Earth Science Applications of Space Based Geodesy

DES-7355

Tu-Th 9:40-11:05

Seminar Room in 3892 Central Ave. (Long building)

Bob Smalley
Office: 3892 Central Ave, Room 103
678-4929
Office Hours – Wed 14:00-16:00 or if I'm in my office.

http://www.ceri.memphis.edu/people/smalley/ESCI7355/ESCI_7355_Applications_of_Space_Based_Geodesy.html

Class 22

DEFNODE

DEFNODE is a Fortran program to model elastic lithospheric block rotations and strains, and locking or coseismic slip on block-bounding faults.

Quote of the day.

I make no guarantees whatsoever that this program will do what you want it to do or what you think it is doing.

The program can solve for

- interseismic plate locking or coseismic slip distribution on faults,
- block (plate) angular velocities,
- uniform strain rates within blocks, and
- rotation of GPS velocity solutions relative to reference frame.

Data to constrain the models include

- GPS vectors,
- surface uplifts,
- earthquake slip vectors,
- spreading rates,
- rotation rates,
- fault slip rates,
- transform azimuths,
- surface strain rates, and
- surface tilt rates.

RUNNING:

```
% defnode
```

the program will ask for the control file name and the model name.

Enter the control file name as a command line argument:

```
% defnode control_file_name
```

Enter model name as second command line argument:

```
% defnode control_file_name model_name
```

Runtime messages are all output to the screen. Many files are generated.

Directories:

All output will be put into a directory specified by the MO: (model) command.

The program also produces a directory called 'gfs' (or a user-assigned directory) to store the Green's function files.

Poles (angular velocities) and blocks:

You can specify many poles and many blocks (dimensioned with MAX_poles, MAX_blocks).

There is NOT a one-one correspondence between poles and blocks.

More than one block can be assigned the same pole (ie, the blocks rotate together) but each block can be assigned only one pole.

Poles can be specified as (lat,lon,omega) or by their Cartesian components (*Wx, Wy, Wz*).

Strain rates and blocks:

The strain rate tensors (SRT) for the blocks are input in a similar way as the rotation poles.

Each SRT is assigned an index (integer) and blocks are assigned a SRT index.

As with poles, more than one block can be assigned to a single SRT.

Velocities are estimated from the SRT using the block's centroid as origin (default) or a user-assigned origin; if multiple blocks use the same SRT assign an origin for this SRT (see ST: option)

Faults and blocks:

Faults along which backslip is applied are specified and must coincide point-for-point at the surface with block boundary polygons.

However, not all sections of block boundaries have to be specified as a fault.

If the boundary is not specified as a fault it is treated as free-slipping and will not produce any elastic strain (ie, there will be a step in velocity across the boundary).

By specifying no faults, you can solve for the block rotations alone.

Fault nodes:

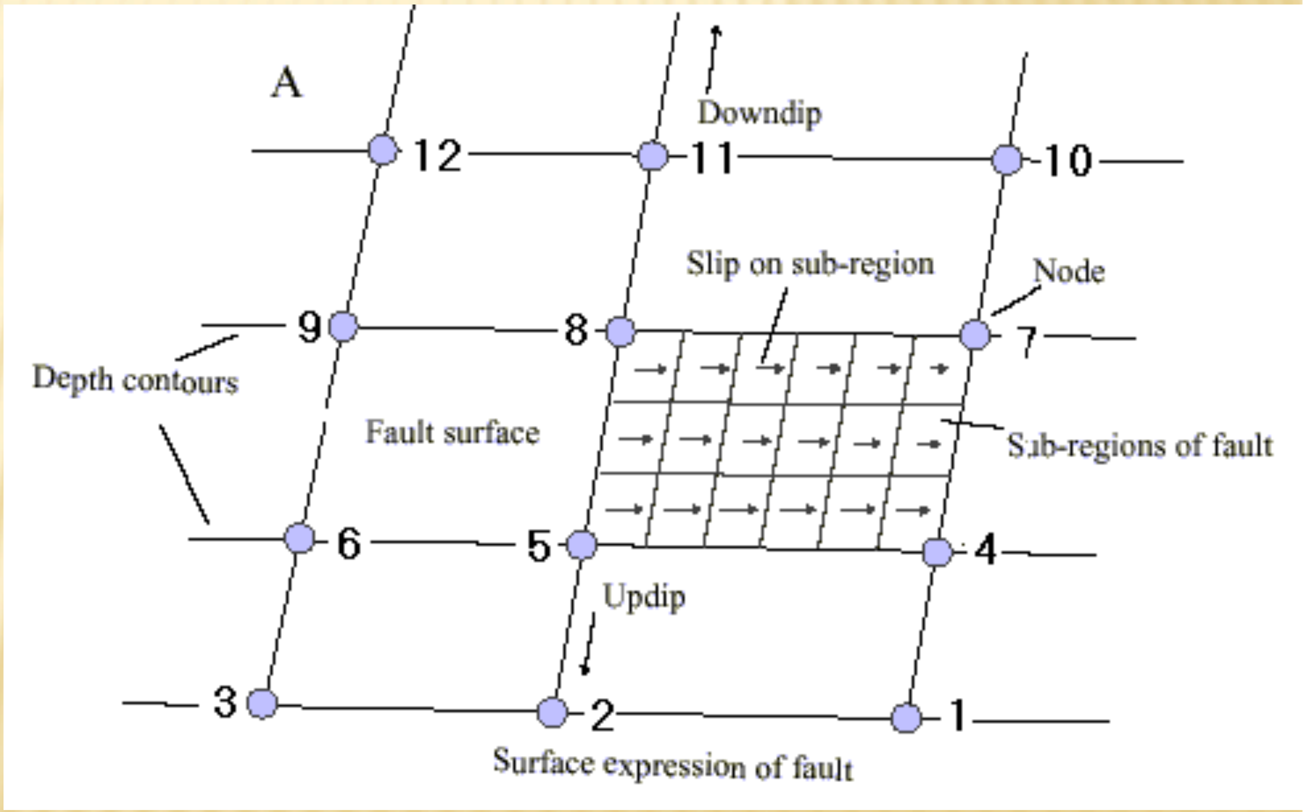
Fault surfaces are specified in 3 dimensions by nodes which are given by their long and lat (in degrees) and depth (in km, positive down).

