Programming and Modelling (week 40)

C. Thieulot

Institute of Earth Sciences

September 2015

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)
4. write code piece by piece. Compile/test along the way.

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)
4. write code piece by piece. Compile/test along the way.
5. benchmark your code

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)
4. write code piece by piece. Compile/test along the way.
5. benchmark your code
6. fine tune, vary parameters, try end members, push your code

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)
4. write code piece by piece. Compile/test along the way.
5. benchmark your code
6. fine tune, vary parameters, try end members, push your code
7. produce relevant datas

Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)
4. write code piece by piece. Compile/test along the way.
5. benchmark your code
6. fine tune, vary parameters, try end members, push your code
7. produce relevant datas
8. filter/analyse/plot datas

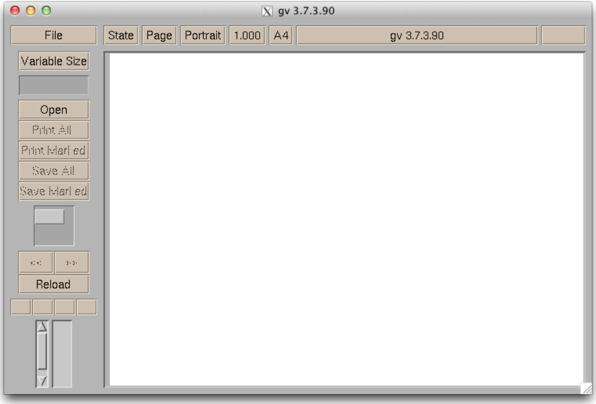
Philosophy

1. state the problem in mathematical terms
   (relevant variables, corresponding conservation equations)
2. think about data types and structures
   (integer, real, complex ? number, array ?)
3. think about output: 2D vs 3D, lines, histograms, ...
   $\rightarrow$ file format (which plotting software ?)
4. write code piece by piece. Compile/test along the way.
5. benchmark your code
6. fine tune, vary parameters, try end members, push your code
7. produce relevant datas
8. filter/analyse/plot datas
9. discuss figures/graphs

GhostView (1)

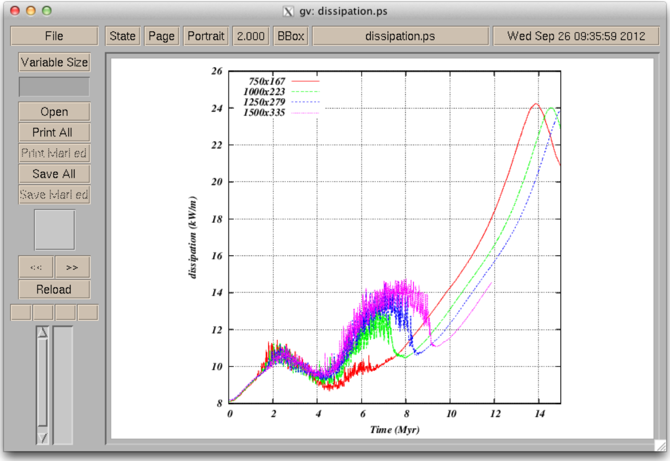
In the terminal:

> gv



GhostView (2)

```
> gv dissipation.eps
```



(press 'Q' to quit the application)

ImageMagick

Postscript (or encapsulated postscripts) isn't a format that text processors (Word, OpenOffice) accept. Here is how to convert plots to different formats:

```
> convert dissipation.ps dissipation.png
> convert dissipation.ps dissipation.jpg
> convert dissipation.ps dissipation.pdf
```

etc...

`convert` is part of ImageMagick , available for Windows, Mac, Linux and even iOS. (www.imagemagick.org)

`> man convert` → lists all options

Makefile (1)

- If your fortran program consists of a unique `.f90` file, compilation is done as follows:

```
> gfortran program.f90
```

Makefile (1)

- If your fortran program consists of a unique `.f90` file, compilation is done as follows:

```
> gfortran program.f90
```

- If your fortran program consists of many `.f90` file, compilation can be done as follows:

```
> gfortran compute_stuff.f90 solver.f90
output_data.f90 module_struc.f90 program.f90
```

Makefile (1)

- If your fortran program consists of a unique `.f90` file, compilation is done as follows:

```
> gfortran program.f90
```

- If your fortran program consists of many `.f90` file, compilation can be done as follows:

```
> gfortran compute_stuff.f90 solver.f90
output_data.f90 module_struc.f90 program.f90
```

What if the code comprises dozens or hundreds of fortran files ?

