# Algorithmically Efficient Ray Tracing for the Simulation of Wall Heating in Particle Accelerator Structures

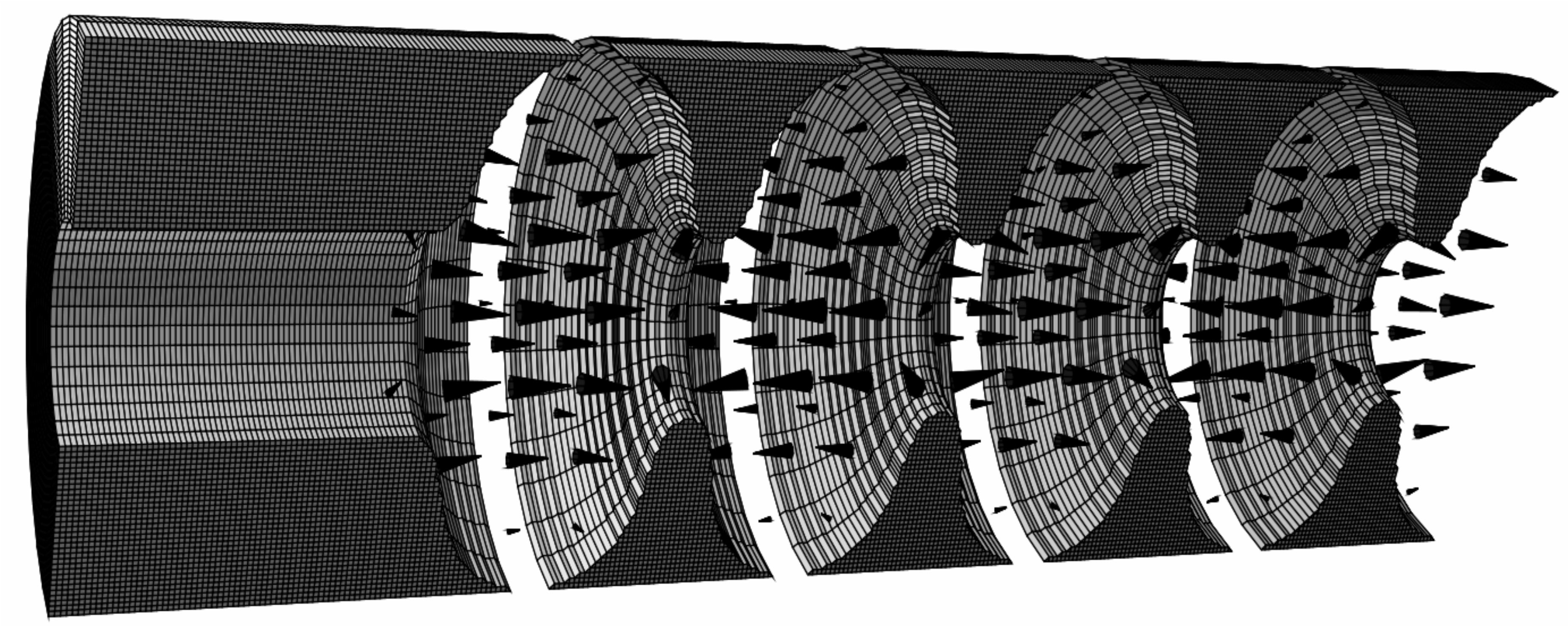
Eike M. Scholz

# Outline

- What are wake fields
- Special problems of high frequency wake fields
- The Cryoloss 2 project  overview
  - Construction overview
  - Tracing
  - Evaluation
  - Parallelization
- Some results
- Questions

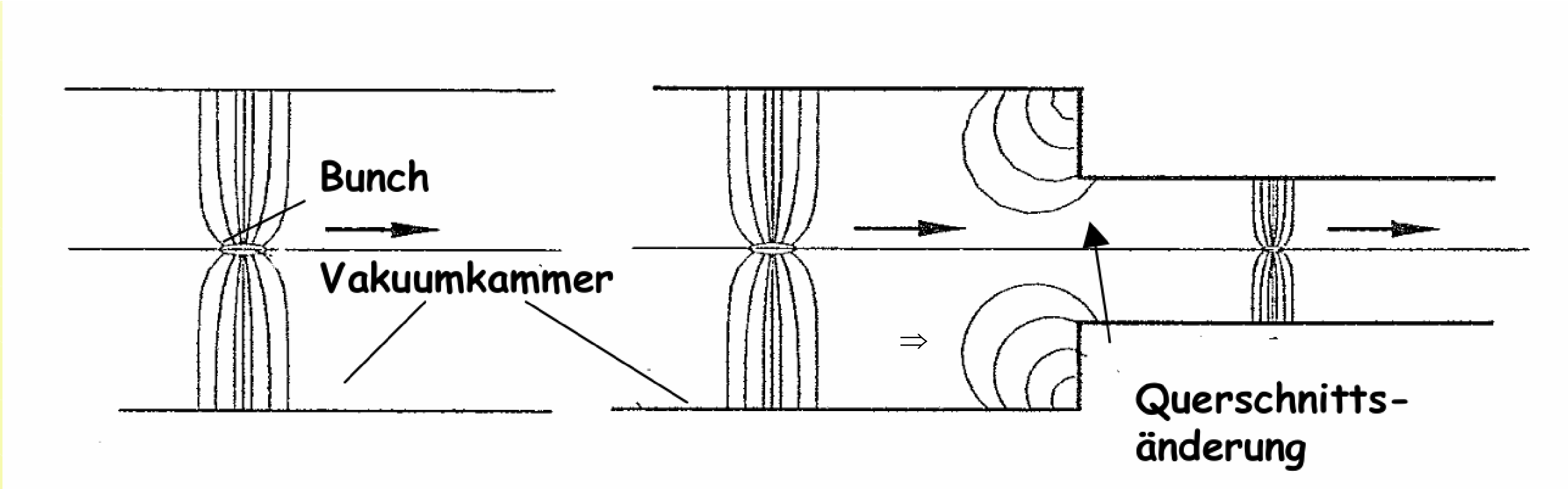
# Cavity Field

fundamental electric field of an accelerator cavity



Picture: Monopole, Dipole Quadrupole Passbands of the TESLA 9-cell Cavity, R. Wanzenberg

# Wake Fields Caused by Specific Geometries (heuristic)




- change of the cross section

=> acceleration of mirror charges

=> acceleration of surface charges

=> "additional" electromagnetic radiation

# High Frequency Fields

- Primary Problem for Cryogenic Accelerators: Heating
- Approximable, by plane waves, because:

| Maxwell Equations (Vacuum) | | EM-Potentials and Lorenz Gauge | | Wave-Equation(s) |
|---|---|---|---|---|
| $rot(\vec{E})+\frac{1}{c}\frac{\partial\vec{B}}{\partial t}=0$ <br> $rot(\vec{B})+\frac{1}{c}\frac{\partial\vec{E}}{\partial t}=\frac{1}{c}\vec{j}$ <br> $div(\vec{B})=0$ <br> $div(\vec{E})=\rho$ | **with** | $\vec{E}=-grad(\varphi)-\frac{\partial\vec{A}}{\partial t}$ <br> $\vec{B}=rot(\vec{A})$ <br> $div(\vec{A})+\frac{1}{c^2}\frac{\partial\varphi}{\partial t}=0$ | **yields** | $\Delta\vec{A}-\frac{1}{c^2}\frac{\partial^2\vec{A}}{\partial t^2}=-\mu_0\vec{j}$ <br> $\Delta\varphi-\frac{1}{c^2}\frac{\partial^2\varphi}{\partial t^2}=-\frac{1}{\varepsilon_0}\rho$ |