Nodes are placed along depth contours of the faults and each depth contour has the same number of nodes.

Nodes are numbered in order first along strike, then down dip.

Fault nodes:
Strike is the direction faced if the fault dips off to your right.
Faults cannot be exactly vertical (90o dip) as the hangingwall and footwall blocks must be defined.

The fault geometry at depth can be built either by specifying all the node coordinates individually or by using the DD: and ZD: options.



Green's functions:

If you are performing an inversion, the program uses unit response (Green's) functions (GFs) for the elastic deformation part of the problem since the inversion method (downhill simplex) has to calculate numerous forward models.

Once you have calculated GFs for a particular set of faults you can use these in inversions without recalculating them (see option GD:).

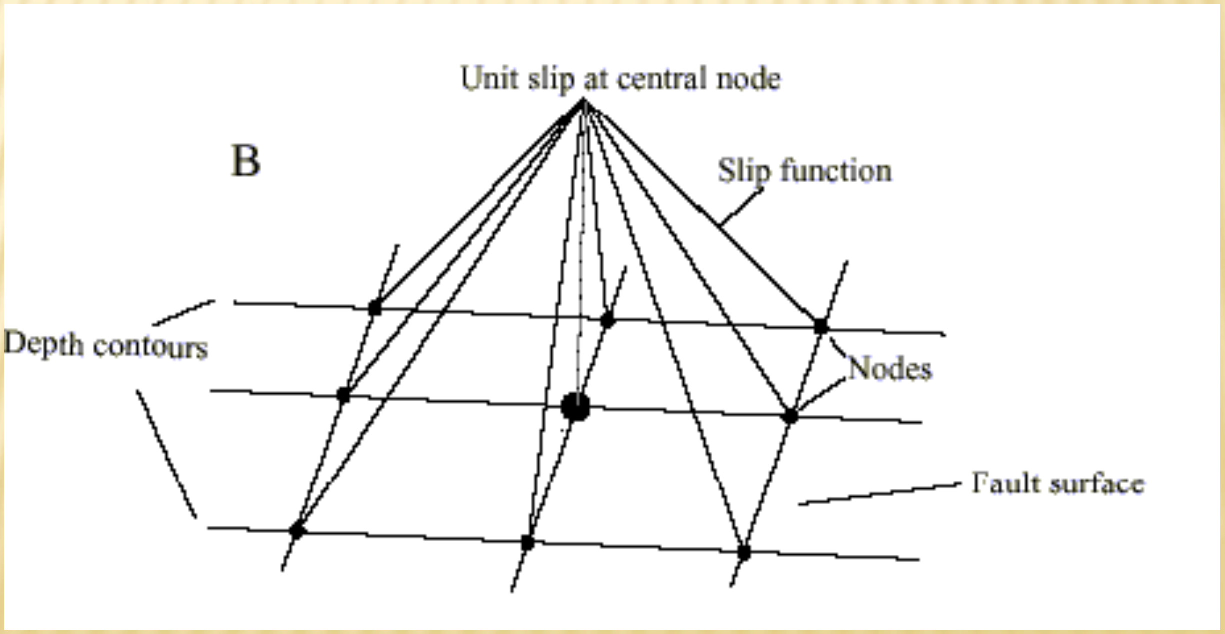
The GFs are based on the node geometry, GPS data, uplift data, strain tensor data, and tilt rate data so if you change the node positions or ADD data, you need to re-calculate GFs.

If you REMOVE data, you do not need to recalculate GFs.

