

# Simple example of a tectonic Inverse Problem (inspired by the rifting crisis in Afar)

Solving the problem using least squares.

*Albert Tarantola, February 2006*

## Forward Problem

These expressions solve the problem of evaluating the displacements caused when a fissure opens in a 2D "elastic" medium (see the PDF text for details).

```
Ux = x - X; Uy = y - Y; U = sqrt(Ux^2 + Uy^2);
Delta = Exp[delta];
f1 = Exp[-U/Delta];
fx = Sin[psi]; fy = Cos[psi];
beta = ArcCos[(fx Ux + fy Uy)/U];
alpha = pi/2 - beta;
f2 = Sin[alpha]^2;
Q = Exp[q];
ux = Q f1 f2 (Ux/U); uy = Q f1 f2 (Uy/U); uH = sqrt(ux^2 + uy^2); uz = uH/7;
```

The final expressions (for the three components of the displacements) are now used to define the functions solving the forward problem.

```
uxCAL[x_, y_, X_, Y_, delta_, psi_, q_] = ux;
uyCAL[x_, y_, X_, Y_, delta_, psi_, q_] = uy;
uzCAL[x_, y_, X_, Y_, delta_, psi_, q_] = uz;
```

## Generating the synthetic data

We shall have (simulated) observations at six points, whose coordinates are as follows.

```
x1 = 11.0; y1 = 24.5; x2 = 13.0; y2 = 28.5; x3 = 14.0; y3 = 21.5;
x4 = 15.0; y4 = 17.5; x5 = 07.0; y5 = 16.5; x6 = 08.0; y6 = 24.5;
```

To simulate our observations, we arbitrarily take some "true model".

```
Xtrue = 10.0; Ytrue = 20.0;  $\delta$ true = 1.0;  $\psi$ true = 0.5; qtrue = -0.5;
```

To the computed values, we shall add some random "errors", whose "size" is  $\epsilon = 0.01$ .

```
 $\epsilon$  = 0.01;
random := Random[Real, {- $\epsilon$ ,  $\epsilon$ }]
SeedRandom[123]
```

We compute the synthetic data, and, to give some realism to the values, we round them leaving only three decimals.

```
K = 1000.;
```

```
 $\omega$  = uxCAL[x1, y1, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uxOBS1 = Round[K ( $\omega$  + random)] / K;
Print["ux1=", %%, " , ux1(rounded)=", %]
 $\omega$  = uyCAL[x1, y1, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uyOBS1 = Round[K ( $\omega$  + random)] / K;
Print["uy1=", %%, " , uy1(rounded)=", %]
 $\omega$  = uzCAL[x1, y1, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uzOBS1 = Round[K ( $\omega$  + random)] / K;
Print["uz1=", %%, " , uz1(rounded)=", %]
```

```
ux1=0.0222763 , ux1(rounded)=0.016
```

```
uy1=0.100243 , uy1(rounded)=0.099
```

```
uz1=0.0146698 , uz1(rounded)=0.022
```

```
 $\omega$  = uxCAL[x2, y2, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uxOBS2 = Round[K ( $\omega$  + random)] / K;
Print["ux2=", %%, " , ux2(rounded)=", %]
 $\omega$  = uyCAL[x2, y2, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uyOBS2 = Round[K ( $\omega$  + random)] / K;
Print["uy2=", %%, " , uy2(rounded)=", %]
 $\omega$  = uzCAL[x2, y2, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uzOBS2 = Round[K ( $\omega$  + random)] / K;
Print["uz2=", %%, " , uz2(rounded)=", %]
```

```
ux2=0.0071395 , ux2(rounded)=0.016
```

```
uy2=0.0202286 , uy2(rounded)=0.015
```

```
uz2=0.0030645 , uz2(rounded)=0
```

```

 $\omega$  = uxCAL[x3, y3, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uxOBS3 = Round[K ( $\omega$  + random)] / K;
Print["ux3=", %%, " , ux3(rounded)=", %]
 $\omega$  = uyCAL[x3, y3, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uyOBS3 = Round[K ( $\omega$  + random)] / K;
Print["uy3=", %%, " , uy3(rounded)=", %]
 $\omega$  = uzCAL[x3, y3, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uzOBS3 = Round[K ( $\omega$  + random)] / K;
Print["uz3=", %%, " , uz3(rounded)=", %]

```

ux3=0.0676067 , ux3(rounded)=0.075

uy3=0.0253525 , uy3(rounded)=0.028

uz3=0.0103149 , uz3(rounded)=0.006

```

 $\omega$  = uxCAL[x4, y4, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uxOBS4 = Round[K ( $\omega$  + random)] / K;
Print["ux4=", %%, " , ux4(rounded)=", %]
 $\omega$  = uyCAL[x4, y4, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uyOBS4 = Round[K ( $\omega$  + random)] / K;
Print["uy4=", %%, " , uy4(rounded)=", %]
 $\omega$  = uzCAL[x4, y4, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uzOBS4 = Round[K ( $\omega$  + random)] / K;
Print["uz4=", %%, " , uz4(rounded)=", %]

