

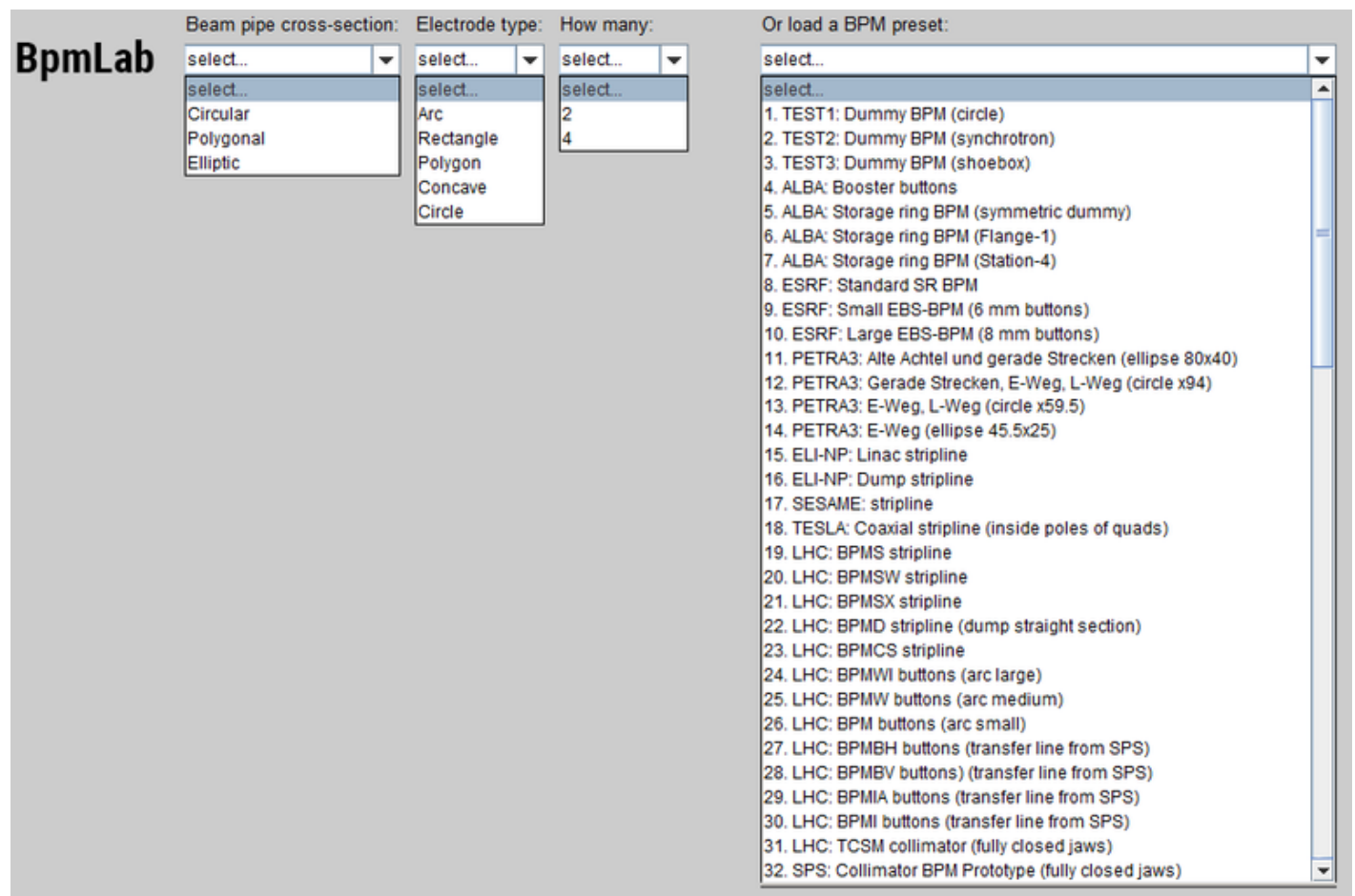
BpmLab

BpmLab is a 2D finite element-based software for Matlab aimed at simulating the electrode response of Beam Position Monitors (BPMs) of arbitrary geometry to emulated beam excitation. The code utilizes an open-source tetrahedral mesh generator DistMesh, combined with a short implementation of an electrostatic FEM with linear basis functions and boundary electric potential excitation (ref1, ref2). BpmLab offers various methods to correct the geometrical nonlinearity of the response map by calculating linear scaling coefficients, 1D/2D polynomial coefficients, or directly inverting voltages-to-positions using numerical optimization.

It can also model the electrode misalignment, distort individual electrode signals (add noise, attenuation or amplification), or import actual BPM voltages to apply correction methods to the resulting dataset.

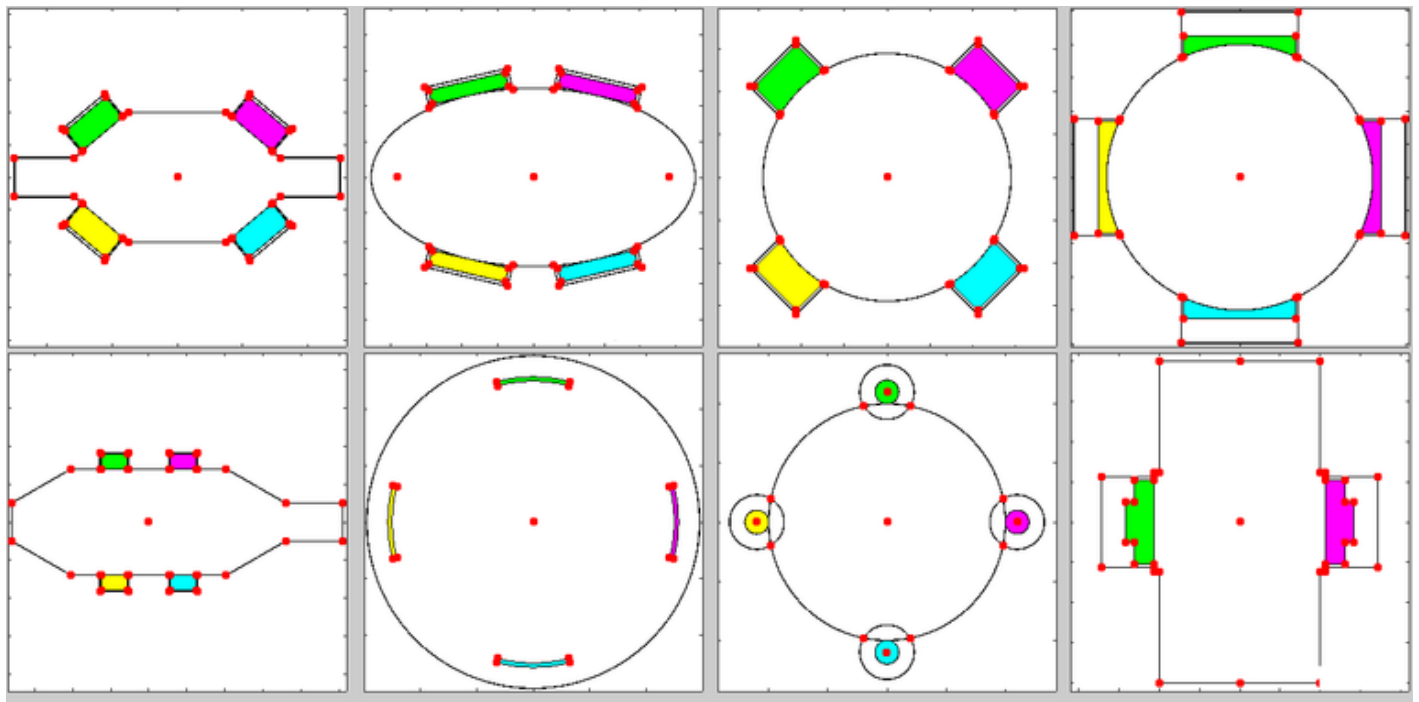
The main intention for BpmLab is to provide a fast and robust way to observe and treat the nonlinear distortion of a BPM, and check or compare the performance of chosen correction methods over signals from perfect and imperfect BPM models or from real BPMs. All results calculated with BpmLab are stored in Matlab's workspace and can be accessed for further post-processing.

BpmLab has an intuitive interface for geometry input: it offers combinations of several shapes, most common to BPMs, to construct 2D cross-sections of vacuum chambers and electrodes. As a useful jump-start, many real-life BPMs are directly available as editable presets:



After selecting the shapes or the preset, all geometrical parameters are accessible and changeable in the parameter tree of the GUI.

Several possible BPM geometries are shown here but BpmLab is not limited to them:



Check out the screenshots below to see what the tool is offering precisely and how to use it.

Package info

Download: BpmLab_v123

Current version: 1.23 (Feb 10, 2020).

Main file: Open Matlab, find and right-click on *BpmLab_v123.p* and select RUN.

Requirements: standard installation of **Matlab v2014b up to v2019b**. **Toolbox-free**. Tested and works under Win7x64 and Linux. May be slow and show java-related warnings under v2019.