# Plane Wave Approximation I

- Field propagates in vacuum, reducing the Equations to:

$$\Delta\vec{A} - \frac{1}{c^2}\frac{\partial^2\vec{A}}{\partial t^2} = 0$$

$$\Delta\varphi - \frac{1}{c^2}\frac{\partial^2\varphi}{\partial t^2} = 0$$

– Thus, the solution of this equation has the form:

$$\varphi(x,t) = F(x-ct) + G(x+ct)$$

# Plane Wave Approximation II

- The wave equation is a linear pde
  - The sum of solutions is a solution too
- Define $\xi := x - ct$
  - Orthogonal expand $F(\xi)$ and $G(\xi)$

$$F(\xi) = \sum_{n=-\infty}^{\infty} < F, \xi \mapsto e^{-in\xi} > e^{in\xi}$$

$$=: \sum_{n=-\infty}^{\infty} C_n e^{in\xi}$$

$$\Longrightarrow \quad F(x,t) =: \sum_{n=-\infty}^{\infty} C_n e^{in(x-ct)}$$

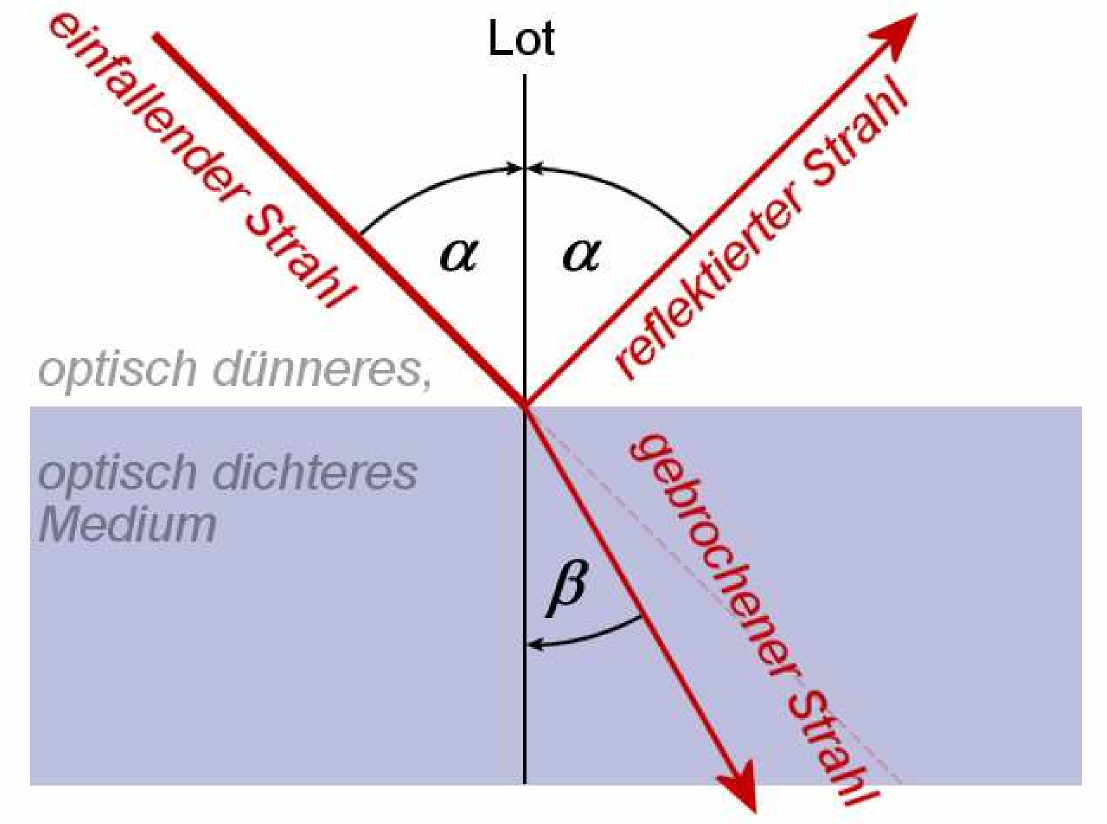
# Plane Wave Approximation III

- Select a finite set of plane waves that provide a sufficient approximations of the field

- Compute the case of field with surface interaction separately for each plane wave
  - High frequencies, allow a flat surface allocation.

# Approximation using Geometric Optics

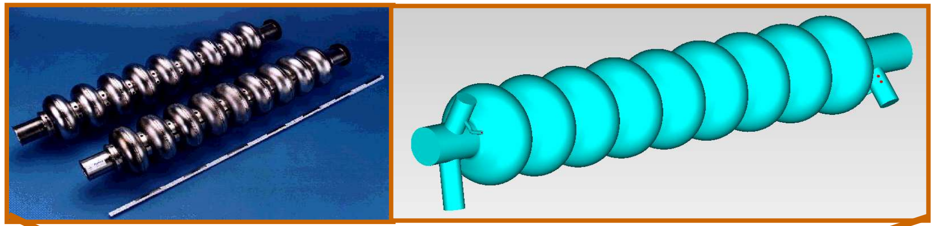
- The problem is very well known:




* In this application the transmitted is treated as absorbed

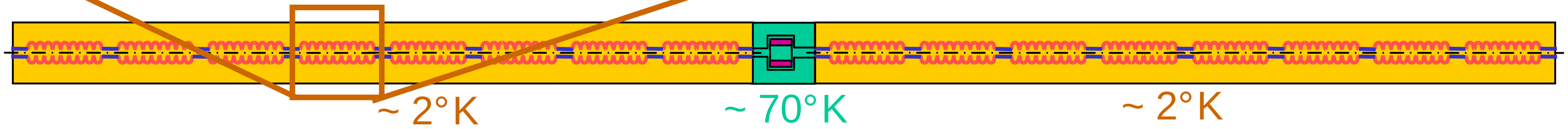
# Ray-Tracing-Methods (Original Cryoloss Program)

## Problem: Wall Losses in Superconducting Cavity-Structures

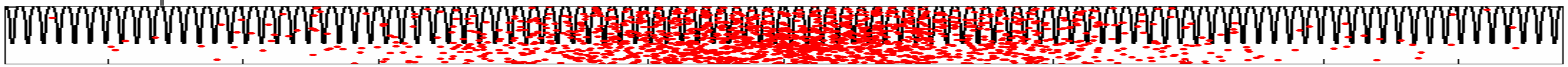


1 Watt Wall Losses at ~ 2°K

➔ 1 kW Cooling Power !