```

ux4=0.000091652 , ux4(rounded)=0.003

uy4=-0.000045826 , uy4(rounded)=0.006

uz4=0.0000146386 , uz4(rounded)=-0.002

```

 $\omega$  = uxCAL[x5, y5, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uxOBS5 = Round[K ( $\omega$  + random)] / K;
Print["ux5=", %%, " , ux5(rounded)=", %]
 $\omega$  = uyCAL[x5, y5, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uyOBS5 = Round[K ( $\omega$  + random)] / K;
Print["uy5=", %%, " , uy5(rounded)=", %]
 $\omega$  = uzCAL[x5, y5, Xtrue, Ytrue,  $\delta$ true,  $\psi$ true, qtrue];
uzOBS5 = Round[K ( $\omega$  + random)] / K;
Print["uz5=", %%, " , uz5(rounded)=", %]

```

ux5=-0.0693041 , ux5(rounded)=-0.077

uy5=-0.0808548 , uy5(rounded)=-0.083

uz5=0.0152132 , uz5(rounded)=0.018

```

ω = uxCAL[x6, y6, Xtrue, Ytrue, δtrue, ψtrue, qtrue];
uxOBS6 = Round[K (ω + random)] / K;
Print["ux6=", %, " , ux6(rounded)=", %]
ω = uyCAL[x6, y6, Xtrue, Ytrue, δtrue, ψtrue, qtrue];
uyOBS6 = Round[K (ω + random)] / K;
Print["uy6=", %, " , uy6(rounded)=", %]
ω = uzCAL[x6, y6, Xtrue, Ytrue, δtrue, ψtrue, qtrue];
uzOBS6 = Round[K (ω + random)] / K;
Print["uz6=", %, " , uz6(rounded)=", %]

```

```
ux6=-0.0148412 , ux6(rounded)=-0.02
```

```
uy6=0.0333928 , uy6(rounded)=0.032
```

```
uz6=0.00522033 , uz6(rounded)=0.009
```

## Plot of the Data

Note: the displacements have been multiplied by 10, to make them visible.

```

MeasuringPoints = Graphics[{
  Point[{x1, y1}], Point[{x2, y2}],
  Point[{x3, y3}], Point[{x4, y4}], Point[{x5, y5}], Point[{x6, y6}]
}];
Arrows = Graphics[{
  Line[{x1, y1}, {x1 + 10 uxOBS1, y1 + 10 uyOBS1}],
  Line[{x2, y2}, {x2 + 10 uxOBS2, y2 + 10 uyOBS2}],
  Line[{x3, y3}, {x3 + 10 uxOBS3, y3 + 10 uyOBS3}],
  Line[{x4, y4}, {x4 + 10 uxOBS4, y4 + 10 uyOBS4}],
  Line[{x5, y5}, {x5 + 10 uxOBS5, y5 + 10 uyOBS5}],
  Line[{x6, y6}, {x6 + 10 uxOBS6, y6 + 10 uyOBS6}]
}];

```

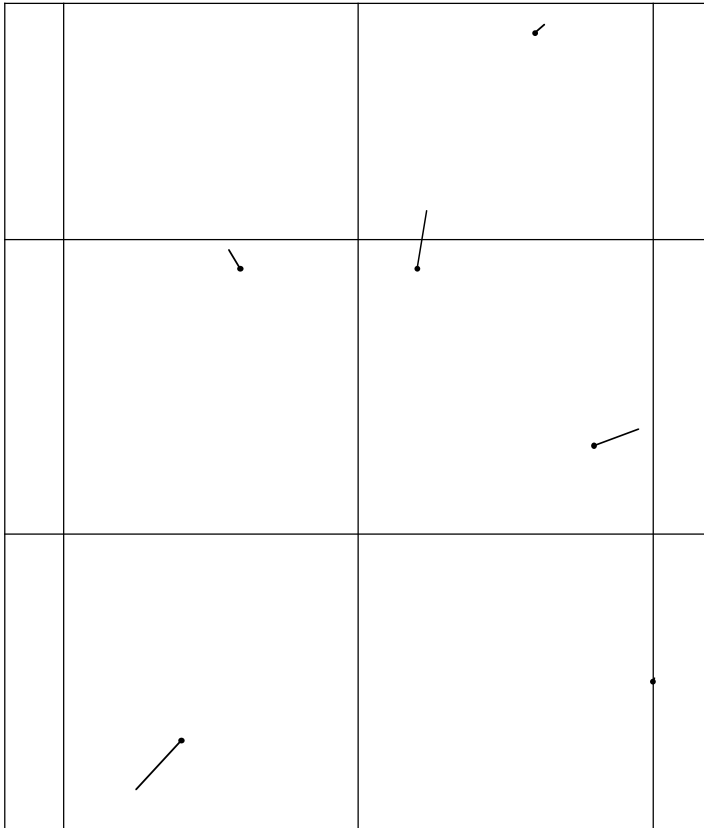
Let us also add some coordinate lines...

```

left = Line[{4, 15}, {4, 29}];
top = Line[{4, 29}, {16, 29}];
right = Line[{16, 29}, {16, 15}];
bottom = Line[{16, 15}, {4, 15}];
vert1 = Line[{5, 15}, {5, 29}];
vert2 = Line[{10, 15}, {10, 29}];
vert3 = Line[{15, 15}, {15, 29}];
hor1 = Line[{16, 20}, {4, 20}];
hor2 = Line[{16, 25}, {4, 25}];
Lines = Graphics[{left, top, right, bottom, vert1, vert2, vert3, hor1, hor2}];

