

a) **Data Directory**: for storing the model output files. SOPALE model results from p690 will need to be transferred to User's home directory (castor, firedrake, etc.) before the IDL post-processing. Users decide the name and the path of this directory to suit their directory trees. The path of the "Data Directory" should be specified in the IDL script "session.pro". If the path of 'Data Directory' is not specified in the IDL script 'session.pro', users can still read the model output after typing the directory path name during the post-process model results (see Section 4.4).

b) **"IDL Scripts" Directory**: for storing Sergei's IDL_M-S scripts and User's modified scripts. We call them "IDL Software" Directory and "IDL Post-processing" Directory in this document. Users can give these directories any names as long as the directory names and paths are specified clearly in the file "session.pro", which will be stored in the "IDL Postprocessing" Directory.

"IDL Software" Directory: for storing the original IDL_M-S scripts. An overview of IDL_M-S scripts is presented in Section 4.2.2. A list of IDL_M-S scripts (source code) and their classifications are summarized in Table 1. Currently (January 2007), the source code can be copied from the castor directory /usr/local/idlscripts/.

"IDL Post-processing" Directory: From this directory, you will run the IDL scripts to read the SOPALE outputs, which are stored in the "Data Directory", and run some IDL scripts to plot diagrams.

- Users will need to modify some parameters for their plots such as plot dimension, color scheme, etc.
- User will need to copy the related programs from 'IDL_M-S' Software Directory to the Post-processing Directory and make changes here. This will allow Users to save their own settings and the changes will not affect the main software directory. Two files should be copied into User's Post-processing Directory are 'session.pro' and 'default.pro'.
- Users can modify IDL scripts for specific applications and rename these scripts with appropriate names. For example, you can modify the file 'post.pro' to plot the Thrust-and-Fold beft models and rename the script as 'post-TFB.pro'.
- The postscript files created from IDL_M-S programs will be stored in this "IDL Post-processing" directory. Users have options to name, save and/or print their postscript files (see Figure 6).

Sergei's descriptions of some programs are included in Section 6.

We suggest two examples of the directory structures, which you can choose whichever work best for you. Figures 5a and 5b illustrate examples of the directory structures. You can decide the directory names and the path of these directories. The IDL_M-S source code is currently located on castor, directory /usr/local/idlscripts/.

Structure A has two "IDL Scripts" directories – one directory for storing source code or main programs (Software Directory) and one directory for storing the IDL scripts which are modified by

users and for storing postscript files. You will run the IDL script from the "IDL post-processing" directory if you choose to use this structure. The IDL post-processing procedures discussed here will be based on the Directory Structure A.
Advantages: It is very clear which files you have modified and the source code is untouched.

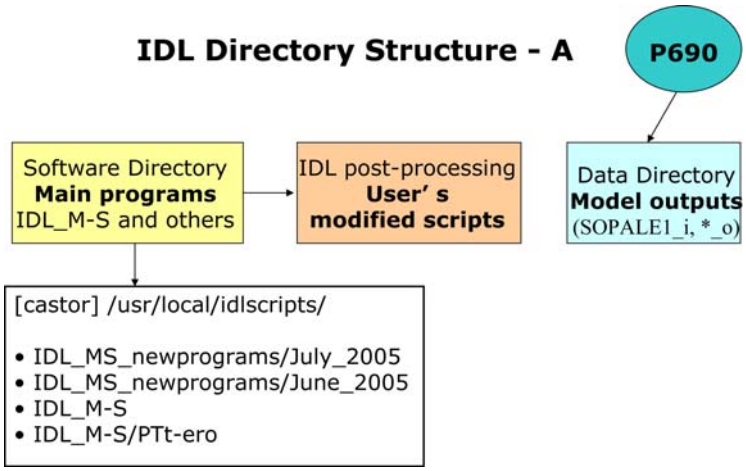


Figure 5a. IDL scripts are stored in two directories. The Software Directory stores the source scripts and the "IDL post-processing directory stores User's modified script.

Structure B has one "IDL Scripts" directory only. All IDL_M-S source code and User's modified scripts are stored in the same directory. You can run the post-processing from this directory. If you find it is hard to search through a long list of IDL scripts to find your modified scripts, it is good to make subdirectories to store your modified IDL scripts for specific purposes. For example: Scripts for Thrust-and-Fold Belt models, Salt models, IDL scripts for animation, IDL scripts for plotting injected Lagrangian particles, etc.

Advantages: You have a set of IDL scripts that will work for a specific task and you do not have to worry about which file is the most current one.
Disadvantages: There are many IDL scripts in the same directory. After the postscript files are created, they will also be in this directory. Since the all the source code and modified IDL scripts are stored in the same directory, it is important to keep track of your modifications to any scripts.

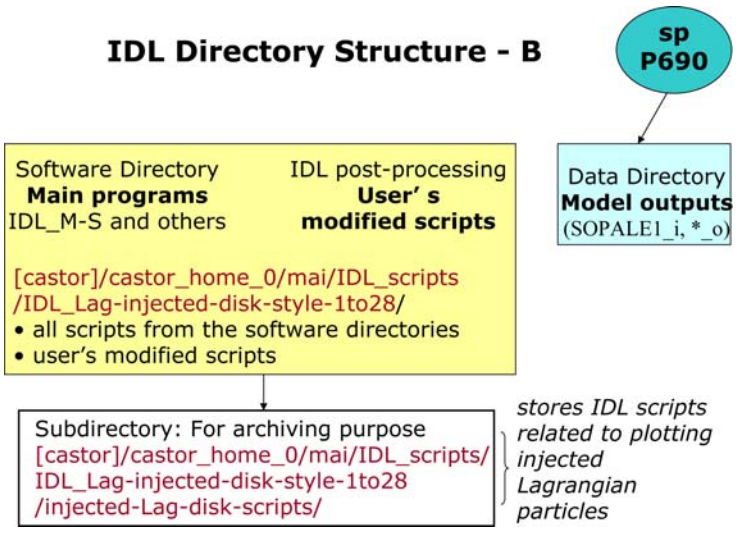
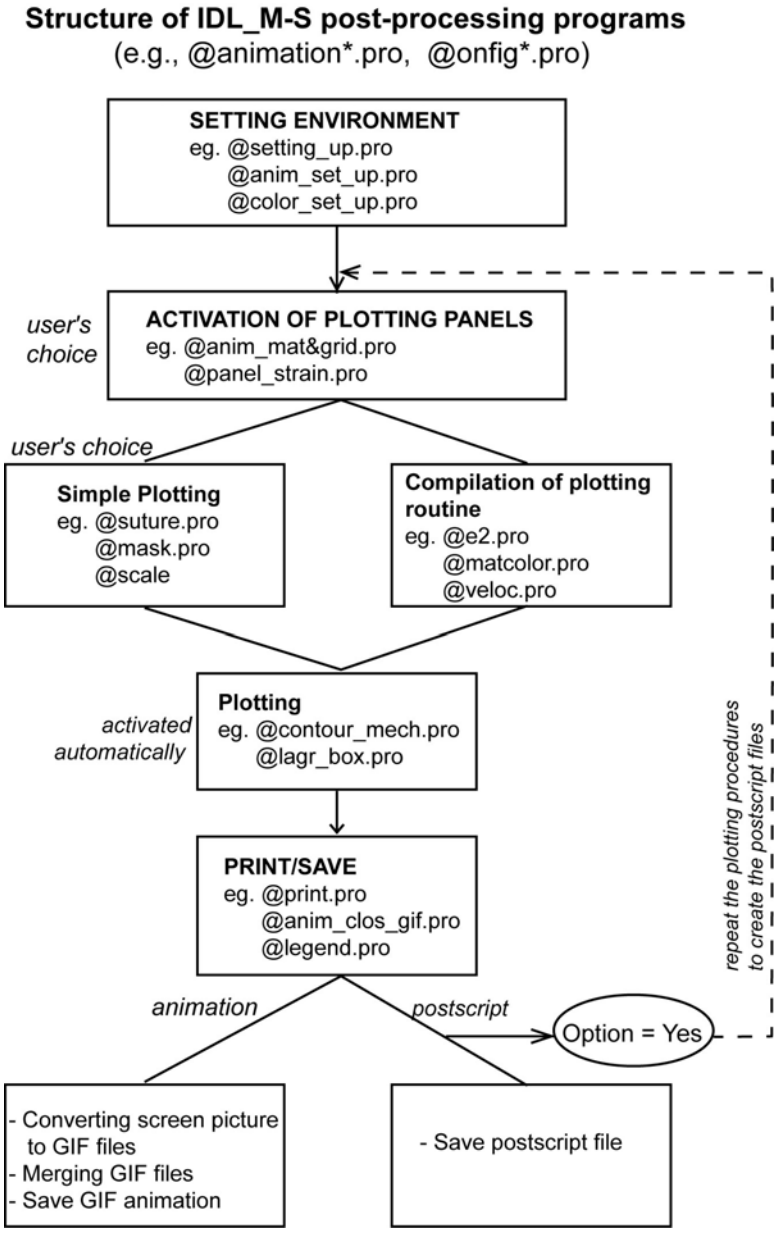


Figure 5b. The IDL_M-S source code is stored in the same directory with the User's modified IDL scripts. The subdirectory "injected-Lag-disk-scripts" stores all IDL scripts related to this type of plotting.

4.2.2 IDL Scripts Overview:

A set of IDL scripts were written for post-processing model results of SOPALE and Microfem models and we call them as IDL_M-S software. If you already have a template of the IDL scripts for plotting certain routine plotting, the post-processing procedures can be fairly simple.



The IDL procedures can be summarized as follows:

IDL> .run read.pro

- Read the data from SOPALE output files or Matlab output files.
- Put the data into arrays in memory. The script "download.pro specifies the names of arrays for each field. The unit conversion is also done in this step.

IDL> .run post*.pro

- Run the "plotting" program, for example, the script onfig*.pro or animation*.pro. Figure 5 illustrates the structure of post-processing routines and the actions when you run the "plotting" scripts. The steps including setting up the environment (color scheme, size of plots, styles of axis, font size, etc.), compilation of plotting routine and plotting the diagrams on screen using X Window display.
- Users will be asked to select options for saving and/or printing the postscript file. The postscript files can contain one single frame or multiple frames.

Figure 6: Flow chart showing the structure of the IDL plotting scripts.

All programs in the IDL_M-S Software Directory are summarized in Table 1. The IDL scripts are grouped into five different types based on the structure shown in Figure 5. This grouping is done in an attempt to help users, who want to reorganize their directory and to modify/create new IDL

scripts. The field data can be plotted in the same diagram with model output (Type 5). Sergei included two example files in Type 5 but these particular files were not used post processing the SOPALE output.

Table 1: Types of programs in IDL_M-S Software Directory (classified by Sergei Medvedev, 2002).

Type 1: Compilation program/routines
1) anim_bc.pro (include,s,80%, 1)
2) anim_clos_gif.pro (include,ms,30%, 1)
3) anim_clos_mpg.pro (include,ms,?%, 1)
5) anim_panel_grid&mat.pro (include,ms,40%, 1)
6) anim_panel_mat.pro (include,ms,40%, 1)
9) animation.pro (programme,ms,100%, 1)
22) e2.pro (include,ms,70%. 1)
33) mat_box (include,ms,50%, 1)
34) matcol.pro (include,ms,50%, 1 and 4)
35) panel_templ.pro (include,ms,100%, 1)
40) read.pro (include,ms,10%, 1)
60) session.pro (include,ms,80%, 1 and 2)
64) Tcontour.pro (include,ms,60%, 1)
65) total_strain_e.pro (include,ms,60%, 1)
66) total_strain_l.pro (include,ms,60%, 1)
67) veloc.pro (include,ms,60%, 1)
68) veloc_bc.pro (include,m,5%, 1)
69) veloc_g.pro (include,m,5%, 1)
70) veloc_mantle.pro (include,m,5%, 1)

Type 2: Data Calculation/Manipulation
4) anim_panel_e2.pro (include,ms,20%, 2)
11) bc_analysis.pro (include,ms,20%, 2)
13) chris_read.pro (include(batch),ms,0%, 2)
19) conv.pro (programme,m,10%, 2)
21) downloads.pro (include,ms,90%, 2)
24) exponent.pro (subroutine,ms,10%, 2)
26) ground_0.pro (programme,ms,10%, 2)
27) inversion.pro (programme,ms,10%, 2)
28) lagr_box.pro (include,ms,10%, 2 and 4)
29) lagr_limits.pro (include,ms,10%, 2)
41) read_description.pro (include,ms,2%, 2)
42) read_dialog.pro (include,ms,2%, 2)
43) read_init_restart.pro (include,s,10%, 2)
44) read_m_data.pro (include,m,10%, 2)
45) read_m_erosion.pro (include,m, 10%, 2)
46) read_m_file.pro (include,m,2%, 2)
47) read_m_finish.pro (include,m,10%, 2)
48) read_m_info.pro (include,m,10%, 2)
49) read_m_mech_box.pro (include,m,10%, 2)
50) read_m_therm_box.pro (include,m,10%, 2)
51) read_m_therm_propr.pro (include,m,10%, 2)
52) read_p_data.pro (include,s,20%, 2)
53) read_s_depth.pro (include,s,10%, 2)
54) read_s_eul.pro (include,s,10%, 2)
55) read_s_finish.pro (include,s,10%, 2)
56) read_s_info.pro (include,s,25%, 2)
57) read_s_lagr.pro (include,s,10%, 2)
58) read_s_mech_box.pro (include,s,10%, 2)
60) session.pro (include,ms,80%, 1 and 2)
62) smoothing.pro (function,ms,10%, 2)

Type 3: Setting Environment
7) anim_setup.pro (include,ms,20%, 3)
17) color_set_up.pro (include(batch),ms,40%, 3)
20) default.pro (include(batch),ms,100%, 3)
36) print.pro (include,ms,10%, 3)
37) print_auto.pro (include,ms,10%, 3)
61) setting_up.pro (include,ms,30%, 3)
73) zz_1x1.pro (include,ms,2%, 3)

Type 4: Plotting Routines
8) anim_time.pro (include,ms,20%, 4)
10) arr.pro (subroutine,ms,2%, 4)
15) circle.pro (subroutine,ms,2%, ?)
16) color_present.pro (programme,*,0%, 4)
18) contour_mech.pro (include,ms,20%, 4)
23) erosion_front.pro (include,m,10%, 4)
25) finish_plot.pro (include,ms,10%, 4)
28) lagr_box.pro (include,ms,10%, 2 and 4)
30) legend.pro (include,ms,50%, 4)
31) lgrid_l.pro (include,ms,30%, 4)
32) mask.pro (include,ms,10%, 4)
34) matcol.pro (include,ms,50%, 1 and 4)
39) radcolor.pro (include,ms,30%, 4)
59) scale.pro (include,ms,80%, 4)
63) suture.pro (include,m,20%, 4)
71) velocity_scale.pro (include,ms,25%, 4)
72) zebra.pro (include,ms,40%, 4)

Type 5: Field data or other data
12) chris_pictures.dat (data,ms,0%, 5)
14) cig.pro (programme,*,0%, 5)

Since IDL_M-S was original written for Microfem models, some of the files only work for the Microfem model output and was noted as '**m**'. We are currently working on bringing some of the Microfem features into SOPALE code and will update the IDL scripts for post-processing SOPALE output. Sergei Medvedev described the IDL scripts as the following notation (Table 1):
'**m**': routines for Microfem models
'**s**': routines for Sopale models
'**program**': programs (end with "end" operator); use .run to run in IDL environment
'**include**': included files in programs or included in other 'include files' or in 'Batch file'.
'**subroutine**': Functions can be called by Main Level and Second Level programs
'**batch**': General purpose program has to be run as shell script (@default.pro, etc.)
'**%**': Sergei estimated the percentage of chance that the files will be modified by users.
Files categorized as more than 50% will likely need to be modified by Users to suit the specific purpose.
'data': field data for comparing to model results or other data.