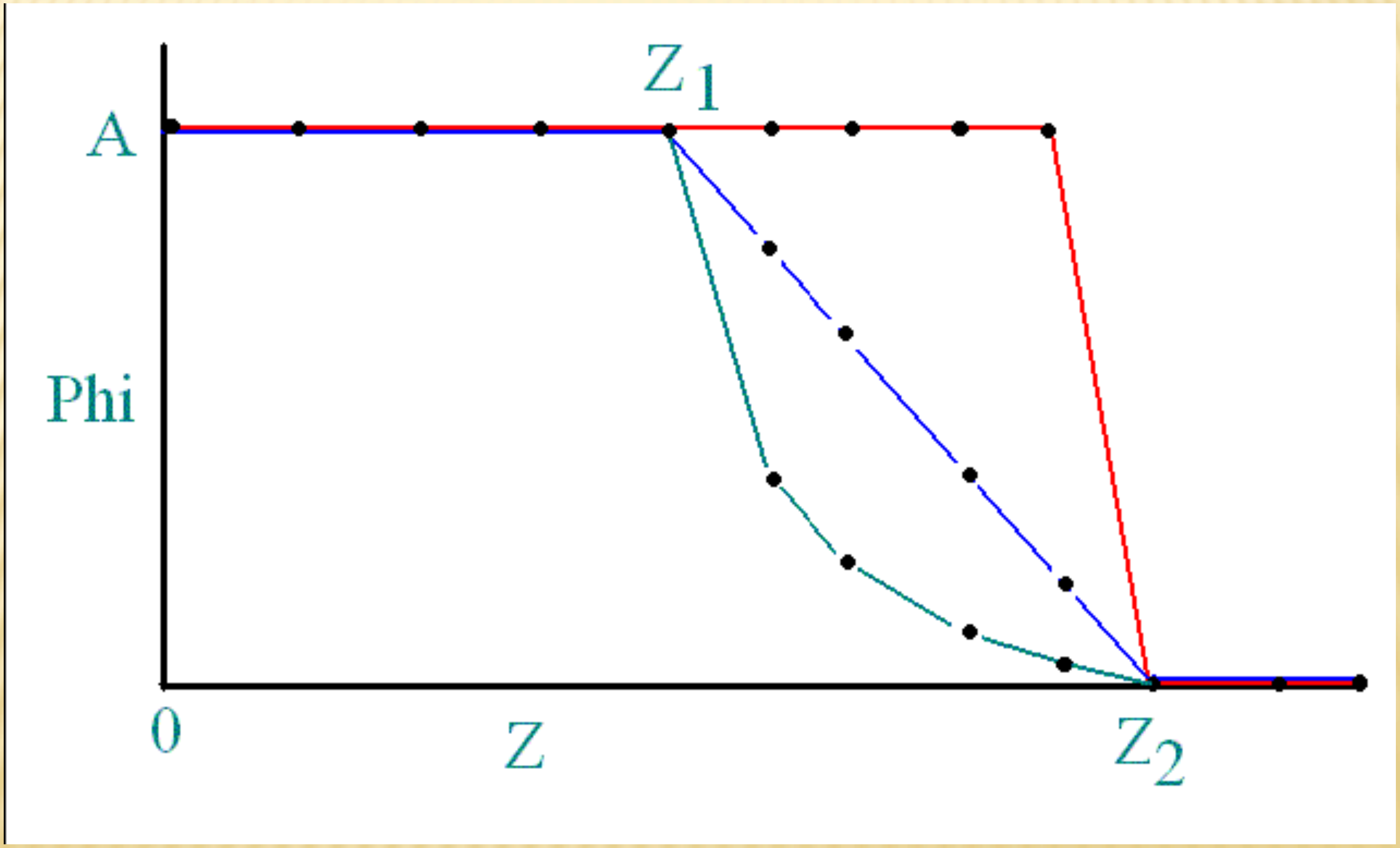
If you add GPS vectors, the program will not detect that and GFs may not be calculated. In this case, re-calculate all the GFs.

The GFs are the responses at the surface observation points to a unit velocity (or displacement) in the North and East directions at the central node.

The slip velocity is tapered to zero at all adjacent nodes.



Specifying how slip dies off.



CONTROL FILE

The program reads the model and all controls from an control file.

Lines in the control file comprise a keyword section and a data section.

CONTROL FILE

The keyword section starts with a 2-character keyword (in the first 2 columns) and ends with a colon (:).

Normally only the first 2 characters of the keyword are used so in general any characters between the 3rd character and the : are ignored. (Sometimes the third character specifies a format.)

THE KEYWORD MUST START IN THE FIRST COLUMN OR THE LINE IS IGNORED.

Case does not matter.

The data section of the input line goes from the colon to the end of the line and its contents depend on the keyword.

In a few cases the data section comprises multiple lines (i.e., always BL: and FA:, and sometimes others).

For example, the key characters for a fault are 'FA' and this has two arguments, the fault name and the fault number, so the following lines are correct:

fa: JavaTrench 1

fault: JT 1

fault (Java trench) : JavaTr 1

FA: JT 1

It is advisable and good practice to start comment lines with a space, *, # or some other character outside the range A - Z (the program has many undocumented options and you may trigger one by accident).

```
smalley-14:costa_rica_example robertsmalley$ more cr.dfn
# Costa Rica example

# flag to set random number seed to 1 to reproduce test case
#  delete from real runs
fl: +rs1

## name the model
Model: crc1

## where to store model parameters
pf: "crc1/pio" 3

## green's function controls
gd: g1d 4 2 0 1.0 1.0 2000.

em:

# data from Lundgren et al. 1997, downweight it
gps data: LUND "lundgren_1997.vec"  1  2  0 0 0

# data from Norabuena et al. 2003
gps data: NORA "norabuena_2003.vec" 2  1  0 0 0

## rotate LUND data into NORA's Carib ref frame
gi: 1  2

# uplift rates from same
uplift data: "lundgren_1997.upz" 1
```

```
# slip vector data from quakes
sv data: costa_rica.svs FORE  COCO  10

## simulated annealing controls
sa: 0  40 0.0 1.0

## grid search controls
gs: 75  0.1  4 2

## run through inversion twice
ni: 2

## set flags: set downdip constraint, estimate parameter uncertainties, do forward run at end
flag: +ddc +cov +for

## solve for pole 3, the forearc
pi: 3

## interpolate faults with 4km x 4 km grid at end
in: 4 4

## CARI is reference frame
re: CARI

## starting poles
pole COCO-CARI:  2  21.9  -123.1  1.26
pole FORE-CARI:  3 9.0 273.8  1.55

# profiles to calculate
pr: 1 273.5  10.0 100 .03 45 30
pr: 2 274.09 9.36 100 .03 40 30
```

```
### Blocks ###

block: CARI 1
 9999
 271.1989  13.3500
 273.2309  16.0077
 280.5192  10.5942
 278.6573 8.1589
 276.4709 9.7829
 274.8961  10.7273
 272.6565  12.2669
 271.1989  13.3496
 9999 9999

block: FORE 3
9999
 271.1989 13.3496
.
.
.
```