Benchmarked: against analytical wall current method, boundary element method, electrostatic and wakefield solvers of CST, time domain solver T3P of ACE3P.

Main reference (with details and benchmarks): A. A. Nosych, U. Iriso, J. Olle, "*Electrostatic finite-element code to study geometrical nonlinear effects of BPMs in 2D*", Proceedings of IBIC-2015, TUPBO47 [PDF].

Acknowledgements: Marek Gasior & Manfred Wendt (CERN BI), Guenter Rehm (DLS), Kees-Bertus Scheidt (ESRF) and Ubaldo Iriso (ALBA) for inspiration, discussions, and priceless feedback. Special thanks to Jan Olle for his valuable contribution to the project at its birth. BpmLab was significantly upgraded in 2018-2019 following extensive requests by ESRF to be used with commissioning of the EBS ring in 2020.

Demo

This demo explains the BpmLab GUI by modeling a sample BPM on a beam pipe of circular cross-section with 4x rectangular concave electrodes of 30° span angle each.

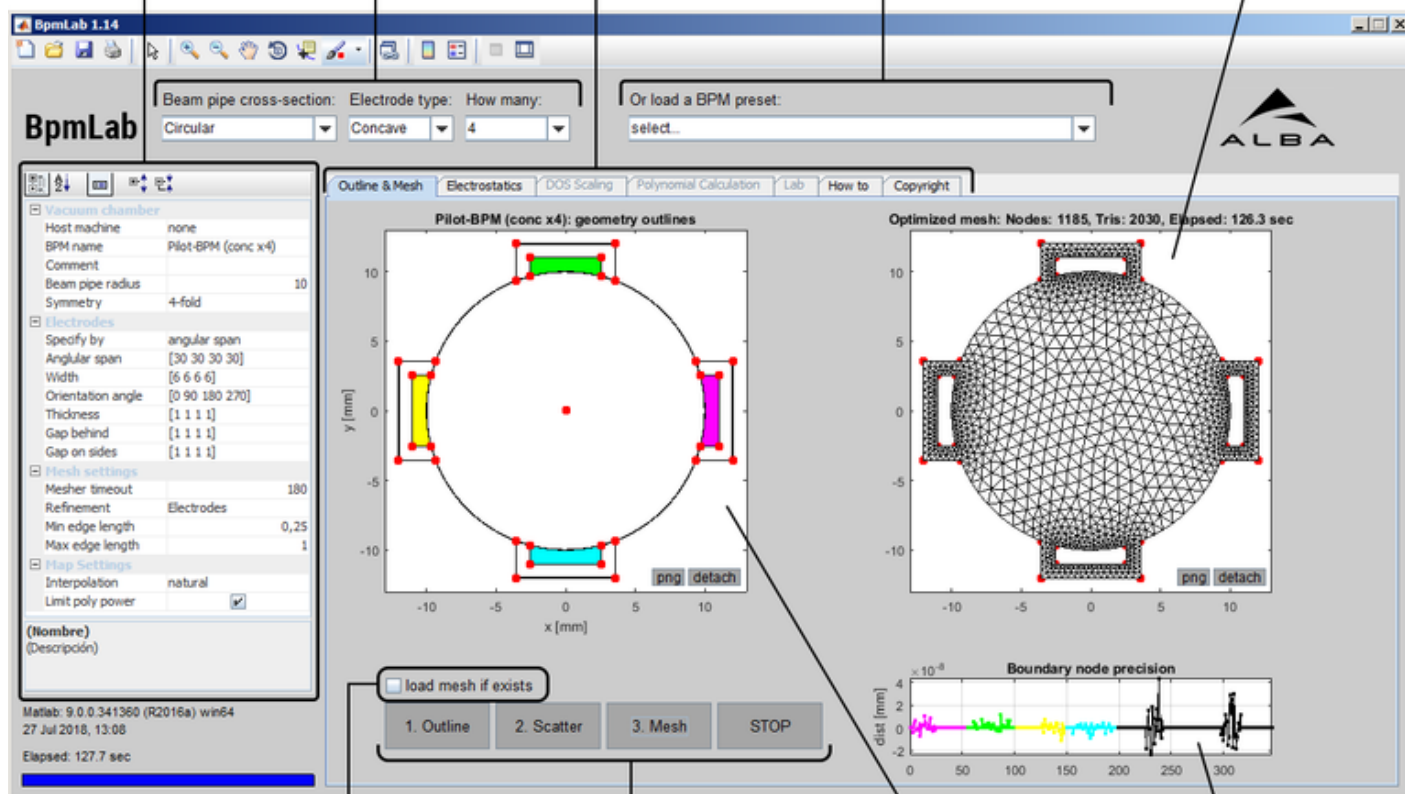
Left-hand property tree lists all changeable variables of the current BPM model

BPM modeling stages are grouped into Tabs

Right axes show real-time mesh optimization process and the final mesh

Selectors for initial geometrical layout of a BPM

Some existing BPMs are already designed and can be found in presets



Save some time by reusing a good mesh once it has been built

Step-by-step geometrical visualization and mesh generation

Precision plot of the boundary mesh nodes is useful to analyze the final mesh quality

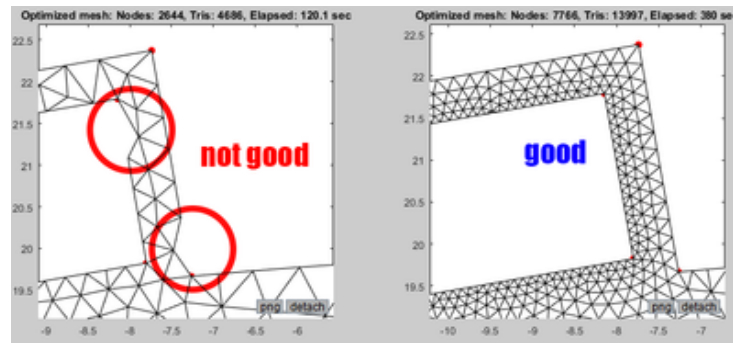
Left axes show geometrical outlines of a BPM with subsequent node spread for initial mesh generation

Geometry: After selecting the shapes combination (Circular+Concave+4) the default outlines are automatically created and can be further modified in the property tree on the left-hand side of the GUI. To apply any changes, press the Outline button.

It is important to note, that to ensure correct treatments of the signals, all electrode-related parameters must be entered in a particular sequence, corresponding to the electrode orientation: either *[Right, Top, Left, Bottom]* or *[RightTop, LeftTop, LeftBottom, RightBottom]*.

E.g. the Angular Span = [20 30 40 50] corresponds to [Right, Top, Left, Bottom] electrode having 20, 30, 40 and 50 degrees of angular span respectively.

Mesh: When the geometry is set, the 2D model then needs to be scattered with points (initial non-optimized mesh nodes) and meshed with equilateral triangles. The mesh needs a bit of time (and sometimes several attempts) to generate and converge. A uniform mesh can be used, as well as a variable mesh (recommended) by tuning the min/max edge sizes and areas of refinement, e.g. around the electrodes. After the final mesh is generated, it is important to give it a closer look and search for missing boundary elements:



If any, this can be corrected by re-meshing as is, or by tuning the *min/max edge length* and *mesher timeout* for generating a better converged mesh.