Routines that create data for plotting legend are as follows (usually that are routines that create colored fields/contours):
e2.pro; matcol.pro; radcolor.pro; Tcontour.pro; total_strain_e.pro; total_strain_l.pro; veloc.pro

4.2.3 Setting up IDL

Before you start IDL post-processing:

1. Set-up IDL software communications with other software and hardware
   - printers (see comments inside routine print.pro)
   - animation (see comments in anim_clos_gif.pro)
2. Locate the Software Directory, where 'IDL_M-S software' package is stored.
As of January 2007, the main 'IDL_M-S software' directories are on castor. They are listed below in the order of new to old directories.

[castor] /usr/local/idlscripts/ IDL_MS_newprograms/July_2005/
[castor] /usr/local/idlscripts/ IDL_MS_newprograms/June_2005/
[castor] /usr/local/idlscripts/ IDL_M-S/
[castor] /usr/local/idlscripts/ IDL_M-S/PTt-ero/

IDL will use the most current version of IDL_M-S scripts if you list the directories in the following order in the file "session.pro".
software_dir='/usr/local/idlscripts/ IDL_MS_newprograms/July_2005/: / IDL_MS_newprograms/June_2005/:/usr/local/idlscripts/ IDL_M-S/:/usr/local/idlscripts/ IDL_M-S/PTt-ero/
If there are files with the same filename in more than one directory, the file in the directory listed first in "software_dir" will overwrite the other files.

3. Set up User's Data Directory to store the SOPALE outputs (see Figure 5a).

4. Set up the "IDL post-processing" Directory to store the IDL scripts which you modify for a specific project (see Figure 5a).

4.3 IDL POST-PROCESSING PROCEDURES

We will use the Directory Structure A (Figure 5a) in this section. The common procedures, when you want to post-processing a new set of SOPALE model results using the IDL_M-S scripts, are as follows:

4.3.1 Transferring model results to "Data Directory"

1. Change directory to the "Data Directory" and transfer model results from P690 to User's "Data Directory" (see Fig. 5a).
2. Once in a while, you might want to post-process the restart files (*f00_o) or frames *p00_f99_o of a SOPALE model in order to analyze the model results with problems. You will need to do the following tasks:
- rename frames g01_p00_f99_o and g02_p00_f99_o to frame numbers other than *f00 or *f99 and the existing frames numbers.
- modify the time step output in the file SOPALE1_i in order to have the correct time step printed on the plots.

4.3.2 Copy files from IDL_M-S Software Directory to "IDL Post-processing Directory" and Modify IDL scripts:

1. Prepare files for User's "IDL Post-processing" Directory:
- copy "session.pro" and "default.pro" from IDL_M-S Software Directory to "IDL Post-processing Directory". Users should always have these two scripts in the "IDL Post-processing Directory".
- consider which files you will need to modify based on your model types and applications.
- copy those IDL scripts, which required editing, from IDL_M-S Software Directory to User's Working Directory.

2. Edit IDL scripts in User's Working Directory to get the desired settings for your applications (see Section 4.4).

Examples of User's Modified scripts: These following files are in the IDL working directory with some modifications for post-processing SOPALE output of Thin-skinned Thrust-and-Fold Belts models.

default.pro
downloads.pro
e2_s.pro (modified from "e2.pro" to plot strain rate)
post_ts.pro (for post-processing Sopale Thin-skinned Thrust Belts models)
print.pro
session.pro (specified the location of the Software Directory and Data directory)
setting_up.pro
veloc_s.pro (modified from veloc.pro)

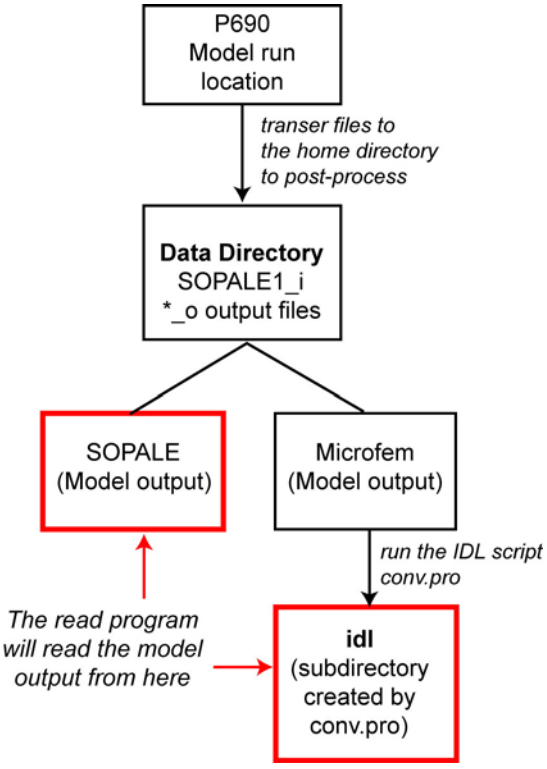
Notes:
The rest of IDL_M-S scripts were not modified and were stored in the "Software Directory".

The files in the User's Working Directory will overwrite the one with the same filename in the 'IDL_M-S' Software Directory. You can modify “session.pro” to set a different option.

In the above example, files “default.pro”, “downloads.pro”, “print.pro”, “session.pro”, “setting_up.pro” will overwrite the files with the same names in the “Software Directory”. Files “e2_s.pro” and “post_ts.pro” were modified from e2.pro and post.pro and were saved as new filenames.

4.3.3 Conversion of Microfem output

Only Microfem model output need to be converted before the post-processing (Fig. 6); SOPALE models: skip this step.



* change directory to User's Working Directory;
* type: idl session (IDL prompt will be shown as: IDL>)
* **For Microfem models**:
  IDL> .run conv.pro

The program convert standard outputs from Microfem model results into data that can be read by the 'IDL_M-S' Software.
Each set of model result will be converted once, and the converted results are stored in the subdirectory "idl" in the 'User Working Directory' (see Figure 6).
The program read.pro will read the data from the 'idl' subdirectory for further post-processing.

Figure 6: Flow Chart showing the subdirectory “idl”, which was created after running the IDL script “conv.pro. The “read.pro” script will read the data from SOPALE output files, or converted Microfem output files.

- The subdirectory 'idl' contains the following files:

e2
eul
grid
info
lagr
lagrmax
material
mecset_L_1_i
microfem_e_i
press
temp
therm_mat
thermprop_i
therset_E_2_radio_i
therset_L_1_i
topo
tvel
velo_dist
veloc
visc

4.3.4 Read Data (once per post-processing session of a set of model output)

Each time you want to post-process a new model, the model results should be read by running the 'read.pro' IDL script. After reading the data, you can then run the IDL plotting scripts or animation programs (Fig. 7). For example, if you want to post-process two models A and B, you will type the following commands:

IDL> .rnew read	(read data from Model A)
IDL> .run post*.pro	(run the post.pro script to plot a diagram for Model A)
IDL> .rnew read	(read data from Model B)
IDL> .run post*.pro	(run the post.pro script to plot a diagram for Model B)
IDL> .rnew read	(read data from Model A because you now want to plot another diagram for Model A)
IDL> .run post*.pro	(run the post.pro script to plot a diagram for Model A)

- The program 'read.pro' should be activated by using command '.rnew' to clear the memory from the previous read session.
- The 'read.pro' can recognize three types of data: Microfem output after they are converted by the conversion process; traditional output from SOPALE; and "Matlab" type of output from SOPALE. The script automatically recognizes the type of data and reads it accordingly.
- The "read.pro" script reads the data from User's Data Directory (or Microfem subdirectory 'idl') and save the data in memory. The array for each field will be named (eg. The 'tex' array holds the field Eulerian co-ordinate, the 'mu' array holds the values for viscosity, etc.). Users can specify the fields to be read by 'read.pro' in the file 'downloads.pro. In Sergei's IDL scripts, the read.pro includes all model output frames in arrays. The units are converted during the reading data such as the conversion of distance unit from meters to kilometers or conversion of temperature in Kelvin to Celcius. The arrays of data are used for further plotting/calculations.

Notes:

There is not much for Users to change inside this compilation program (and in all its subroutines **read*.pro*).

Two files can control read.pro:

session.pro - set directories to check for data

downloads.pro – define the array to hold the data of selected fields

Bonny modified the read*.pro scripts and IDL scripts can read one frame at a time. The script 'downloads.pro' was edited so it allocates the arrays to hold the data for all fields instead of selected fields. Members of the GeoDynamics Group use this set of IDL scripts to post-process Salt, Shale and some Deep model series. The commands for reading model output files and plotting are as follows:

IDL> .run newread.pro (instead of IDL> .rnew read.pro)

IDL> .run post*.pro

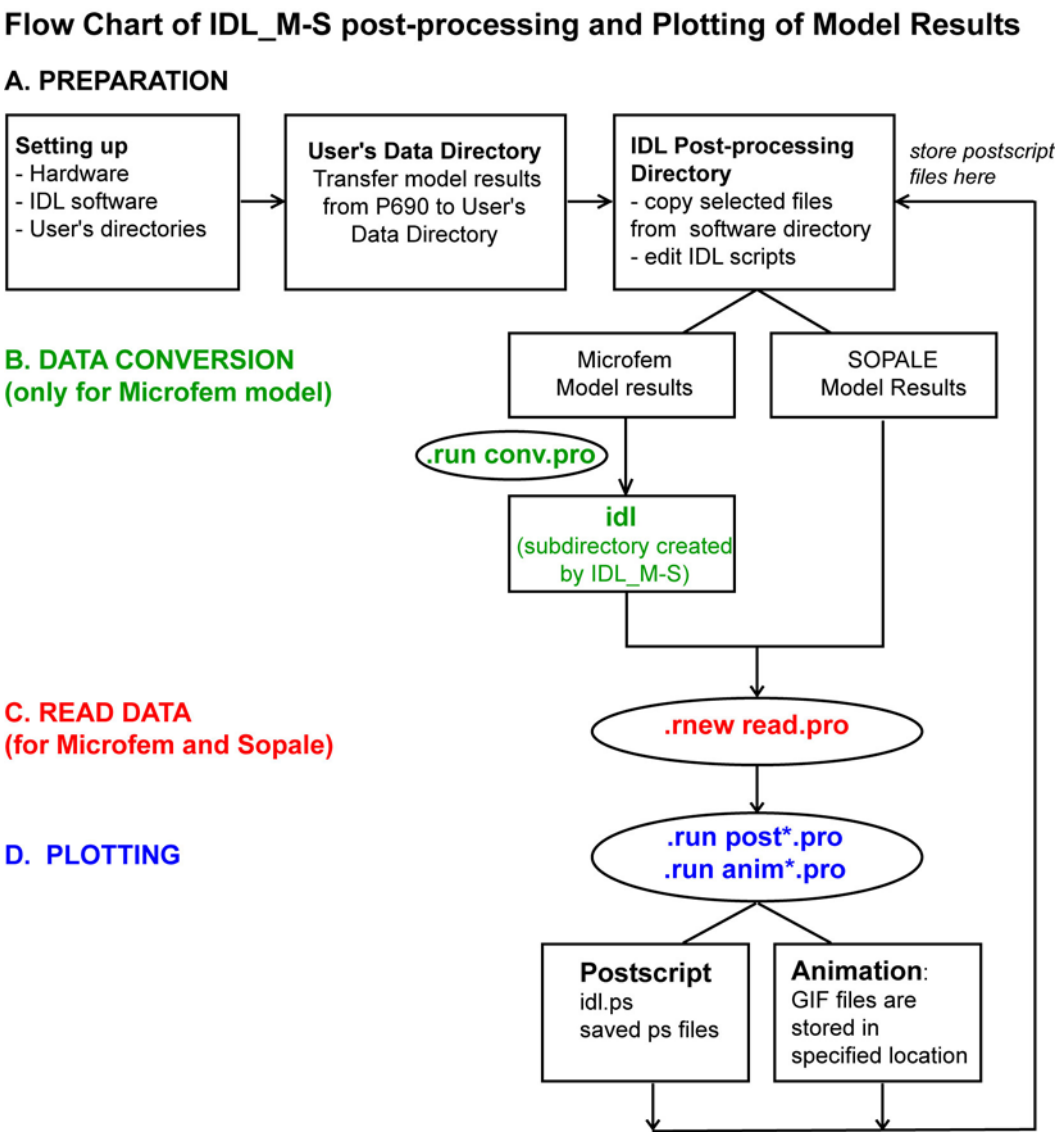


Figure 6: Flow chart of the IDL_M-S Post-processing routines.

4.3.5 Plotting Model Results

a) After data have been read, the field values are stored in arrays. User can run the plotting scripts by enter at IDL prompt ".run program name".

IDL> .rnew read.pro
IDL> .run post.pro (run the IDL script post.pro)

It is not necessary to type the extension '.pro'; therefore, the following command has the same effect as the above command line.

IDL> .run post

b) After the plotting is completed, Users have the option to save and/or print their postscript files and name the postscript files. These options are specified in the IDL script 'print.pro' and can be modified to match with the printer and paper setting.
Options are as follows:

y: print on 8.5x11 paper size, black and white printer
l: print on 11x17 paper size, black and white printer
c: print on 8.5x11 paper size, color printer
s: save as postscript file, the program will then ask for file name (User's choice)

ls: print on 11x17 paper and save as postscript file

The postscript files are named from idl0.ps to idl10.ps to avoid interference that may cause by slow print piping.
If you select the option "s" (save) as main or second symbol in your response, you will be able to name the postscript file. If you decide not to give the postscript file a name, the default file name will be given to the saved postscript file such as 'idl0.ps'. All postscript files will be stored in the 'IDL post-processing' directory, which is the directory where you type commands or run IDL scripts.

Example of the prompt on the screen after the postscript file is created:
Do you want to print? press "enter" for [no] or:
y-print, s-save PS, e-save EPS, c-color print, t-transparency, l-large print?
: ls
Give name to file without the extension .ps (idl by default)
: RBRTP-23_lag_mat
File to print: RBRTP-23_lag_mat.ps

4.4 Common plotting tasks
Currently, the IDL_M-S software has the ability to produce a variety of diagram and animation files. When you are familiar with the IDL scripts and what they can plot, you should be able to plot a combination of various parameters. The 'lines' should be plotted after the 'fill coloring' so the colors will not cover the contours/lines. We present here only two examples of the common plotting routines which we often plot for all model results. These two options are include in the IDL script 'post-TFB-figure-Sept2006.pro' for post-processing the Thin-Skinned Thrust-and-Fold Belt model results. This IDL script is included in Section 4.4.4.
- Option 'l': plotting Mechanical Material Coloring and Lagrangian Grids;
- Option 's': plotting Strain rate and Velocity vectors..