Continue till all blocks defined

```
275.0271  3.4103
 9999 9999

### Faults ###

ft: 1 1

Fault:  MidAmTr 1
 7  5  FORE COCO  1  0 0
 3.00
  275.5262 8.5473
  274.5473 9.0604
  274.2448 9.2626
  273.9441 9.4521
  273.6727 9.7100
  273.4306 9.9912
  272.7812 10.9354
 12.3
  275.7426 8.8303
  273.6500 12.0719
```

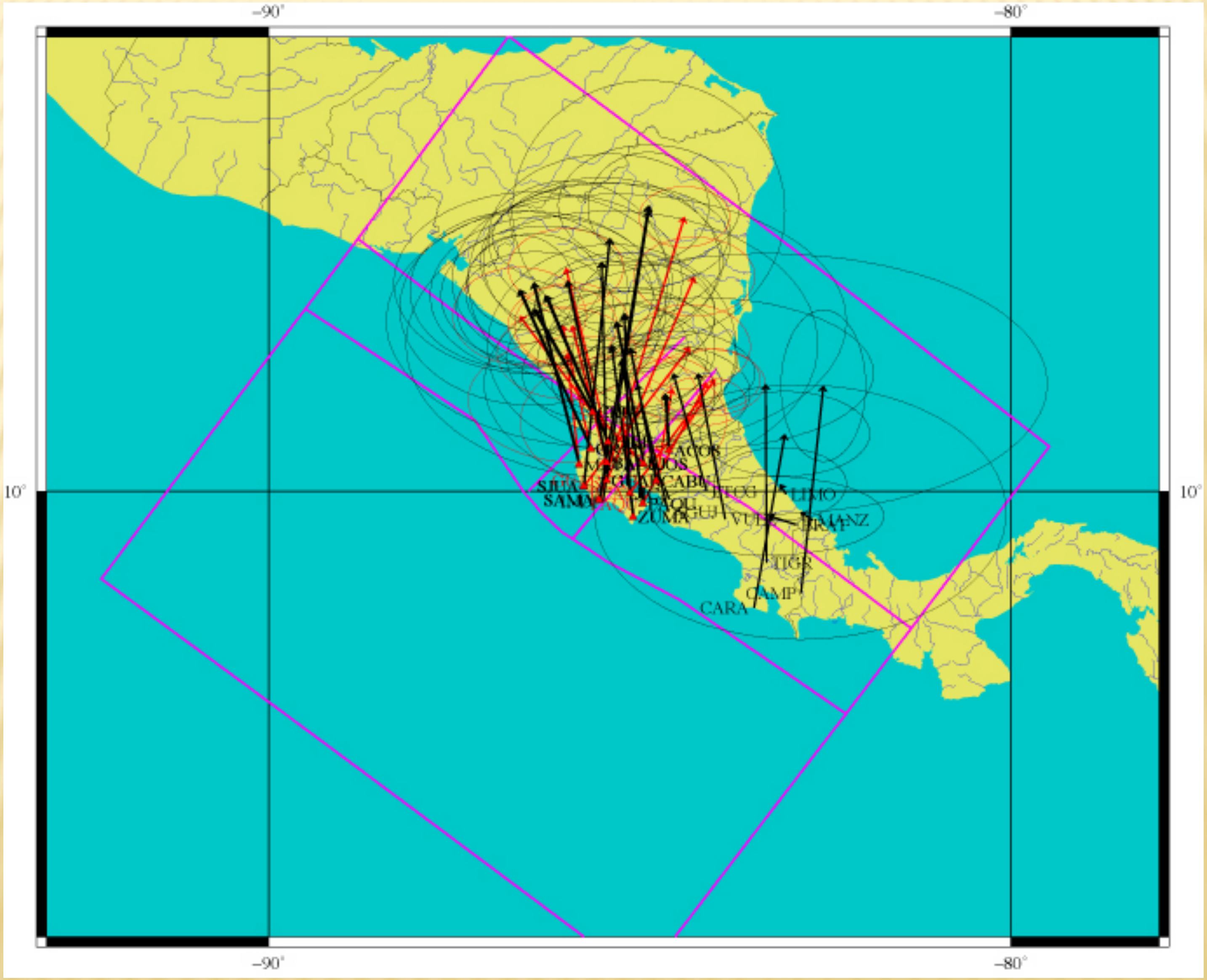
5 sections of 7 segments

```
## node indices for fault 1
NNg: 1  7  5
 1  1  1  1  1  1  1
 2  3  4  5  6  7  8
 9 10 11 12 13 14 15
16 17 18 19 20 21 22
 0  0  0  0  0  0  0

## starting phi values corresponding to fault 1 node indices
NV: 1  1.0  .9  .9  .9  .4 .4 .4  .1 .1 .1

## near vertical fault along arc
fault: Arc_SS 2
4 2 CARI FORE 1 0
0.0
  278.6573 8.1589
  276.4709 9.7829
  274.8961  10.7273
  272.6565  12.2670
zd: 15.0 88.0

## node indices for fault 2
## this fault will have uniform phi at all nodes
nn: 2  1 1 1 1 1 1 1 1

## starting phi values corresponding to fault 2 node indices
nv: 2  1.0


end:
```