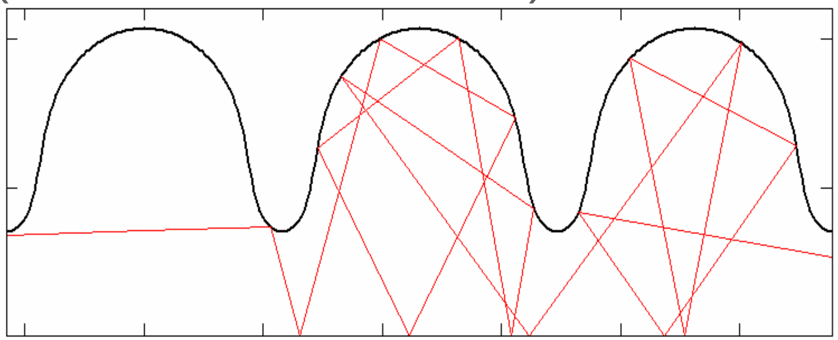


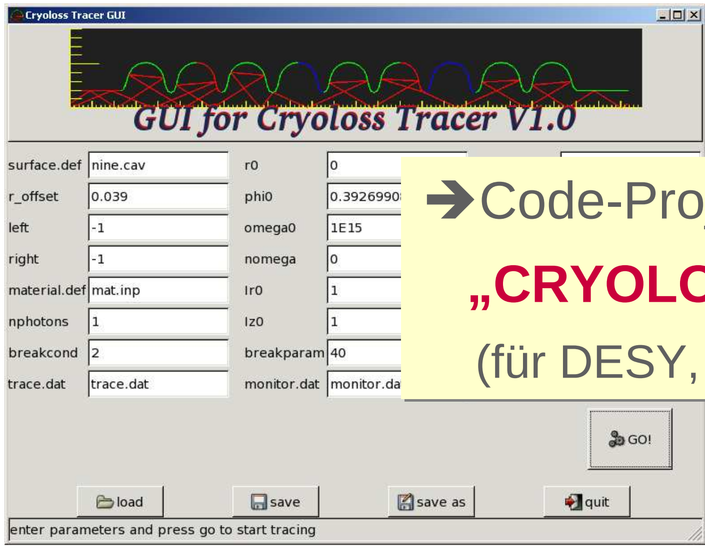

➔ Computation of Wall Losses with a Photon-Model



## Photon-Modell using Geometric Optics

(Dr. Martin Dohlus, DESY)