```

```
Show[{MeasuringPoints, Arrows, Lines}, AspectRatio -> Automatic]
```



## Resolution of the inverse problem using the least-squares method

### Introducing the vector of calculated observations

```
dcal = {uxCAL[x1, y1, X, Y, δ, ψ, q],
  uyCAL[x1, y1, X, Y, δ, ψ, q], uzCAL[x1, y1, X, Y, δ, ψ, q],
  uxCAL[x2, y2, X, Y, δ, ψ, q], uyCAL[x2, y2, X, Y, δ, ψ, q],
  uzCAL[x2, y2, X, Y, δ, ψ, q], uxCAL[x3, y3, X, Y, δ, ψ, q],
  uyCAL[x3, y3, X, Y, δ, ψ, q], uzCAL[x3, y3, X, Y, δ, ψ, q],
  uxCAL[x4, y4, X, Y, δ, ψ, q], uyCAL[x4, y4, X, Y, δ, ψ, q],
  uzCAL[x4, y4, X, Y, δ, ψ, q], uxCAL[x5, y5, X, Y, δ, ψ, q],
  uyCAL[x5, y5, X, Y, δ, ψ, q], uzCAL[x5, y5, X, Y, δ, ψ, q], uxCAL[x6, y6, X,
  Y, δ, ψ, q], uyCAL[x6, y6, X, Y, δ, ψ, q], uzCAL[x6, y6, X, Y, δ, ψ, q]};
```

## Introducing the vector of observed values and the associated covariance matrix

```
dobs = {uxOBS1, uyOBS1, uzOBS1, uxOBS2, uyOBS2, uzOBS2, uxOBS3, uyOBS3, uzOBS3,
        uxOBS4, uyOBS4, uzOBS4, uxOBS5, uyOBS5, uzOBS5, uxOBS6, uyOBS6, uzOBS6};
CD =  $\epsilon^2$  IdentityMatrix[18];
ICD = Inverse[CD];
```

## The a priori model and the covariance matrix describing uncertainties in the a priori model .

We assume that we have the following a priori information on the model parameters:

$$X = 12 \pm 20 ; Y = 12 \pm 20 ; \delta = 2 \pm 1 ; \psi = \text{ArcTan}[1] \pm 1 ; q = 0 \pm 2 .$$

This is a quite weak information, but it is useful for two reasons: i) sometimes - when we have low quality data - it may help obtaining reasonable solutions; a ii) the algorithm uses a metric in the model space, and the convergence is more rapid when this metric is reasonable.

```
Xprior = 12.; Yprior = 12.;  $\delta$ prior = 2.;  $\psi$ prior = ArcTan[1.]; qprior = 0;
 $\sigma$ X = 20.;  $\sigma$ Y = 20.;  $\sigma$  $\delta$  = 1.;  $\sigma$  $\psi$  = 1.;  $\sigma$ q = 2;
m = {X, Y,  $\delta$ ,  $\psi$ , q};
mprior = {Xprior, Yprior,  $\delta$ prior,  $\psi$ prior, qprior};

CM =  $\begin{pmatrix} \sigma X^2 & 0 & 0 & 0 & 0 \\ 0 & \sigma Y^2 & 0 & 0 & 0 \\ 0 & 0 & \sigma \delta^2 & 0 & 0 \\ 0 & 0 & 0 & \sigma \psi^2 & 0 \\ 0 & 0 & 0 & 0 & \sigma q^2 \end{pmatrix}$ ;

ICM = Inverse[CM];
```

## The misfit function (with a priori information) .

Warning: for some strange reason, the developers of *Mathematica* use the same symbol (a dot) to designate a product of two matrices and the (elementary) scalar product of two vectors. The expression below should be written  $S2 = \text{Transpose}[(\text{tcal} - \text{tobs})].\text{Inverse}[\text{CD}].(\text{tcal} - \text{tobs})$ , but *Mathematica* would misunderstand the expression.