After running get a directory full of output.

```
rsmalley-14:crc1 robertsmalley$ ls
crc1.fault_detail  crc1.poles     crc1.summary      crc1_blk3.gmt             crc1_model.input          crc1_sa.out
crc1.moment        crc1.res       crc1.svs          crc1_blocks.out           crc1_p01.out              loc2_dn.tmp
crc1.net           crc1.rot       crc1.ups          crc1_control.backup       crc1_p02.out              loc3_dn.tmp
crc1.nod           crc1.slp       crc1.vec          crc1_flt_atr.gmt          crc1_parameter.tmp        loc_dn.tmp
crc1.obs           crc1.str       crc1_blk.gmt      crc1_lin.gmt              crc1_pio.tmp              pio
crc1.omr           crc1.strain    crc1_blk2.gmt     crc1_mid.vec              crc1_removed.vec
```

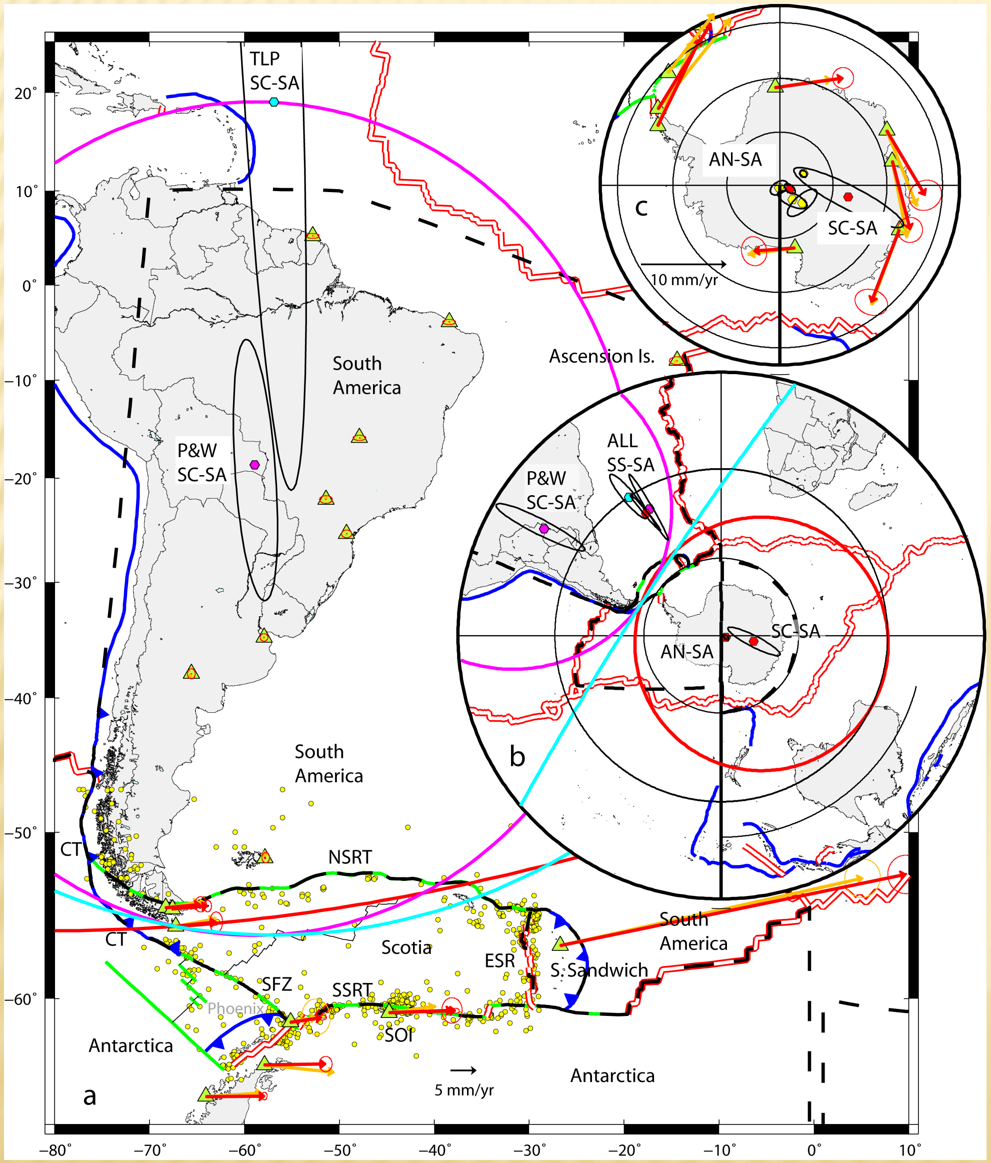
Red – measurements Black – model.