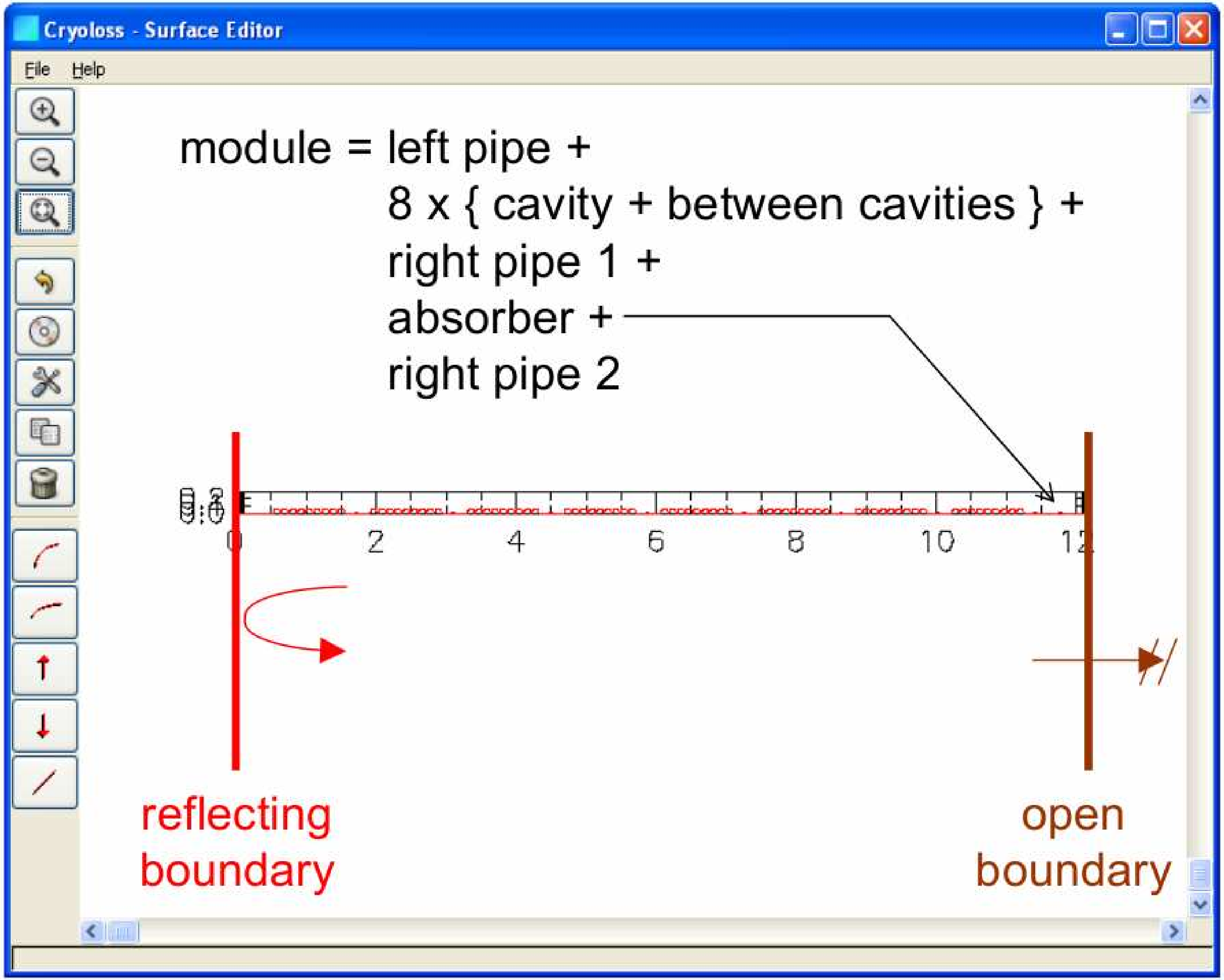



➔ Code-Projekt

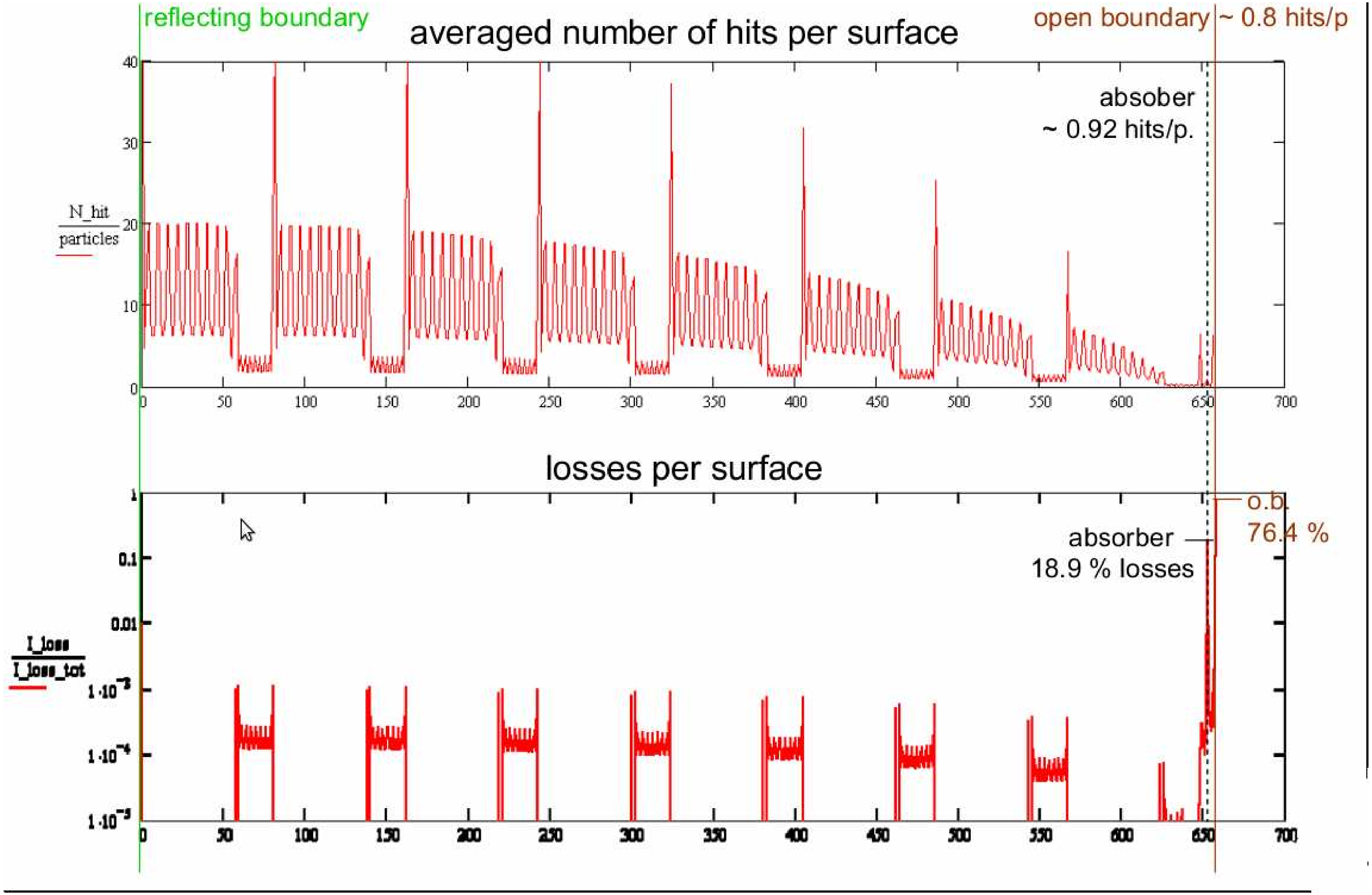
**„CRYOLOSS“**

(für DESY, HH)

# Cryoloss: XFEL Example



# Cryoloss: XFEL Example Results



# Cryoloss Problems:

- Primary: GUI-Integration
  - GUI ➔ complex code!
  - Quite seldom usage ➔ typical SOTA Problems
- Secondary: General approach for a special problem
  - In principle very extensible, but complex code as well

➔ all in all, the code is hard to understand and maintain

# Cryoloss 2 Project

- Simplification of the original cryoloss project

- Basically: Reduction to a simple C-Library
  - 1 file for the code with the primary functionality
  - 2 further files for other needed functionality

- Small simple script to compile and link simulation description programs (“scripts”).

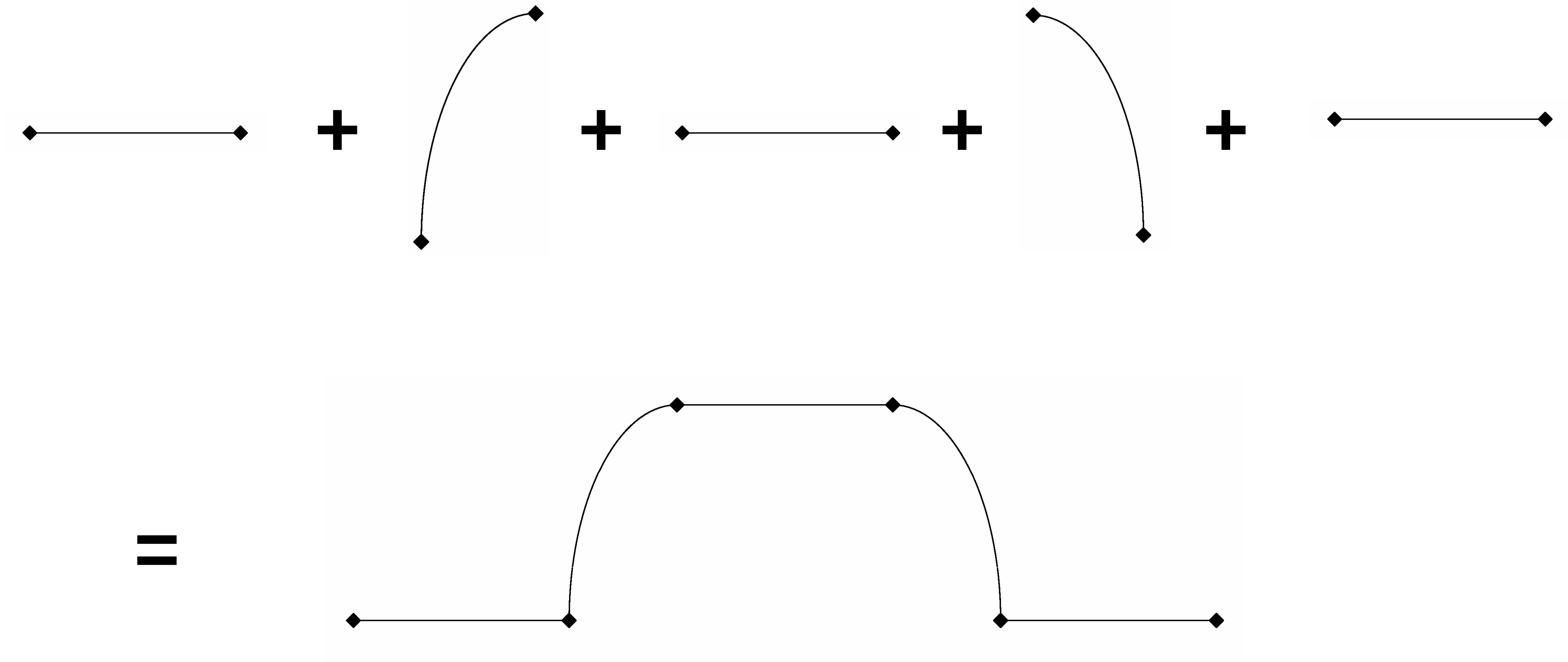
# Construction of an Accelerator Structure