Makefile (2)

Example: the *elefant* code.

geogarfield — thieulot@gaia:~/ELEFANT/3Dbig/code — ssh — 190×58

```
[thieulot@gaia code]$ ls *.f90
advect_cloud2.f90  define_bc_213.f90  define_bcT_400.f90  material_layout_317.f90  read_exp_226.f90  read_materials_227.f90  stretch_201.f90
advect_cloud.f90  define_bc_214.f90  define_bcT.f90  material_layout_318.f90  read_exp_227.f90  read_materials_228.f90  stretch_205.f90
advect_landscape.f90  define_bc_215.f90  diffuse_surface.f90  material_layout_320.f90  read_exp_228.f90  read_materials_229.f90  stretch_210.f90
advect_lsurface.f90  define_bc_216.f90  diffusion_sp.f90  material_layout_321.f90  read_exp_229.f90  read_materials_230.f90  stretch_211.f90
advect_tracers.f90  define_bc_217.f90  directsolver.f90  material_layout_322.f90  read_exp_230.f90  read_materials_231.f90  stretch_212.f90
advect_vgrid.f90  define_bc_218.f90  elefant.f90  material_layout_400.f90  read_exp_231.f90  read_materials_232.f90  stretch_213.f90
analyse_run.f90  define_bc_219.f90  erosion_232.f90  material_layout_401.f90  read_exp_232.f90  read_materials_233.f90  stretch_215.f90
analytical_solution.f90  define_bc_220.f90  evolve_grid.f90  material_layout.f90  read_exp_233.f90  read_materials_234.f90  stretch_217.f90
bcvel.f90  define_bc_221.f90  find_cell.f90  matrix_profile.f90  read_exp_234.f90  read_materials_235.f90  stretch_218.f90
benchmark.f90  define_bc_222.f90  footer.f90  matrix_setup.f90  read_exp_235.f90  read_materials_300.f90  stretch_224.f90
boundary_cloud_228.f90  define_bc_223.f90  free_memory.f90  modify_grid.f90  read_exp_300.f90  read_materials_301.f90  stretch_228.f90
boundary_cloud_232.f90  define_bc_224.f90  grid_setup.f90  module_structures.f90  read_exp_301.f90  read_materials_302.f90  stretch_232.f90
boundary_cloud.f90  define_bc_225.f90  header.f90  nu_layout_211.f90  read_exp_302.f90  read_materials_303.f90  stretch_301.f90
cailloux_setup.f90  define_bc_226.f90  initialise_mumps_P.f90  nu_layout_232.f90  read_exp_303.f90  read_materials_305.f90  stretch_302.f90
check_cloud.f90  define_bc_227.f90  initialise_mumps_T.f90  nu_layout.f90  read_exp_304.f90  read_materials_306.f90  stretch_303.f90
clean.f90  define_bc_228.f90  initialise_mumps_V.f90  open_files.f90  read_exp_305.f90  read_materials_307.f90  stretch_305.f90
close_files.f90  define_bc_229.f90  interpolate_vel_on_pt2D.f90  output_ascii.f90  read_exp_306.f90  read_materials_315.f90  stretch_306.f90
cloud_setup2.f90  define_bc_230.f90  int_to_char.f90  output_bin.f90  read_exp_307.f90  read_materials_316.f90  stretch_307.f90
cloud_setup.f90  define_bc_231.f90  landscape_setup.f90  output_mats.f90  read_exp_315.f90  read_materials_317.f90  stretch_315.f90
cloud_setup_old.f90  define_bc_232.f90  lsurface_setup.f90  paint_stripes.f90  read_exp_316.f90  read_materials_318.f90  stretch_316.f90
compute_cloudmax.f90  define_bc_233.f90  make_matrix.f90  phase_change_232.f90  read_exp_317.f90  read_materials_320.f90  stretch_317.f90
compute_convergence.f90  define_bc_234.f90  make_matrixT.f90  pslib.f90  read_exp_318.f90  read_materials_321.f90  stretch_318.f90
compute_elemental_values.f90  define_bc_235.f90  material_layout_200.f90  psplot_cloud.f90  read_exp_320.f90  read_materials_322.f90  stretch_400.f90
compute_element_centers.f90  define_bc_300.f90  material_layout_202.f90  psplot_cloud_zoom.f90  read_exp_321.f90  read_materials_400.f90  stretch_401.f90
compute_elsize.f90  define_bc_301.f90  material_layout_203.f90  psplot_grid2D.f90  read_exp_322.f90  read_materials_401.f90  stretch.f90
compute_fluxes.f90  define_bc_302.f90  material_layout_204.f90  psplot_grid2D_zoom.f90  read_exp_400.f90  read_materials.f90  subtiming.f90
compute_heatflux.f90  define_bc_303.f90  material_layout_210.f90  psplot_grid_sandbox.f90  read_exp_401.f90  read_n_compute_parameters.f90  temperature_layout_208.f90