Electrostatic emulation of beam excitation

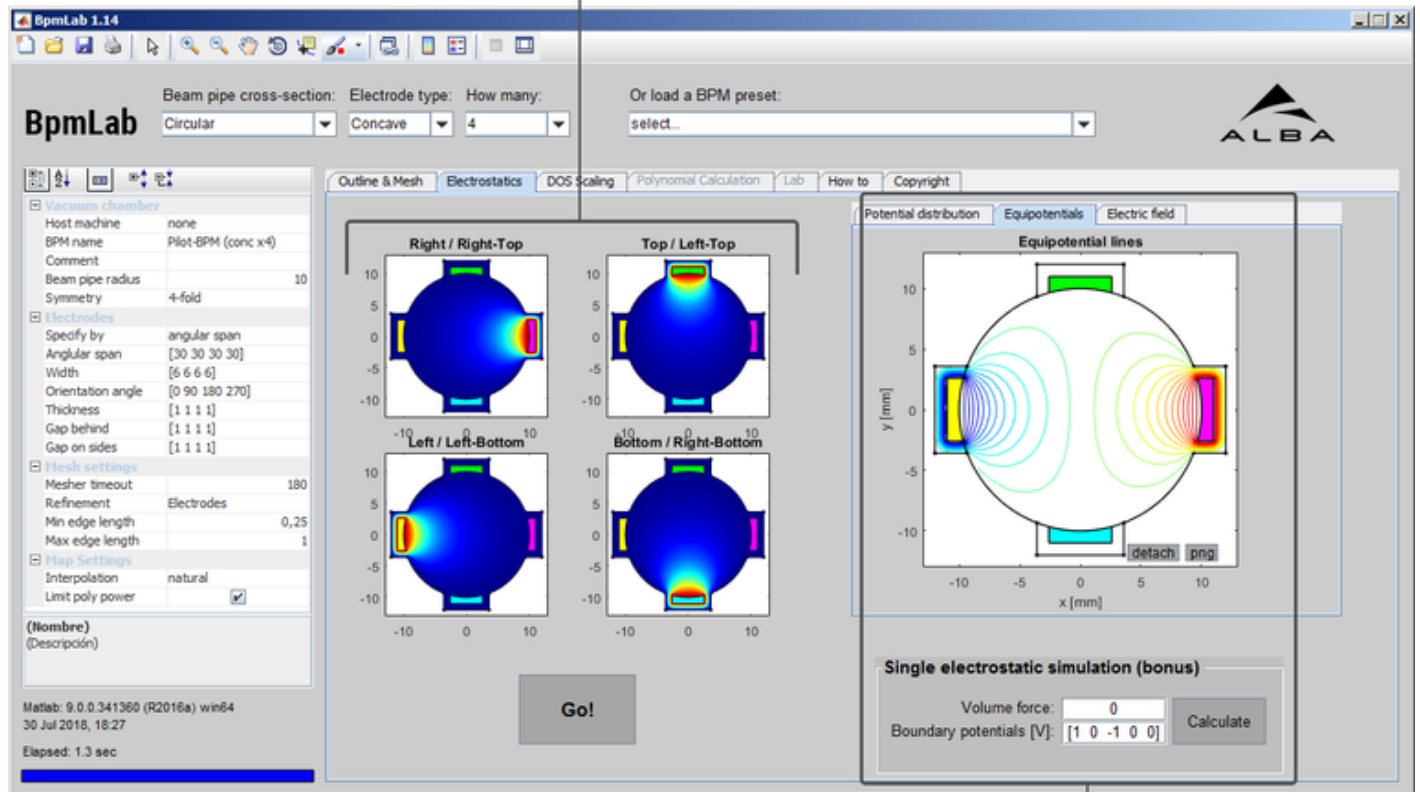
The following Poisson equation is solved in 2D with Dirichlet boundary conditions and $f=0$:

$$\begin{aligned} -\Delta u &= f & \text{in } \Omega, \\ u &= u_D & \text{on } \Gamma_D. \end{aligned}$$

The vacuum chamber is grounded, and the electrodes are excited with 1 V of potential one by one, while keeping the others also grounded. The FEM solver calculates the 4 electrostatic potential distributions in each mesh node based on these boundary conditions.

Using the *Green's reciprocity theorem*, which states that *the surface charge induced on an electrode due to a test charge at (x,y) is proportional to the potential at that same position when the test charge is absent and the electrode is excited by a potential V*, the BPM response is calculated for all possible excitation positions in only 4 calculations:

Combination of electrostatic potential
excitations of each electrode and grounded others
emulates beam excitation in any location in the beam pipe

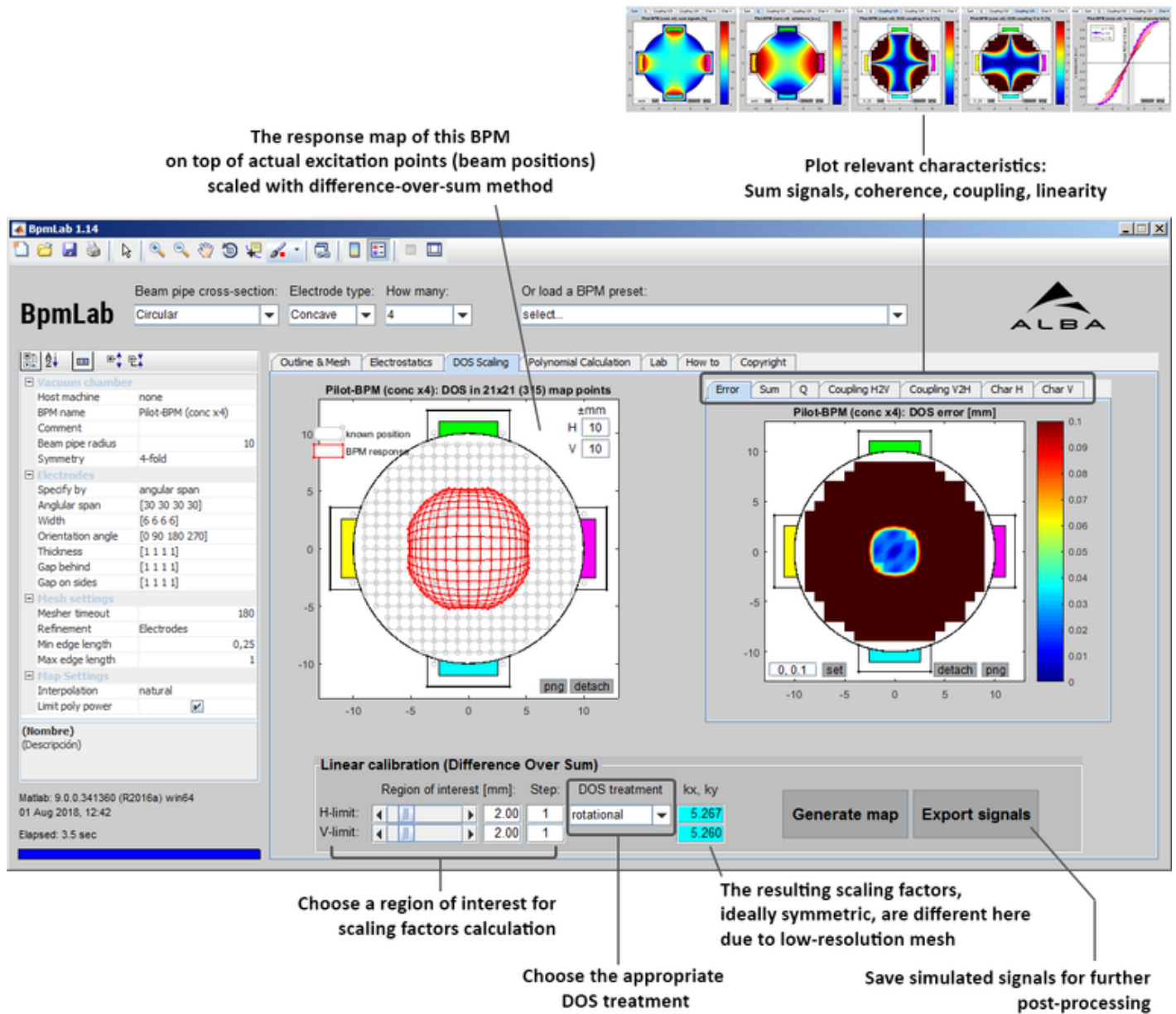


Electrostatic playground (unrelated to BPM studies)
for solving various boundary value problems with FEM.
This feature is here since the dawn of BpmLab

The BPM response has the form of normalized difference-over-sum (DOS) ratios of potentials. They are calculated in every mesh node inside the BPM geometry. The values beyond mesh nodes are found by interpolation.

Response map scaling

To calibrate the DOS ratios from $[-1, 1]$ to mm, BpmLab extract the linear scaling factors from the selected region-of-interest (ROI) area of the point map.



Several types of DOS treatment are available. The appropriate one would depend on the BPM symmetry type and purpose:

For BPMs with rotational 4-fold symmetry (circular beam pipes with electrodes centered at 0/90/180/270 degrees)

rotational:

$$x_m = k_x \times \frac{V_R - V_L}{V_R + V_L}$$

$$y_m = k_y \times \frac{V_T - V_B}{V_T + V_B}$$

log10:

$$x_m = k_x \times \log_{10} \left(\frac{V_R}{V_L} \right)$$

$$y_m = k_y \times \log_{10} \left(\frac{V_T}{V_B} \right)$$

For BPMs with mirrored 2-fold symmetry (skewed, elliptic, synchrotron-type BPMs)

mirrored:

$$x_m = k_x \times \frac{V_{TR} + V_{BR} - V_{TL} - V_{BL}}{V_{TR} + V_{BR} + V_{TL} + V_{BL}}$$

$$y_m = k_y \times \frac{V_{TR} + V_{TL} - V_{BR} - V_{BL}}{V_{TR} + V_{BR} + V_{TL} + V_{BL}}$$

diagonal:

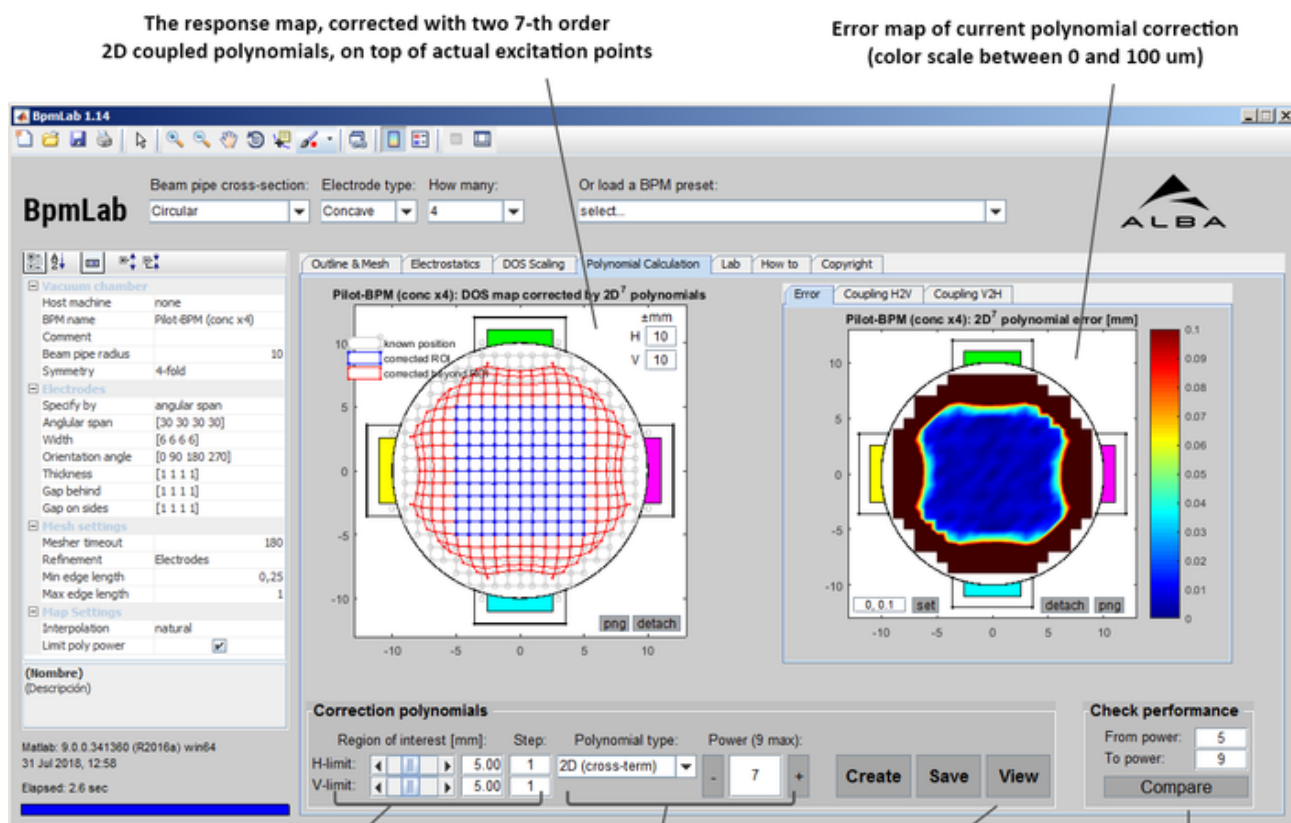
$$x_m = \frac{k_x}{2} \times \left[\frac{V_{TR} - V_{BL}}{V_{TR} + V_{BL}} - \frac{V_{TL} - V_{BR}}{V_{TL} + V_{BR}} \right]$$

$$y_m = \frac{k_y}{2} \times \left[\frac{V_{TR} - V_{BL}}{V_{TR} + V_{BL}} + \frac{V_{TL} - V_{BR}}{V_{TL} + V_{BR}} \right]$$

Response map correction with polynomials

1D and 2D polynomial coefficients are calculated in BpmLab for all possible powers (which depend on the ROI size and point density) to correct for horizontal and vertical nonlinear distortion of the BPM response.

Find the optimal polynomial set by selecting the ROI (or the beam acceptance area) and listing through all powers. Lower-degree polynomials are often sufficient for good on-the-fly correction:

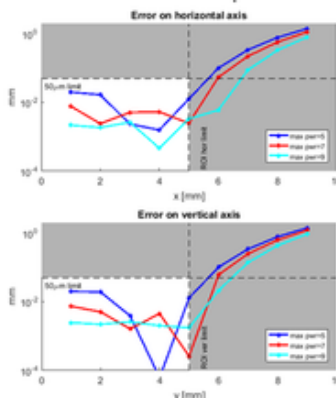
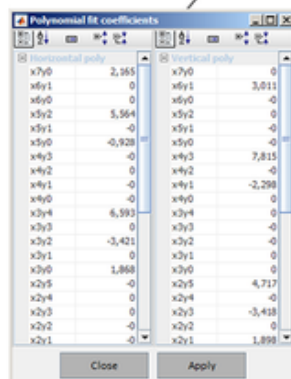


The correction polynomials are efficient within the selected region of interest (blue points). Beyond this area the beam position correction is not controlled (red points)

Choose the appropriate polynomial dimension and list through its orders for optimal correction

The coefficients can be viewed, exported or altered on the fly

Precise plots of error-on-axes for a range of polynomial powers for comparison



Lab tab: workbench for merging the BPM model with reality

The Lab tab is made for testing the robustness and performance of the developed BPM model on various data. It is thought to be used in three different ways:

A) Generate a perfect map (uniform or scattered), distort it with added noise, attenuation or amplification of BPM channels, and then observe effects of correcting it with 3 available correction types (DOS, polynomials, and voltage inversion);

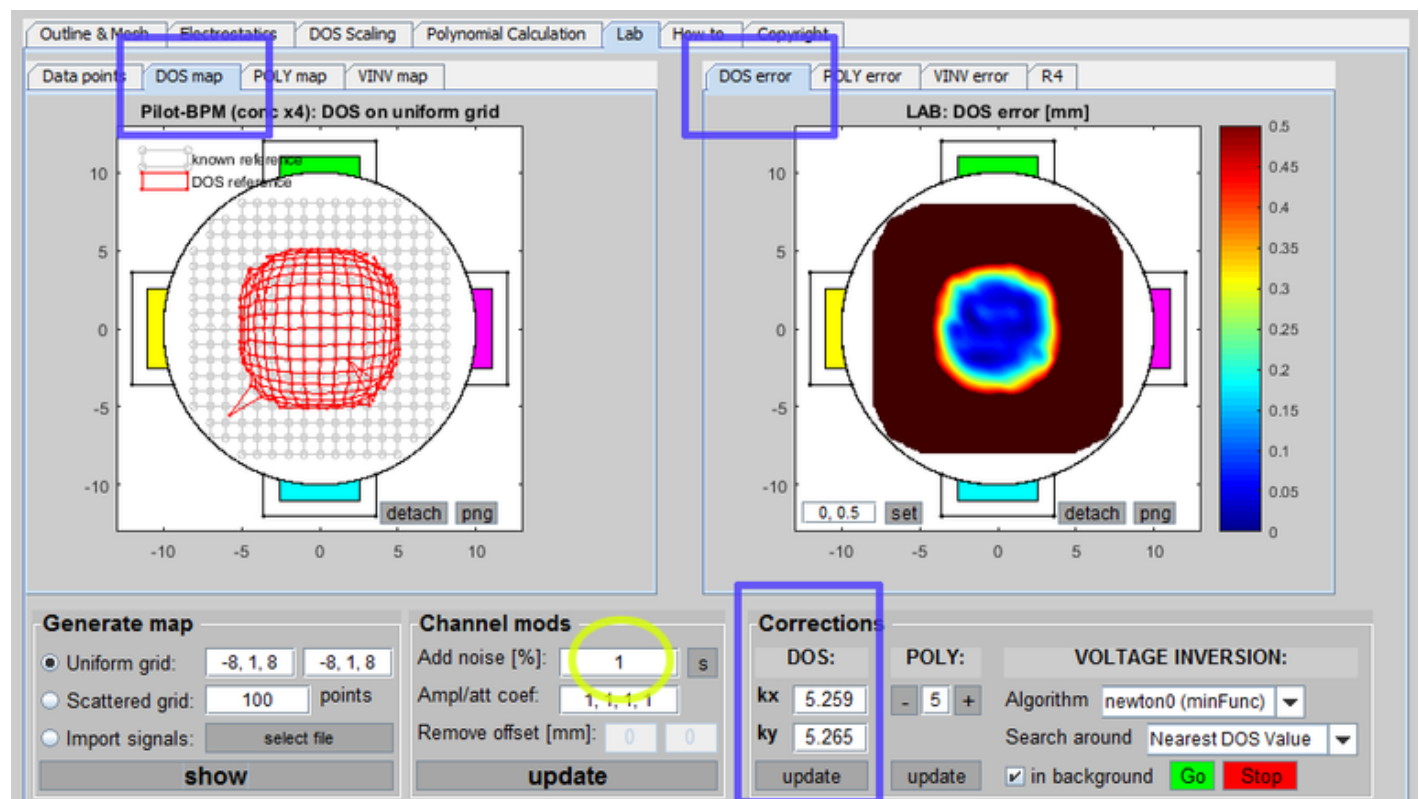
B) Build a geometrically "imperfect" BPM model (e.g. wry, tilted or asymmetric in some other way) and save its signals matrix. Build another "perfect" BPM model, import the "imperfect" signals and study how the corrections handle the imperfect signals matrix.