4.4.1 Common modifications:
IDL will ignore all text after the semicolon (;) and you can comment out an option by adding the semicolon in front of the line. The most common changes when you plot diagrams are as follows:

Select output frames for plotting
frames=indgen(nt1)
;frames=[1,3,6,8]
Specify the model window for plotting
horizontal_intervals=[0.,200.]
Specify the color/greyscale scheme to color the mechanical materials, thermal materials, strain rate, viscosity, stress, etc
material_colors=[1,2,magenta,yellow,blue,red]
;material_colors=[1,2,gray(70),gray(60), gray(50)]

str_rates_col=[gray(0), gray(50),gray(60),gray(80)]

Select the paper size/orientation and numbers of panel to plot on a page

```
xpr = 41. & ypr = 26  ;*** size of paper for print (in cm)
```

4.4.2 Strain rate and Velocity Plot

This option plots the strain rate as fill color and velocity vectors on top of the fill.

style='fill'
col=str_rates_col
@e2_s
@mask

style='contour_line'
@veloc_s
@finish_plot

@legend

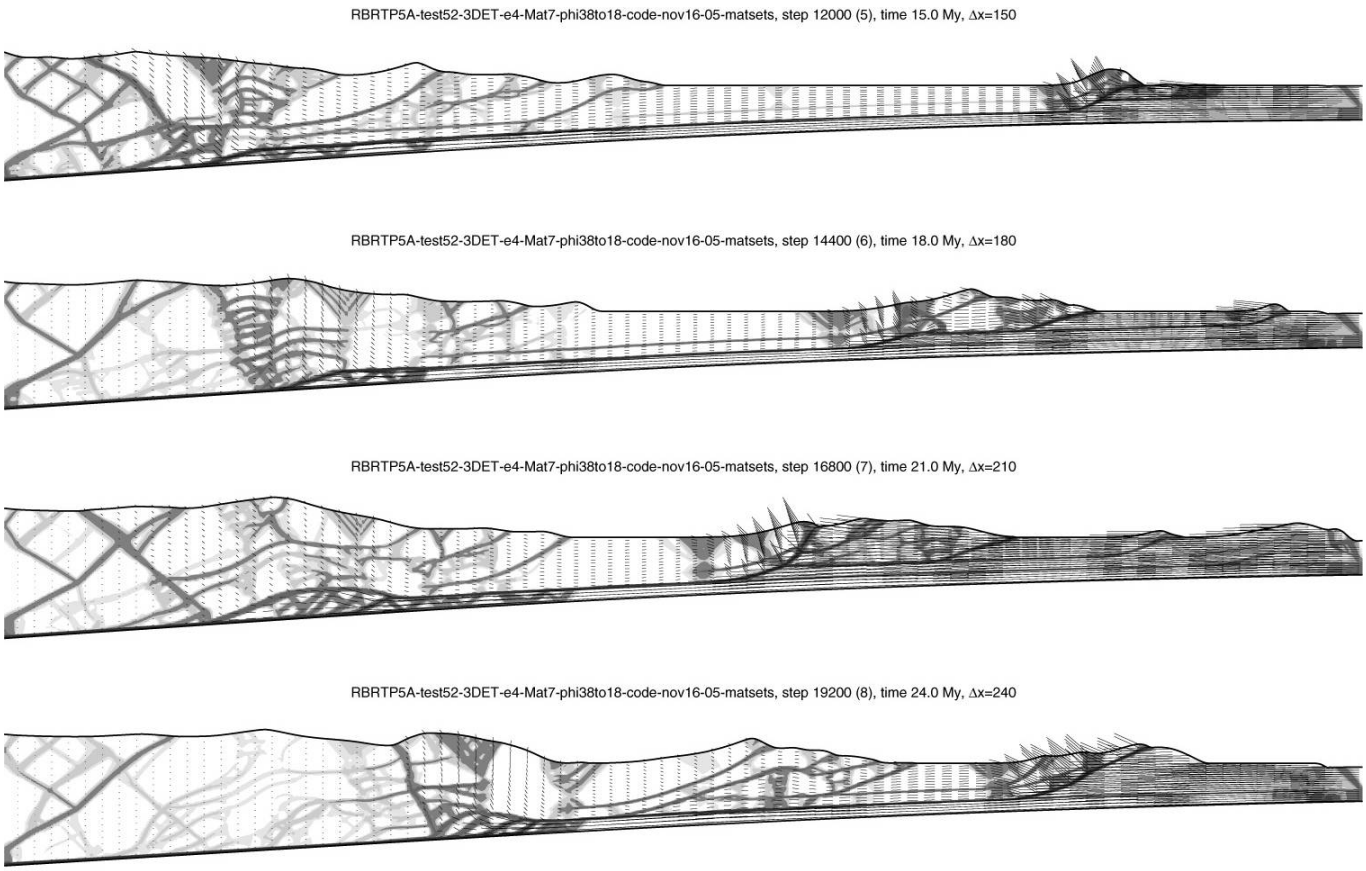


Figure 7 Strain rate and Velocity plot of a Thinned-skinned Thrust-and-Fold belt model.

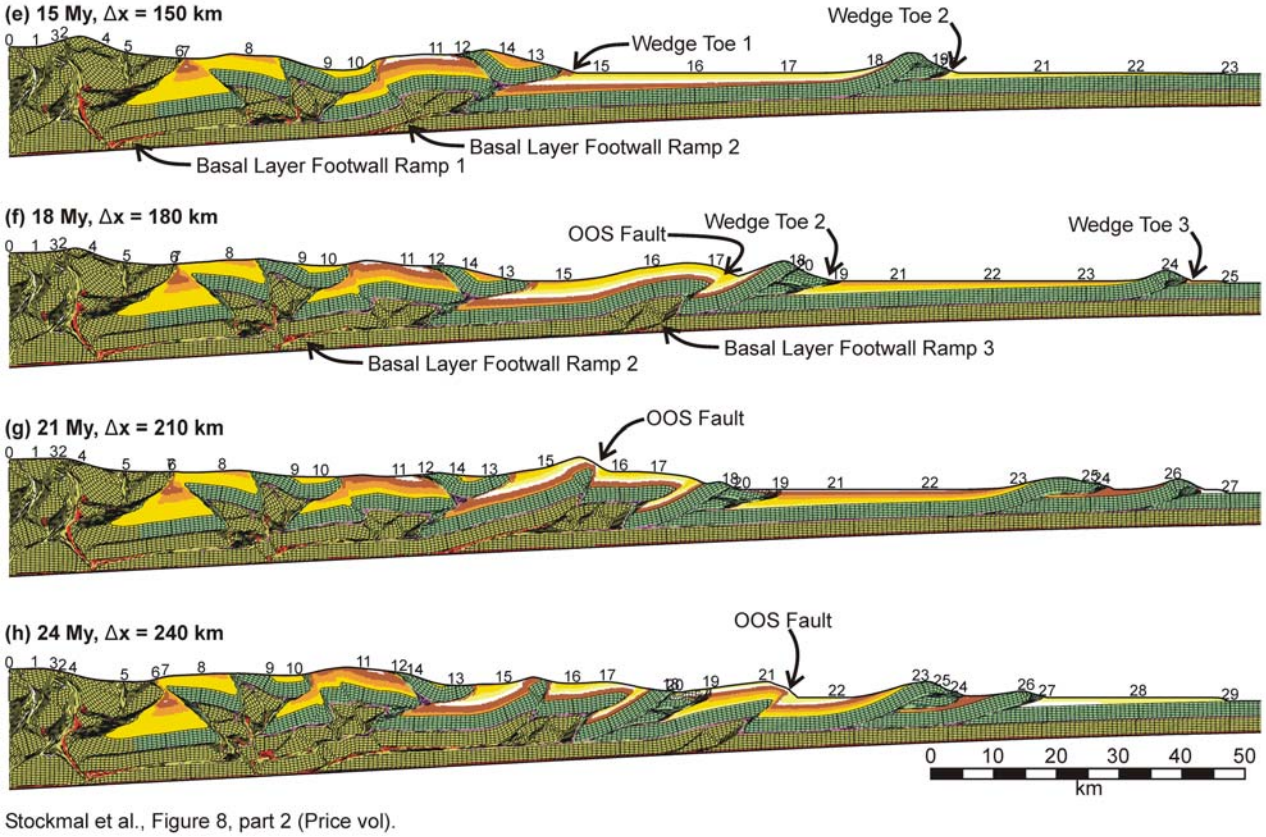
4.4.3 Mechanical Material Coloring and Lagrangian Grid plot

This option plots the mechanical material coloring (Eulerian or Lagrangian coloring) and the Lagrangian grids plotted on top of the color fill.

style='fill'
style_col=1
;style_col=20
@matcol
@mask

style='contour_line'
lgrid_stepx=3
lgrid_stepy=3
thick_line_freq=20
@lgrid_l

@finish_plot
@legend
@scale



Stockmal et al., Figure 8, part 2 (Price vol).

Figure 8 Eulerian mechanical material colors and Lagrangian grids of a Thin-Skinned Thrust-and-Fold Belt model. There are no grids on top of the sediment layers.

4.4.4 IDL scripts for the standard plotting 'post-TFB-figure-Sept2006.pro'

All text following the semicolon (;) are comments only because IDL will ignore these text.
The descriptions of most variables are summarized in Table 2. The variables which users might like to modify for plotting diagrams are as follows:
frames, horizontal_intervals, yrang, material_colors, str_rates_col, plot, plot_n, xaxis_style, yaxis_style, landscape, xpr, ypr, one2one_adjust, legend_position_y, legend_new_panel, legend_resize, style, style_col. See Section 4.4.1 for explanations of some common changes.

There are also some IDL variables which users do not need to be modified such as multi, ikon, iqq (see descriptions in Table 2).

The 'batch' files are general purpose scripts, which have to run as shell scripts (see Section 4.2.2). In the IDL script, these files start with'@' sign. In the IDL script 'post-TFB-figure-Sept2006.pro', the 'batch' files are listed as follows:

@setting_up.pro	(for setting up)
@zz_1x1.pro	(plot V:H = 1:1 scale)
@e2_s.pro	(plot strain rate)
@mask.pro	(mask the area outside model plot)
@veloc.pro	(plot the velocity vectors, the style, size can be edited in veloc.pro)
@matcol.pro	(plot the mechanical material coloring)
@lgrid_l.pro	(plot the Lagrangian grids)
@finish_plot.pro	(plot the think lines for the top and bottom of models, plot S point for Microfem models)
@legend.pro	(plot legends of Strain rate fill or Mechanical material fill colors)
@scale.pro	(plot the scale bar, style of scale bar can be edited in scale.pro)

```
;*****************************************************************
; FILE NAME: post-TFB-figure-Sept2006.pro
; MODIFIED BY: MAI NGUYEN
; DATE: Sept. 19, 2006
; USER WILL BE ASKED TO ENTER THE OPTION OF PLOTTING AT THE IDL PROMPT:
; OPTION 'l': PLOT LAGRANGIAN GRID AND MATERIAL COLORING
; OPTION 's': PLOT STRAIN RATE AND VELOCITY

; Procedures:
; STEP 1: SELECT THE FRAMES FOR PLOTTING (frames)
; STEP 2a & 2b: SELECT THE WINDOW TO PLOT (horizontal_interval and yrang)
; STEP 3: SPECIFY THE COLOR OR GRAYSCALE FOR MATERIAL COLORING (material_colors)
; STEP 4: SPECIFY COLOR OR GRAYSCALE FOR STRAIN RATE COLORING (str_rates_col)
; STEP 5a: SELECT PAPER SIZE (plot)
;      5b: SPECIFY NUMBER OF PANELS PER PAGE (plot_n)
;      5c: SPECIFY THE SIZE OF PLOT ON PAPER (USE OPTION FROM STEP 5a)
; STEP 6: PLOT OPTION "s" - Strain rates and velocities

; NOTES: - all text after ';' are comments, will not be read by the program
;       - comment the unused option by putting ';' at the beginning of the text.

;*****************************************************************
```

```
;************************************
; THE FOLLOWING DIALOG WILL APPEAR ON USER'S SCREEN AFTER TYPING ".run post-TFB-
figure-Sept2006.pro":
;************************************
print,'Which postprocessing routine do you want to perform?'
print,' "l" - for Lagrangian grid/mat.coloring,'
print,' "s" - for strain-rates/velocities,'
read, ookk   ; (ookk: data type = string; ookk is a general variable for reading
user input; ookk is ‘l’ or ‘s’)


;************************************
; STEP 1: SELECT FRAMES TO BE PLOTTED, FRAME 1 IS LISTED AS 0; THE NUMBERING OF
FRAMES IS BASED ON THE LIST OR OUPUT FILES STORED IN USER’ DATA DIRECTORY, NOT
THE FRAME NUMBER OF THE SOPALE OUTPUT FILE. For example: There are four model
ouput files in User’s Data directory: g01_p00_f01_o, g02_p00_f01_o,
g01_p00_f04_o, g02_p00_f04_o. Specify: frames=[0] to plot g01_p00_f01_o or
g02_p00_f01_o and frames=[1] to plot g01_p00_f04_o or g02_p00_f04_o
;************************************
;frames=indgen(nt1) ;*** PLOTS ALL MODEL OUTPUT FRAMES IN USER’S DATA DIRECTORY
;frames=[1,2,4] ;*** PLOT SELECTED FRAMES
frames=[8] ;*** PLOT ONE FRAME

;************************************
; STEP 2a: SELECT THE WINDOW TO PLOT BY EDITING 'horizontal_interval'; If
frames=[1,2,4] and horizontal_intervals=[0.,100.,100.,200.], IDL will plot two
set of diagrams, the first set of plots (frames 0, 1 and 3) from 0 to 100km and
the second set of plots (frames 0, 1 and 3) from 100 to 200km.
;************************************
horizontal_intervals=[0.,200.] ;*** (PLOT THE WINDOW FROM 0 TO 200 KM)
;horizontal_intervals=[0.,100.,100.,200.] ;*** (PLOT 2 windows: 0-100km, 100-
200km)

;************************************
; STEP 2b: PLOT WILL BE 1:1 SCALE, EDIT yrang WILL NOT CHANGE THE 1:1 SCALE
; NEED TO MODIFY yrang WHEN USER CHANGES THE PLOT WINDOW OR THE MODEL DEPTH
; THE SECOND VALUE IN yrang IS THE DISTANCE FROM THE TOP OF THE y-AXIS TO THE
; TOP OF THE MODEL PLOT
;************************************
yrang=[-0.5,10.] ; 10.=The distance from the top of the y-axis to the top of the
model plot (in km)

;************************************
; STEP 3: SPECIFY MECHANICAL MATERIAL COLORING, COLORS OF MATERIALS ARE
; SPECIFIED IN THE SCRIPT ‘color_set_up.pro’
; YOU NEED TO LIST ALL MATERIALS IN THE INPUT FILES EVEN IF SOME MATERIALS
; ARE NOT USED. THE VALUE gray(70) MEANS GRAYSCALE=70%
;************************************
material_colors=[1,2,magenta,Material4,Material5,pale_green,Material7,red,sedime
nt1,sediment3,sediment4,sediment5,sediment5B] ; Materials 1 and 2 are listed in
the model inputs but are not used for any mechanical materials boxes.

;************************************
; STEP 4: SPECIFY STRAIN RATE COLORING
;************************************
str_rates_col=col_scale ; col_scale=indgen(30)+1 (specify in default.pro)
;str_rates_col=[gray(0),gray(10),gray(20),gray(50),gray(60),gray(80)]
```

```
;*****************************************
; STEP 5a: SELECT PAPER SIZE AND NUMBER OF PANELS PER PAGE
; NOTE: ONLY NEED TO DO STEP 5c FOR THE PAPER SIZE ‘small_land’

;xaxis_style or yaxis_style: (1: box type; 5: suppress axis; 9: x&y axis only)
;Notes: 1=Force exact axis range, box type; 2=Extend axis range; 4=Suppress
;entire axis; 8=Suppress box style axis (draw only one axis); 16=Inhibit setting
;the Y axis starting value to 0 (Y axis only).
;*****************************************
;plot='big_port'   ;*** 11x17 paper, portrait
;plot='big_land'    ;*** 11x17 paper, landscape
plot='small_land' ;*** 8x11 paper, landscape
;plot='small_port' ;*** 8x11 paper, portrait

xaxis_style=1 & yaxis_style=1 ;(1: box type; 5: suppress axis; 9: x&y axis only)

;*****************************************
; STEP 5b: SPECIFY THE NUMBER OF PLOTS PER PAGE BY EDITING PARAMETER 'plot_n'
;*****************************************
plot_n=4 ;*** PLOTS 4 PANELS ON ONE PAGE

;*****************************************
; STEP 5c: SPECIFY THE SIZE OF PLOT ON 8X11 (LANDSCAPE) BY EDITING xpr and ypr
;*****************************************
landscape='y'  ; orientation of plot = landscape
   xpr = 26. & ypr = 17.0   ;*** size of the plot in cm

   multi=[0,1,plot_n,0,1]
   ikon=n_elements(frames)

;*****************************************
; STEP 5c: SPECIFY THE SIZE OF PLOT ON 11x17 (PORTRAIT) BY EDITING xpr & ypr
;*****************************************
if ikon gt 4 and plot eq 'big_port' then begin
   landscape='big'
   multi=[0,1,plot_n,0,1]

   xpr = 26. & ypr = 41. ;*** size of paper for print (in cm)

endif

;*****************************************
; STEP 5c: SPECIFY THE SIZE OF PLOT ON 11x17 (LANDSCAPE) BY EDITING xpr & ypr
;*****************************************
if plot eq 'big_land' then begin
   landscape='y'
   multi=[0,1,plot_n,0,1]

   xpr = 41. & ypr = 26  ;*** size of paper for print (in cm)

endif

;*****************************************
; STEP 5c: SPECIFY THE SIZE OF PLOT ON 8x11 (PORTRAIT) BY EDITING xpr and ypr
;*****************************************
if plot eq 'small_port' then begin
   landscape=''
   multi=[0,1,(ikon)*plot_n,0,1]
```

```
   xpr = 20. & ypr = 25   ;*** size of paper for print (in cm)

endif
;******************

another=n_elements(horizontal_intervals)/2
if another gt 1 then if  (horizontal_intervals(1)-horizontal_intervals(0)) ne
(horizontal_intervals(3)-horizontal_intervals(2)) then print,'HORIZONTAL SCALES
SHOULD BE EQUAL!!!'

xrang=horizontal_intervals(0:1)

@setting_up
;one2one_adjust='bottom'  ;
one2one_adjust='top'      ;
@zz_1x1                   ;*** PLOT 1:1 SCALE

;*************************************
;STEP 6: PLOT OPTION "s" (Strain rates and velocities)
;*************************************

;*************************************
;STEP 6a: SELECT LEGEND SIZE AND POSITION FOR "s" PLOT BY EDITING
"legend_position_y, legend_resize"
;*************************************
legend_position_y=0.5   ;***  0 : no legend;
                        ;*** >0 : separate panel

legend_new_panel='yes'
legend_resize=1         ;*** 1 is normal size
yaxis='z, km'           ;*** LABEL OF Y AXIS

aa=strpos(ookk,'s')
if aa ne -1 then begin
   for iqq=0,ikon*another-1 do begin
      i2=frames(iqq mod ikon)
      xrang=horizontal_intervals(0:1)
      if iqq gt ikon-1 then xrang=horizontal_intervals(2:3)
      if iqq eq ikon then !p.multi(0)=0

      ttt=model(0)+', step '+strtrim(string(convst(0,i2),format='(i5)'),2)+' ('
+strtrim(i2,2)+'), time '+strtrim(string(timee(0,i2),format='(f5.1)'),2)+' My,
!4D!3x='+strtrim(round(conve(0,i2)),2) ; *** title of plot
      xaxis=''

      ;*************************************
      ; LABEL OF X-AXIS CAN BE CHANGED BY EDING PARAMETER "xaxis"
      ;*************************************
      if !p.multi(0) eq 1 or iqq eq ikon then xaxis='x, km (S=0)'


plot,[0,1],[0,1],xrange=xrang,yrange=yrang,xst=xaxis_style,yst=yaxis_style,title
=ttt,/nodata,xtitle=xaxis,ytitle=yaxis

      ;*************************************
      ;STEP 6b: USERS SELECT FIELDS TO PLOT FOR OPTION "s"
      ;NOTES: "@mask" and "@finish_plot" SHOULD NOT BE COMMENTED OUT.
```

```
      ;**************************************
      style='fill'
      col=str_rates_col
      @e2_s                      ;*** strain rate coloring
      @mask

      style='contour_line'     ;*** solid lines
      @veloc                   ;*** velocities
      @finish_plot

   endfor
   @legend

endif

;**************************************
;STEP 7: PLOT OPTION "l" (Material color, lagrangian grid)
;**************************************
aa=strpos(ookk,'l')
if aa ne -1 then begin
   !p.multi=multi

   ;**************************************
   ;STEP 7a: SELECT LEGEND SIZE AND POSITION FOR OPTION "l" PLOT BY EDITING
"legend_position_y, legend_resize"
   ;**************************************
   legend_position_y=0.5   ;***  0 : no legend;
                           ;*** >0 : separate panel

   legend_new_panel='yes'

   for iqq=0,ikon*another-1 do begin
      i2=frames(iqq mod ikon)
      xrang=horizontal_intervals(0:1)
      if iqq gt ikon-1 then xrang=horizontal_intervals(2:3)
      if iqq eq ikon then !p.multi(0)=0

      ttt=model(0)+', step '+strtrim(string(convst(0,i2),format='(i5)'),2)+' ('
+strtrim(i2,2)+'), time '+strtrim(string(timee(0,i2),format='(f5.1)'),2)+' My,
!4D!3x='+strtrim(round(conve(0,i2)),2) ; *** title of plot
      xaxis=''

      ;**************************************
      ; LABEL OF X-AXIS CAN BE CHANGED BY EDING PARAMETER "xaxis"
      ;**************************************
      if !p.multi(0) eq 1 or iqq eq ikon then xaxis='x, km (S=0)'


plot,[0,1],[0,1],xrange=xrang,yrange=yrang,xst=xaxis_style,yst=yaxis_style,title
=ttt,/nodata,xtitle=xaxis,ytitle=yaxis

      ;**************************************
      ;STEP 7b: SELECT FIELDS TO PLOT FOR OPTION "l"
      ;NOTES: OPTIONS "@mask" and "@finish_plot" SHOULD NOT BE COMMENTED OUT.
      ;**************************************
      style='fill'
      style_col=1            ;*** Eulerian colors
      ;style_col=20          ;*** Lagrangian colors
```

```
      @matcol                ;*** material coloring
      @mask

      ;*************************************
      ; OPTIONS FOR PLOTTING LAGRANGIAN GRID (@lgrid_l)
      ; SELECT OPTIONS FOR "lgrid_stepx, lgrid_stepy, thick_line_freq"
      ; CHOOSE DENSER GRID FOR ZOOM IN IMAGE, LESS DENSE GRID FOR WIDER PLOTS
      ; thick_line_freq=0: NO THICK LINES AND NUMBERING OF LAGRANGIAN GRIDS
      ;*************************************
      style='contour_line'   ;*** solid lines
      lgrid_stepx=3          ;*** PLOT EVERY 3 HORIZONTAL GRID LINES
      lgrid_stepy=6          ;*** PLOT EVERY 3 VERTICAL GRID LINES
      thick_line_freq=20
      thick_line_numbers='yes'  ; to plot the number of vertical lines
      @lgrid_l

      @finish_plot

   endfor
   @legend
   @scale

endif
;*************************************
@print
end
```