compute_hydr_pressure.f90  define_bc_304.f90  material_layout_211.f90  qshep2d.f90  read_materials_200.f90  renumber_nodes.f90  temperature_layout_209.f90
compute_pressure.f90  define_bc_305.f90  material_layout_212.f90  read_exp_200.f90  read_materials_201.f90  scan_for_arguments.f90  temperature_layout_212.f90
compute_pressure_fem.f90  define_bc_306.f90  material_layout_213.f90  read_exp_201.f90  read_materials_202.f90  set_default_values.f90  temperature_layout_213.f90
compute_qcoords.f90  define_bc_307.f90  material_layout_214.f90  read_exp_202.f90  read_materials_203.f90  smooth_pressure.f90  temperature_layout_225.f90
compute_stress_profile.f90  define_bc_315.f90  material_layout_215.f90  read_exp_203.f90  read_materials_204.f90  SolCx.f90  temperature_layout_232.f90
compute_tensors.f90  define_bc_316.f90  material_layout_216.f90  read_exp_204.f90  read_materials_205.f90  SolKz.f90  temperature_layout_235.f90
compute_timestep.f90  define_bc_317.f90  material_layout_217.f90  read_exp_205.f90  read_materials_206.f90  solve.f90  temperature_layout_303.f90
compute_Trms.f90  define_bc_318.f90  material_layout_218.f90  read_exp_206.f90  read_materials_207.f90  solveP.f90  temperature_layout_305.f90
compute_Trms_old.f90  define_bc_320.f90  material_layout_219.f90  read_exp_207.f90  read_materials_208.f90  solvePfem.f90  temperature_layout_306.f90
compute_viscous_dissipation.f90  define_bc_321.f90  material_layout_220.f90  read_exp_208.f90  read_materials_209.f90  solveT.f90  temperature_layout.f90
compute_volume.f90  define_bc_322.f90  material_layout_221.f90  read_exp_209.f90  read_materials_210.f90  spacer.f90  template.f90
compute_vrms.f90  define_bc_400.f90  material_layout_222.f90  read_exp_210.f90  read_materials_211.f90  spas.f90  time_nl.f90
compute_xyzq.f90  define_bc_401.f90  material_layout_223.f90  read_exp_211.f90  read_materials_212.f90  spline.f90  timevisu_setup.f90
cshep2.f90  define_bc.f90  material_layout_224.f90  read_exp_212.f90  read_materials_213.f90  strain_history_203.f90  tracers_setup_212.f90
define_bc_200.f90  define_bcT_206.f90  material_layout_226.f90  read_exp_213.f90  read_materials_214.f90  strain_history_210.f90  tracers_setup_216.f90
define_bc_201.f90  define_bcT_207.f90  material_layout_228.f90  read_exp_214.f90  read_materials_215.f90  strain_history_211.f90  tracers_setup.f90
define_bc_202.f90  define_bcT_208.f90  material_layout_229.f90  read_exp_215.f90  read_materials_216.f90  strain_history_214.f90  trim_cloud.f90
define_bc_203.f90  define_bcT_209.f90  material_layout_232.f90  read_exp_216.f90  read_materials_217.f90  strain_history_215.f90  trim_landscape.f90
define_bc_204.f90  define_bcT_210.f90  material_layout_233.f90  read_exp_217.f90  read_materials_218.f90  strain_history_301.f90  tshep2.f90
define_bc_205.f90  define_bcT_211.f90  material_layout_234.f90  read_exp_218.f90  read_materials_219.f90  strain_history_302.f90  update_cloud.f90
define_bc_206.f90  define_bcT_213.f90  material_layout_302.f90  read_exp_219.f90  read_materials_220.f90  strain_history_303.f90  vgrid_setup.f90
define_bc_207.f90  define_bcT_225.f90  material_layout_303.f90  read_exp_220.f90  read_materials_221.f90  strain_history_306.f90  write_parameters.f90
define_bc_208.f90  define_bcT_232.f90  material_layout_304.f90  read_exp_221.f90  read_materials_222.f90  strain_history_315.f90
define_bc_209.f90  define_bcT_235.f90  material_layout_305.f90  read_exp_222.f90  read_materials_223.f90  strain_history_316.f90
define_bc_210.f90  define_bcT_303.f90  material_layout_306.f90  read_exp_223.f90  read_materials_224.f90  strain_history.f90
define_bc_211.f90  define_bcT_305.f90  material_layout_315.f90  read_exp_224.f90  read_materials_225.f90  strainweaken.f90
define_bc_212.f90  define_bcT_306.f90  material_layout_316.f90  read_exp_225.f90  read_materials_226.f90  stretch_200.f90
[thieulot@gaia code]$
```

