Advanced numerical engineering

... around the Piton de la Fournaise

Y. Richet

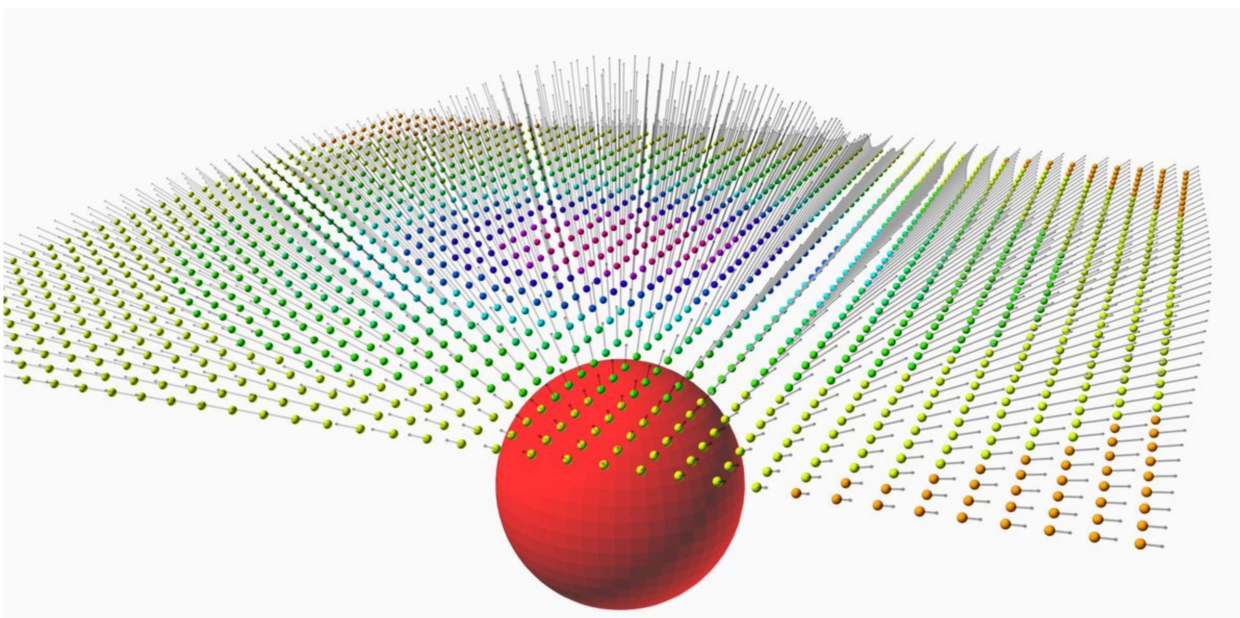
2017

- Understand the problem: model & data
- Build a consistent engineering approach: optimization & inversion
- Use R packages dedicated to design of experiments, kriging, optimization & inversion
- Use R packages suitable to plot & visualize N-dimensional data
- ... produce a "reproducible" report using RMarkdown

Case study - Mogi model & InSAR measured displacement

Following is a short reminer of detailed information available in the "volcan_test_case.pdf" file.

Mogi model



```
#' MOGI(G,nu,xs,ys,zs,a,p,xi,yi,zi) compute surface displacements and tilts created by
#' a point source located beneath a topography. To account for topography, a first order
#' solution in which the actual source to ground surface point is taken into account.
#' @author V. Cayol, LMV, sept 2017 (translated into R by R. Le Riche)
#' @example [uxi,uyi,uzi] = mogi_3D(G,nu,xs,ys,zs,a,p,xi,yi,zi)
#' @param G = shear modulus in MPa, G = E/2(1+nu)
#' @param nu = Poisson's ratio
#' @param xs, ys, zs = source position (z axis is positive upward),
#' @param a = source radius,
#' @param p = source overpressure in MPa,
#' @param xi, yi, zi = location of ground surface points
```

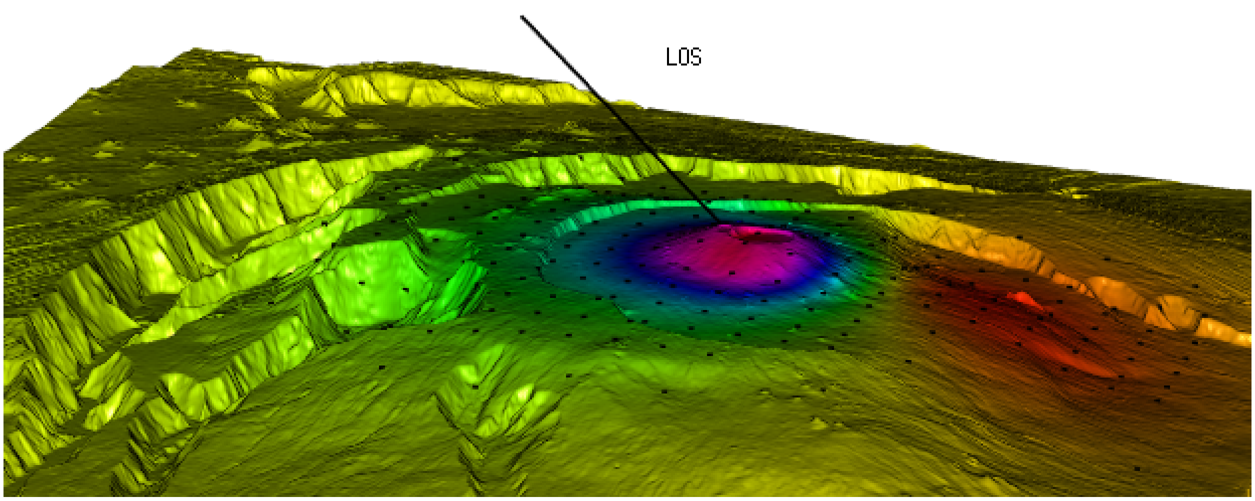
```r
#' @return displacement U(x,y,z) following Mogi's model
mogi_3D <- function(G = 2000,nu = 0.25,xs,ys,zs,a,p,xi,yi,zi){
  DV = pi*a^3*p/G
  C = (1-nu)*DV/pi
  r = sqrt((xi-xs)^2+(yi-ys)^2)
  f = r^2+(zi-zs)^2
  uzi = C*(zi-zs)/(f^(3/2))
  ur = C*r/(f^(3/2))
  theta = atan2(yi-ys,xi-xs)
  uxi = ur*cos(theta)
  uyi = ur*sin(theta)
  U = list(x=uxi,y=uyi,z=uzi)
  return(U)
}
```