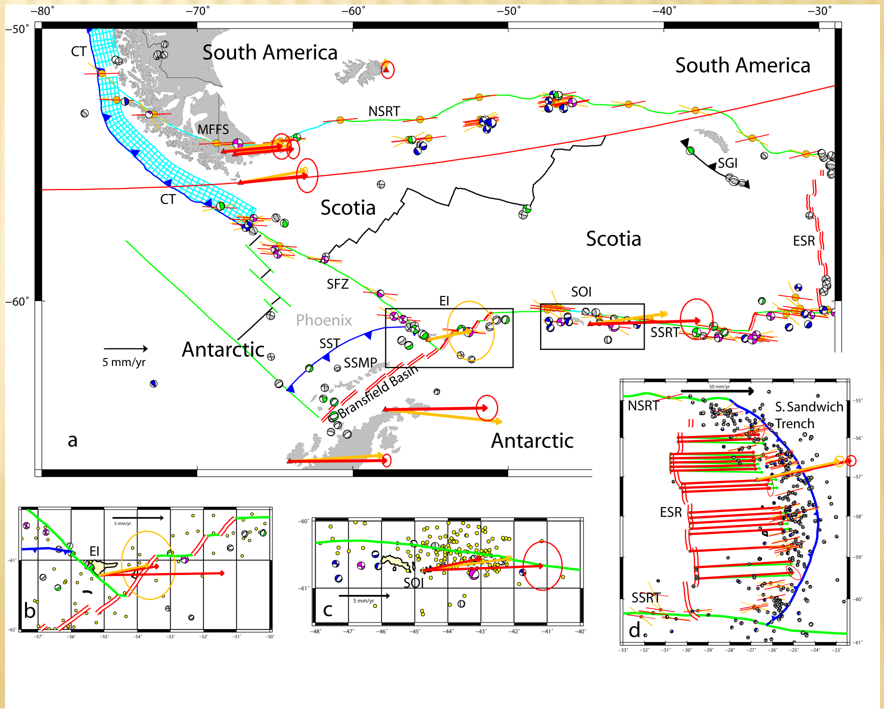
Example II

Combine GPS and geologic data to estimate Euler pole for Scotia plate.

Results for GPS-Geologic combination for Scotia Arc.



Use Combination of GPS (velocity and azimuth, focal mechanisms (azimuth), Scotia-South Sandwich spreading.



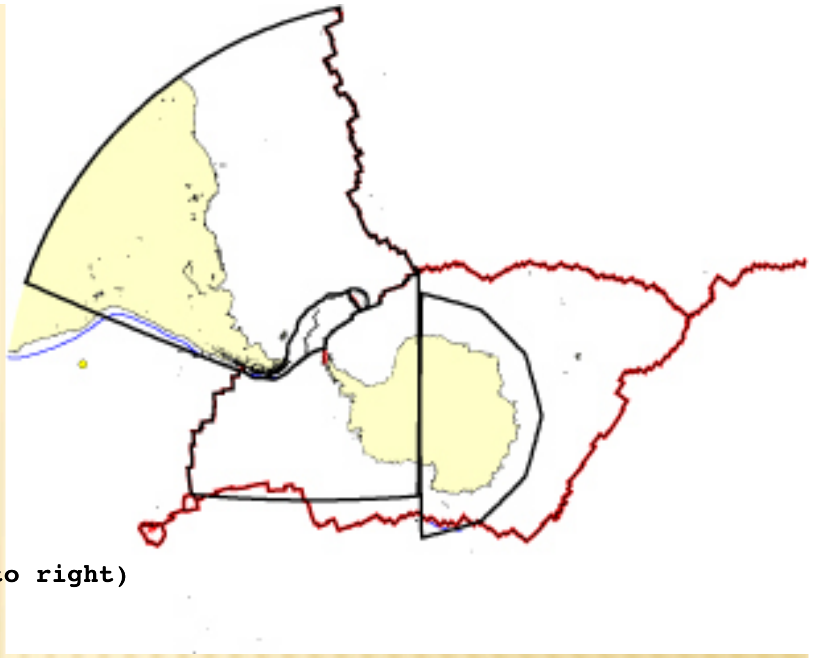
Smalley et al., 2007

First – write a bunch of programs to make the input files.

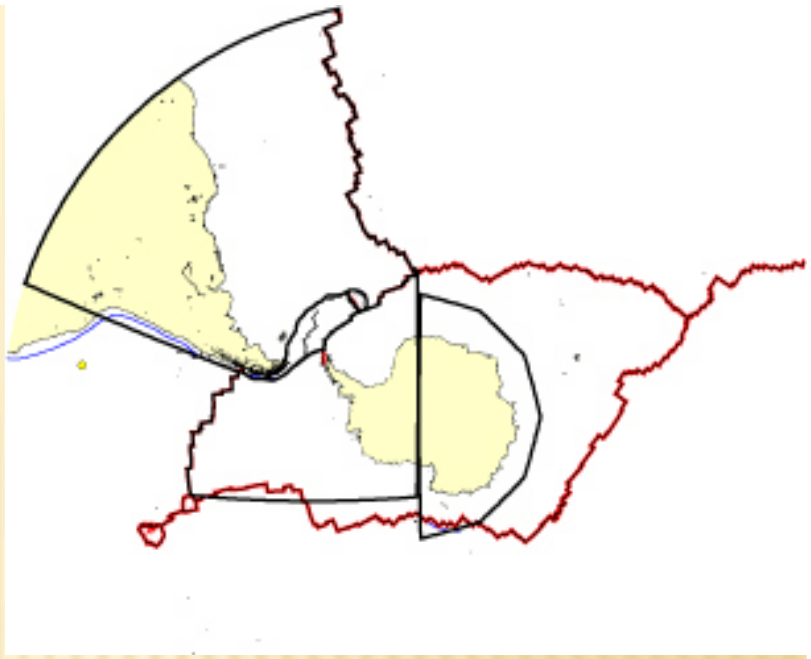
Erase old files.
Set up environment variables.
Make blocks.

```
rsmalley-14:defnode_stuff robertsmalley$ more scot.sh
#!/bin/sh

EXP=scot
DEFNINFILE=$EXP.dfn
#erase stuff in output directories
#picks up greens functions in gsc and uses them - may be wrong, from previous different run, etc.
\rm -r $EXP
\rm -r gsc
\rm -r $EXP.vec
touch $EXP.vec
\rm -r ${EXP}_no_rescerr.vec
touch ${EXP}_no_rescerr.vec

DATA=/gaia/home/rsmalley/defnode_stuff

#make block files from plate boundary data
#use breakitup to break up scot.pb.gmt file from UT into individual segments
#use anatwblock.sh, samblock.sh, scotblock.sh, ssandblock.sh to make the block files
#based on texas plate boundaries -- but too detailed and high freq -- making of
#blocks also filters and decimates and produces file that can be used to generate
#faults for defnode pulls out appropriate sections and puts in *.dfn

antwblock.sh
samblock.sh
scotblock.sh
ssandblock.sh
```

