R/Q map showing code. Calculations of Resolution/Resolution error ratio

Dmytro Dmytriiev

06.09.2018

1 Program overview

This program consists of three modules where PlotCST module builds 1D and 2D plots of the R/Q values, ResFreq module creates resolution map of the cavity and SuperposField module added fields of two cavities to calculate the influence of one cavity on another. Simulations were made in CST2018.

2 Class PlotCST

Constructor of the class creates several variables which will use data entered by user. Calling command:

python3 PlotCST.py /home/skye/Programs/myscripts/CSTDATA/GSICav/Backupdata/14x3-allXforY.txt h 0.0 0.02 0.12 21 -0.015 0.015 21

python3 - your calling command for python 3

PlotCST.py - name of the class. It has some check function which will show you three plots.

/home/skye/Programs/myscripts/CSTDATA/GSICav/Backupdata/14x3-allXforY.txt - path to your datafile from CST

h - direction in which you want to check linearity

0.0 - at which coordinate you want to check linearity

0.02 - x start coordinate of your datafile

0.12 - x stop coordinate of your datafile

21 - number of sweeps through x coordinate parameter

-0.015 - y start coordinate of your datafile

0.015 - y stop coordinate of your datafile

21 - number of sweeps through y coordinate parameter

Imagine that you have R/Q data inside the rectangular pipe. According to the CST algorithms data was taken as it shown on the Table 1

Table 1: Cst saved data "All X for Y"

R/Q(X1,Y1)	R/Q(X2,Y1)	...	R/Q(Xn,Y1)
R/Q(X1,Y2)	R/Q(X2,Y2)	...	R/Q(Xn,Y2)
...	...	...	...
R/Q(X1,Yn)	R/Q(X2,Yn)	...	R/Q(Xn,Yn)

Also it is possible that you will have another data structure, when sweeping goes first through another parameter. Your data then will look as it shown inTable 2

Table 2: Cst saved data

R/Q(X1,Y1)	R/Q(X1,Y2)	...	R/Q(X1,Yn)
R/Q(X2,Y1)	R/Q(X2,Y2)	...	R/Q(X2,Yn)
...	...	...	...
R/Q(Xn,Y1)	R/Q(Xn,Y2)	...	R/Q(Xn,Yn)

2.1 Symbol change method

Symbol change method allows user to change all commas to dots in datafile. This is needed because python doesn‘t treat numbers with commas as a floats, so instead of numbers in your array you will have "NaN" - Not A Number value. Special word "with" takes care about all exceptions, so no need to write all of them. If something goes wrong with reading or writing to file - you will know exactly what happens. For every line in the text file this method reads line, changes comma to dot and adds that changed line to an array which later will be used for all data manipulations. Input file remains unchanged. New file with a new name appears. For example "data.txt" will become "datadot.txt".

2.2 Get CST data method

This method returns XY coordinates for mesh and reshaped R/Q data. Idea is to skip first 16 lines of input file, to use internal *numpy* method $genfromtxt$ to create a *numpy* array from the text file. 15 lines - is a description of an exported file, line number 16 is a data, taken from the first simulation where electromagnetic field was simulated. Only after that simulation one can export any data from the simulation. So, first point is useless for our calculations. Data file contains

two columns, first is numbers of simulations, second is related R/Q data. *genfromtxt* reads data vertically, so first part of array will be numbers of simulations. We have to remove numbers of simulations from our array. That's why we use internal method *reshape* and reshape our array from shape (882, 1) to (441,2) and remove the first part. After that we create two arrays with X and Y coordinates of our simulation points using the data with start points, stop points and amount of steps between. Then we reshape our data array so it fits our coordinate arrays and transpose it because of working algorithm of *meshgrid* method. Get CST data method returns 1D array with X-coordinates, 1D array with Y-coordinates and 2D array with a R/Q values which fits our coordinate system. We already can produce a plot from that data which is done in a Check method.

2.3 Linear fit method

This method allows us to do linear fit and get errors of parameters. When user calls a command he is able to choose vertical or horizontal direction of linearity check. Then method picks all coordinates along the certain line (vertical or horizontal) and creates an array which is used for storage of an R/Q values. Later on lambda function is used for linear fitting, from which we have parameters of the fit and covariance matrix. Square root from diagonalized covariance matrix give us an error of fitting parameters. Later on we create an array which contains points of fit. Method returns us 1D array with coordinate along which we check linearity, 1D array of simulated R/Q values along that coordinate, 1D array of points which are related to our linear fit, parameters of the fit and their errors(float values).

2.4 Check method

This method builds an R/Q maps and also data with its fit. Just as opportunity to check is everything works correctly. As a result user will have two consequent plots. Contour and contourf plot of R/Q map and linearity fit on the choosen coordinate. Both will be saved as .pdf. Example is shown on Fig. 1