C) Import raw signals set (i.e. voltages or ADC bins) from any source, e.g. real BPM data, and correct them with a proper correction method. Can be useful to study bumps and aperture scans for large beam drifts.

Example 1: correcting distorted signals

We design a simple BPM with no imperfections. After all successful steps of meshing and DOS and polynomial calculations, we proceed to the Lab tab to play with this model.

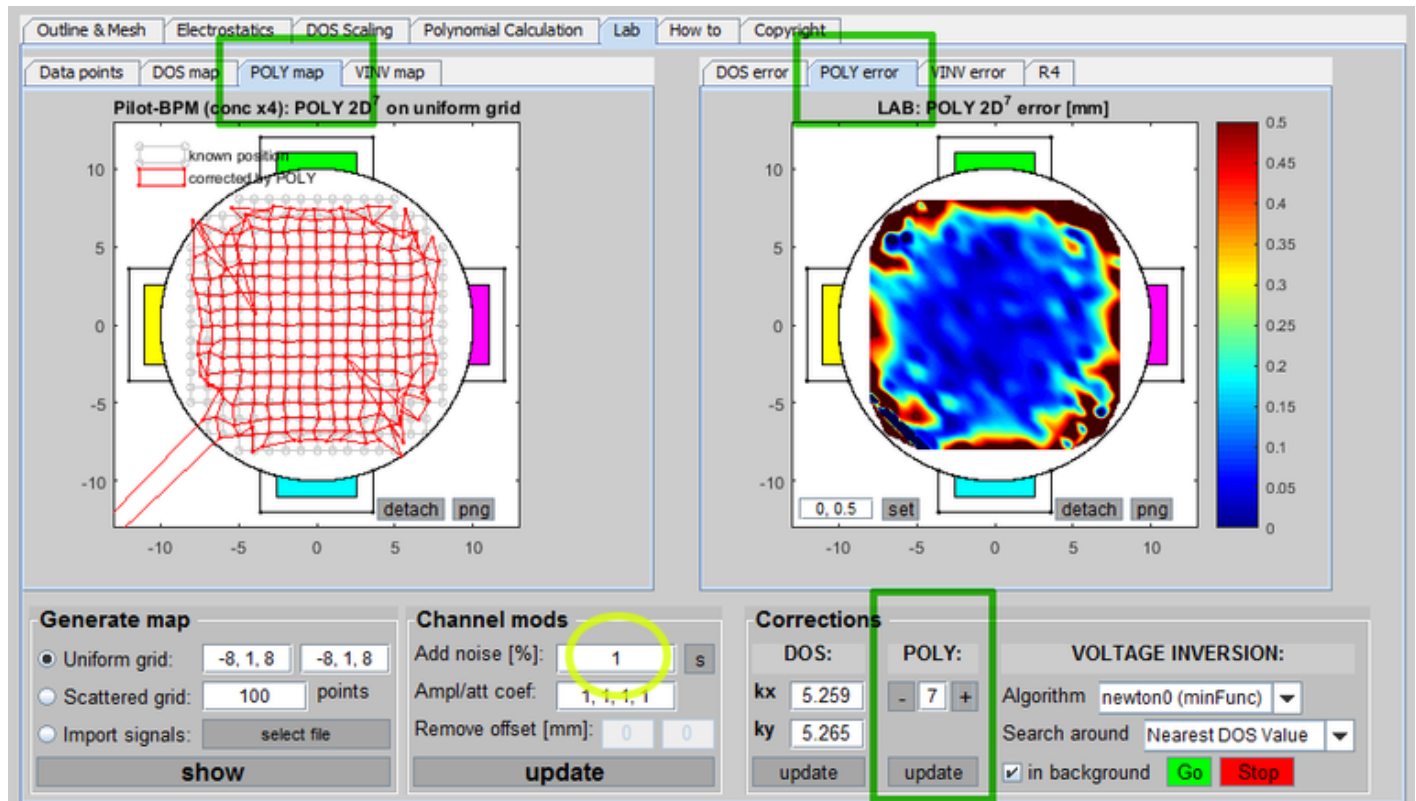
Here, a self-generated map is distorted with 1% random noise (meaning distorting signals from each electrode), and corrected with scaled difference-over-sum method using the previously calculated scaling factors:



Add noise: the noise values specified in percent, i.e. 100% = signal strength in the center. Value = [1,2,3,4] means it will add 1%, 2%, 3% and 4% to A, B, C and D correspondingly. Value = 1 means it will add 1% to each channel. Noise per channel is randomized with std(value).

We can also apply a 7-th order 2D polynomial to the noisy dataset. The polynomials were calculated for the perfect

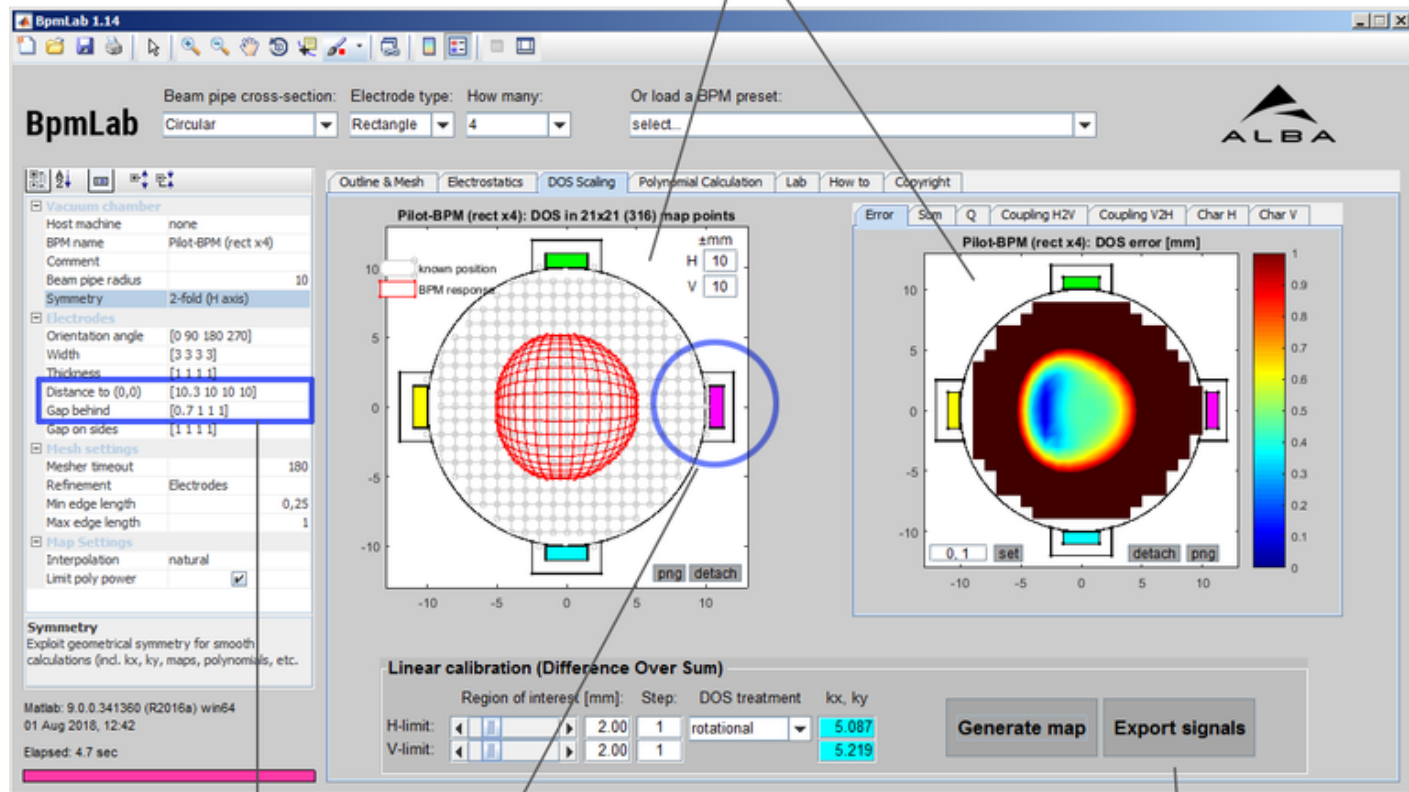
model, and we can list through different powers to see their effect on noisy data:



Example 2: studying button retraction (wryness)

Firstly, we design and simulate a **wry BPM** with the right button retracted by 300 μm . The effect of a single button retraction is clearly visible in the asymmetry of the response map:

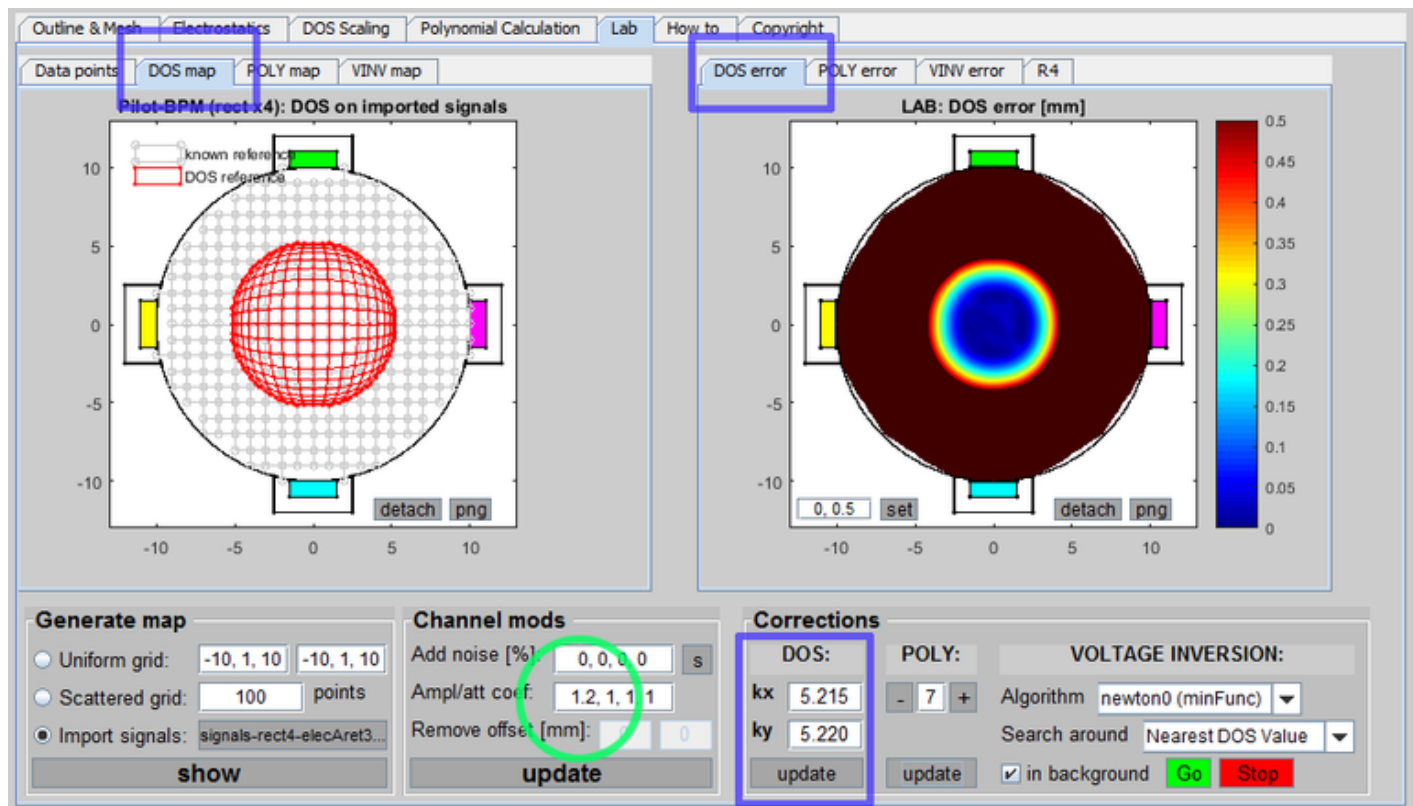
The button retraction results in
a "twist" of the response map and its error



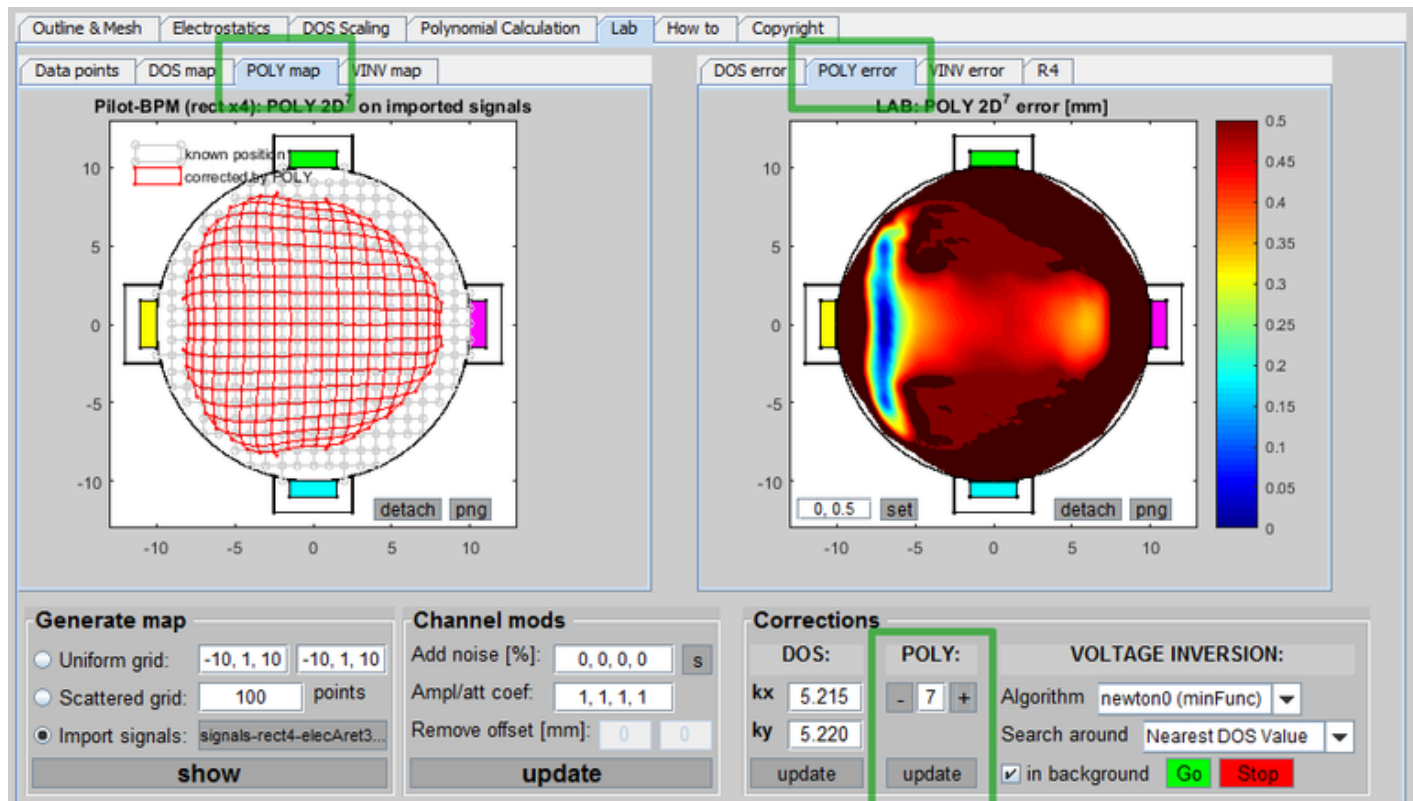
Build a model of an imperfect BPM with
the Right button being retracted by 300 um