Blocks have to be closed polygons whose sides are traversed in order. You may have to piece them from pre-existing data files.



```
rsmalley-14:defnode_stuff robertsmalley$ more antwblock.sh
#!/bin/sh
#goes cw around west antarctica block (antarctica - hanging wall - to right)
OF=newantw.block
\rm $OF
touch $OF
#remove first line (file id) from all files
echo 9999 >> $OF
sed '1,1d' scot.pb.gmt.03 | smoothbound 5 | nawk '{print $0, NR}' >> $OF

#also remove second line (first point) from 2nd through end file to not duplicate points
sed '1,2d' scot.pb.gmt.04 | smoothbound 15 | nawk '{print $0, NR}' >> $OF
sed '1,2d' scot.pb.gmt.11 | nawk '{print $0, NR}' >> $OF
sed '1,2d' scot.pb.gmt.12 | nawk '{print $0, NR}' >> $OF

#remove header line, reverse it, then delete new first point (or remove last pt
#before reversal)
sed '1,1d' scot.pb.gmt.08.orig | sed '1!G;h;$!d' | sed '1,1d' | nawk '{print $0, NR}' >> $OF
sed '1,1d' scot.pb.gmt.07 | sed '1!G;h;$!d' | sed '1,1d' | nawk '{print $0, NR}' >> $OF

#also remove second line (first point) from 2nd through end file to not duplicate points
sed '1,2d' scot.pb.gmt.09 | nawk '{print $0, NR}' >> $OF

cat samant.pb.gmt >> $OF
cat antsplit.pb.gmt >> $OF

#delete last 3 lines of chile ridge file
sed '$d' /gaia/home/rsmalley/ptect/f066 | sed '$d' | sed '$d' >> $OF
echo 9999, 9999 >> $OF
```



Blocks only have to agree with plates, etc. where there is data and/or you are trying to estimate behavior.

Blocks can't include pole as interior point – Antarctica composed of two blocks.

Set flags for what to process.

```
if [ $selection = everything ]
then

echo everything
SAM_SCO_SV=1
SAN_SCOT_SV=1
SAM_SAN_SV_tlp=1
SAM_SAN_SV_mt_tlp=1
ANT_SAN_SV=1
ANT_SCO_SV=1
NSR_SYNTH_SV=1

SCOT_SAN_SSV=1
SAN_SCOT_TA_tlp=1

FAULTS=1
NSR_SS=1
SSR_SS=1
ANT_B_SAM=1
ANT_B_SCOT=1
```

Build input control file.

Define which poles to find.
Put in geologic data (slip vectors)

```
#setup or rerun
#NEW=0
#with selection of solution - have to do setup each time
NEW=1
if [ $NEW = 1 ]
then
echo build ${DEFNINFILE}
\rm -r ${DEFNINFILE}
cat ${DEFNINFILE}.form > ${DEFNINFILE}
#pole 2 scotia, 3 sandwich, 4 antarctica
#echo pi pole: 2 3 4 >> ${DEFNINFILE}
echo pi pole: ${POLES} >> ${DEFNINFILE}

if [ $SAM_SCO_SV = 1 ]
then
echo eq slip vector data north scotia ridge paw, tlp and new, SAM_SCO${SIGMA}.slip
echo sv: SAM_SCO${SIGMA}.slip SCOT SAMR 1 >> ${DEFNINFILE}
echo sv: tdf1949.slip SCOT SAMR 1 >> ${DEFNINFILE}
fi
```

Build input control file.

Put in faults

```
if [ $FAULTS = 1 ]
then
 echo add faults to ${DEFNINFILE}
#have to only include faults with GPS data
# makedefnodefault filename lowleftlon lowerleftlat upperrightlon upperrightlat faultdep dip faultno
#               faultname hangingwall footwall
# cutdefnodefault
#have to be careful that fault goes correct direction hanging wall to right
#footwall correct and unique on fault

#define faults and put in dfn file

 if [ $NSR_SS = 1 ]
 then
#newsam.block goes ccw around sam, bounding block on right - scot - is hangingwall - make go other way, switch
#ll strike slip on nsr
#sector of NSR corresponding to Magallanes-Fagnano fault
  cutdefnodefault newsam.block -75.9962  -51.8223 -60.0172  -53.6962 15. 89. 1 SAMR-SCOT SCOT SAMR >> ${DEFNINFILE}
# Greens function controls - directory name 3 char only, x spacing, down dip spacing, fault id #
  echo gd: gsc 20 15 1 >> ${DEFNINFILE}
 fi
```

Build input control file.

Specify types output and positions to calculate it.

```
if [ $CALCRELVEC = 1 ]
then
 echo specify points to calc velocity A wrt B

#cant smooth scot-sand boundary easily
#sctually dont need to smooth to find ponts to determine vel
#(only need to smooth is want unailaised resampling or azimuth info)
 if [ $SCOT_SAND = 1 ]
 then
  nawk '{ print "fsp: SCOT SAND", $1, $2}' <<END>> ${DEFNINFILE}
-30.20 -57.39
-30.32 -57.29
.
.
.
-29.62 -59.25
-29.59 -59.52
END
 fi
```

Build input control file.

Control file done, now work on input data.

```
echo build ${DEFNINFILE} done, now make gps input data file

#have to remove segment identifiers and duplicate points from the segments
#the endpoints between adjacent segements are common, and defnode
#does not want blocks closed
```

Build input control file.