$$S = \frac{1}{2} ( (\text{dcal} - \text{dobs}).\text{Inverse}[\text{CD}].(\text{dcal} - \text{dobs}) + (\text{m} - \text{mprior}).\text{Inverse}[\text{CM}].(\text{m} - \text{mprior}) );$$

The matrix of partial derivatives  $G^{\wedge}i_{\alpha} = \delta g^{\wedge}i / \delta m^{\wedge}\alpha$  .

```

a01 = uxCAL[x1, y1, X, Y, δ, ψ, q];
a02 = uyCAL[x1, y1, X, Y, δ, ψ, q];
a03 = uzCAL[x1, y1, X, Y, δ, ψ, q];
a04 = uxCAL[x2, y2, X, Y, δ, ψ, q];
a05 = uyCAL[x2, y2, X, Y, δ, ψ, q];
a06 = uzCAL[x2, y2, X, Y, δ, ψ, q];
a07 = uxCAL[x3, y3, X, Y, δ, ψ, q];
a08 = uyCAL[x3, y3, X, Y, δ, ψ, q];
a09 = uzCAL[x3, y3, X, Y, δ, ψ, q];
a10 = uxCAL[x4, y4, X, Y, δ, ψ, q];
a11 = uyCAL[x4, y4, X, Y, δ, ψ, q];
a12 = uzCAL[x4, y4, X, Y, δ, ψ, q];
a13 = uxCAL[x5, y5, X, Y, δ, ψ, q];
a14 = uyCAL[x5, y5, X, Y, δ, ψ, q];
a15 = uzCAL[x5, y5, X, Y, δ, ψ, q];
a16 = uxCAL[x6, y6, X, Y, δ, ψ, q];
a17 = uyCAL[x6, y6, X, Y, δ, ψ, q];
a18 = uzCAL[x6, y6, X, Y, δ, ψ, q];

```

```

G = {
  {D[a01, X] D[a01, Y] D[a01, δ] D[a01, ψ] D[a01, q]
  D[a02, X] D[a02, Y] D[a02, δ] D[a02, ψ] D[a02, q]
  D[a03, X] D[a03, Y] D[a03, δ] D[a03, ψ] D[a03, q]
  D[a04, X] D[a04, Y] D[a04, δ] D[a04, ψ] D[a04, q]
  D[a05, X] D[a05, Y] D[a05, δ] D[a05, ψ] D[a05, q]
  D[a06, X] D[a06, Y] D[a06, δ] D[a06, ψ] D[a06, q]
  D[a07, X] D[a07, Y] D[a07, δ] D[a07, ψ] D[a07, q]
  D[a08, X] D[a08, Y] D[a08, δ] D[a08, ψ] D[a08, q]
  D[a09, X] D[a09, Y] D[a09, δ] D[a09, ψ] D[a09, q]
  D[a10, X] D[a10, Y] D[a10, δ] D[a10, ψ] D[a10, q]
  D[a11, X] D[a11, Y] D[a11, δ] D[a11, ψ] D[a11, q]
  D[a12, X] D[a12, Y] D[a12, δ] D[a12, ψ] D[a12, q]
  D[a13, X] D[a13, Y] D[a13, δ] D[a13, ψ] D[a13, q]
  D[a14, X] D[a14, Y] D[a14, δ] D[a14, ψ] D[a14, q]
  D[a15, X] D[a15, Y] D[a15, δ] D[a15, ψ] D[a15, q]
  D[a16, X] D[a16, Y] D[a16, δ] D[a16, ψ] D[a16, q]
  D[a17, X] D[a17, Y] D[a17, δ] D[a17, ψ] D[a17, q]
  D[a18, X] D[a18, Y] D[a18, δ] D[a18, ψ] D[a18, q]
};

```

```
GT = Transpose[G];
```

Programming the Quasi-Newton algorithm .

We define, the correspondence between our basic variables and the components of the vector "mcurrent":

```

refresh := {X → mcurrent[[1]], Y → mcurrent[[2]],
           δ → mcurrent[[3]], ψ → mcurrent[[4]], q → mcurrent[[5]]}

```

Here, the algorithm is initialized at the point `mprior`, and the convergence is obtained after 16 iterations. Note: I have not tried to nicely write the algorithm; I have rather used an explicit (and, I hope, easily understandable) notation. When starting the iterations at the prior point, I have had to use an attenuation factor  $\epsilon = 0.5$ ; if not, the algorithm oscillates too much.

```

ε = 0.5;
mcurrent = mprior;
Do[{
  Print["m = ", mcurrent, "    S = ", S /. refresh],
  dataresiduals = (dcal - dobs) /. refresh,
  modelresiduals = (mcurrent - mprior) /. refresh,
  g = G /. refresh,
  gt = GT /. refresh,
  gradient = gt.ICD.dataresiduals + ICM.modelresiduals,
  direction = Inverse[gt.ICD.g + ICM].gradient,
  mnew = mcurrent - ε direction,
  mcurrent = mnew},
{20}]