- Assumptions:
  - Rotation symmetry
  - One curve sufficient as description
  - The cure is a assembly of simple basic curves, that is of lines and segments of ellipses:

$$\gamma : I \subset \mathbb{R} \to \mathbb{R}^2$$

$$\gamma = t \mapsto \begin{cases} \gamma_0(t) \text{ if } t \in [0,1) \\ \gamma_1(t-1) \text{ if } t \in [1,2) \\ \gamma_2(t-2) \text{ if } t \in [2,3) \\ \vdots \end{cases}$$

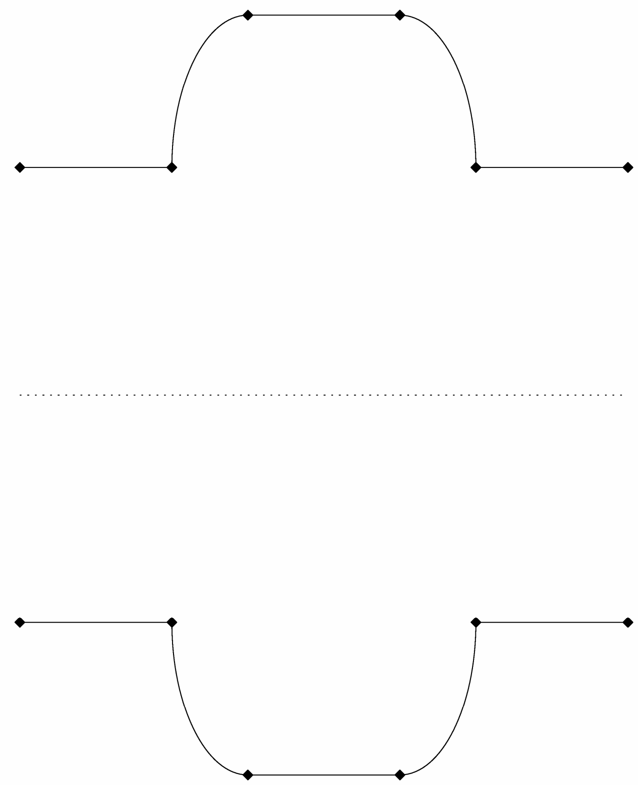
# Assembly of a Structure Visualized



# Assebly of the above structure in cryoloss 2 code

```
left_ellipse = ellipse_new(1,2,M_PI,-0.5*M_PI);
right_ellipse = ellipse_new(1,2,0.5*M_PI,-0.5*M_PI);
line = line_new(2,0);

bp = beampipe_new(RIGHT_OPEN,3,0.4);
beampipe_append(bp,line,mat0);
beampipe_append(bp,left_ellipse,mat1);
beampipe_append(bp,line,mat1);
beampipe_append(bp,right_ellipse,mat1);
beampipe_append(bp,line,mat0);
```



# Most Important Implementation Technique

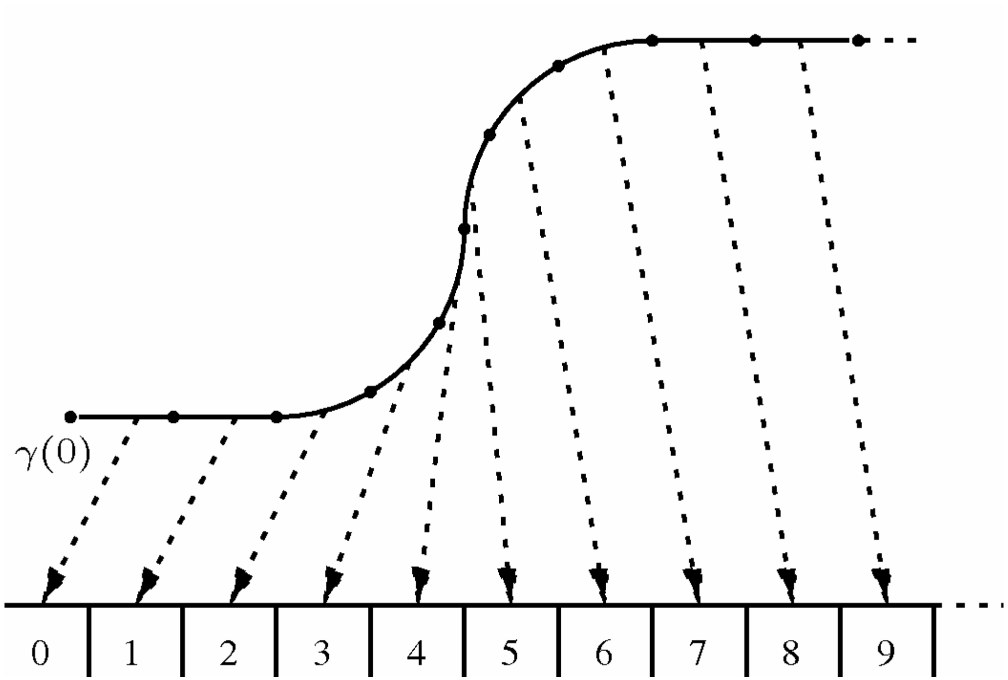
- Mimicry of Algebraic Data Types in C for Segments

```
union segment_t {
  enum { ELLIPSE_SEG, LINE_SEG } type;
  struct {
    size_t padding;
    … /* ellipse data */
  } ellipse;
  struct {
    size_t padding;
    … /* line data */
  } line;
};
```

- Storing all segments in an Array, thus allowing O(1) access time. Especially to accordingly sorted data.

# Structures for Data evaluation and Processing

- Separate the strucure defining curve into equidistand parts.
  - These „tiles“ allow the above mentioned storing of the data in an array




  - But: Tile borders may not match segment borders.
    - Divide tiles into a start and end part where necessary

# Tracing

- Create a trace-structure with
  - Start position
  - Number of the maximum of ray-parts/reflections
    - Which is the only abort criterion for simplicit

