Designing a flexible treatment plan optimization tool for proton therapy based on Python environment and GPU acceleration

D. Borys[1,2], J. Gajewski[1], **A. Ruciński[1]**

[1] Cyclotron Centre Bronowice, Institute of Nuclear Physics PAN, Kraków, Poland
[2] Silesian University of Technology, Department of Systems Biology and Engineering, Gliwice, Poland

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

Aim of the work

- **Main goal:**
    - Proton radiotherapy dose optimization < 1 min
    - Simultaneous dose and LET optimization
    - Optimization using micro/nano dosimetric quantities
- **Features:**
    - Python based
    - Extensible
    - Flexible
    - Fast (GPU accelerated)

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

Mainstream optimization tools

- RayStation (clinical, dose + LET)
- matRad (Matlab based, open-source, *ipopt* based)
- others:
  - Paul Sherrer Institute (PSI) - Java based, GPU accelerated dose optimizer
  - …

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

ipopt: Software for therapy optimization

- ipopt - Interior Point OPTimizer, a software package for large-scale nonlinear optimization, EPL open-source license (Eclipse Public License)
- https://coin-or.github.io/Ipopt/
- solve general nonlinear programming problems of the form:
  - x : are the optimization variables
    - possibly with lower and upper bounds $x^L <= x^U$
  - functions f(x), g(x) are : objective function and general nonlinear constraints
    - can be linear or nonlinear and convex or non-convex
    - g(x) have lower and upper bounds $g^L <= g^U$

$$\begin{array}{ll} \min\limits_{x \in \mathbb{R}^n} & f(x) \\ \text{s.t.} & g^L \leq g(x) \leq g^U \\ & x^L \leq x \leq x^U, \end{array}$$

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

Monte Carlo dose recalculation

- **FRED** - **F**ast pa**R**ticle th**E**rapy **D**ose evaluator,
- Fast Monte-Carlo platform for particle transport in heterogeneous media
- Allow a rapid recalculation of dose deposition in the context of Particle Therapy
- FRED can run on CPU hardware exploiting multi-core parallelism as well as on single or multiple **GPU** cards using OpenCL.
- Example: time for one patient plan simulation to obtain Dij matrix 231s (3.9min) [6 fields, 10818 PBs, 1E4 protons/PB]
- fred-mc.org

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

FREDopt - Specification

- **Goal:**
  - Proton dose optimization
  - Dose + LET
  - Micro / nano dosimetric approaches
  - Fast -> optimise clinical setup + adaptive therapy

- We are using Python to simplify the development process
- Based on fredTools library (fredtools.ifj.edu.pl) (sitk, numpy, pandas)
- FRED MC software is a basic tool to calculate Dij matrix

Gajewski, et al. *Commissioning of GPU–Accelerated Monte Carlo Code FRED for Clinical Applications in Proton Therapy.* Front. Phys. 8, 403 (2021) https://doi.org/10.3389/fphy.2020.567300

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

FREDopt - our idea

- Current version:

A. Optimizer algorithm: DDO - dose-difference optimization

$$N_{i,k+1} = N_{i,k} \cdot \left[1 + F_{\mathrm{DDO}} \left(\frac{\sum_{j \in \mathrm{PTV}} w_j d_{i,j}^2 \frac{\hat{D}_j}{D_{j,k}} + \sum_{j \in \mathrm{OAR}} w_j d_{i,j}^2 \frac{\hat{D}_j}{D_{j,k}} \Theta(\hat{D}_j - D_{j,k})}{\sum_{j \in \mathrm{PTV}} w_j d_{i,j}^2 + \sum_{j \in \mathrm{OAR}} w_j d_{i,j}^2 \Theta(\hat{D}_j - D_{j,k})} - 1 \right) \right]$$