4.4.5 Other Examples of Plotting Tasks

4.4.5.1 Injected Lagrangian Particle Plots

The related IDL scripts are stored in the directory /castor_home_0/mai/IDL_scripts/IDL_Lag-injected-disk-style-1to28/injected-Lag-disk-scripts/.
Parameters:
style='fill' or 'contour'
style_col=1: Eulerian coloring
style_col=20, 22, 23, 24, 27, 28: Lagrangian coloring
materials_to_color: materials for coloring
radius: radius of particles

Files which can be modified by users:
onfig_discs_test*.pro (or post*.pro)
mat_box.pro
phase_transition_patch.pro (specify the materials after the phase change)

Files which should NOT be modified by users:
matcol.pro
lagr_box.pro
lagr_limits.pro
injected_read.pro

phase_transition_patch.pro

```
; ************************************************************************
; File: phase_transition_patch.pro
; List the material numbers that do not appear in the initial set-up, but expected to appear during the
; run. Then run this patch after reading the data, but before drawing programmes

add_boxes=[13,14,15,16,17,19,20,21]   ; add boxes for the materials after the phase change
add_length=n_elements(add_boxes);
mec_boxes_add=fltarr(mec_boxes_no+add_length,8)
mec_colors_add=fltarr(mec_boxes_no+add_length)

mec_boxes_add(0:mec_boxes_no-1,*)=mec_boxes;
mec_colors_add(0:mec_boxes_no-1)=mec_colors;

i_add=0;
for ib=0,add_length-1 do begin
aaa=where(mec_colors eq add_boxes(ib),count);
if count eq 0 then begin
mec_boxes_add(mec_boxes_no+i_add,*)=mec_boxes(mec_boxes_no-1,*)*0;
mec_colors_add(mec_boxes_no+i_add)=add_boxes(ib);
i_add=i_add+1;
endif
endfor

mec_boxes_no=mec_boxes_no+i_add-1;
mec_boxes=mec_boxes_add(0:mec_boxes_no-1,*);
mec_colors=mec_colors_add(0:mec_boxes_no-1);
end
; ************************************************************************
```

Users' common modifications for plotting Lagrangian particles

```
xrang=[1550,1650]          ; width of model windows
yrang=[-20.,670.]
landscape="
xpr = 17. & ypr = 24       ; size of paper for print (in cm)
plots=1                    ; number of panels to plot on one page
multi=[0,1,plots,0,1]

i2=1<stope(0)              ; Select frames to plot
material_colors=[white,gray(10),gray(30),sea_green,brown,pale_yellow,pink,pink,lavender,
pale_green,gray(70),gray(90)] ; color or greyscale scheme for material coloring
style='fill'
style_col=24               ; Eul.=1;  Lag. coloring: 20, 21, 22, 23, 24, 27, 28
radius=1                   ; radius of disks
@matcol                    ; plotting material coloring
@finish_plot
```

IDL postprocessing steps:

1. Change directory to the data directory
2. transfer model outputs from p690 to your data directory
3. Change directory to the IDL post-processing directory
4. review readme_injected-Lag-disks.pro
5. modify session.pro
6. modify scripts 'onfig*.pro' or 'post*.pro'
7. type 'idl session' and press enter

IDL> .rnew read *(read the output data)*
IDL> .run phase_transition_patch *(if you want to plot the materials after phase change)*
IDL> .run onfig*.pro *(run the IDL plotting scripts)*
Save and/or Print postscript files

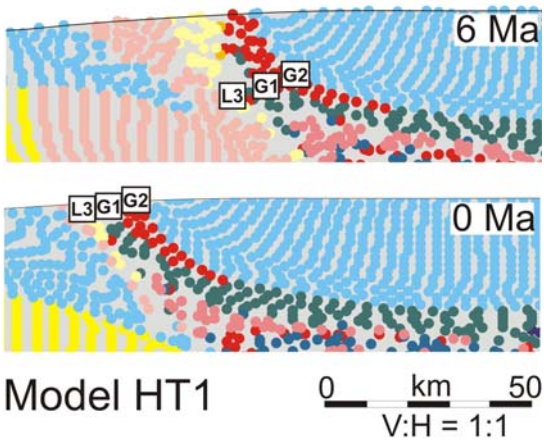


Figure 9 Lagrangian coloring of mechanical materials of a Microfem model (HT1). (Source: Jamieson et al., 2006)

4.4.5.2 Temperature contour (or fill) Plots

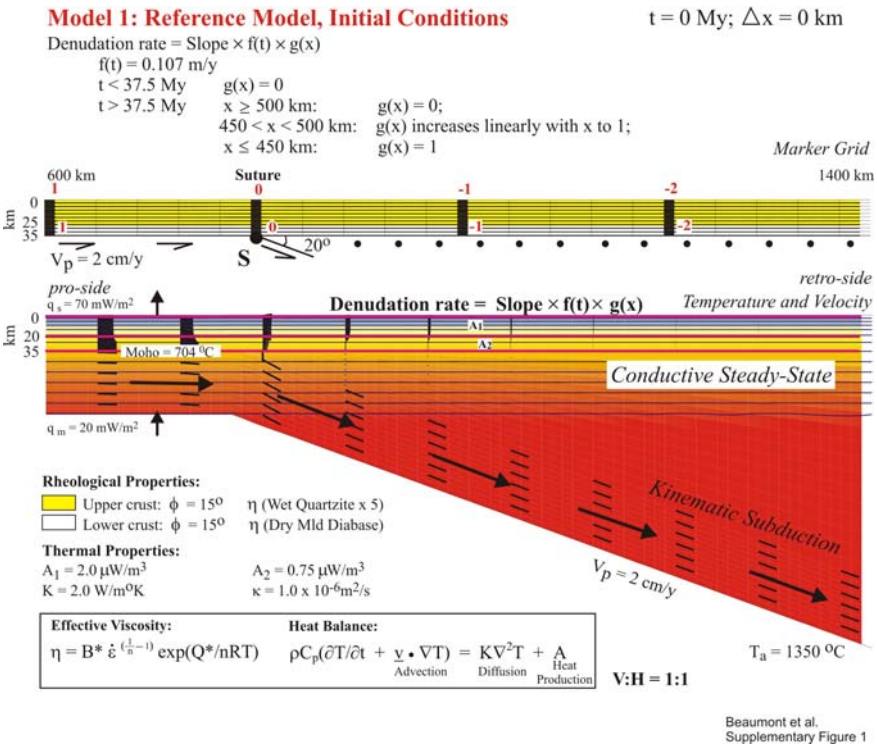


Figure 10 Temperature field was plotted as fill coloring.

4.4.5.3 Topography and Erosion Plots

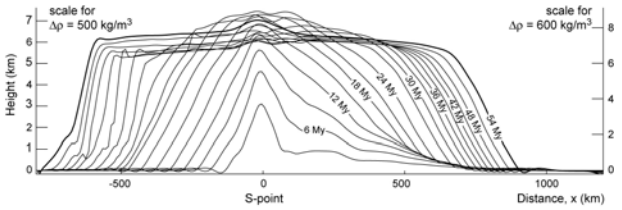


Figure 11a Evolution of the topography of a Microfem model HT1.
Source: Beaumont et al. (2004).

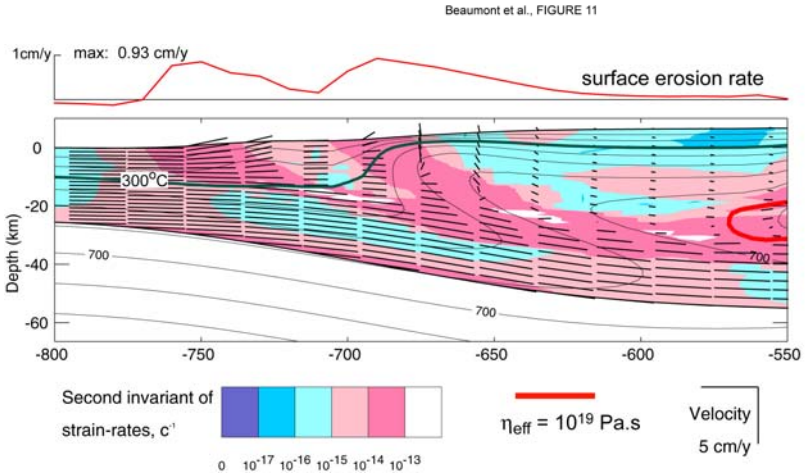


Figure 11b A diagram shows the strain rate (plotted as fill coloring), temperature contours, viscosity contours and the surface erosion plot. Source: Jamieson et al., HKT meeting, 2006.

4.4.5.4 Animations

The animations of model results can be created by various options (Figures 12 and 13).

- The simplest way is to post process the model results using the IDL scripts and obtain GIF animation files. Users can set the numbers of frames per second, the dimension of animation windows, numbers of panels and types of plots. The GIF animation files have the screen resolution.
- High resolution movies (*.mov) can be created from merging jpeg files in QuickTime. Users run IDL scripts to obtain a set of postscript files and then manipulate these postscript files on the personal computers using softwares such as Adobe Photoshop, CorelPhotoPaint, QuickTime, GIF Constructions, etc.