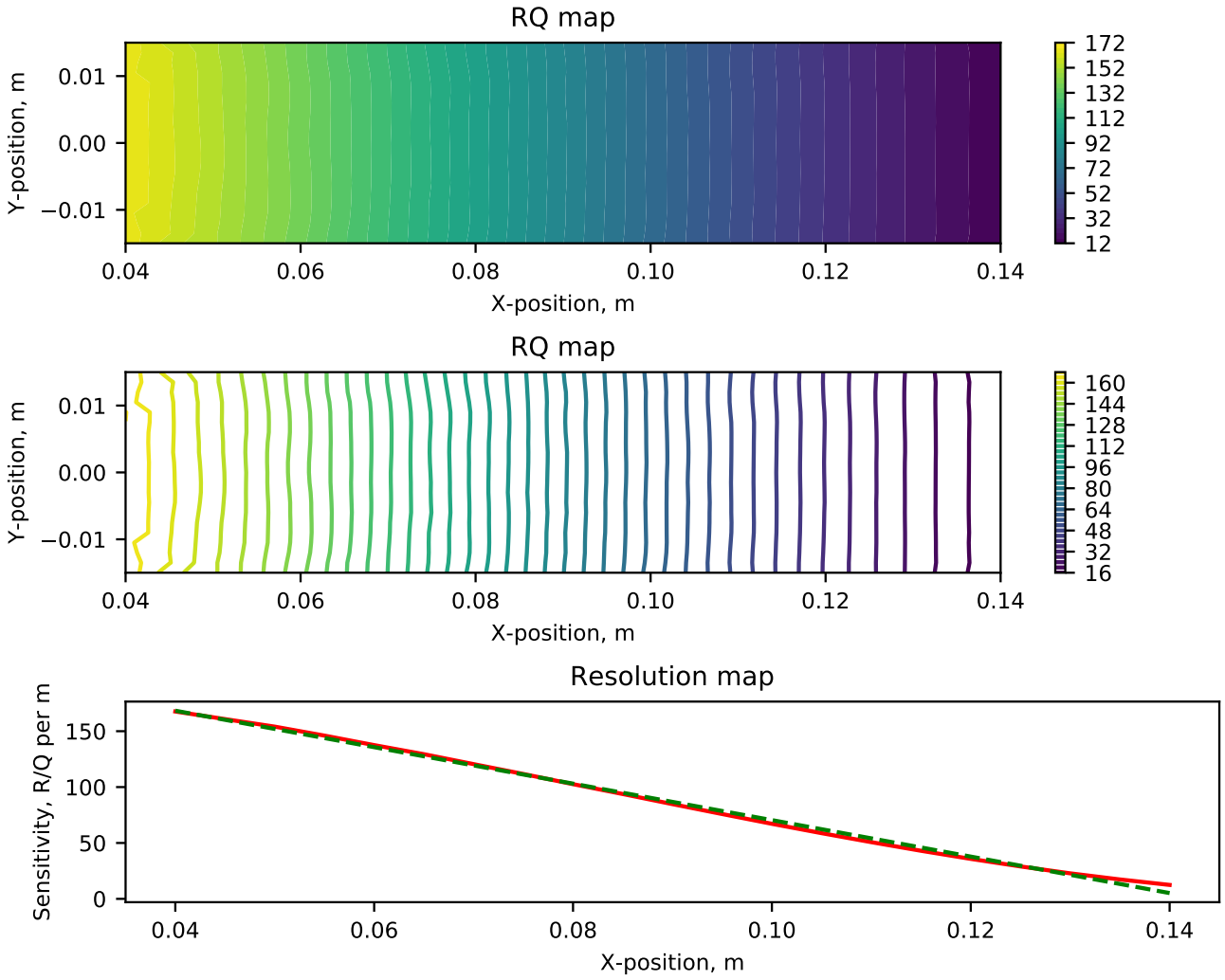


Fig. 1: Resolution to resolution error ratio for different cavity sizes

3 init.py

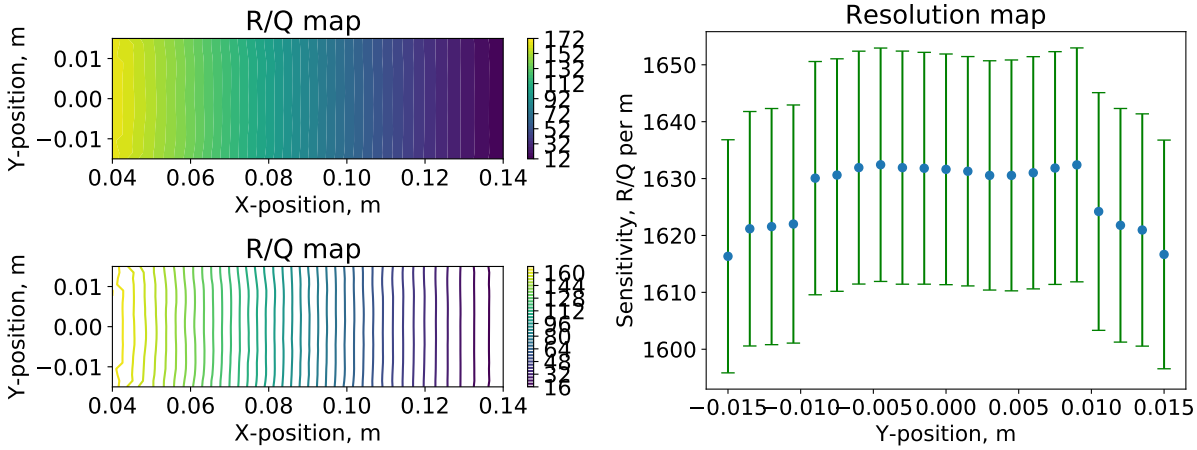
This script reads data file and using linear fit method from PlotCST for every horizontal or vertical line depends on user‘s choise. For every line it saves parameters of the fitting and their error, which later are used for plotting a resolution to resolution ratio map. As a result user will have two consequent plots. Contour and contourf plot of R/Q map and resolution map. Both will be saved as $datafilename + H.pdf$ for resolution map using horizontal lines or $datafilename + V.pdf$ for resolution map using vertical lines. Results of its work are shown on Fig. 2

3.1 Input data and starting command

Starting command:

python3 init.py /home/skye/Programs/myscripts/CSTDATA/GSICav/Test/90x34-10x3-0.09.txt h 0.04 0.14 21 -0.015 0.015 21

3.2 Results of the program



(a) R/Q map from example data.

(b) Resolution map example.

Fig. 2: R/Q and resolution maps