- Performing a Trace

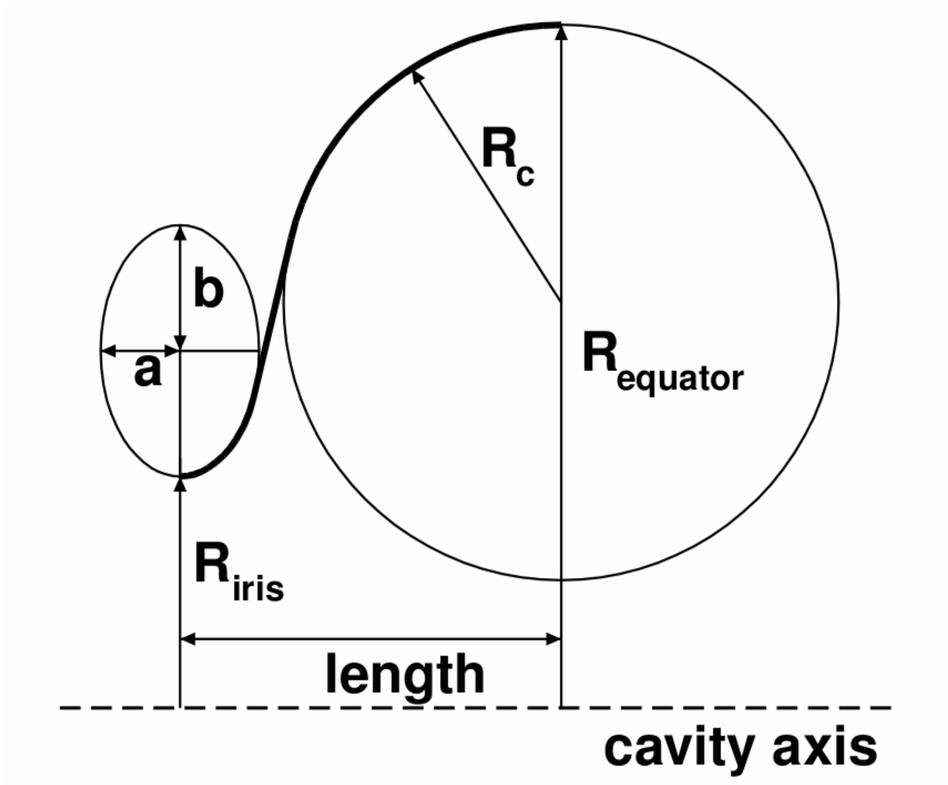
Example:

```
tr = trace_new(4,0,0,1,2);

raytrace( tr, bp );
```

# Problems with real Structures

- At junctions the curve has to be smooth

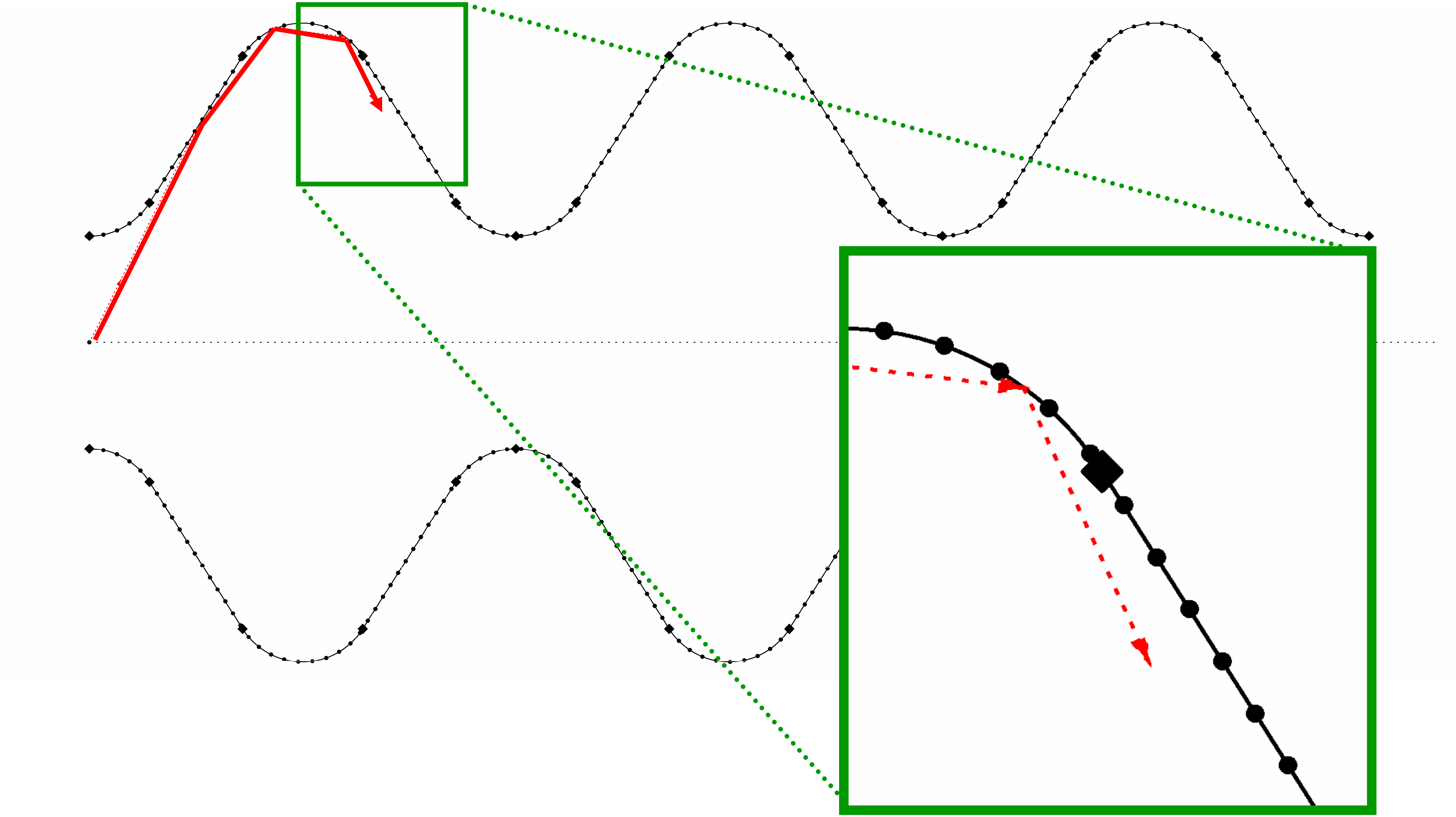



- Solution: A utility library providing an adequate interface to the core library

# Evaluation I

- Tiles are again algebraic data structures with the constructers
    - PLAIN : Whole tile is on one segment
    - JUNCTION: Tile contains a junction point of to segments.