(373 fortran files, 40,000 lines of code)

Makefile (3)

Example: the gravity modelling exercise for this week
The whole programs comprises the following fortran files:

```
write_two_columns.f90
write_three_columns.f90
program.f90
```

Makefile (3)

Example: the gravity modelling exercise for this week
The whole programs comprises the following fortran files:

```
write_two_columns.f90
write_three_columns.f90
program.f90
```

The subroutines are given to you, but you have to write the main program.

Makefile (4)

Compiling all the routines and assembling them all into the executable `grav`:

```
> gfortran write_two_columns.f90
write_three_columns.f90 program.f90 -o grav
```

Makefile (4)

Compiling all the routines and assembling them all into the executable `grav`:

```
> gfortran write_two_columns.f90
write_three_columns.f90 program.f90 -o grav
```

- not practical

Makefile (4)

Compiling all the routines and assembling them all into the executable `grav`:

```
> gfortran write_two_columns.f90
write_three_columns.f90 program.f90 -o grav
```

- not practical
- if you modify one file, this approach still requires you to recompile all fortran files !

Makefile (5)

We then create the following file: `Makefile` (See appendix C)

```
.SUFFIXES:.out .o .s .c .F .f .f90 .e .r .y .yr .ye .l .p .sh .csh .h

default: code

F90      = gfortran

FLAGS= -c -ffree-line-length-none

OPTIONS =

INCLUDE =

OBJECTS = \
write_two_columns.o\
write_three_columns.o\
program.o

.f90.o:
	$(F90) $(FLAGS) $(INCLUDE) $*.f90
.f.o:
	$(F90) $(FLAGS) $(INCLUDE) $*.f

code:	$(OBJECTS)
	$(F90) $(OPTIONS) $(OBJECTS)  -o grav
```

Makefile (5)

We then create the following file: `Makefile` (See appendix C)

```
.SUFFIXES:.out .o .s .c .F .f .f90 .e .r .y .yr .ye .l .p .sh .csh .h

default: code

F90      = gfortran

FLAGS= -c -ffree-line-length-none

OPTIONS =

INCLUDE =

OBJECTS = \
write_two_columns.o\
write_three_columns.o\
program.o

.f90.o:
	$(F90) $(FLAGS) $(INCLUDE) $*.f90
.f.o:
	$(F90) $(FLAGS) $(INCLUDE) $*.f

code:	$(OBJECTS)
	$(F90) $(OPTIONS) $(OBJECTS)  -o grav
```

To compile the whole code:

`> make`

Wrapping things up: Key concepts

- data types (integer, real, character, ...)
- data structures (numbers, static arrays, allocatable arrays)
- if then else
- do loop
- subroutines and functions
- intrinsic functions
- modules, formats
- open/close file
- compile vs run
- makefile
- plotting (xmgrace, gnuplot)
- shell commands (ls, cd, pwd, ...)

Things you HAVE to know (for the exam)

How to declare

- an integer, a real
- an array (static, or allocatable)

How to write

- a program
- a subroutine, a function
- a do-loop
- an if-then-else statement

How to

- open a file
- write in a file
- close a file
- call a subroutine/function
- pass an array as argument

salt tectonics(1)

Marine and Petroleum Geology 18 (2001) 779–797

Marine and Petroleum Geology

www.elsevier.com/locate/marpetgeo

Salt diapirs in the Dead Sea basin and their relationship to Quaternary extensional tectonics

Abdallah Al-Zoubi[a,1], Uri S. ten Brink[b,*]

[a]*Woods Hole Oceanographic Institution, Woods Hole, MA 02543, USA*
[b]*US Geological survey, Woods Hole Field Center, 384 Woods Hole Road, Woods Hole, MA 02543, USA*

Received 21 March 2001; received in revised form 23 May 2001; accepted 23 May 2001

salt tectonics(2)