Export these signals for further studies

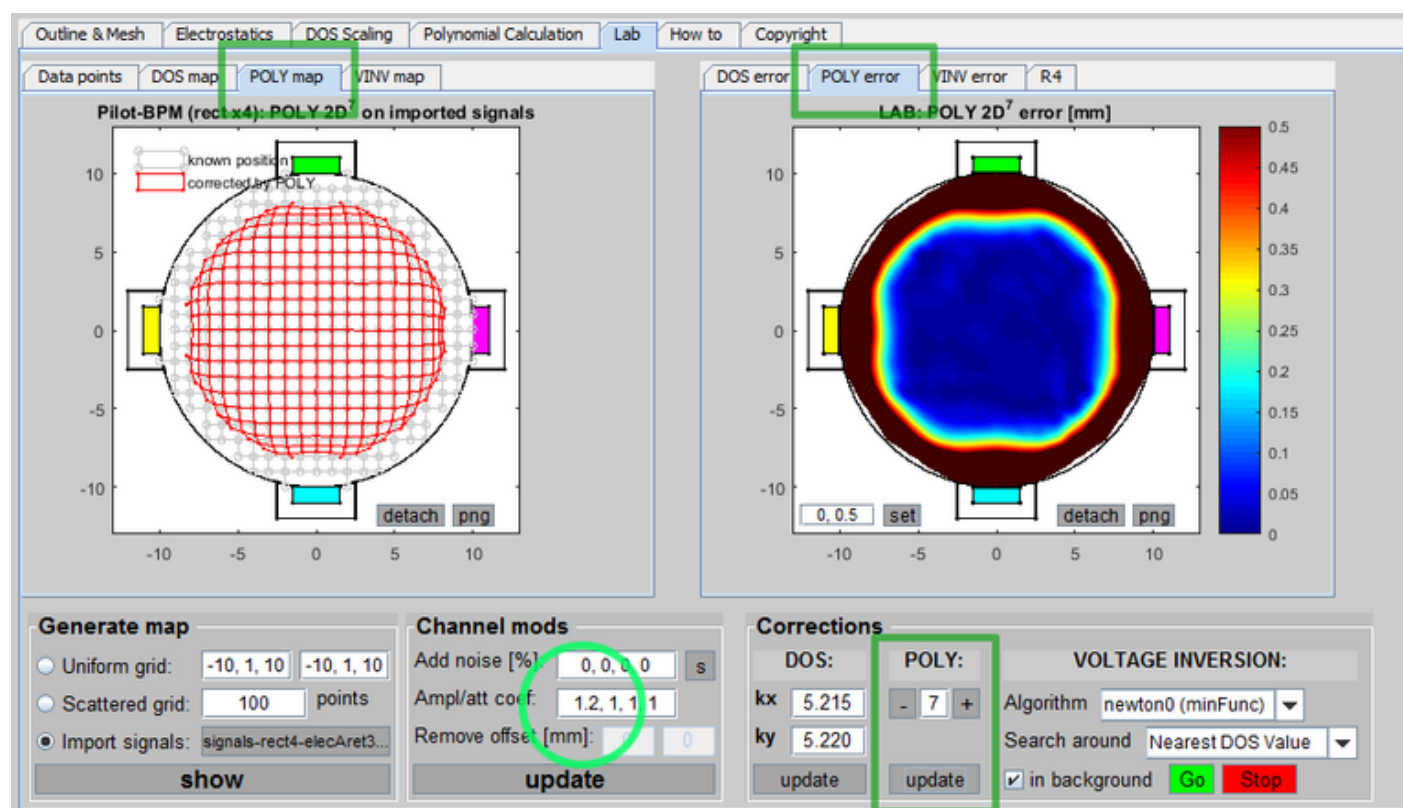
We save the signals of this imperfect design and proceed to making a new **"perfect"** model of this BPM with **all buttons aligned**. In the Lab tab of the perfect model, we import the wry signals and compensate the retraction effect by picking and adding an amplification coefficient of 1.2 to the first channel. It results in correcting the symmetry of the DOS map:



For comparison, let's see how the polynomials, calculated for the perfect model, can handle it. First, the wry dataset is corrected by 7-th order 2D polynomials:



After adding the same amplification coefficient, the result becomes much better:



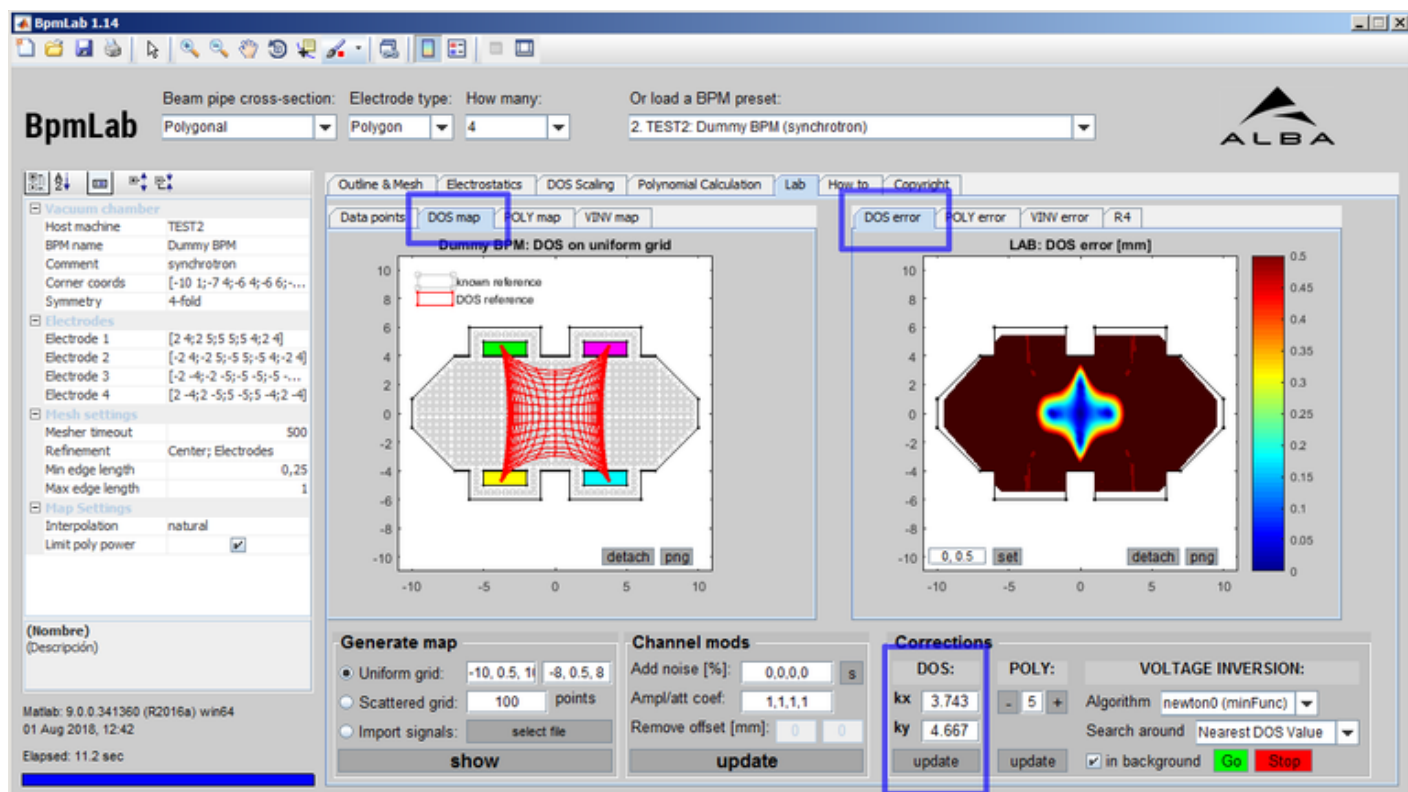
Example 3: direct voltage-to-position inversion, including real BPM data

Another way of correcting the geometrical nonlinear distortion of BPM signals is using numerical optimization to iteratively invert voltages to beam position. This approach is called "*Newton optimization*" and is much more accurate than DOS scaling or polynomial correction. It only requires a model behavior of a BPM, which in case of BpmLab, is the electrostatic model.

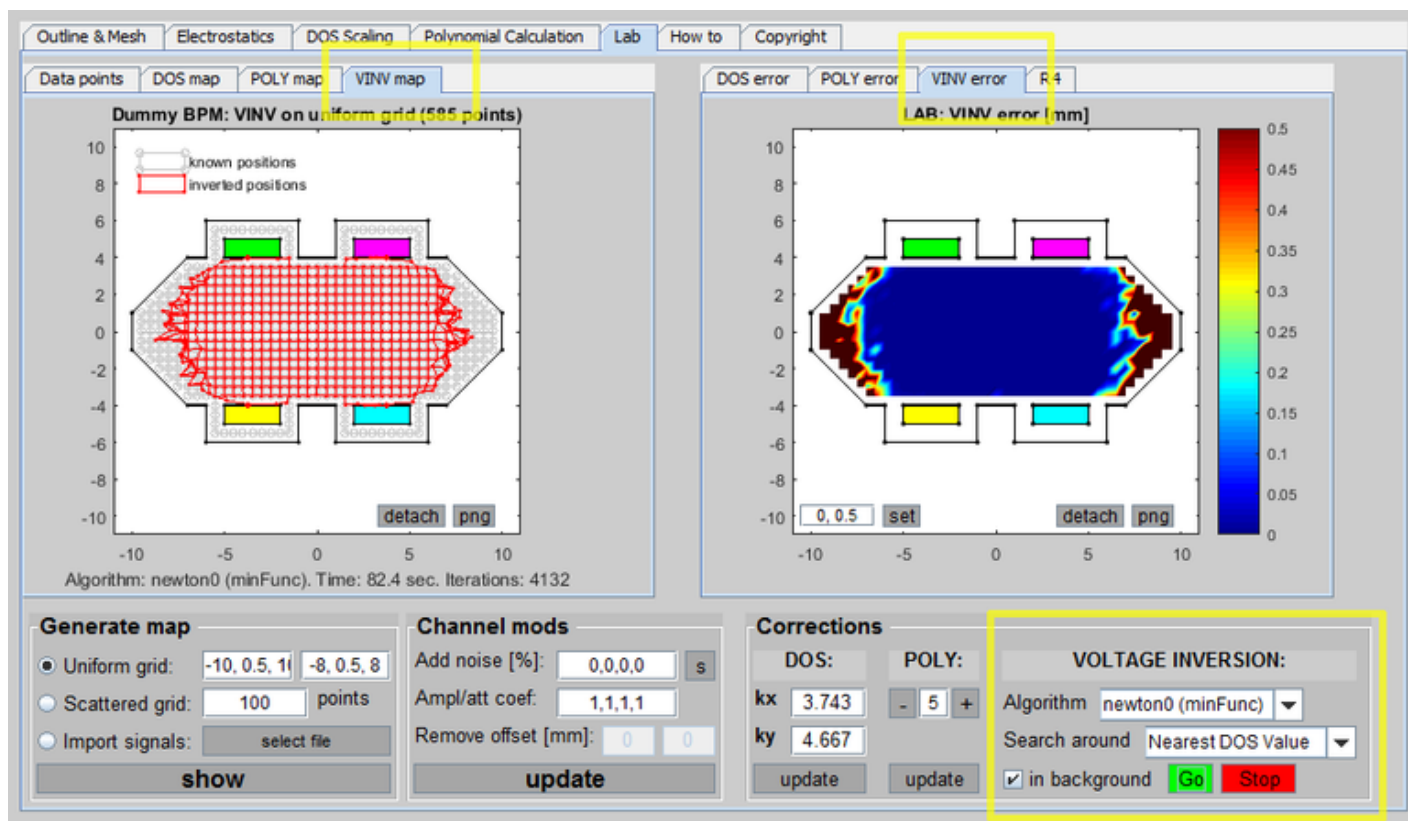
The inversion process can not be used on-the-fly: numerical optimization requires time to minimize the target function for each set of 4x voltage signals per beam position, so voltage inversion is an offline way to accurately post-process BPM data.