# Example Tesla Cavity (Scheme)

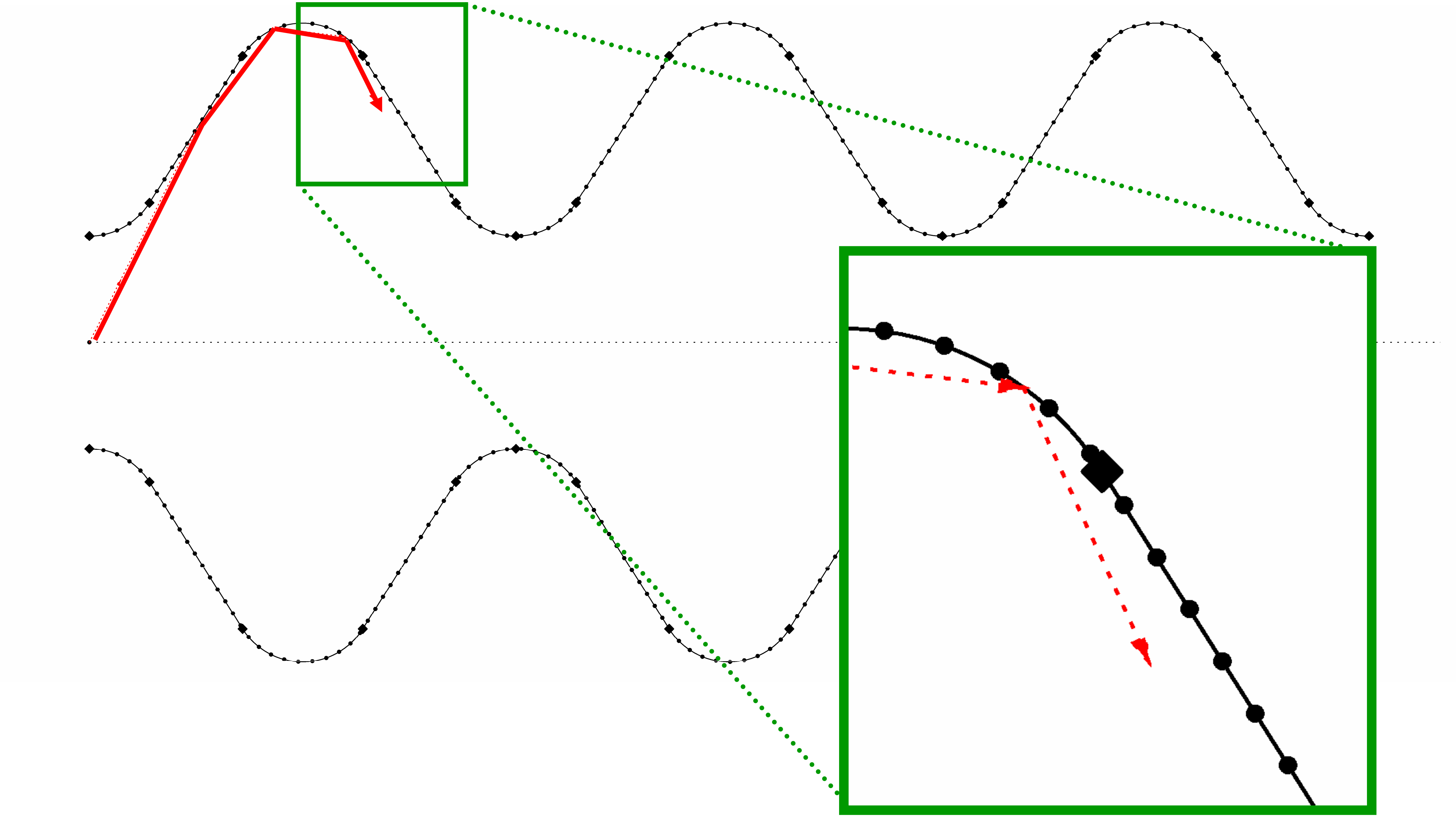


# Evaluation II

- Every tile refers to the segment and material data that that is/are applicable
- Tiles are indexed by the arc length divided of the structure curve through the arc length of a tile

- Resulting in O(1) access time on the structure when processing and evaluating data

# Example Tesla Cavity (Scheme)



# Parallelization of the Tracing I

- Create an Array von Traces with different initial conditions, using one of the provided helper functions

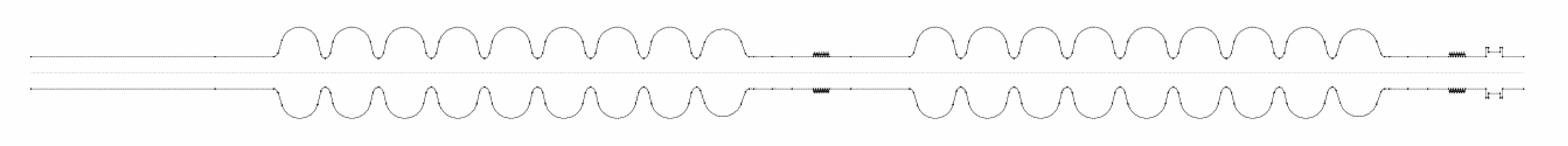
- Call parallel_raytrace with the following parameters:
  - The array of traces with initial conditions
  - Number of threas/CPU used to perform the tracing

# Parallelization of the Tracing II

- Every thread accesses the geometry data read-only
  - No caching conflicts
- Every thread writes its result data into a thread dedicated structure
  - No caching conflicts too
- Evaluation could be parallelized in the same manner. However it is so fast in comparison with the tracing it self, that its not really necessary.

# Some Results |

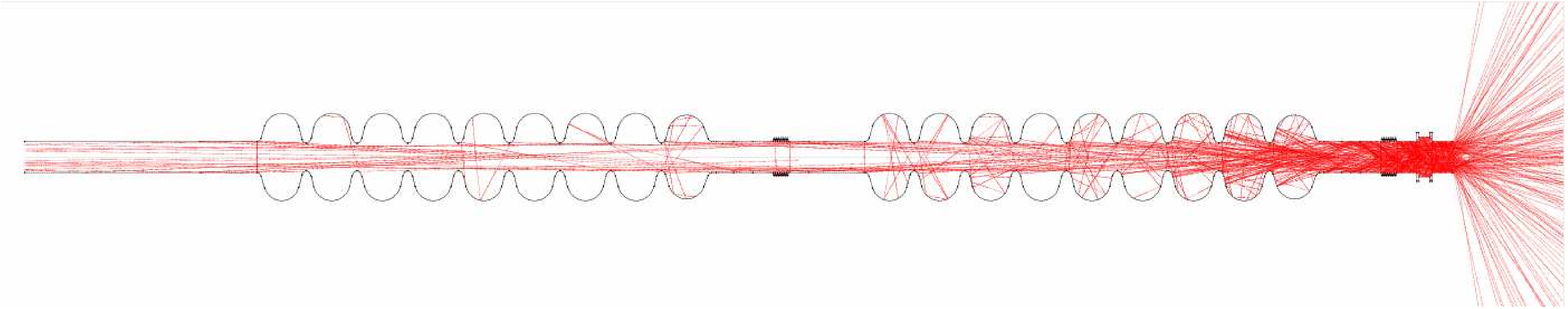
- Tested with XFEL-Style structure:



- Boundaries:
  - Maximal 5000 reflections per ray:
  - Left: Closed (Ideal Reflecting)
  - Right: Open

# Some Results II

- Results



  - Number of Traces/Particles:

    1564

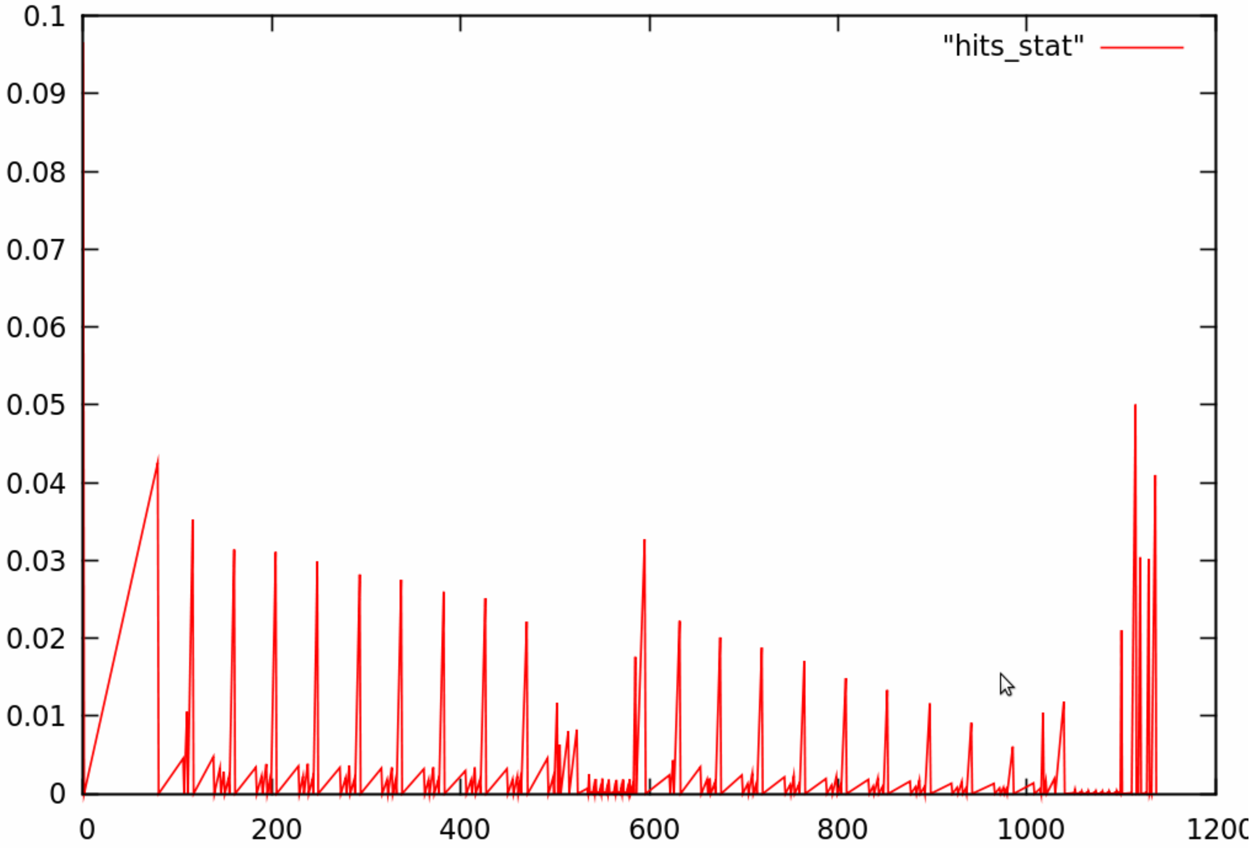
  - Mean reflections before leaving the ray:

    ~512

  - Percentage of Particles not left after 5000 hits:

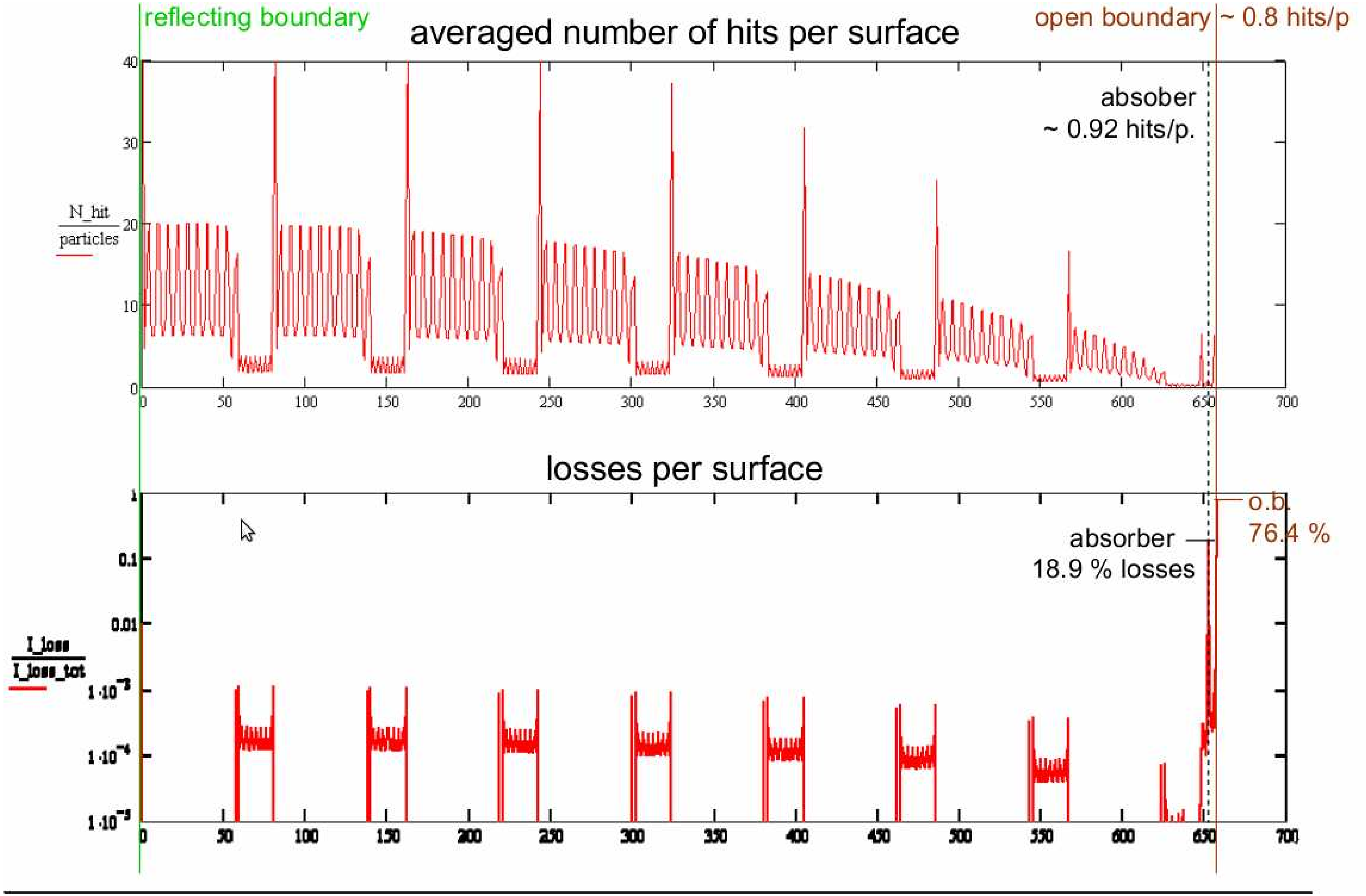
    ~3%

# Some Results III




- Y-Axis: hits/tota_hits    X-Axis: Tile-ID
  - Reproduces Pattern of earlier simmulations
- No further results jet … sorry

# Cryoloss: XFEL Example Results



# Questions?