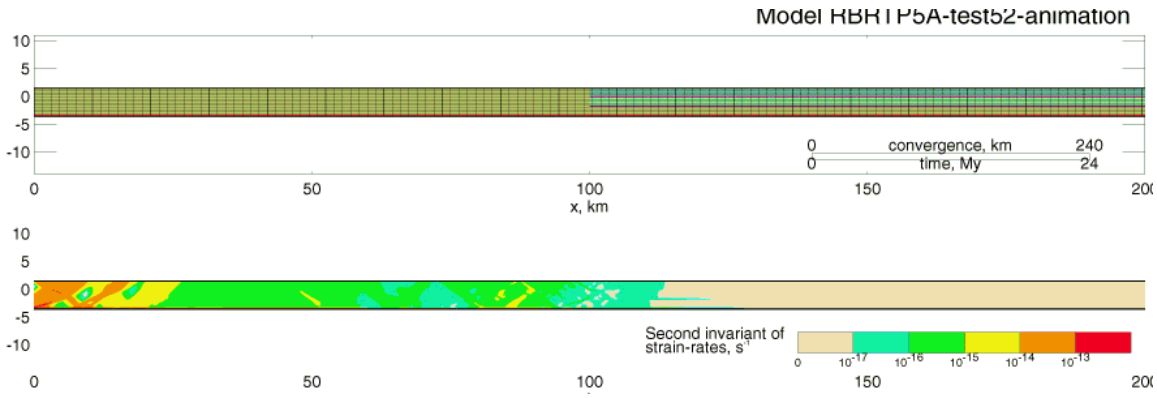


Figure 12 Frame 1 of a thin-skinned thrust-and-fold belts model animation. The first panel shows the Lagrangian grids and mechanical material coloring. The second panel shows the strain rate as fill coloring, and the strain rate legend.

Some common modifications from users are as follows:

```
Modify anim*.pro (model width and height, output frames, strain rate or
material colors, density of Lagrangian grids, velocity, etc.)
      xrang=[0,200]
      yrang=[0.5,11.]
      frames=indgen((nt1)<stope(0))    ; plots all saved frames
      plots=3                          ; Number of panels to plot
      material_colors=[1,2,magenta,red,sediment1,sediment3,sediment4,sediment
      5,sediment5B
      @anim_panel_grid&mat             ; select the type of panel
      @anim_panel_e2
```

IDL postprocessing steps:

1. Run model asking for matlab outputs and more output frames
2. Modify anim*.pro (model width and height, output frames, strain rate or material colors, density of Lagrangian grids, velocity, etc.)
3. Run the IDL plotting scripts

IDL> .rnew read
IDL> .run anim*.pro
Specify name of postscript (*.ps) or animation *.GIF files and save

4. Transfer postscript files or GIF animation files to the personal computers. Figure 12 shows various options of merging the animation files.

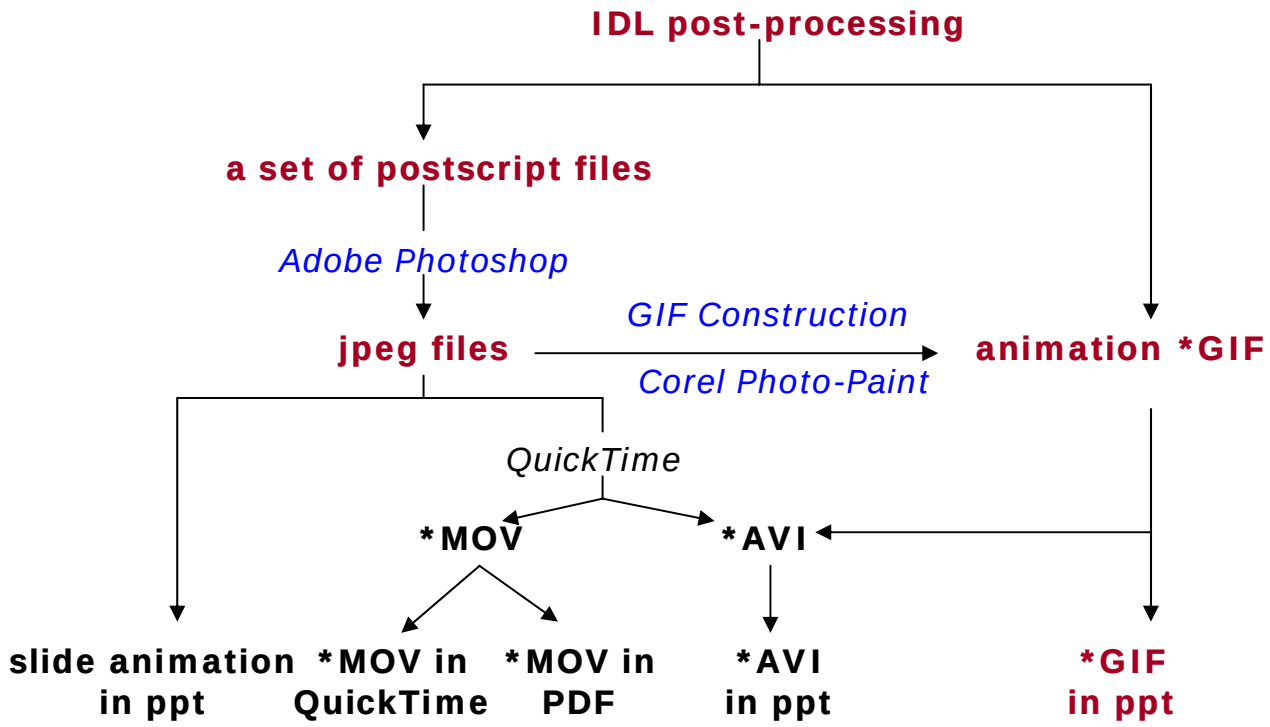


Figure 13 Creating animation files from model results using IDL, QuickTime, GIF Construction.

- IDL post-processing => to obtain the animation *.GIF. Users can run the IDL scripts to obtain the GIF animation files, which can be inserted to PowerPoint file for presenting.This is the simplest way but the GIF animation files have only the screen resolution .
- IDL post-processing => to obtain postscript files => jpeg files => insert jpeg files into Power Point and set animation slide show using menu Slide Show
- IDL post-processing => to obtain postscript files => jpeg files => create movie files (*mov, *avi) in QuickTime Pro => show animation in QuickTime or Adobe Acrobat PDF
- IDL post-processing => to obtain postscript files => jpeg files => create movie files (*GIF) using GIF Construction Set (Animation Wizard available) or Corel Photo-Paint => insert animation GIF files into PowerPoint

The following information is not directly related to IDL post-processing but it will assist you in making the animations and inserting the movies to your presentations using various softwares.

a) Transform postscript to jpeg format using Adobe Photoshop

Step 1: Recording the Action:

- ***Window*** > ***Show Actions*** >
  - Right click on the arrow at the top right corner to select ***New Action***.
    - Name the New Action and Set.
    - Type the directories for input directory (postscript files), output directory (store jpeg files)
  - Start Recording: From that point on you are in "record" mode and whatever you do is recorded as a set of actions.
    - ***File*** > ***Open***: Open the first postscript file
    - Rasterize Generic EPS (select resolution: 150 dpi; Mode: RGB)
    - ***Image*** > ***Rotate Canvas*** (Rotate the image 90 degree to get the correct orientation)
    - ***Layer*** > ***Flatten Image***
    - ***Crop***
    - ***File*** > ***Save As***: Save (save as jpeg: Format Option > Base Line (standard); Quality=10 (Maximum))
    - Close

Stop the recording.

Step 2: Transform all postscript files in a folder to jpeg format by using Batch Menu:

- ***File*** > ***Automate*** > ***Batch*** menu and you see a window where you have to set the input directory, the ouput directory and the action to be applied to all files in the directory.
  - ***Play*** >
    - Set (Select the Set that you recorded earlier)
    - Action (Select the Action that you recorded earlier)
    - Source (Folder where the postscript files are stored)
  - check the boxes "***Override Action Open commands***" and "***Override action save in commands***" so that you do not need to confirm opening each image.
  - Click OK (All actions recorded will be performed for every postscript files stored in the “Source” folder. The jpeg files will be saved in the output directory, which you specify during the Action recording.

b) Create Quick Time movies (*.mov) from *.jpeg in QuickTime Pro

Put all the jpeg files in a folder.
Each file should have the same name followed by a number; for example, "picture1," "picture2."

- In QuickTime Player, choose ***File*** > ***Open Image Sequence***, Browse and then select the first jpeg file.
- In the ***Image Sequence Settings*** pop-up menu, choose a frame rate.
- QuickTime Pro creates the movie, which shows each picture in sequence.
- Choose ***File*** > ***Save*** to name and save the movie.

c) Adding movies (*.mov) to a PDF file in Adobe Acrobat

- Choose ***Tool*** > ***Advance Editing*** > ***Movie Tool***
- Double-click on the page (in the pdf file) where you want to insert the upper left edge of the animation.
- ***Add Movie*** dialog box > Browse to select the movie clip
- ***Add Movie*** > ***Movie Properties*** => select
  - new content's compatibility => *Select Acrobat 6 Compatible Media* for all movie options; *Select Acrobat 5* for readers without Acrobat 6
  - *Embed Content in Document* => the PDf file include the movie (not the link only)
  - *Snap to Content Proportions* => the movie remains proportional
  - *Retrieve Poster from Movie* => the first frame as a still image when the movie is not playing
  - Playback (right click): select *Show Player Control*
  - *Appearance*: Invisible rectangle
  - Click *OK*

d) Adding movie (*GIF, *AVI) to a PowerPoint file

- Open the PowerPoint file
- GIF Animation: ***Insert*** > ***Picture*** > ***From file*** (Browse to select the animation GIF file)
- AVI animation: ***Insert*** > ***Movie*** > ***Movie from File*** (Browse to select the animation *.avi file)

e) PowerPoint slide show as animation

- ***Insert*** > ***Picture*** > From file: insert postscript or jpeg files into the PowerPoint presentation
- ***Format*** > ***Picture*** > ***Slide Position***: All diagrams should have the same position
- ***Slide Show*** > ***Slide Transition***:
  - Advance slide: Automatically after (select the desire interval time)
  - Modify transition
- ***Slide Show*** > ***Animation Schemes***
- ***Slide Show*** > ***Custom Animation***

5 DESCRIPTIONS OF IDL_M-S VARIABLES AND SCRIPTS

5.1 Parameters and Variables

Table 2a shows the parameters and variables of some IDL scripts.
Bonny Lee also summarizes the variable names, data type of variables and their description in Table 2b. The variable names in bold are ones which contain the data read in from the SOPALE output. The variable names in blue are ones which the user might modify for plotting. Variables are listed in alphabetical order.

Table 2a Summary of IDL_M-S variable descriptions (prepared by Bonny Lee, 2006).

variable name	data type	IDL routine	description
aa	string array	read_s_lagr.pro, read_s_eul.pro	list of files (including full pathname) In read_s_lagr.pro, aa contains the list of Lagrangian output files in the user's chosen data directory In read_s_eul.pro, aa contains the list of Eulerian output files in the user's chosen data directory
aa_m	string array	read_dialog.pro	list of full pathnames for idl/info files
aa_m_z	string array	read_dialog.pro	list of full pathnames for idl/info.gz files
aa_s	string array	read_dialog.pro	list of full pathnames for SOPALE1_i files
aa_s_z	string array	read_dialog.pro	list of full pathnames for SOPALE1_i.gz files
another	integer	"first level" routines	number of horizontal windows per frame to be plotted (e.g. if "another" = 2, then each frame will be plotted in 2 panels, one for x=horizontal_interval[0] to horizontal_interval[1], the second for x=horizontal_interval[2] to horizontal_interval[3])
bbco	integer array	lagr_limits.pro, lgrid_l.pro, matcol.pro, radcolor.pro, read_s_depth.pro zebra.pro	array listing the indices of the points (either Lagrangian or Eulerian) which fall within the plotting window
bblue	byte array	color_set_up.pro	Blue values for RGB colour table (see rred and ggreen)
bound_cond	integer array	read.pro	type of boundary condition
c_m	integer	read_dialog.pro	number of idl/info files found in directory
c_m_z	integer	read_dialog.pro	number of idl/info.gz files found in directory
c_s	integer	read_dialog.pro	number of SOPALE1_i files found in directory
c_s_z	integer	read_dialog.pro	number of SOPALE1_i.gz files found in directory
cell_size	integer array	set in bc_analysis.pro, used in lagr_box.pro, lagr_limits.pro, lgrid_l.pro, zebra.pro	cell_size[0] = max width (x) of all Eulerian cells in this frame cell_size[1] = max height (y) of all Eulerian cells in this frame
code	string array	read.pro	type of model ("Microfem" or "Sopale")
col_scale	integer array	default.pro	array containing integers 1 - 10; used for indexing into colormap?
coord	double array	read_s_lagr.pro	Equivalent to x2 and y2 (records 1 and 2) of SOPALE Lagrangian output file. Contains coordinates of nodes in Lagrangian grid for all frames to be processed; units used are kilometres, NOT metres! coord(n,0,framenum) is value of x coordinate for Lagrangian node n, frame=framenum coord(n,1,framenum) is value of y coordinate for Lagrangian node n, frame=framenum

conv	double array	read_p_data.pro	Convergence (in km) of Lagrangian grid conv(ik,it) is convergence for frame number it (ik will usually be 0)
convst_1	long integer array	read_s_info.pro, read_s_eul.pro	Array of timestep numbers for saved Eulerian output (corresponds to array isave_1 in file SOPALE1_i)
convst_2	long integer array	read_s_info.pro, read_s_eul.pro	Array of timestep numbers for saved Lagrangian output (corresponds to array isave_2 in file SOPALE1_i)
convtt	double array	read_s_eul.pro	Timestep numbers of Lagrangian output files convtt(ik,it) is value for output set ik (usually ik will be 0) and frame number it
data_directories	common	read.pro, session.pro	common data area
data_dirs	string array	read.pro	list of full pathnames for data directories; last entry is the name of the 'description file'
descriptionfile	string	read.pro	last entry in string array pref; name or location of description file
discribe	string array	read.pro	
exx	double array	read_s_eul.pro	Equivalent to e_fx1 (record 14) in SOPALE Eulerian output file. Contains horizontal strain rate at Eulerian nodes for all frames to be processed. inv2(j,i,frame) is value for node at column j, row i, frame = index of frame
exy	double array	read_s_eul.pro	Equivalent to e_fy1 (record 15) in SOPALE Eulerian output file. Contains shear strain rate at Eulerian nodes for all frames to be processed. inv2(j,i,frame) is value for node at column j, row i, frame = index of frame
ezz	double array	contour_mech.pro	field of values to be contoured
frames	integer array	"first level" routines, e.g. post_LHO-LS.pro	Contains a list of frame numbers to be plotted. For Sergei's original scripts, the numbering of the frames is based on the list of output files in the chosen data directory: 0 is the first file in the list, 1 is the second file in the list, and so on. This may be totally unrelated to the "frame number" in the name of the Sopale output file. Later scripts by Sergei may have added a feature that allowed the user to specify the frame numbers to be consistent with the Sopale output filename.
ggreen	byte array	color_set_up.pro	Green values for RGB colour table (see rred and bblue)
gray	integer array	color_set_up.pro	Array of indices into the RGB color table (see rred, ggreen, bblue). Given a gray level, say 70%, gray(70) would give an index n such that the RGB triplet [rred(n),ggreen(n),bblue(n)] would give a 70% gray level.
hint	string	read_dialog.pro	general variable for reading user input (?)