A standard desktop PC with quad-core 3GHz CPU, 8 Gb RAM, and Matlab is able to perform up to **10 inversions per second** for beam positions away from highly-nonlinear regions of the response map (those regions take a bit longer to crunch).

To demonstrate the performance of this method, we build a model of a simple synchrotron BPM and compare the performance of DOS scaling



with the resulting map of positions inverted directly from button signals:



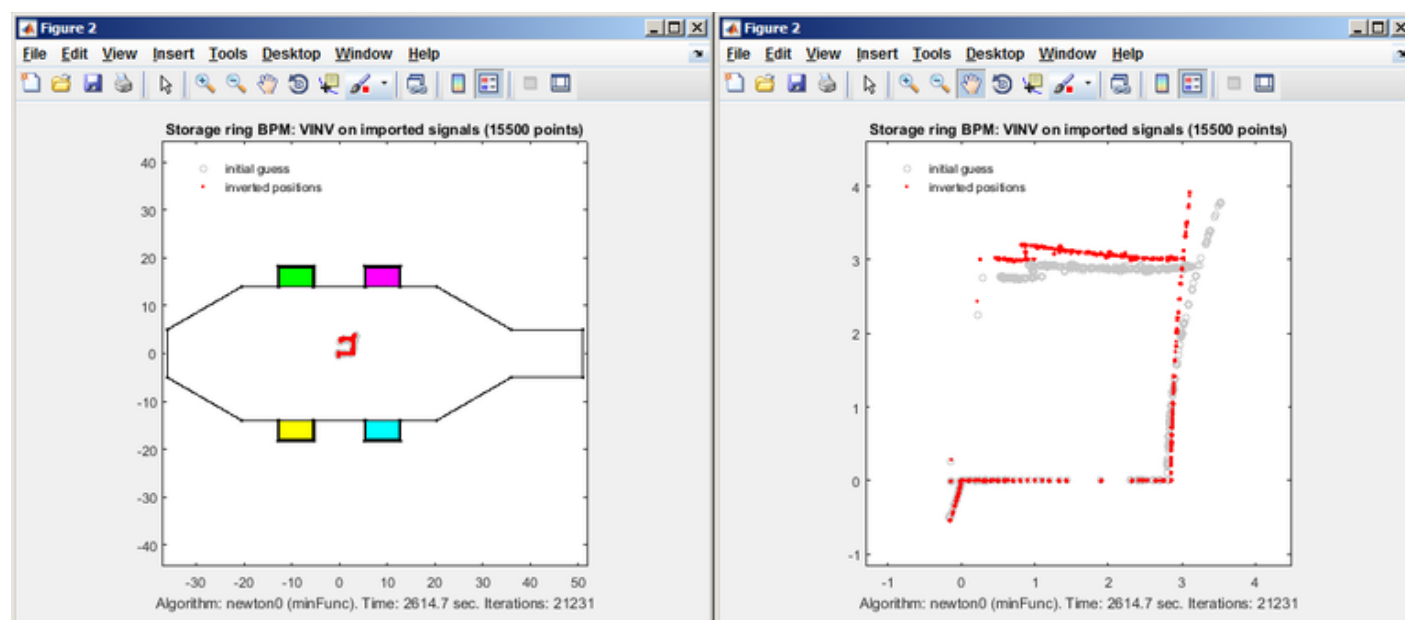
This example was tested on self-generated BPM signals, so this is sort of a **backwards convergence test** of BpmLab. Here, the voltage inversion is capable of resolving signal differences far away from the center of the BPM to invert them back to original positions with machine precision. The area of effective inversion is usually within 90% of the total BPM area, which by far outperforms the DOS or polynomial correction.

The default optimization algorithm in BpmLab is *limited-memory BFGS* (a quasi-Newton algorithm, supplied with exact Jacobian matrix and a low-rank Hessian approximation) from a powerful third-party optimization toolkit *minfunc* (ref). It makes BpmLab toolbox free; however, it is still possible to use the Matlab's original optimization toolbox and its options in the "Algorithm" dropdown menu. To date, the use of *minfunc()* has always been showing better performance with fewer iterations and less elapsed time, as compared to the built-in optimization toolbox leading to the same result.

Application to real data

1) The voltage inversion was applied to off-line BPM data acquired from beam bumps with 4 orbit-corrector magnets in the ALBA synchrotron storage ring. The beam was forced to have a maximum offset of $[+3,+3]$ mm in several H and V steps of ~ 100 μm . At these offsets the BPM response starts to enter the nonlinear behavior so we can already observe and correct its effects:

The images below show the beam transverse path as it was doing bumps. During this test the voltage signals were acquired from one particular BPM in slow acquisition mode, and then inverted with BpmLab.

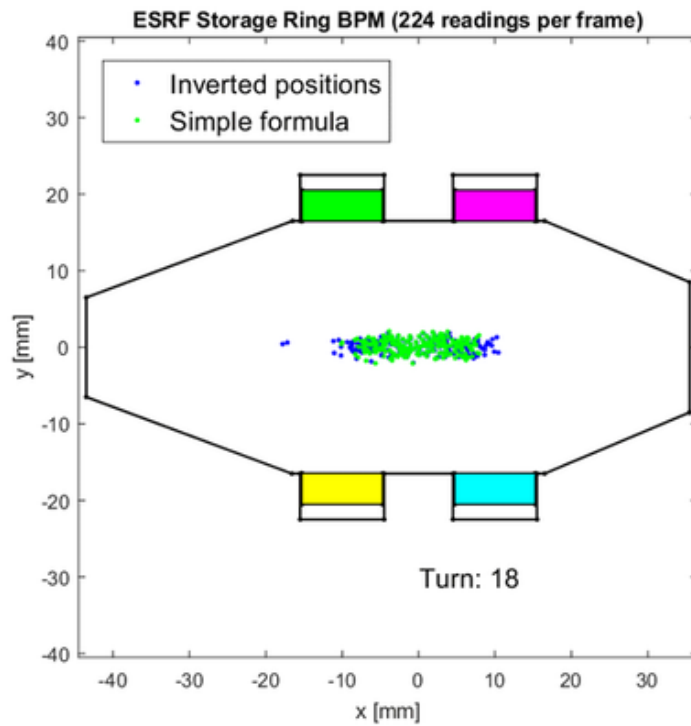


In grey, the DOS response (also used as the initial guess for voltage inversion) follows the non-straight pincushion distortion lines, typical for such BPMs.

In red are the positions, inverted from voltages. The most significant difference is seen when the beam was bumped up from the rightmost location. The path also starts to bend slightly as it gets towards the farthest diagonal position, however, this is where the kick of the corrector magnet also becomes nonlinear.

2) Injection into the ESRF storage ring with RF off. In this animation the beam is making 58 spiraling turns with the RF system being OFF, after finally crashing towards the inside of the chamber. The animation combines data from turn-by-turn measurements of all 224 BPMs in the ring.

The DOS response (in green) is plotted vs. the inverted positions (in blue) which show more realistic beam path as it spirals inwards:



(click on the image to see the animation)

Such big spread from BPM to BPM (spread between points per frame) is due to beam orbit oscillations increasing in amplitude until beam is lost.

Your feedback is needed

This tool has potential for further development and expansion, but its growth will depend on the response and needs of the accelerator diagnostics community.

The author is grateful for all feedback from the users of BpmLab and is open to any suggestions, ideas and comments. For any inquiry please contact **Andriy Nosych** at anosych@cells.es.

Copyright © 2024 CELLS

Feedback - Contact