B. Only Dose is optimized

$$D_j(\mathbf{N}) = \sum_{i \in \mathrm{PB}} d_{i,j} \cdot N_i$$

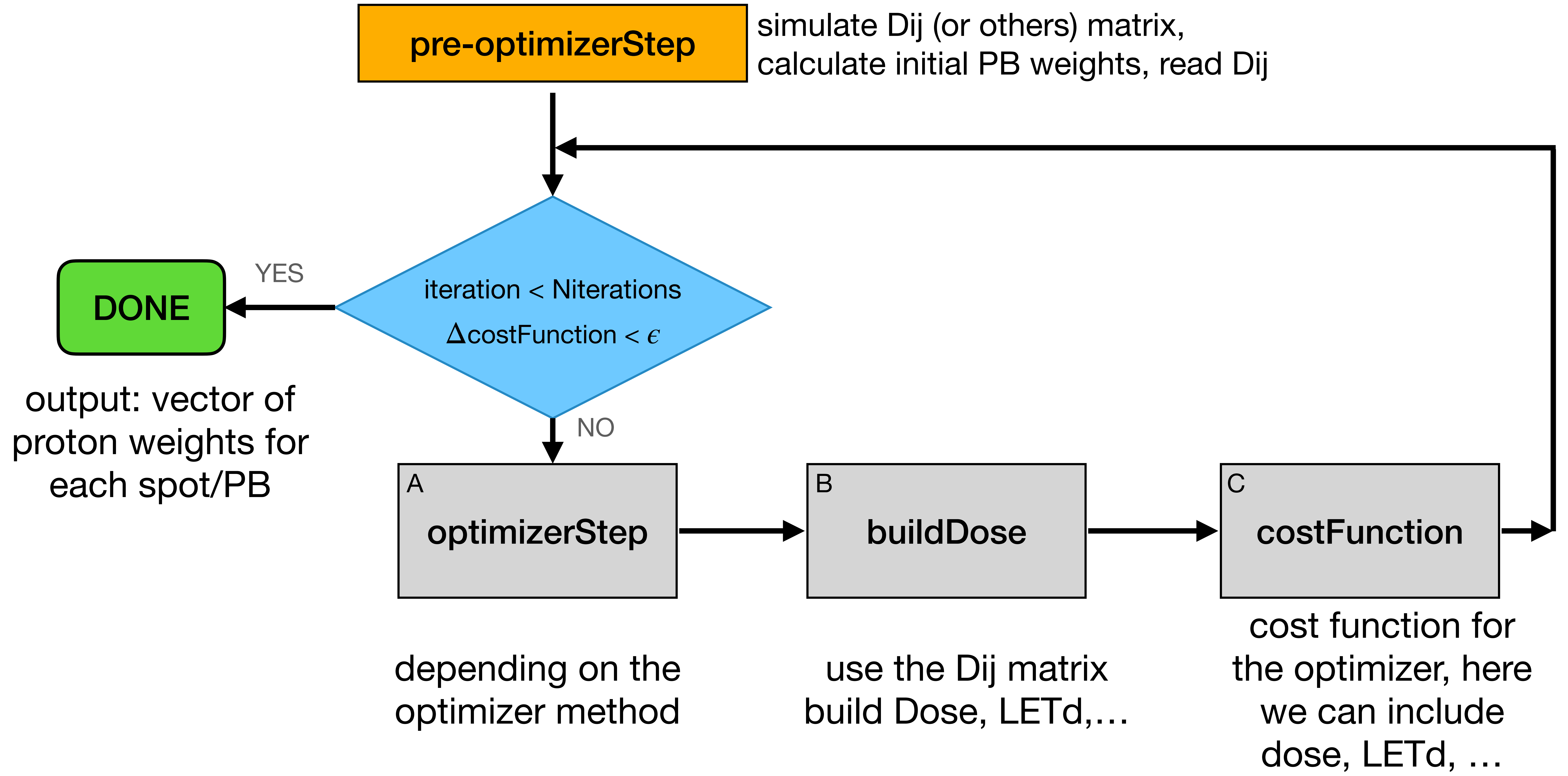
C. Cost Function

$$\chi^2(\mathbf{N}) = \sum_{j \in \mathrm{PTV}} \frac{w_j (\hat{D}_j - D_j)^2}{\hat{D}_j^2} + \sum_{j \in \mathrm{OAR}} \frac{w_j (\hat{D}_j - D_j)^2}{\hat{D}_j^2} \Theta(\hat{D}_j - D_j)$$