horizontal_intervals	real array	"first level" routines, e.g. post_LHO-LS.pro	Defines x coordinates for plot windows; coordinates are taken in pairs, i.e. first window begins at x=horizontal_intervals[0] and ends at x=horizontal_intervals[1], second window begins at x=horizontal_intervals[2] and ends at x=horizontal_intervals[3], etc.
i2	integer	"first level" routines, e.g. post_LHO-LS.pro	frame number (used as index to array frames)
ik	integer	read.pro	index of model that is being processed (initialized to 0 at beginning of read.pro) - but it doesn't look like scripts are set up to loop over ik...
ikon	integer	"first level" routines, e.g. post_LHO-LS.pro	number of elements in array frames (i.e. number of frames to be plotted)
inv2	double array	read_s_eul.pro	Equivalent to f1_sr (record 13) in SOPALE Eulerian output file. Contains second invariant of strain rate at Eulerian nodes for all frames to be processed. inv2(j,i,frame) is value for node at column j, row i, frame = index of frame
inv2_sig	double array	read_s_eul.pro	Equivalent to f1_sd (record 11) in SOPALE Eulerian output file. Contains second invariant of deviatoric stress (J2D) at Eulerian nodes for all frames to be processed. inv2_sig(j,i,frame) is value for node at column j, row i, frame = index of frame
iqq	integer	"first level" routines, e.g. post_LHO-LS.pro; lgrid_l.pro	loop counter; indexes the current plot
isolables	real	setting_up.pro	size of labels on isolines
ke	integer	read.pro	Number of nodes in horizontal dimension of Eulerian grid
kl	integer	read.pro	Number of nodes in horizontal dimension of Lagrangian grid
kx	integer	read.pro	not sure, but is set to same as ke to start with (i.e. number of nodes in horizontal dimension of Eulerian grid)
kz	integer	read.pro	Number of nodes in vertical dimension of Eulerian grid
kzl	integer	read.pro	Number of nodes in vertical dimension of Lagrangian grid
kzt	integer	read.pro	Number of nodes in vertical dimension of Eulerian temperature grid
landscape	string	"first level" routines, e.g. post_LHO-LS.pro	"yes" if the plot is to be in landscape mode

legend_new_panel	string	"first level" routines, e.g. post_LHO-LS.pro	"yes" ; to plot legend as a separate panel
legend_position_x		"first level" routines, e.g. post_LHO-LS.pro	
legend_position_y		"first level" routines, e.g. post_LHO-LS.pro	0 - no legend >0 - separate panel (need to use legend_new_panel='yes') <0 append to the bottom panel
legend_resize	integer	"first level" routines, e.g. post_LHO-LS.pro	1 is normal size
lgrid_stepx	integer	"first level" routines, e.g. post_LHO-LS.pro; lgrid_l.pro	grid spacing when plotting vertical grid lines (i.e. lgrid_stepx = 10 would plot every 10th vertical gridline)
lgrid_stepy	integer	"first level" routines, e.g. post_LHO-LS.pro; lgrid_l.pro	grid spacing when plotting horizontal grid lines (i.e. lgrid_stepy = 10 would plot every 10th horizontal gridline)
llii	string	read_dialog.pro	
ltd	double array	read_s_lagr.pro	Lagrangian temperature (in degrees Celsius) and depth (in km) ltd(n,0,framenum) is temperature for Lagrangian node n, frame=framenum, EXCEPT where the Lagrangian particle has left the Eulerian grid. In this case, the value of ltd(n,0,framenum) is the number of the last Eulerian cell that the particle occupied before it was lost. [ltd(n,0,framenum) is equivalent to t2 (record 9) of SOPALE Lagrangian output file, except for units used.] ltd(n,1,framenum) is depth for Lagrangian node n, frame=framenum, EXCEPT where the Lagrangian particle has left the Eulerian grid. In this case, the value of ltd(n,1,framenum) is either -1 or its depth minus 1, which ever is the most negative.
lveloc	double array	read_s_lagr.pro	Equivalent to vx2 and vy2 (records 3 and 4) of SOPALE Lagrangian output file. Contains x and y components of velocities for nodes in Lagrangian grid for all frames to be processed; units used are cm/yr, NOT m/s. lveloc(n,0,framenum) is value of x component of velocity for Lagrangian node n, frame=framenum lveloc(n,1,framenum) is value of y component of velocity for Lagrangian node n, frame=framenum

mat	integer array	read_s_eul.pro	Equivalent to color1 (record 16) in SOPALE Eulerian output file. Contains mechanical materials of Eulerian elements for all frames to be processed. mat(j,i,frame) is mechanical material number for element at column j, row i, frame = index of frame
mat_lagr	integer array	read_s_lagr.pro	Equivalent to color2 (record 5) in SOPALE Lagrangian output file. Contains mechanical materials of Lagrangian nodes for all frames to be processed. mat_lagr(n,frame) is mechanical material number for node n, frame = index of frame
mat_t	integer array	read_s_eul.pro	Equivalent to color1t (record 17) in SOPALE Eulerian output file. Contains thermal materials of Eulerian elements for all frames to be processed. mat_t(j,i,frame) is thermal material number for element at column j, row i, frame = index of frame
material_colors	integer	"first level" routines, e.g. post_LHO-LS.pro	array defining plot colours for materials. material_colors[i] gives the colour number to use for material (i+1) [since IDL indexes its arrays starting at 0]. The colour number is an index into the colour table defined in color_set_up.pro. There are some predefined values, so you can set up your colours like this: material_colors=[red, green, blue, brown]
mec_boxes	real array	set in read_m_mech_box.pro and read_s_box.pro, used in matcol.pro	Coordinates in km for the four corners of the mechanical material boxes mec_boxes[n,0] = x1 mec_boxes[n,1] = y1 mec_boxes[n,2] = x2 mec_boxes[n,3] = y2 mec_boxes[n,4] = x3 mec_boxes[n,5] = y3 mec_boxes[n,6] = x4 mec_boxes[n,7] = y4 where n is the number of the mechanical material box. Sergei says it's "adjusted to S point"
mec_boxes_no	integer	set in read_m_mech_box.pro and read_s_box.pro, used in matcol.pro	Number of mechanical material boxes, as defined in SOPALE1_i
mec_colors	integer array	set in read_m_mech_box.pro and read_s_box.pro, used in matcol.pro	Colours to use for plotting the mechanical material boxes mec_colors[i] = colour number to use for mechanical box i
model	string array	read.pro	name of model
mu	double array	read_s_eul.pro	Equivalent to vy1r (record 5) in SOPALE Eulerian output file. Contains Eulerian nodal viscosities for all frames to be processed. mu(j,i,frame) is value of viscosity of node for column j, row i, frame = index of frame

multi	integer array	"first level" routines	stores values for setting system variable !P.MULTI !P.MULTI0] is number of plots remaining on the page !P.MULTI[1] is number of plot columns per page !P.MULTI[2] is number of rows of plots per page !P.MULTI[3] is number of plots stacked in the Z dimension !P.MULTI[4] is 0 to make plots from left to right (column major), and top to bottom, and is 1 to make plots from top to bottom, left to right (row major).
ncolor22	integer	matcol.pro, matbox.pro, lagr_box.pro	Number of regions to color
nhe	integer array	read.pro	horizontal dimension for Eulerian grid (nodes in X)
nhl	integer array	read.pro	horizontal dimension for Lagrangian grid (nodes in X)
nst	integer	conv.pro, read_s_info.pro	number of timesteps for microfem output; also the number of frames of Eulerian output saved in Sopale (equivalent to nsave1 in SOPALE1_i)
nst_lagr	integer	read_s_info.pro	number of frames of Lagrangian output saved in Sopale (equivalent to nsave2 in SOPALE1_i)
nt	integer	read.pro	total number of frames to process
ntt	integer	read.pro	total number of frames to process
nt1	integer	read.pro	total number of frames to process
number	integer	read_s_eul.pro	Frame number of Eulerian output file (used within a loop in read_s_eul.pro)
nve	integer array	read.pro	vertical dimension for Eulerian grid (nodes in Y)
nvl	integer array	read.pro	horizontal dimension for Lagrangian grid (nodes in Y)
nvt	integer array	read.pro	vertical dimension for Eulerian temperature grid (nodes in Y)[Microfem only]
nxe	integer	conv.pro	horizontal dimension for Eulerian grid (nodes in X) for microfem output
nye	integer	conv.pro	vertical dimension for Eulerian grid (nodes in Y) for microfem output
nxl	integer	conv.pro	horizontal dimension forLagrangian grid (nodes in X) for microfem output
nyl	integer	conv.pro	vertical dimension for Lagrangian grid (nodes in Y) for microfem output
nyt	integer	conv.pro	vertical dimension for thermal grid (nodes in Y) for microfem output
ookk	string		general variable for reading user input
p	double array	read_s_eul.pro	Equivalent to nodpres (record 6) in SOPALE output file. Contains Eulerian nodal solid pressure for all frames to be processed. p(j,i,frame) is value of solid pressure for node at column j, row i, frame = index of frame
path	string array	read.pro	full path of directory which contains data to be processed
plot	string	"first level" routines, e.g. post_LHO-LS.pro	Defines paper size and orientation for plot: 'big_port' for 11 x 17 paper, portrait 'big_land' for 11x17 paper, landscape 'small_land' for 8x11 paper, landscape 'small_port' for 8x11 paper, portrait

plot_n	integer	"first level" routines, e.g. post_LHO-LS.pro	number of plot panels per page
pr	string	setting_up.pro, print.pro	'y' = print plot 'e' = save plot as EPS 's' = save plot as Postscript 'v' = view plot in GhostView 'c' = colour print 'l' = print on 11x17 paper 't' = print on transparency
predif	integer	read.pro	number of elements in pref (list of pathnames to search for data), minus 2
pref	string array	read.pro	same as data_dirs (list of directories to search for data files)
processed	string	read_dialog.pro	'y' if model(ik) has been selected for processing ?
reference	string array	read.pro	
rred	byte array	color_set_up.pro	Red values for RGB colour table (see ggreen and bblue)
ryad	long integer	read.pro	total number of nodes in Lagrangian grid
spoint	double array	read.pro	position of S-point
stop	integer array	read.pro	number of frames of saved topography data
stope	integer array	read.pro	number of frames of Eulerian field data
strain_e	double array	read_s_eul.pro	Equivalent to strain1 (record 18) in SOPALE Eulerian output file. Contains second invariant of total strain for Eulerian elements for all frames to be processed. strain_e(j,i,frame) is value for element at column j, row i, frame = index of frame
strain_l	double array	read_s_lagr.pro	Equivalent to strain2 (record 7) of SOPALE Lagrangian output file. Contains second invariant of total strain for nodes in Lagrangian grid for all frames to be processed. strain_l(n,0,framenum) is value for Lagrangian node n, frame=framenum
str_rates_col	integer array	"first level" routines, e.g. post_LHO-LS.pro	array defining plot colours for strain rate plot. str_rates_col[i] gives the colour number to use for the range specified by level[i] ? in e2.pro ? The colour number is an index into the colour table defined in color_set_up.pro.
string_bc	string array	read.pro	string contained parameters of b.c.
style	string	contour_mech.pro, "first level" routines, e.g. post_LHO-LS.pro	defines style of plotting fields as contours "fill" means plot filled contours "true_fill" means plot filled contours using a cell-fill algorithm (is slower than the other algorithm but works better for contours which are not closed) "contour_dot" means plot contours, using dotted line "contour_line" means plot contours, using solid line

style_col	integer	lagr_box.pro, "first level" routines, e.g. post_LHO-LS.pro	defines style of plotting material colours [these are Sergei's notes] 0 - set manual horizontal levels for material boundaries; this is not much used now 1 - plot Eulerian coloring; i.e. color each Eulerian cell according to its mechanical material value 20 - plot Lagrangian material boxes based on initial model setup; this method needs the Lagrangian material boxes from the initial model setup and maps those boundaries to the current location of the corresponding Lagrangian particles. Note that this method requires a file containing the initial Lagrangian configuration, and it does not work very well for high deformations. 21 -like style_col 20, but includes only points that are definitely inside the Lagrangian box 22 - like style_col 21 with the addition of one more element around the outside 23 - disk coloring of Lagrangian particles: plot a filled circle at the position of each Lagrangian particle where the color of the circle is based on the material value. 24 - disk coloring using initial Lagrangian boxes; plot Lagrangian points within each Lagrangian box as filled circles.
te	double array	read_p_data.pro	Contains coordinates of surface nodes (top row) of Eulerian grid; units used are kilometres, NOT metres! te(model,j,0,frame) is X coordinate for surface node at column j, frame = index of frame, model=index of model [usually 0] te(model,j,1,frame) is Y coordinate for surface node at column j, frame = index of frame, model=index of model [usually 0] te(model,j,2,frame) is Y coordinate for bottom node at column j, frame=index of frame, model=index of model [usually 0]
tex	double array	read_s_eul.pro, read_p_data.pro	Equivalent to x1 and y1 (records 1 and 2) in SOPALE Eulerian output file. Contains coordinates of nodes in Eulerian grid for all frames to be processed; units used are kilometres, NOT metres! tex(j,i,0,frame) is X coordinate value for node at column j, row i, frame = index of frame tex(j,i,1,frame) is Y coordinate value for node at column j, row i, frame = index of frame
temp	double array	read_s_eul.pro	Equivalent to t1 (record 9) in SOPALE Eulerian output file. Contains Eulerian nodal temperatures for all frames to be processed. temp(j,i,frame) is value of temperature for node at column j, row i, frame = index of frame
timet	double array	read_s_eul.pro	Time (in My) for Sopale output files timet(ik,it) is value for output set ik (usually ik will be 0) and frame number it
total	integer	read.pro	total number of data sets or models (hardwired to 1 at beginning of read.pro)

tv	double array	read_s_eul.pro	Equivalent to vx1 and vy1 (records 3 and 4) in SOPALE Eulerian output file. Contains Eulerian velocities for all frames to be processed; units used are cm/yr, NOT m/s. tv(j,i,xy,frame) is value of velocity component for node at column j, row i, xy=0 for X component, xy=1 for Y component, frame = index of frame
type	string array	read.pro	
uu	integer		generic file unit number
vconv	double array	read.pro	convergent velocity
vert_margin	real array	setting_up.pro	values for setting system variable !Y.MARGIN !Y.MARGIN[0] is bottom margin, in units of character size !Y.MARGIN[1] is top margin, in units of character size
xpr	real	"first level" routines, e.g. post_LHO-LS.pro	width of paper in cm
xrang	real array	"first level" routines, e.g. post_LHO-LS.pro	Defines x coordinates (i.e. min x, max x) for current plot window
xss	real	setting_up.pro	ratio of width/height for the plot device
yaxis	string	"first level" routines, e.g. post_LHO-LS.pro	label for y axis ?
ypr	real	"first level" routines, e.g. post_LHO-LS.pro	length of paper in cm
yrang	real array	"first level" routines, e.g. post_LHO-LS.pro	Defines y coordinates (i.e. min y, max y) for current plot window. Comment from script: THE FIRST VALUE IN yrang IS THE DEPTH FROM MODEL SURFACE TO THE BOTTOM X-AXIS ; THE SECOND VALUE IN yrang IS THE DISTANCE FROM THE TOP X-AXIS TO THE SURFACE OFTHE MODEL

