POWDEr : Physics Online Visualization & Verification Data Explorer

Manfred Klaffenböck*
University of Vienna

Philipp Sturmlechner†
University of Vienna

ABSTRACT

POWDEr : Physics Online Visualization & Verification Data Explorer.

Index Terms: H.5.0 [Information Interfaces and Presentation]: General—Experimentation; J.2 [Computer Applications]: Physical Sciences and Engineering—Physics; G.1.2 [Numerical Analysis]: Approximation—Linear Approximation;

1 INTRODUCTION

This project was motivated by Dr. Charles Rogers. Dr. Rogers does powder X-ray diffraction on crystalline structures filled molecules with a certain electrostatical behaviour. He attempts to create materials with interesting characteristics but cannot be sure what the molecular structure looks like until he examined the scattered X-ray spectrum and found a crystal and molecule which generates the same spectrum in a simulation.

Our tool is meant to help with the search by allowing the user to sample the parameter space, calculate discrepancy to the measurement, look at the spectrum, and look at the four dimensional parameter space in a comprehensible way.

The users of this tool are on the one hand all researchers who need to compare measurement data in the form of a spectrum to simulation data, where the simulation takes multiple input parameters. On the other hand scientific data often needs to be made public and a tool like this one could help present the findings to a broad audience.

There are two main data files needed by this application. One must describe the measurement, it takes the form of a .cvs file. The other is the configuration for the simulation and depends on the simulation implementation. Usually a description of the crystal and one of the molecule is needed.

2 RELATED WORK

Our tool is certainly mostly related to a tool called Tuner (see [2]). Tuner is a tool especially developed to explore multidimensional parameterspaces. One of the key differences between Tuner and POWDER is that Tuner uses a rather small number of random sample points, whereas POWDER uses 10000 exactly calculated cartesian samplepoints. Tuner uses for interpolation a gaussian process model, whereas POWDER uses linear interpolation between these 10000 samplepoints. Another difference, more on a lowlevel implementation aspect, is that Tuner is a Desktop tool, but POWDER is an online tool. The idea is to make collaboration between locally separated researchers easier.

Another tool which we drew inspiration from is World Lines (see [3]). This tool also tries to combine the exploration of a multiple parameter space with the detailed depiction of a certain set of parameters. It differs from our tool in the way that it is actually designed for timed data and realtime scenarios, whereas POWDER is mainly for static data and the comparison between sampled and measured data.

Even though not directly related work, we took inspiration for the design of the hypersclices from the paper Continious Scatterplots (see [1]). In this paper the authors suggest to colorcode the density of a certain element in form of a heatmap. We took this basic idea and use it now to represent the χ^2 value on the hyperslices as a colorvalue mapped onto a heatmap.

3 APPROACH

Our design evolved over the duration of development from a detail only to an overview first approach. The key to this tool is the navigation of the four dimensional sample space. On the left side of screen there is a view that shows your current location in this high dimensional space via hyperslices. You can also see a very reduced piece of information about every other point in the slices in the form a goodness of fit metric.

Once a point has been chosen the user shifts her attention to the right side, where a detail view is shown. The exact model at the chosen point gets calculated is is overlain by the measurement data. Differences are easily visible, each peak contributing to the simulation output can be marked and thereby tracked should the user choose to pick a new point in the parameter space.

The reason for the shift from detail only to overview was partly the papers mentioned in related work but for the most part a discussion with an experianced visualization professor who helped us see the main challenge the user has to overcome in the whole process.

4 IMPLEMENTATION

4.1 User interaction

The user has to provide the following:

- a .pbd file, representing a crystal
- a .xyz file, representing a molecule
- a script, representing the model which operates on the crystal and molecule files
- a .csv file, containing the results from the X-ray powder diffraction experiment

The model is dependent on the crystal and the molecule to calculate the reflection for this pair. In order for the model to work you have to specify four parameters:

- Rotation: the rotation of the molecule inside the crystal.
- Depth: how deep is the molecule inside the crystal
- Density: how high a proportion of the wells in the crystal are actually populated with molecules
- Crystal wall rotation: if the molecule is big, the crystal wall can be distorted, which can be described as a rotation.

*e-mail: manfred.klaffenboeck@univie.ac.at
†e-mail: a0802706@unet.univie.ac.at

Within these 6 hyperslices, you can drag a cross around to vary the parameters. A line in the scale to the left shows the colorvalue of the current selection, representing the goodness of this particular point. Once the selection is done and the mousebutton is released, an asynchronous call to the server is undertaken, sending the current selection of parameters. For these parameters, an exact calculation is done and the results are sent back to the client. These are then represented in the detailview for further inspection.

One of the shortcomings in formerly existing visualization solutions for this problem was, that individual peaks can not be seen the entire time, since it would leave to visual cluttering. So we extracted each peak into its own little window, where they can be spotted individually the entire time. You can hover over and select these mini peaks, in order for them to appear in the main window, together with the graphs of the overall reflection ((see Figure 3).

4.2 Notes on interpolation

We do cartesian sampling with 10 sample points per parameter, yielding 10000 samples in total.