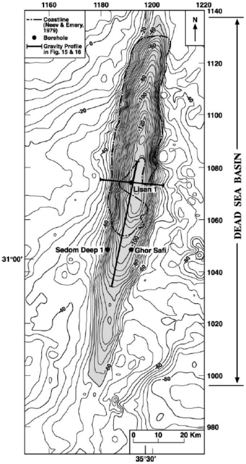
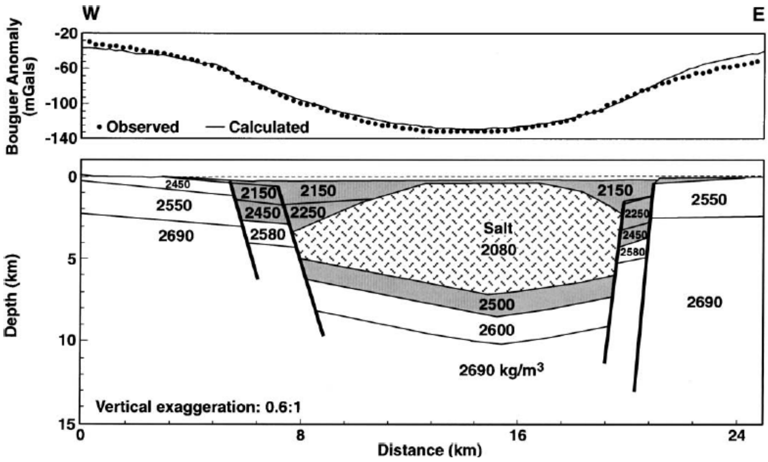


Fig. 14. Bouguer gravity anomaly map of the study area. Contour interval is 3 mGal. Location of the gravity profiles in Figs. 15 and 16. Dashed-dotted line — 1967 coastline of the Dead Sea (after Neev & Hall, 1979). Black dots — location of wells.

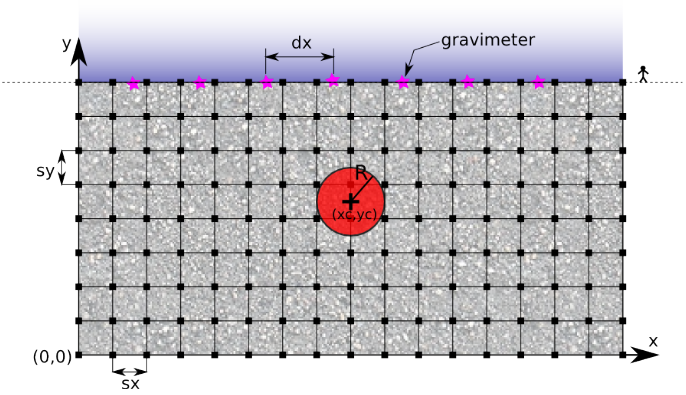
salt tectonics(3)

A. Al-Zoubi, U.S. ten Brink / Marine and Petroleum Geology 18 (2001) 779–797



Grav (1)

The modelling program `grav` computes the gravity anomaly at the Earth's surface of a number of spherical density anomalies in the subsurface.



Grav (2)

A key idea in numerical modelling: **benchmarking**

Grav (2)

A key idea in numerical modelling: **benchmarking**

- Your code runs and produces beautiful, tangible results 

Grav (2)

A key idea in numerical modelling: **benchmarking**

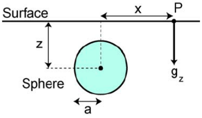
- Your code runs and produces beautiful, tangible results 
- How do you know that you haven't forgotten a minus sign somewhere ? a factor 2 ?

→ You can run your code on typical experiments/problems to which we know an analytical solution
→ You can run your code and a commercial/mature code on the same problem and compare results
→ You can run your code on a problem and compare its results with real life experimental results

Grav (3) - benchmarking the program

A sphere has the same gravitational pull as a point mass located at its centre: it allows us to calculate its gravitational pull.
Simple mathematics (See Turcotte and Schubert) can be used to show that at Point P, the vertical component of g is given by

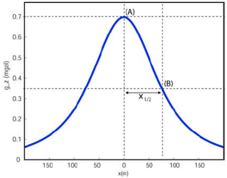
$$g_z = \frac{G M_S z}{(x^2 + z^2)^{\frac{3}{2}}}$$

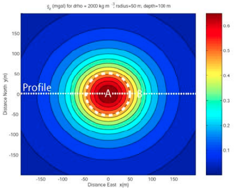


Suppose:

Radius, a	= 50 m	Depth, z	= 100 m
Density contrast , Δρ	= 2000 kg m^{-3}	Excess mass, M_S	= 10^9 kg

The variation in g_z can be plotted on a profile and map





Grav (4) - benchmarking the program