```

```

m = {12., 12., 2., 0.785398, 0}    S = 981.926
m = {10.2673, 8.95779, 2.11663, 0.889017, -0.0635489}    S = 451.768
m = {12.6696, 14.5346, 2.16264, 0.344174, -1.81926}    S = 159.688
m = {9.10826, 10.2419, 2.16706, 0.315647, -1.58497}    S = 173.169
m = {11.818, 17.9069, 2.37172, 0.0298606, -2.64171}    S = 103.955
m = {9.23099, 21.2811, 1.10319, 0.568782, -1.45768}    S = 63.4426
m = {9.67771, 19.8664, 1.04041, 0.596848, -0.797587}    S = 10.8065
m = {9.9448, 19.8661, 1.02796, 0.52946, -0.627091}    S = 4.26857
m = {10.018, 19.8525, 1.01371, 0.506228, -0.53557}    S = 2.76893
m = {10.0455, 19.8469, 1.0054, 0.496744, -0.490014}    S = 2.41156
m = {10.0575, 19.8447, 1.00061, 0.492442, -0.46669}    S = 2.32413
m = {10.063, 19.8439, 0.997888, 0.490391, -0.454582}    S = 2.30246
m = {10.0656, 19.8436, 0.996355, 0.489389, -0.448266}    S = 2.29705
m = {10.0669, 19.8435, 0.995505, 0.488892, -0.444969}    S = 2.29569
m = {10.0676, 19.8435, 0.995038, 0.488644, -0.443251}    S = 2.29535
m = {10.0679, 19.8435, 0.994784, 0.48852, -0.442358}    S = 2.29527
m = {10.068, 19.8435, 0.994648, 0.488457, -0.441894}    S = 2.29525
m = {10.0681, 19.8435, 0.994575, 0.488425, -0.441654}    S = 2.29524
m = {10.0681, 19.8435, 0.994537, 0.488409, -0.44153}    S = 2.29524
m = {10.0681, 19.8435, 0.994516, 0.488401, -0.441466}    S = 2.29524

```

The low value attained above by the misfit function ( $S = 2.3$ ) suggests that we have the absolute minimum. But let us check this by starting the iterations at an arbitrary point.

```

mtest = {11., 21., 2., 0.8, -0.8};
mcurrent = mtest;
ε = 0.6;
Do[{
  Print["m = ", mcurrent, "    S = ", S /. refresh],
  dataresiduals = (dcal - dobs) /. refresh,
  modelresiduals = (mcurrent - mprior) /. refresh,
  g = G /. refresh,
  gt = GT /. refresh,
  gradient = gt.ICD.dataresiduals + ICM.modelresiduals,
  direction = Inverse[gt.ICD.g + ICM].gradient,
  mnew = mcurrent - ε direction,
  mcurrent = mnew},
{15}]

```

```

m = {11., 21., 2., 0.8, -0.8}    S = 180.258
m = {10.6335, 20.708, 1.94703, 0.779005, -1.14382}    S = 50.0455
m = {10.2857, 20.2664, 1.6674, 0.697159, -1.20803}    S = 12.3365
m = {10.0713, 19.9587, 1.31965, 0.605531, -0.983832}    S = 4.30384
m = {10.0605, 19.8827, 1.12116, 0.537617, -0.707147}    S = 2.94314
m = {10.0645, 19.859, 1.04431, 0.507917, -0.554982}    S = 2.43544
m = {10.0665, 19.85, 1.01471, 0.496295, -0.488709}    S = 2.32061
m = {10.0675, 19.8463, 1.00282, 0.491634, -0.460995}    S = 2.29957
m = {10.0679, 19.8447, 0.997941, 0.489729, -0.449498}    S = 2.29596
m = {10.068, 19.844, 0.995924, 0.488944, -0.444743}    S = 2.29536
m = {10.0681, 19.8437, 0.995087, 0.48862, -0.442778}    S = 2.29526
m = {10.0681, 19.8436, 0.99474, 0.488487, -0.441967}    S = 2.29524
m = {10.0682, 19.8435, 0.994596, 0.488432, -0.441633}    S = 2.29524
m = {10.0682, 19.8435, 0.994536, 0.488409, -0.441495}    S = 2.29524
m = {10.0682, 19.8435, 0.994511, 0.488399, -0.441438}    S = 2.29524

```

We have converged to the same point. O.K.

```

mpost = mnew
replace =
  {X → mpost[[1]], Y → mpost[[2]], δ → mpost[[3]], ψ → mpost[[4]], q → mpost[[5]]};