The result is an array of peaks (in our case 32, dependent on the researched crystal and molecule) corresponding to refraction angles in the experiment, that is further compared with the according peaks in the experimental data.

In a next step the model has to be sampled at exactly the same points where the experimental data was taken. These are in our case 1041 points.

The comparison of goodness between the two (sampled and measured) is done by a χ^2 test. All the individual sample points are evaluated against this test and the result is a scalar value, indicating the goodness of the chosen input params against the measured experimental data. The value of a single χ^2 test gets mapped to a value between 0 and 1, where 0 indicates worst correlation and 1 indicates perfect correlation.

The four dimensional space, described by the four formerly described variables, is displayed as 6 hyperslices, where each slice represents one unique hyperplane. Each slice shows the variation in goodness over two dimensions, leaving the remaining dimensions fixed. The scalar value is represented in form of a continuous colorvalue, where the colors are mapped onto a heatmap. Since there are only a finite number of sample points, but an infinite number of possibilities, some form of interpolation between the samplepoints must be undertaken. Because of the rather vast amount of sample points, we chose to do linear interpolation, because we thought it might yield the most approximate results.

4.3 Architectural desicions

Since this project was planned for web browsers, we had to go with a client server architecture. On the front end there is no choice as to what language should be used, JavaScript is the only one with wide support. jQuery was used for the ease of selection and simple HTML manipulation. D3 helped with the drawing of the graphs. The parameter exploration part of the view uses webGL to directly render on the graphics card and do interpolation between points. For the markup we included Twitter Bootstrap.

On the server side we decided to use python and the django web application framework. It allows easy database access, generating web pages from templates, with it's own template language, and session management amoung other things. We chose to use the sqlite3 database since it enjoys native support from python.

The sampling of the parameter space was to slow in python and therefore had to be implemented in C with python bindings via ctypes. The C implementation uses OpenMP to fully utilize the CPU powder for the most costly operations (very low sampling frequency still yields approximately 320 million sampling steps).

On the server side the parameter space sampling was by far the most extensive task. Bulk reads and updates from and to the database were the first steps to make these task run in reasonable time, this moved the bottleneck from disk access to computation time. An implementation in C with python bindings helped to reduce the runtime from that point. Finally the usage of OpenMP to use all available cores gave a quadruple speedup and made the sampling feasible.

5 Results

Step one the user picks the experiment data set and the simulation sampling batch run she wants to compare (see Figure 1).

Step two the user navigates the parameter space to a point she is interested in, the view on the right side updates on the fly (see Figure 2).

Step Three the user shifts her attention to the right side and selects a peak she find interesting by clicking it's miniature representation on the bottom or clicking the peak directly (see Figure 3).

Step Four the user picks other positions in the paramter space by clicking or dragging the haircross in the hyperslices and watches the peaks vary. The selection makes sure the user can still easily track the chosen peak (see Figure 4).

5.1 Further notes

Due to the implementation in WebGL on the frontend and C on the back end the performance of the system is amazing. Even on a three year old laptop the transitions of hyperslices in the frontend run absolutely smoothly.

6 Discussion

One of the major strengths of the implementation is it's performance. It runs smooth on a laptop, it is implemented in a browser and thereby platform independent and the server does a lot of the heavy lifting reducing system requirement for the user further.

One of the weak points of our implementation is that it is tailored to this specific usecase, extending the current implementation will proof to be a difficult task.

The main lesson of this project is focus on the major problem. Try to see what the user really has to accomplish and where she could struggle. The shift in design gave us this insight. Besides that we found that Python with the Django framework and a JavaScript front end application play very nice together. Dividing responsibilities into back end and front end worked very well, we would opt to divide workload this way again.

Acknowledgements

The authors wish to thank Charles T. Rogers and Torsten Möller for their support during this project. They would also like to thank Thomas Torsney-Weir and Mike Philipps for some very good technical suggestions regarding the effective implementation of the heatmap.

References

[1] S. Bachthaler and D. Weiskopf. Continuous scatterplots. *Visualization and Computer Graphics, IEEE Transactions on*, 14(6):1428–1435, Dec. 2008.

[2] T. Torsney-Weir, A. Saad, T. Moller, H.-C. Hege, B. Weber, and J.-M. Verbavatz. Tuner: Principled parameter finding for image segmentation algorithms using visual response surface exploration. *IEEE Transactions on Visualization and Computer Graphics*, 17(12):1892–1901, 2011.

[3] J. Waser, R. Fuchs, H. Ribicic, B. Schindler, G. Bloschl, and M. Groller. World lines. *Visualization and Computer Graphics, IEEE Transactions on*, 16(6):1458–1467, Dec. 2010.