A Mairani, et al., **A Monte Carlo-based treatment planning tool for proton therapy**; Phys. Med. Biol. 58 (2013) 2471–2490 **doi:10.1088/0031-9155/58/8/2471**

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

FREDopt - block diagram / algorithm

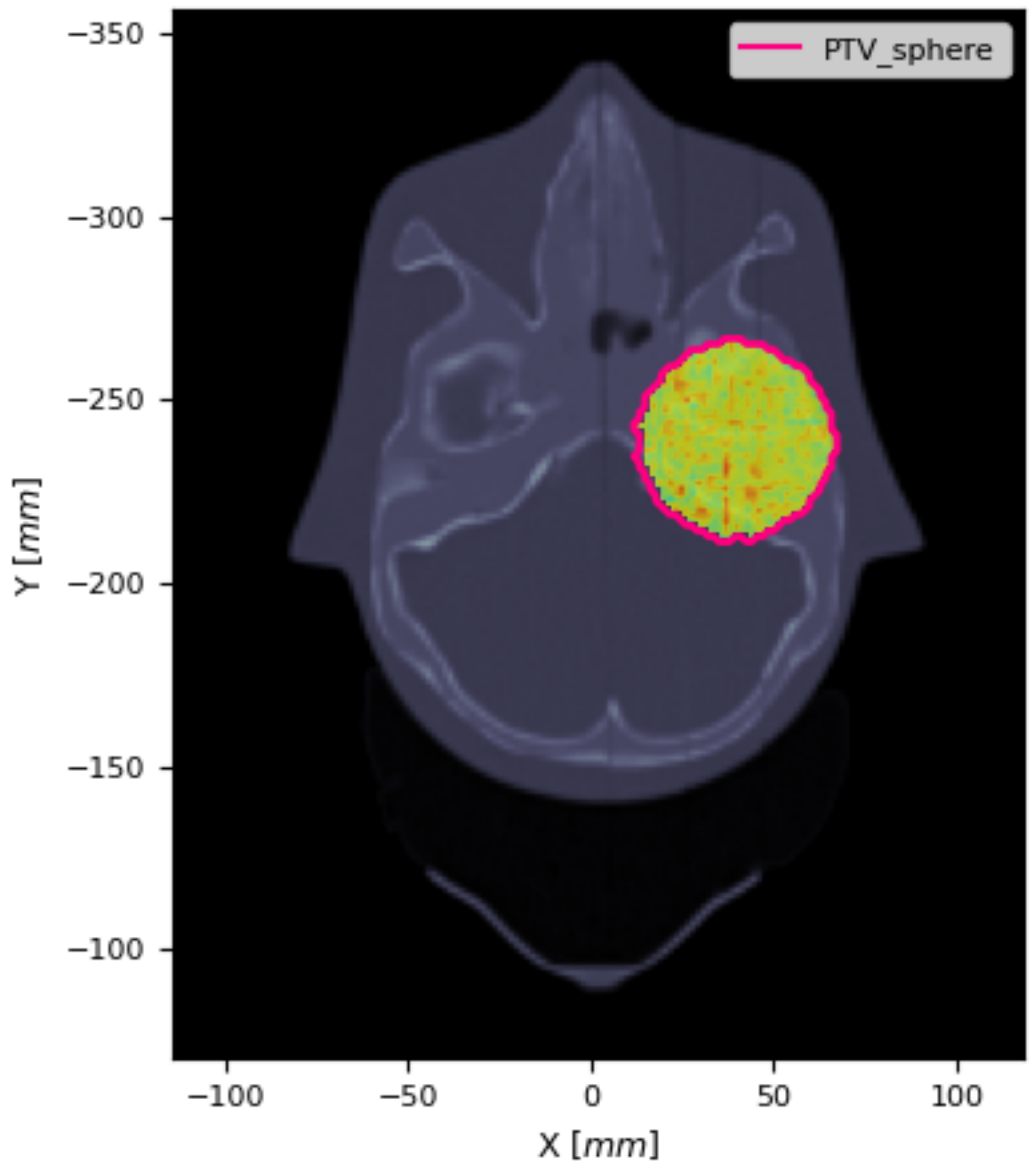


step	time [s]
pre-opt FRED Dij calculate	~230
pre-opt initial PB weights	~0.45
pre-opt read Dij	~18
optimizer Step	~0.34
build Dose	~0.22
costFunction	~6E-4