```

```
{10.0682, 19.8435, 0.994501, 0.488396, -0.441415}
```

Comparison between the "observed" and the computed data (and evaluation of the final residuals).

```

m = {X, Y,  $\delta$ ,  $\psi$ , q};
{Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost} = m /. replace;
dfinal = {
  a01 = uxCAL[x1, y1, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a02 = uyCAL[x1, y1, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a03 = uzCAL[x1, y1, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a04 = uxCAL[x2, y2, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a05 = uyCAL[x2, y2, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a06 = uzCAL[x2, y2, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a07 = uxCAL[x3, y3, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a08 = uyCAL[x3, y3, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a09 = uzCAL[x3, y3, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a10 = uxCAL[x4, y4, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a11 = uyCAL[x4, y4, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a12 = uzCAL[x4, y4, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a13 = uxCAL[x5, y5, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a14 = uyCAL[x5, y5, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a15 = uzCAL[x5, y5, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a16 = uxCAL[x6, y6, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a17 = uyCAL[x6, y6, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost],
  a18 = uzCAL[x6, y6, Xpost, Ypost,  $\delta$ post,  $\psi$ post, qpost]};
dobs
Round[K (dfinal)] / K
Round[K (dfinal - dobs)] / K

```

```
{0.016, 0.099, 0.022, 0.016, 0.015, 0, 0.075, 0.028, 0.006,
  0.003, 0.006, -0.002, -0.077, -0.083, 0.018, -0.02, 0.032, 0.009}
```

```
{0.02, 0.1, 0.015, 0.007, 0.02, 0.003, 0.074, 0.031,
  0.011, 0, 0, 0, -0.076, -0.083, 0.016, -0.015, 0.034, 0.005}
```

```
{0.004, 0.001, -0.007, -0.009, 0.005, 0.003, -0.001, 0.003, 0.005,
  -0.003, -0.006, 0.002, 0.001, 0, -0.002, 0.005, 0.002, -0.004}
```

## Evaluation of the a posteriori uncertainties

We now evaluate the posterior covariance matrix:

```

Wpost = gt.ICD.g + ICM;
CMpost = Inverse[Wpost];
MatrixForm[CMpost]

```

$$\begin{pmatrix} 0.034931 & -0.013336 & -0.00203915 & -0.00940779 & 0.00860566 \\ -0.013336 & 0.0655971 & -0.00674565 & 0.0154005 & 0.00584505 \\ -0.00203915 & -0.00674565 & 0.0720219 & 0.0052986 & -0.124863 \\ -0.00940779 & 0.0154005 & 0.0052986 & 0.00848156 & -0.0120313 \\ 0.00860566 & 0.00584505 & -0.124863 & -0.0120313 & 0.220756 \end{pmatrix}$$

The standard deviations are:

```

 $\sigma_1 = \sqrt{\text{CMpost}[[1, 1]]}$ 
 $\sigma_2 = \sqrt{\text{CMpost}[[2, 2]]}$ 
 $\sigma_3 = \sqrt{\text{CMpost}[[3, 3]]}$ 
 $\sigma_4 = \sqrt{\text{CMpost}[[4, 4]]}$ 
 $\sigma_5 = \sqrt{\text{CMpost}[[5, 5]]}$ 

```

0.186898

0.256119

0.268369

0.0920954

0.469847

The basic results are:

$X = 10.1 \pm 0.2$  ;  $Y = 19.8 \pm 0.3$  ;  $\delta = 1.0 \pm 0.3$  ;  $\psi = 0.49 \pm 0.1$  ;  $q = -0.44 \pm 0.5$  .

This is to be compared with the "true model", that was (remember that we contaminated the "true observed values" with some errors):

$X = 10.$  ;  $Y = 20.$  ;  $\delta = 1.$  ;  $\psi = 0.5$  ;  $q = -0.5$  .

Remember also the the a priori model was

$X = 12. \pm 20.$  ;  $Y = 12. \pm 20.$  ;  $\delta = 2. \pm 1.$  ;  $\psi = 0.8. \pm 1.$  ;  $q = 0. \pm 2.$  .

We see that all the a posteriori uncertainties are smaller that the a priori ones, so, our data has actually brought information on all the parameters.