Table 2a List of variables of some IDL scripts.
The numbering of the IDL scripts is based on Table 1.

```
*******************************************************************************
10) arr.pro (subroutine,ms,2%, 4)
Parameters used from veloc.pro:
   (xb,yb) - coordinates
   (vx,vy) - vector
   vvv - vector-scale
   xss - x/y ratio of the plot (calculated automatically)
**********************************
19) conv.pro (programme,m,10%, 2)
- INFO: eulerian grid (nxe,nye)
- INFO: thermal grid (nyt) and time steps (nst)
- INFO: lagrangian grid (nxl,nyl) and topo-time (ntt)
**********************************
25) finish_plot.pro (include,ms,10%, 4)
Parameter in default.pro:
otline_thickness: thickness of outter line
radi: radius of circle on S-point
**********************************
28) lagr_box.pro (include,ms,10%, 2 and 4)
mat_box(ncolor22,8),
lagr_color(ncolor22),
erotime, style_col (20, 21, or 22)
 - to set before call address to subroutine: @lagr_box
**********************************
29) lagr_limits.pro (include,ms,10%, 2)
bbco - #s of lagrangian points,
l_x, l_y - coords
**********************************
45) read_m_erosion.pro (include,m, 10%, 2)
   erostyle - integer:
      =1 (eros prop to topo)
      =2 (eros prop to uplift vel)
      =3 (erosion prop to slope)
   ero - integer: amount of history events
   erostages - arrey(ero): time boundaries between events
   erovvalues - arrey(ero): value of erosion coefficient on the event boundaries
   erospaces - integer: amount of erosion provinces
   erospace - arrey(erospaces): boundaries between provinces in km (S=0)
   erovvalues - arrey(erospaces): value of erosion weight on the province boundaries
**********************************
46) read_m_file.pro (include,m,2%, 2)
   hint - name of file
   path(ik) - directory of data, ended with ../idl
**********************************
49) read_m_mech_box.pro (include,m,10%, 2)
- Boxes (mec_boxes) are defined as in input files:
  [x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)
- Outputs: mec_boxes_no - number of mechanical boxes
          mec_colors(mec_boxes_no) - number of color to each box
          mec_boxes(mec_boxes_no,8) - coordinates of boxes in km, adjusted to S-point
**********************************
50) read_m_therm_box.pro (include,m,10%, 2)
- Boxes (ther_boxes) are defined as in input files:
   [x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)
```

```
- Outputs: ther_boxes_no - number of thermal boxes
          ther_colors(ther_boxes_no) - number of color to each box
          ther_boxes(ther_boxes_no,8) - coordinates of boxes in km, adjusted to S-point
****************************************
58) read_s_mech_box.pro (include,s,10%, 2)
- Boxes (mec_boxes) are defined as in input files:
   [x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)
- Outputs: mec_boxes_no - number of mechanical boxes
          mec_colors(mec_boxes_no) - number of color to each box
          mec_boxes(mec_boxes_no,8) - coordinates of boxes in km, adjusted to S-point
**********************************
"read_sopal"

total=1
path=strarr(total)
model=path
type=model

discribe=strarr(total)
reference=strarr(total)

nhe=intarr(total) ; horizontal dim for eul
nve=intarr(total) ; vertical -"-
nvt=intarr(total) ; vertical dim for eul T
nhl=intarr(total) ; horizontal dim for lagr
nvl=intarr(total) ; vertical -"-
stope=intarr(total)  ; saved field data
stop=intarr(total)  ; saved topography data
spoint=dblarr(total) ; position of S-pont
vconv=dblarr(total)  ; converggent velocity
bound_cond=intarr(total)  ; type of b.c. (1 - for Microfem)
code=strarr(total)  ; type of code ('microfem' or 'sopale')

ke=max(nhe)
kl=max(nhl)
nt=count_eul;  max(stop)+1
ntt=count_eul;  max(stop)+1
kx=ke
kz=max(nve)
kzt=max(nvt)
kzl=max(nvl)
nt1=count_eul;   max(stope)+1

conv=fltarr(total,nt)
timet=fltarr(total,nt)
convtt=lonarr(total,nt)
conve=fltarr(total,nt1)
timee=fltarr(total,nt1)
convst=lonarr(total,nt1)

;Topography
te=fltarr(total,ke,4,nt)
tl=fltarr(total,kl,4,nt)

;Velocity
tv=fltarr(kx,kzt,2,nt1)
tex=fltarr(kx,kzt,2,nt1)
```

```
mu=fltarr(kx,kz,nt1)
Inv2=fltarr(kx,kz,nt1)
exx=fltarr(kx,kz,nt1)
exy=fltarr(kx,kz,nt1)
mat=intarr(kx,kz,nt1)
temp=fltarr(kx,kzt,nt1) ; only if tcompute eq 1

ryad=long(kl)*kzl
coord=fltarr(ryad,2,nt1+1)
ltd=fltarr(ryad,2,nt1+1)+5  ; just to make that >0, values <0 assigne for eroded
;mat_lagr=fltarr(ryad,2)

;erostyle - integer:
;   =1 (eros prop to topo)
;   =2 (eros prop to uplift vel)
;   =3 (erosion prop to slope)
;ero - integer: amount of history events
;erostages - arrey(ero): time boundaries between events
;erovvalues - arrey(ero): value of erosion coefficient
;            on the event boundaries
;erospaces - integer: amount of erosion provinces
;erospace - arrey(erospaces): boundaries between provinces in km (S=0)
;erovvalues - arrey(erospaces): value of erosion weight
;            on the province boundaries

 mec_boxes_no - number of mechanical boxes
;         mec_colors(mec_boxes_no) - number of color to each box
;         mec_boxes(mec_boxes_no,8) - coordinates of boxes in km,
;                   adjusted to S-point

 ther_boxes_no - number of therhanical boxes
;         ther_colors(ther_boxes_no) - number of color to each box
;         ther_boxes(ther_boxes_no,8) - coordinates of boxes in km,
;                   adjusted to S-point

mt=intarr(kx,kzt,nt1)
radio=fltarr(fix(aa)+1)

Parameters:

upp_slab=fix(nve(0)-1)

printall
print
landascape

Plotting parameters:
xaxis_style=1 - for plotting axis (default)
                  =5 - to suppress axis
************************************************************
```

Notes: Sergei Medvedev described the IDL scripts as the following notation (Tables 1 and 2a):
'**m**': routines for Microfem models
'**s**': routines for Sopale models
'**program**': programs (end with "end" operator); use .run to run in IDL environment
'**include**': included files in programs or included in other 'include files' or in 'Batch file'.
'**subroutine**': Functions can be called by Main Level and Second Level programs

'**batch**': General purpose program has to be run as shell script (@default.pro, etc.)
'**%**': Sergei estimated the percentage of chance that the files will be modified by users.
Files categorized as more than 50% will likely need to be modified by Users to suit the specific purpose.
'data': field data for comparing to model results or other data.

5.2 Description of IDL_M-S Scripts (written by Sergei Medvedev, 2002)

The text in this section was taken from Sergei's notes (with few minor typo corrections) at the beginning of the IDL scripts; however, not all files have explanations. These scripts should be stored in the IDL_M-S Software Directory (see Fig. 5a). The programs are listed in an alphabet order in this section. A summary of their classification is presented in Table 1.

1) anim_bc.pro (include,s,80%, 1)
To illustrate boundary condition (bc) on the animation.
Notes:
- Subroutine to animate bc. This very much depends on type of bc. Check this routine, and *bc_analysis.pro* and SOPALE User guide if you want to improve/apply this subroutine.
- In version of 31 Aug, 2002, only bc#12 (no isostasy, no S-point) for Sopale implemented (see top panel on http://adder.ocean.dal.ca/sergei/thrmal/RBRTP-17_web_fast.gif : moving bottom strip, moving arrows at the base, backstop, material moves out in the left-vbottom corner)

2) anim_clos_gif.pro (include,ms,30%, 1)
Subroutine that wraps up the animation: save each frame, converts it to GIF, merges all frames into GIF animation, removes all intermediate files, and places it in proper place after permission changed to see it world-wide.
Notes:
- The purpose of this subroutine is to read from the screen, save the screen in graphic format, close the frames loop and then compile frames into animation file.
- We implemented here gif-animation. The procedure is simple: read screen (tvrd()), save frames in separate gif files and then merge them into gif animation file. The procedure is simple if the version of IDL in use has GIF license.

- The version presented here oriented on the "IDL without GIF license" (e.g.: firedrake and viper). So, a complicated gif conversion is performed using another computer, which has GIF license(e.g.: adder):
* we read screen in IDL internal format,
* save it remotely on another computer which has GIF license (adder),
* save color map there (we need it for GIF files),
* create procedure (batch file) to run this particular animation on another computer (adder),
* and then run the compilation of the animation from the remote computer (adder)
See details of GIF animator on http://www.lcdf.org/gifsicle/
* Clean extra files in anim_dir.
* Please, make sure that compile_dir and anim_dir are separate directories, and should be visible from adder, otherwise you have good chance to remove the resulting animated file.

3) anim_clos_mpg.pro (include,ms,?%, 1)

No success with nice MPEG animation so far - if someone wants to continue, this file can substitute the above in closing and saving animations. It successfully creates JPEG (24 byte color map), but merged MPEG does not look nice and it is big

4) anim_panel_e2.pro (include,ms,20%, 2)
Panel included into animation.pro to plot contours of the second invariant of str-rates, format and plot legend (see bottom panel in http://adder.ocean.dal.ca/sergei/thrmal/RBRTP-17_web_fast.gif)

5) anim_panel_grid&mat.pro (include,ms,40%, 1)
Panel included into animation.pro to plot contours of materials (matcol.pro) and lagrangian grid (lgrid_l.pro), illustrate bc and time (anim_bc.pro and anim_time.pro)(see top panel in http://adder.ocean.dal.ca/sergei/thrmal/RBRTP-17_web_fast.gif)

6) anim_panel_mat.pro (include,ms,40%, 1)
Panel included into animation.pro to plot contours of materials (matcol.pro), format and plot legend (see middle panel in http://adder.ocean.dal.ca/sergei/thrmal/RBRTP-17_web_fast.gif)

7) anim_setup.pro (include,ms,20%, 3)
Sets up technical parameters and arrange environment for animation (font type, plotting window etc.). Similar to setting_up.pro
Notes:
- setting the plot parameters, used also by print.pro (link to label "print:"
- Manipulating with colors, devices (window) etc.
- Frames loop started here and finished in anim_clos.pro

8) anim_time.pro (include,ms,20%, 4)
Colored time scale (see top panel on http://adder.ocean.dal.ca/sergei/thrmal/RBRTP-17_web_fast.gif)
Notes:
- This subroutine is build on the basis of scale.pro and legend.pro
- Simply coloring the time past

9) animation.pro (program,ms,100%, 1)
Most of the parameters and steps of animation are set here. Many parameters explicitly specified here: program calls anim_setup.pro, anim_panel_* (examples of several panels are presented here, more can be created using examples as templates), anim_clos_gif.pro.
Notes:
- Main Level Program animation for Sopale and Microfem
- It is set up by anim_setup.pro and wrapped by anim_clos.pro.
- Panels are plotted by subroutines anim_panel_*.pro.
- Several supporting subroutines with name started with anim_* (see list).
- See anim_clos.pro for details of output procedures.
- Output directory (anim_dir) is specified in this program.
- Important parameter: size of the window (bigger size results in bigger animation size)
- Parameters of animation run (in 1/100 sec):
  time to hold initial picture, time per frame,
  time to hold the last picture

10) arr.pro (subroutine,ms,2%, 4)
Subroutine to plot vector fields
Notes:
Parameters used from veloc.pro:
(xb,yb) - coordinates
(vx,vy) - vector
vvv - vector-scale
xss - x/y ratio of the plot (calculated automatically)

11) bc_analysis.pro (include,ms,20%, 2)
Extracts (not read) bc data from Sopale1_i, analyses boundary velocities (to define valoc_scale_global - characteristic velocity; and specifies centers of restricted areas where boundary velocity changes rapidly), specify characteristic cell-sizes. Called from read.pro and asia.pro (the letter to redefine characteristic scale for velocity). Boundary conditions are not entirely analyzed yet.
Notes:
- Subroutine to analyze bc of the model (called from program read.pro).
The type of bc is already known to that point (bound_cond):
Microfem is always 1,
Sopale has several different options.
- This subroutine is still under construction and has to (at some point) recognize most of the boundary data and how this data can affect plotting. Some functions are already implemented here:
- This subroutine extracts parameters of bc for Sopale and counts their number.
If you know the order of the bc parameters and know their meaning - you already can use them. See example in anim_bc.pro (bc #12 is used there).
- Be aware that bc parameters are not converted to conventional like all other data (cm/y, My, km, etc.).
It assign global scale for velocity: max along the boundary at init time or (if previous is 0) by total max of velocity. Later, in veloc plots you can chose between global scale (assigned here) and local scale.
It also searches for points of 0 velocity along the boundaries that has non-zero neighbors and assigns these points as restricted.
It also calculates characteristic size of a cell: to bound the restricted areas and to set limits for line removal in lagrangian grid plots.
- Program procedures:
Extracting b.c. from Sopale data
Analyzing specific b.c. (examples) for Sopale main target here- to understand how we should evaluate convergence in the system
Setting lagr cell size - to use in lagr grids and in restricted areas - to remove very long lines
Setting characteristic velocity : velo_scale_global (note that velocity, tv, is already in cm/y)
Analyze bc for discontinuity, create restricted areas

12) chris_pictures.dat (data,ms,0%, 5)
Picture of "Chris on top of his favorite orogens". Should be read by program chris_read.pro

13) chris_read.pro (include(batch),ms,0%, 2)
To read picture of Chris

14) cig.pro (program,*,0%, 5)
Plots how many cigarettes Sergei smokes (y axis) against time (x axis).

15) circle.pro (subroutine,ms,2%, 2)
Creates 50-points circle

16) color_present.pro (program,*,0%, 4)
Presents colors that defined by this package and can be used in color and black&white plotting.

17) color_set_up.pro (include(batch),ms,40%, 3)
Subroutine to set up the color map of the plots. Colors can be added (see details inside file)
Notes:
- This subroutine is called by "setting_up" before each plot is created.
- After the subroutine applied, you can refer to color by just giving its number.
  For convenience, several color numbers are assigned to color names ("yellow", "red" etc.;).
  The last 101 elements of colormap set the grayscale, you can refer to them by gray(n), where n is desired percentage of gray.
- The palette can be viewed/printed using program color_present.pro
- This routine also changes background color: from black for screen to white in PS.
- You can switch parameter color between 'b/w' and "color" (in subroutine "default.pro").
  That will only affect named colors, but not change the palette.
  For example, if color='color' "yellow" will be yellow, but if color='b/w', "yellow" will be gray with 25% (if we set yellow=gray(25)).
- User can interpret any color to grayscale by your own in the second part of the subroutine.
- User can modify colormap by adding more colors (up to 154) to the end of explicit expression for RGB values (rred,ggreen, bblue arrays). Do not change the 0th element and grayscale.

18) contour_mech.pro (include,ms,20%, 4)
Complicated subroutine to plot contours (line of filled) of mechanical fields
Notes:
- Contouring mechanical field (used in most of subroutines for filled/line contouring)
- This subroutine recognizes type of field (lagr, eul, elemental, nodal), reformat data, smooth, prepare legend.
- Smoothing (Eulerian fields smooth only)

19) conv.pro (program,m,10%, 2)
Complicated program to convert standard output from Microfem into data readable by this software package. Each Microfem run result should be converted once, the converted results are stored in the subdirectory "idl" in User's Data Directory.
Files in this "idl" subdirectory can be recognized by program read.pro
Notes:
Program procedures:
- Directory search
- INFO: Eulerian grid (nxe,nye)
- INFO: thermal grid (nyt) and time steps (nst)
- INFO: Lagrangian grid (nxl,nyl) and topo-time (ntt)
- Initializing the arrays
- Reading temperature fields

- Reading Eulerian arrays
- Reading Lagrangian arrays
- Reading Lagrangian topography
- Eulerian topography
- Obtaining additional information: evolution arrays
- Obtaining additional information: densities
- Obtaining additional information: viscosity limits
- Saving information
- Cleaning

20) default.pro (include(batch),ms,100%, 3)
Sets many parameters, technical and user-defined.
Notes:
- It calls automatically before the end of each program.
- If you change some parameters there, you have to run this file as a batch program (IDL>@default.pro)

21) downloads.pro (include,ms,90%, 2)
Allow you to define which fields you want to download by program read.pro. Especially important for large data sets and small computers.
Notes:
- If read_pro used to read Sopale data, you can chose what fields you want to download - make sure that fields you don't want are commented out (;), and vise versa
- You don't need to worry about each field: it will be no error (with Sopale) if data is not saved, but unmark - it will remain array of '0'
Coordinates (tex and coord) and depth (ltd) should not be commented, though.
- Microfem data should be read entirely
- Lagrangian data: Note here, that lagrangian fields have one more time step assigned than eulerian. This is because the initial position of eulerian grid is not important, while it is very important for lagrangian.
*** Sergei assigns the first time step to both zero and first lagrangian coordinates.
If someone have an idea how to find very initial data for Lagrangian, feel free to implement, or to discuss that with Segei M..

22) e2.pro (include,ms,70%. 1)
Plotting routine to contour (lines or fills) second invariant of str-rates and (if filled) prepare legend parameters. Set ranges, labels, etc manually. There are some traditional default settings.
Notes:
- Contour labels (do not appear if filled contour is chosen)
- to contour_mech

23) egrid (include,ms,10%, 4)
Subroutine to plot Eulerian grid
Notes:
- The regularity of plotted grid defined by lgrid_stepx and lgrid_stepy -
- The same as in Lagrangian grid plotting by lgrid_l.pro

24) erosion_front.pro (include,m,10%, 4)
Program to plot distribution of erosion
- automatically plots if there is erosion
- set the color for plot: erosion_front_color for colored pictures

Procedures:
- Extracting the info from erosion-data
- Calculations of the erosion - distribution
- Plotting

25) exponent.pro (subroutine,ms,10%, 2)
FUNCTION Exponent, number
To present nicely numbers (on the legend mainly) - just give the number
to that subroutine (like: 1.e21 -> 10(21), where number 21 will appear as superscript)

26) finish_plot.pro (include,ms,10%, 4)
Parameter in default.pro: otline_thickness - thickness of outer line
- Program to finish plot:
- outline domain and put S-point

Notes:
radi=1.2 ; radius of circle on S-point
Notes: The size of circle depends on number of plots on the page.