Figure 1: select your data

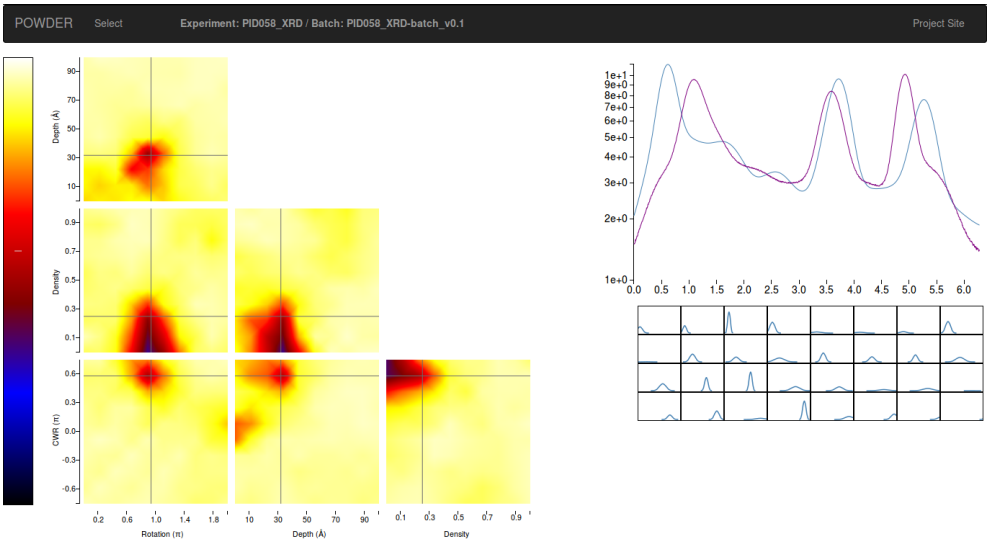

Figure 2: explore the parameterspace

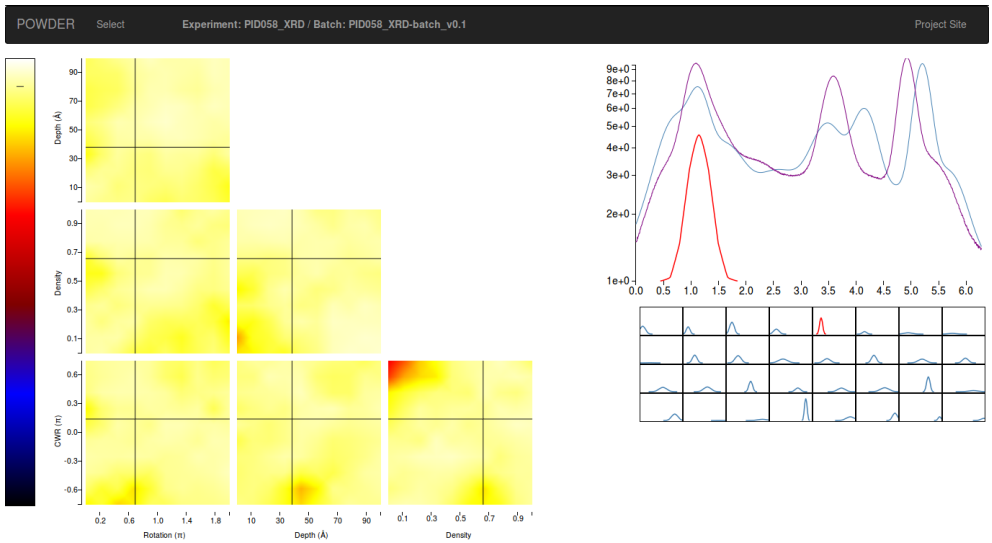

Figure 3: inspect your selection

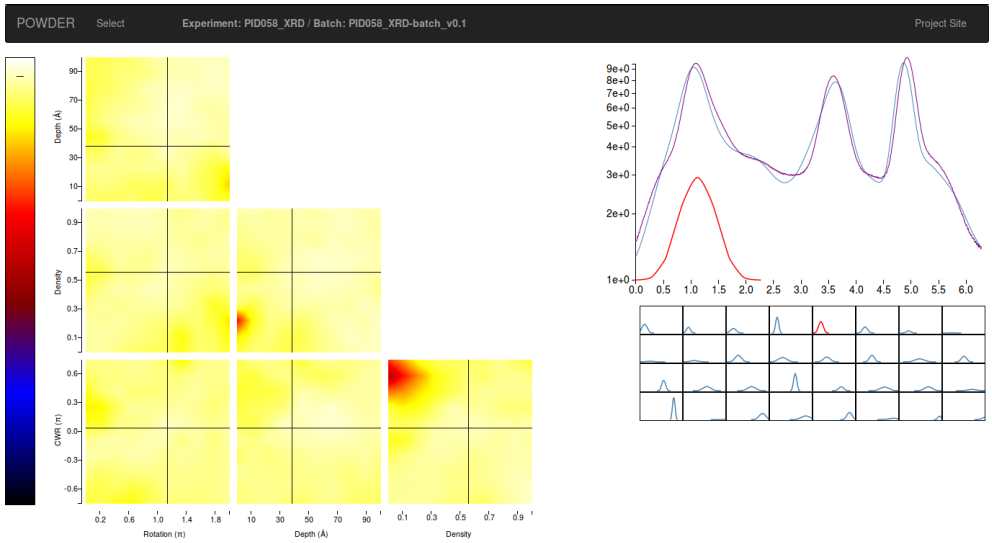

Figure 4: do further exploration