The covariances are:

$$\rho_{12} = \frac{\text{CMpost}[[1, 2]]}{\sigma_1 \sigma_2}; \rho_{13} = \frac{\text{CMpost}[[1, 3]]}{\sigma_1 \sigma_3};$$

$$\rho_{14} = \frac{\text{CMpost}[[1, 4]]}{\sigma_1 \sigma_4}; \rho_{15} = \frac{\text{CMpost}[[1, 5]]}{\sigma_1 \sigma_5}; \rho_{23} = \frac{\text{CMpost}[[2, 3]]}{\sigma_2 \sigma_3};$$

$$\rho_{24} = \frac{\text{CMpost}[[2, 4]]}{\sigma_2 \sigma_4}; \rho_{25} = \frac{\text{CMpost}[[2, 5]]}{\sigma_2 \sigma_5}; \rho_{34} = \frac{\text{CMpost}[[3, 4]]}{\sigma_3 \sigma_4};$$

$$\rho_{35} = \frac{\text{CMpost}[[3, 5]]}{\sigma_3 \sigma_5}; \rho_{45} = \frac{\text{CMpost}[[4, 5]]}{\sigma_4 \sigma_5};$$

$$\text{CorrelationMatrix} = \begin{pmatrix} 1 & \rho_{12} & \rho_{13} & \rho_{14} & \rho_{15} \\ \rho_{12} & 1 & \rho_{23} & \rho_{24} & \rho_{25} \\ \rho_{13} & \rho_{23} & 1 & \rho_{34} & \rho_{35} \\ \rho_{14} & \rho_{24} & \rho_{34} & 1 & \rho_{45} \\ \rho_{15} & \rho_{25} & \rho_{35} & \rho_{45} & 1 \end{pmatrix};$$

`MatrixForm[CorrelationMatrix]`

$$\begin{pmatrix} 1 & -0.278598 & -0.0406548 & -0.546568 & 0.0979991 \\ -0.278598 & 1 & -0.0981407 & 0.65291 & 0.0485723 \\ -0.0406548 & -0.0981407 & 1 & 0.214383 & -0.990249 \\ -0.546568 & 0.65291 & 0.214383 & 1 & -0.278046 \\ 0.0979991 & 0.0485723 & -0.990249 & -0.278046 & 1 \end{pmatrix}$$

## Random generation of posterior models

At this point, my suggestion is as follows. We use the approximation that the posterior probability distribution for the parameters  $\{X, Y, \delta, \psi, q\}$  is a Gaussian distribution, of which we know the mean vector,  $\{10.1, 19.8, 0.91, 0.48, -0.29\}$ , and the covariance matrix,

$$\begin{pmatrix} 0.034931 & -0.013336 & -0.00203915 & -0.00940779 & 0.00860566 \\ -0.013336 & 0.0655971 & -0.00674565 & 0.0154005 & 0.00584505 \\ -0.00203915 & -0.00674565 & 0.0720219 & 0.0052986 & -0.124863 \\ -0.00940779 & 0.0154005 & 0.0052986 & 0.00848156 & -0.0120313 \\ 0.00860566 & 0.00584505 & -0.124863 & -0.0120313 & 0.220756 \end{pmatrix}. \text{ We can use this}$$

Gaussian distribution to generate some random solutions, and we can represent the corresponding results (i.e., draw the randomly generated fissures). This should convey the actual information that we really have, much better than any examination of numbers.

Here is the computation. See the drawing in the PDF document.

I first define the posterior probability density (note that it is normalized so that its maximum value is 1).

```
m = {X, Y, δ, ψ, q};
mpost = m /. replace;
posteriorPDF[X_, Y_, δ_, ψ_, q_] = Exp[-1/2 (m - mpost).Wpost.(m - mpost)];
```

Then, I use the simple rejection method to obtain sample points:

## Sampling the posterior (and printing the results)

```
SeedRandom[321]
k = 3.;
Do[{
  RX = Random[Real, {Xpost - k  $\sigma$ 1, Xpost + k  $\sigma$ 1}],
  RY = Random[Real, {Ypost - k  $\sigma$ 2, Ypost + k  $\sigma$ 2}],
  R $\delta$  = Random[Real, { $\delta$ post - k  $\sigma$ 3,  $\delta$ post + k  $\sigma$ 3}],
  R $\psi$  = Random[Real, { $\psi$ post - k  $\sigma$ 4,  $\psi$ post + k  $\sigma$ 4}],
  Rq = Random[Real, {qpost - k  $\sigma$ 5, qpost + k  $\sigma$ 5}],
   $\alpha$  = posteriorPDF[RX, RY, R $\delta$ , R $\psi$ , Rq],
  TestValue = Random[Real, {0, 1}],
  If[ $\alpha$  > TestValue,
    {Print["{X,Y, $\delta$ , $\psi$ ,q} = ", {RX, RY, R $\delta$ , R $\psi$ , Rq}, " ,  $\alpha$  = ",  $\alpha$ ]}]
}, {25000}]
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{9.86494, 20.0906, 0.879582, 0.608103, -0.305516} ,  $\alpha$  = 0.297818
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{9.70054, 20.2998, 1.13962, 0.57445, -0.794375} ,  $\alpha$  = 0.010289
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.3175, 19.7086, 1.0881, 0.436413, -0.582175} ,  $\alpha$  = 0.353715
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{9.83593, 19.9795, 0.881003, 0.584421, -0.153527} ,  $\alpha$  = 0.0180661
```

```
{X,Y, $\delta$ , $\psi$ ,q} = {10.05, 20.017, 1.30788, 0.491992, -1.01534} ,  $\alpha$  = 0.164651
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.3306, 19.6847, 0.889961, 0.494039, -0.141103} ,  $\alpha$  = 0.0161806
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.15, 19.9575, 0.552222, 0.446119, 0.356096} ,  $\alpha$  = 0.196341
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.0522, 20.1889, 0.928204, 0.532686, -0.291207} ,  $\alpha$  = 0.21896
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{9.98811, 19.9363, 0.821493, 0.462214, -0.23058} ,  $\alpha$  = 0.16852
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.3291, 19.6417, 0.895593, 0.413219, -0.113015} ,  $\alpha$  = 0.0328758
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{9.82264, 19.3974, 0.972244, 0.388959, -0.403887} ,  $\alpha$  = 0.0225489
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.0132, 19.4069, 0.974686, 0.42919, -0.402533} ,  $\alpha$  = 0.146218
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{9.81669, 19.8808, 1.04412, 0.549861, -0.472047} ,  $\alpha$  = 0.111103
```

```
{X,Y, $\delta$ , $\psi$ ,q} =
{10.0682, 20.5177, 1.46444, 0.683745, -1.33969} ,  $\alpha$  = 0.00213303
```

```

{X,Y, $\delta$ , $\psi$ ,q} =
{9.95656, 19.9435, 1.05596, 0.522436, -0.600674} ,  $\alpha$  = 0.631803

{X,Y, $\delta$ , $\psi$ ,q} =
{10.0029, 20.0584, 1.12913, 0.55773, -0.68911} ,  $\alpha$  = 0.583905

{X,Y, $\delta$ , $\psi$ ,q} =
{9.82265, 19.7291, 1.01402, 0.502853, -0.499515} ,  $\alpha$  = 0.289315

{X,Y, $\delta$ , $\psi$ ,q} =
{10.1163, 19.9208, 1.05098, 0.491892, -0.569873} ,  $\alpha$  = 0.71878

{X,Y, $\delta$ , $\psi$ ,q} =
{9.77458, 19.9911, 0.890841, 0.568479, -0.233935} ,  $\alpha$  = 0.110831

{X,Y, $\delta$ , $\psi$ ,q} =
{10.1081, 20.0731, 0.79251, 0.52674, -0.0699913} ,  $\alpha$  = 0.386158

```

## Sampling the posterior (and drawing the results)

```

SeedRandom[321]
k = 3.;
Fissure = Table[0, {100}];
i = 0;
Do[{
  RX = Random[Real, {Xpost - k  $\sigma$ 1, Xpost + k  $\sigma$ 1}],
  RY = Random[Real, {Ypost - k  $\sigma$ 2, Ypost + k  $\sigma$ 2}],
  R $\delta$  = Random[Real, { $\delta$ post - k  $\sigma$ 3,  $\delta$ post + k  $\sigma$ 3}],
  R $\psi$  = Random[Real, { $\psi$ post - k  $\sigma$ 4,  $\psi$ post + k  $\sigma$ 4}],
  Rq = Random[Real, {qpost - k  $\sigma$ 5, qpost + k  $\sigma$ 5}],
   $\alpha$  = posteriorPDF[RX, RY, R $\delta$ , R $\psi$ , Rq],
  TestValue = Random[Real, {0, 1}],
  If[ $\alpha$  > TestValue,
    {i = i + 1,
     center = {RX, RY},
     R $\Delta$  = Exp[R $\delta$ ],
     point1 = {RX + (R $\Delta$  / 2) Cos[R $\psi$ ], RY - (R $\Delta$  / 2) Sin[R $\psi$ ]},
     point2 = {RX - (R $\Delta$  / 2) Cos[R $\psi$ ], RY + (R $\Delta$  / 2) Sin[R $\psi$ ]},
     RQ = Exp[Rq],
     pointQ = {RX + RQ Sin[R $\psi$ ], RY + RQ Cos[R $\psi$ ]},
     lineP = Line[{point1, point2}],
     lineQ = Line[{center, pointQ}],
     Fissure[[i]] = Graphics[{lineP, lineQ}]
    }
  ], {25000}]

```

```

FissureSet = {
  Fissure[[01]], Fissure[[02]], Fissure[[03]], Fissure[[04]], Fissure[[05]],
  Fissure[[06]], Fissure[[07]], Fissure[[08]], Fissure[[09]], Fissure[[10]],
  Fissure[[11]], Fissure[[12]], Fissure[[13]], Fissure[[14]]};
Show[{MeasuringPoints, Arrows, FissureSet, Lines},
  PlotRange  $\rightarrow$  All, AspectRatio  $\rightarrow$  Automatic]
Show[{FissureSet}, PlotRange  $\rightarrow$  All, AspectRatio  $\rightarrow$  Automatic]

```