InSAR measures

```r
## Get measured data from InSAR
data <- R.matlab::readMat('data_nonoise.mat')
Glb_xi <<- as.matrix(data$locdata[,1])
Glb_yi <<- as.matrix(data$locdata[,2])
Glb_zi <<- as.matrix(data$locdata[,3])
Glb_ulos <<- as.matrix(data$locdata[,4])

# calculate data Covariance matrix, store it in a Global variable
# covariance from exponential kernel, var = 5e-4m2, cor_length = 850 m
# and invert it
Xdata <- cbind(Glb_xi,Glb_yi) # z's are not accounted for in kernel
source("kernels.R")
Glb_CXinv <- solve(kExp(Xdata,Xdata,c(5e-4,850,850))) # calculated once for all, used in wls_ulos

nlos = c(-0.664,-0.168,0.728) # vector of direction of line of sight (satellite)
```



Model & data compliance

In this case study, the main interest output value is the accordance of model to measured data (it is most often the case for numerical engineering in life & earth sciences). So the objective function considered is the error between pure Mogi's model and measured data (accounting for their spatial correlation).

```
#' Weighted Least Squares distance function for ulos vectors.
#' The covariance matrix is passed through global variable.
#' @param xyzap array containing xs,ys,zs,a and p
#' @return error (weighted least square) between measure and model
wls_ulos <- function(xyzap){
  G = 2000 # Shear modulus in MPa
  nu = 0.25 # Poisson's ratio
  # Compute surface displacements follwoing Mogi's model
  U <- mogi_3D(G,nu,xyzap[1],xyzap[2],xyzap[3],xyzap[4],xyzap[5],Glb_xi,Glb_yi,Glb_zi)
  # project along LOS
  ulos <- nlos[1]*U$x + nlos[2]*U$y + nlos[3]*U$z
  # calculate weighted least squares error between measure and model
  wls <- t((ulos-Glb_ulos))%*%Glb_CXinv%*%(ulos-Glb_ulos)
  return(wls)
}
```

Numerical engineering - reminder

General model analysis:

- what magma chamber parameters (x_s, y_s, z_s, a, p) are most influent on displacement error ?

→ *parameters screening* (eg. Morris screening on (x_s, y_s, z_s, a, p))

- what displacement error uncertainty due magma chamber parameters (x_s, y_s, z_s, a, p) uncertainty ?

→ *uncertainties propagation* (eg. random sampling on (x_s, y_s, z_s, a, p))

- what magma chamber parameters (x_s, y_s, z_s, a, p) uncertainty are most contributing to displacement error uncertainty ?

→ *sensitivity analysis* (eg. Sobol indices of (x_s, y_s, z_s, a, p) using FAST DoE)

Numerical engineering - next

Standard model optimization/calibration:

- find *one* possible magma chamber parameters (x_s, y_s, z_s, a, p) to reach a displacement close to InSAR measure.

→ *bayesian optimization* (eg. EGO)

Full model identification:

- identify *all* possible magma chambers (x_s, y_s, z_s, a, p) leading to a given displacement (close to InSAR measured one).

→ *bayesian inversion* (eg. Ranjan, tIMSE, SUR)

- identify *all* possible magma chambers (x_s, y_s, z_s, a, p) leading to many given displacements (closer to InSAR measured one).

→ *multi-level bayesian inversion* (eg. SUR)

Penalized model identification:

- identify *all* possible magma chambers positions (a, p) leading to a given displacement (close to InSAR measured one), considering any position (x_s, y_s, z_s).

→ *bayesian robust inversion* (eg. RSUR)

Preliminary sampling of model

0. Build a first design of experiments based on 100 random simulations. Plot.
   - scatter plot: `pairs()`, `pairsD3::pairsD3()`
   - parallel coordinates plot: `MASS::parcoord()`, `parcoords::parcoords()` or `plotly::plot_ly`

Magma chamber optimization

1. Perform an optimization of the magma chamber to find a first calibrated magma chamber ($wls < -1.5$)

Full identification of magma chamber

2. Based on previous design, build a meta-model and plot it. Propose a target level for inversion (considering previous optimization results)

3. Using Bichon criterion (from `KrigInv` package), propose next simulation. Iterate. Plot.

4. Using SUR criterion (from `KrigInv` package), propose next simulation. Iterate by batch (size=10). Plot.

5. Compare and analyse results of 3. & 4.