27) ground_0.pro (program,ms,10%, 2)
Program to make the undeformed upper surface of model to be at y=0

28) inversion.pro (program,ms,10%, 2)
Change sides of model domain (+ -> -)

29) lagr_box.pro (include,ms,10%, 2 and 4)
Subroutine to color lagrangian boxes
Notes:
- Boxes (mat_box) are defined as in input files:
- [x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)
- The function dist is calculated as the minimum distance to the boundary of box (at t=0), with positive sign if lagrangian point is on the same side from boundary as central point of the box (centx, centy - calculated as average of 4 corners of the box). Then positive area of dist is colored.
Dist is smooth function across the boundary -> contour is smooth.
- Parameters: mat_box(ncolor22,8), lagr_color(ncolor22),
erotime, style_col (20, 21, or 22) - to set before call address to subroutine: @lagr_box

30) lagr_limits.pro (include,ms,10%, 2)
Routine to search for lagrangian points inside and in vicinity of the plotting area and to exclude "restricted" area points
Notes:
Result: bbco - #s of lagr. points, l_x, l_y - coords

31) legend.pro (include,ms,50%, 4)
- Parameters in default.pro:

legend_back='no' - The size of legend is not known before legend is finished.
Setting this parameter to 'yes' will force subroutine do the following after legend is finished: cover plotted legend by background color and change legend_back to 'no'. So that second application of legend.pro will be plotted on the background - useful if main plot covers almost entire window and there is no place to put legend outside (see examples on Ritske's animations).
The sequence in the program will be like that:

```
legend_back='yes'
@legend
@legend
```

The default 'no' assumes that you plot legend only once

legend_position_y=0. - Vertical position of the legend (~0 - along the bottom of the plot, 1 - at the top; negative values and values >1 allowed on your own risk). Setting the value exactly to 0 force not to plot legend

legend_position_x=0. - Horizontal position of the beginning of the legend (0 - left of the plot, 1 - right side of the plot). There is no special meaning for value 0, likely in legend_position_y.

legend_resize=1. - you can rescale the legend (including text) proportionally using this parameter

vertical_rescale=1.5 - you can rescale vertical dimension of the legend boxes

legend_new_panel='no' - if 'yes' is set, the legend will be plotted on the new panel, if 'no' - the same as main plot

legend_mantle_labels='no' - leave it to 'no' if you don't work with Microfem models.

velo_legend='no' - do not change this parameter

- Legend will be presented if legend_position_y not 0. Legend uses colored boxes assigned by the last filling subroutine.
- Legend boxes and space between them can be resized using legend_resize (1 by default). It does not affect font size.
- Legend will be plotted on the new panel, if legend_new_panel is 'yes', otherwise it will be plotted using recent panel (you can assign negative value to legend_position_y and legend appears below that recent panel.

32) lgrid_l.pro (include,ms,30%, 4)
This routine plots lagrangian grid. Optionally, it plot certain vertical grid lines thicker; it can also put numbers at the ends of the thick lines. The main parameters of this routine should be specified before call to this routine, or by using parameters in default.pro:

lgrid_stepx=10 - plot every 10th vertical lagrangian line
lgrid_stepy=10 - plot every 10th horizontal lagr. line
thick_line_freq=7 - set it to 0 to suppress thick lines; otherwise each 7th thin line will be thick (the thickness of thick lines can be changed be editing lgrid_l.pro)
thick_line_numbers='yes' - set it to 'no' to suppress numbering the thick lines (it does not have an effect if thick_line_freq=0)

Name of the routine came from "lagrangian grid with limits". If two initially neighboring lagrangian points went too far from each other during deformation, these points will not be connected. The limits can be changed by editing this file.

33) mask.pro (include,ms,10%, 4)
Some of the plotting routine (especially contouring routines) make some noise outside of real plots (due to irregularity of data). The mask.pro put background colored masks outside the targeted plotting area.

34) mat_box (include,ms,50%, 1)
Tracing the lagrangian box, arbitrary set by User. This routine is similar to matcol.pro (with style=2*), except it uses arbitrary set material boxes (not mechanical boxes from Sopale1_i)

35) matcol.pro (include,ms,50%, 1 and 4)
Subroutine to color or outline the materials.
The style_col and is defined now in "default.pro" or it can be defined in the calling plotting routine

style_col - style of coloring:
0 - set manual horizontal levels for material boundaries
(up to 3 colors allowed)
1 - use Eulearian coloring
2* - use Lagrangian boxes from input (does not work very good for high deformations):
20 - contouring distance to the boundaries
21 - coloring from inside (all colored - definitely belong)
22 - coloring from outside (all that belong - definitely colored)
material_colors=[blue,yellow,bottom_grey] - set colors for materials

The routine colors materials if style_col=0 or 1; it calls lagr_box.pro for 2*

36) onfig.pro (program,ms,100%, 1)
This program is similar to other post-processing programs, except it aims to plot only chosen time steps.
It is also attempts to be flexible enough to meet needed demands for certain style and to present certain information.
Notes:
- This program set the parameters of plots and calls for panel_*.pro (see, e.g., panel_template.pro).
- Instead of panel, you can use anim_panel_*.pro

37) panel_templ.pro (include,ms,100%, 1)
This program is a template and will be modified 100 percent by user. This should be a list of plotting routines that can be combined on one post-processing panel.

38) print.pro (include,ms,10%, 3)
Printing subroutine for Sopale/Microfem/IDL post-processing
Notes:

- This subroutine looks for parameters "pr" and "landscape" and label "print:" in the plotting routines (standard "print:" label is hidden in "setting_up" subroutine)
  If printing is chosen, this subroutine sets-up PS devise and plots everything on the PS. Default files to print are idl0.ps through idl4.ps - they are rotating to avoid interference that may caused by slow print piping. You can choose your own name for postscript file using "s" (save) as main or second symbol in your response - in this case title of the file will appear on the plot (you can change that - go down 3-4 lines and check the line with "if ... 's'...")
- I would not recommend saving large sized (11x17) plots...
- This file has 100% chance to be edited by the system administer.

39) print_auto.pro (include,ms,10%, 3)
Check and arrange display:
  if printall not empty -> no display, direct to PS
  if printall is empty -> arrange display

40) PTt-ero (m)
This is directory including the following programs (for Microfem models). The routines of this subdirectory heavily rely on extended metamorphic and erosion related data, which available only for Microfem results.
data1
ero_distrib.pro
erosion.pro
erosion_front.pro
intersect.pro
lagr33.pro
lagr84.pro
meta_points.pro
meta_search.pro
metabox.pro
metamap.pro
post_metamap.pro
pttbox.pro
pttcolor.pro
pttsetup.pro
thermall.pro
timemarks.pro

41) radcolor.pro (include,ms,30%, 4)
Coloring of the materials according to their thermal type. It can be improved similar to matcol.pro if there will be an interest. So far it was not used for Sopale.

42) read.pro (include,ms,10%, 1)
Important: this program should be activated using command rnew:
IDL>.rnew read

This program read the data from User's Data Directory (or Microfem subdirectory 'idl') to save it in memory and use it for further plotting/calculations.

At the moment the program recognizes three types of data: Microfem (postprocessed by conv.pro); traditional output from Sopale; and "matlab" type of output from Sopale. The program automatically recognizes the type of date and reads it accordingly. Overall, it has a good level of automatism.

There is not much to change inside this compilation program. Two files can control read.pro:
session.pro - set directories to check for data
downloads.pro - set the parameters to read

43) read_description.pro (include,ms,2%, 2)
Reading file models (description file). This subroutine is rarely used because nobody has enough time to put few notes about data.

44) read_dialog.pro (include,ms,2%, 2)
Subroutine to chose the data for following post-processing

45) read_init_restart.pro (include,s,10%, 2)
This subroutine tries to find initial (closest to initial) data for lagragian coordinates
Notes:
- It is called from "read_sopal_lagr.pro"

46) read_m_data.pro (include,m,10%, 2)
Reads Microfem data.

47) read_m_erosion.pro (include,m, 10%, 2)
Program to extract parameters of erosion
Parameters:
  erostyle - integer:
    =1 (erosion proportional to topography)
    =2 (erosion proportional to uplift velocity)
    =3 (erosion proportional to slope)
  ero - integer: amount of history events
  erostages - arrey(ero): time boundaries between events
  erovvalues - arrey(ero): value of erosion coefficient on the event boundaries
  erospaces - integer: amount of erosion provinces
  erospace - arrey(erospaces): boundaries between provinces in km (S=0)
  erovvalues - arrey(erospaces): value of erosion weight on the province boundaries

Notes:
- Program works automatically (zip-independent, protection-independent)
- run in readmai.pro

48) read_m_file.pro (include,m,2%, 2)
Subroutine to prepare file to read: too many files to read in microfem - sometime we decide to add one to be downloaded
Parameters:
  hint - name of file
  path(ik) - directory of data, ended with ../idl

49) read_m_finish.pro (include,m,10%, 2)
Conversion to convenient units.

50) read_m_info.pro (include,m,10%, 2)
Reads general information of Micrrofem data (e.g. number of saves, size of arrays etc.)

51) read_m_mech_box.pro (include,m,10%, 2)
Subroutine to read info about mechanical boxes
Notes:
- Boxes (mec_boxes) are defined as in input files:
[x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)

- Outputs: mec_boxes_no - number of mechanical boxes
mec_colors(mec_boxes_no) - number of color to each box
mec_boxes(mec_boxes_no,8) - coordinates of boxes in km, adjusted to S-point
-these boxes trace only init lagr. position - no need to worry about them after inversion

52) read_m_therm_box.pro (include,m,10%, 2)
Subroutine to read info about therm boxes
Notes:
- Boxes (ther_boxes) are defined as in input files:
[x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)
- Outputs: ther_boxes_no - number of thermal boxes
ther_colors(ther_boxes_no) - number of color to each box
ther_boxes(ther_boxes_no,8) - coordinates of boxes in km, adjusted to S-point
-these boxes trace only init lagr. position - no need to worry about them after inversion

53) read_m_therm_propr.pro (include,m,10%, 2)
Reads thermal properties of materials involved in Microfem models

54) read_p_data.pro (include,s,20%, 2)
Reads Sopale "matlab"-type of outputs

55) read_s_depth.pro (include,s,10%, 2)
Program for manual assignment of depth for lagrangian points. In contrast with Microfem data, Sopale does not have depth of Lagrangian points as an output. The depth is roughly estimated here for each Lagr. point for each step saved.

56) read_s_eul.pro (include,s,10%, 2)
Reading from g01_... - eulerian data

57) read_s_finish.pro (include,s,10%, 2)
Assigning additional parameters, conversion to convenient units

58) read_s_info.pro (include,s,25%, 2)
Extracting the info from SOPALE1_i

59) read_s_lagr.pro (include,s,10%, 2)
Reading from g02_... - Lagrangian data

60) read_s_mech_box.pro (include,s,10%, 2)
Subroutine to read info about mechanical boxes from Sopale
Notes:
- Boxes (mec_boxes) are defined as in input files:
  [x1,y1,x2,y2,...] - anticlockwise (does not really matter here, which direction)
- Outputs: mec_boxes_no - number of mechanical boxes
  mec_colors(mec_boxes_no) - number of color to each box
  mec_boxes(mec_boxes_no,8) - coordinates of boxes in km, adjusted to S-point
-these boxes trace only init lagr. position - no need to worry about them after inversion

61) scale.pro (include,ms,80%, 4)
Subroutine to add scale to the bottom of the plot
Notes:
- This subroutine will work only if there is no horizontal axis on the last plot (xaxis_style=5)
- Two styles are implemented: 'line' and 'chess'.
  the last uncommented assignment to scale_type will work.
  If scale_type='line' there are two options: ticks='yes' or 'no':
  'yes': |____|____|____|

  'no': |_____________|

- There is no automatic assignment to length of the scale and length between ticks/chess boxes, it's up to user to define scale_length and scale_ticks, both in km.
- Scale length/height ratio is assigned to 20 in statement: scale_height=scale_length/20.
- you can change the ratio by replacing the numeric parameter

62) session.pro (include,ms,80%, 1 and 2)
Sets common data_directories; sets the paths for other parts of IDL_S-M; sets up color style. Please make sure that all directories with data are included here before you start IDL.

63) setting_up.pro (include,ms,30%, 3)
Setting the plot parameters, used also by print.pro (link to label "print:")

64) smoothing.pro (function,ms,10%, 2)
FUNCTION smoothing - smooths 1 and 2 D arrays.

65) suture.pro (include,m,20%, 4)
Tracing Lagrangian points that initially bound India-Asia
Notes:
Chris suggested to changed program name to lcol.pro or lcolumn.pro)

66) Tcontour.pro (include,s,60%, 1)
Temperature contours. This routine calls contour_mech.pro to plot/contour temperature

67) total_strain_e.pro (include,s,60%, 1)

Total strain contours based on eulerian data. This routine calls contour_mech.pro to plot/contour temperature

68) total_strain_l.pro (include,s,60%, 1)
Total strain contours based on Lagrangian data. This routine calls contour_mech.pro to plot/contour temperature

69) veloc.pro (include,s,60%, 1)
Set-up of velocity vectors and Plotting velocities. By default it plots total velocity vectors dynamically calculated in the program (see other veloc*.pro for other possibilities)

70) veloc_bc.pro (include,m,5%, 1)
Plots velocity that would be result if gravity would be set to 0.

71) veloc_g.pro (include,m,5%, 1)
gravity velocity

72) veloc_mantle.pro (include,m,5%, 1)
plot velocity in crust and mantle (below Moho)

73) velocity_scale.pro (include,ms,25%, 4)
Plot velocity scale
Notes:
- If you want this style of scale - put it right before "legend", if you do opposite - "legend" will take over
- This scale will appear only if you have velocity in your plot; no need to comment this subroutine if you don't plot velocities.

- The same as in "legend", the legend_position defines the position of the scale; but setting legend_position=0 does not suppress this feature (it suppress only "legend")
- You can use veloscale not equal 1 to present scale differently, e.g. when advancing subduction makes it 2.5 cm/y convergence from both sides, there is no need to present scale for 5 cm/y (which is total convergence rate): make veloscale=0.5 and you will get velocity scale for 2.5 cm

74) zebra.pro (include,ms,40%, 4)
Zebra-style lagrangian grid

75) zz_1x1.pro (include,ms,2%, 3)
Ono-to-one scaling. This routine recalculate vertical extend of plotting area to make one-to-one plot; then it changes one of the predefined vertical limits (yrang) to match the desired vertical extend.
The vertical limit is decided by parameter in the default.pro:
one2one_adjust='bottom' - bottom limit is stay (e.g. in thin-skinned models, where bottom is always 0)
one2one_adjust='top' - top limit stay (e.g. in Microfem models, or in deep Sopale models after ground_0.pro applied - you have good sense of what value for upper limit should be chosen)

6 REFERENCES

Fanning, D.W., 2002, Coyote' s guide to IDL Programming, http://www.dfanning.com

Fanning, D.W., 1998, IDL Programming Techniques, Fanning Software Consulting, 325 pp.

http://www.ecdf.org/gifcisle
http://www.dfanning.com/
http://www.informatik.fh-mannheim.de/idl/onlguide.pdf.
http://www.informatik.fh-mannheim.de/idl/refguide.pdf
http://www.informatik.fh-mannheim.de/idl/quickref.pdf
http://geodynamics.oceanography.dal.ca/bonny/docs/idl_scripts/idlrefguide.pdf.