Prepare GPS files.

Have to rescale errors for defnode, but want to leave as are for plotting.

```
if [ $TDF_C = 1 ]
then
CRESCL=5
CRESCL=10
CRESCL=15
#CRESCL=45
#use tdf continuous stations - AUTF and PWMS - in plate boundary deforming zone
CFILES='tdf_gps_unscerr_c_good.vec'
for cfile in $CFILES
do
echo process cfile $cfile rescale errors $CRESCL
nawk '{print $1, $2, $3, $4, $5*'$CRESCL', $6*'$CRESCL', $7, $8}' ${DATA}/$cfile >> $EXP.vec
nawk '{print $1, $2, $3, $4, $5, $6, $7, $8}' ${DATA}/$cfile >> ${EXP}_no_rescerr.vec
done
fi
```

Build input control file.

Prepare GPS files.

Finally run defnode.

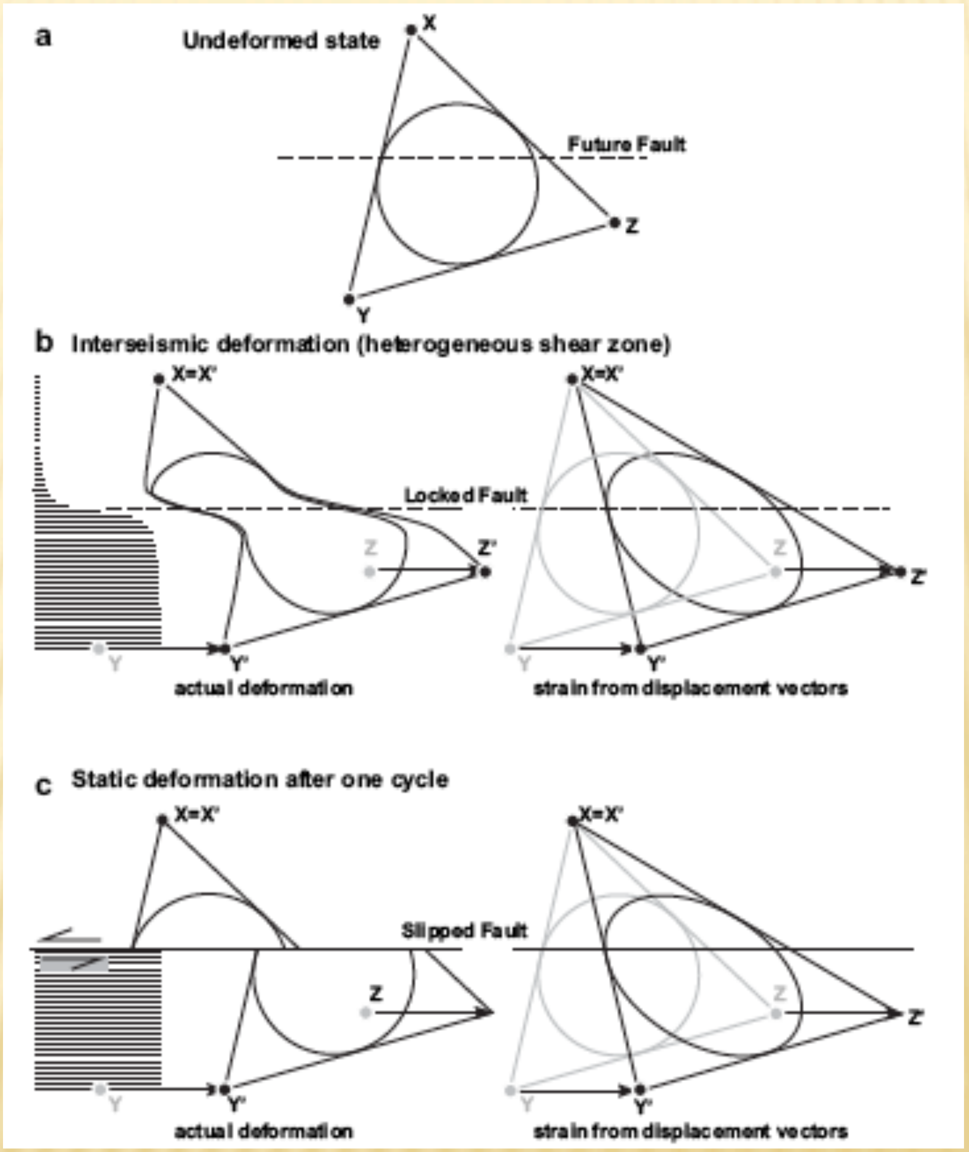
```
defnode ${DEFNINFILE} $EXP
echo done with defnode - make plots
```

1) Build control file, this includes definition of blocks (which can be quite complicated)

2) Prepare various data sets (slip vectors, transform azimuths, spreading directions and rates, GPS/VLBI/SLR/etc.).

(have to keep track of which information goes inside control file – typically geometry, slip, deformation (stuff not being modeled) – and which goes into data files – GPS, slip vectors, etc. (stuff being modeled).

3) Run it


a
Undeformed state
X
Future Fault
Z
Y
b Interseismic deformation (heterogeneous shear zone)
X=X'
Locked Fault
Z
Z'
Y
Y'
actual deformation
X=X'
Z
Z'
Y
Y'
strain from displacement vectors
c Static deformation after one cycle
X=X'
Slipped Fault
Z
Y
Y'
actual deformation
X=X'
Z
Z'
Y
Y'
strain from displacement vectors

Allmendinger et al., 2009