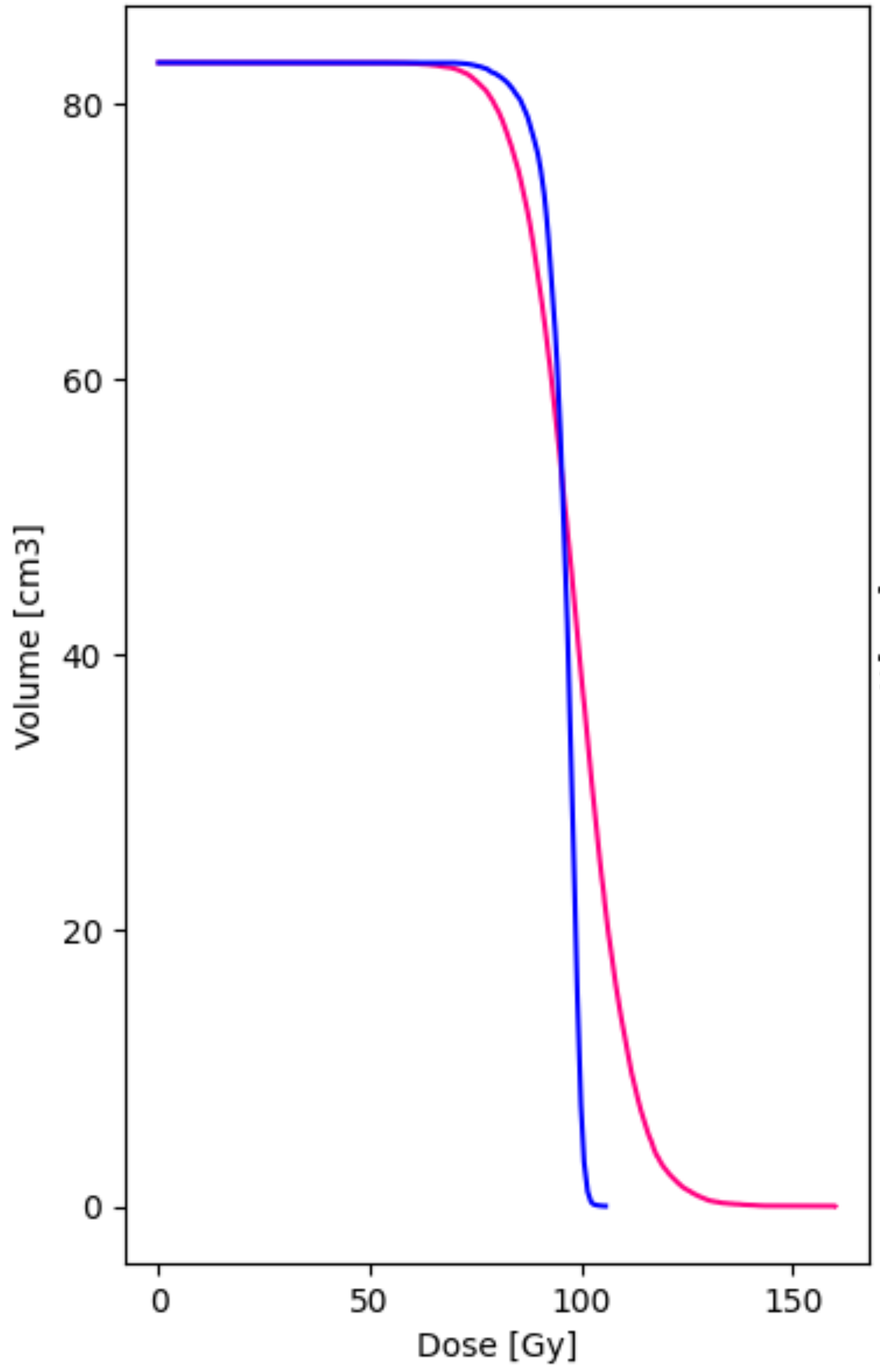
damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

FREDopt - initial results

FREDopt optimization

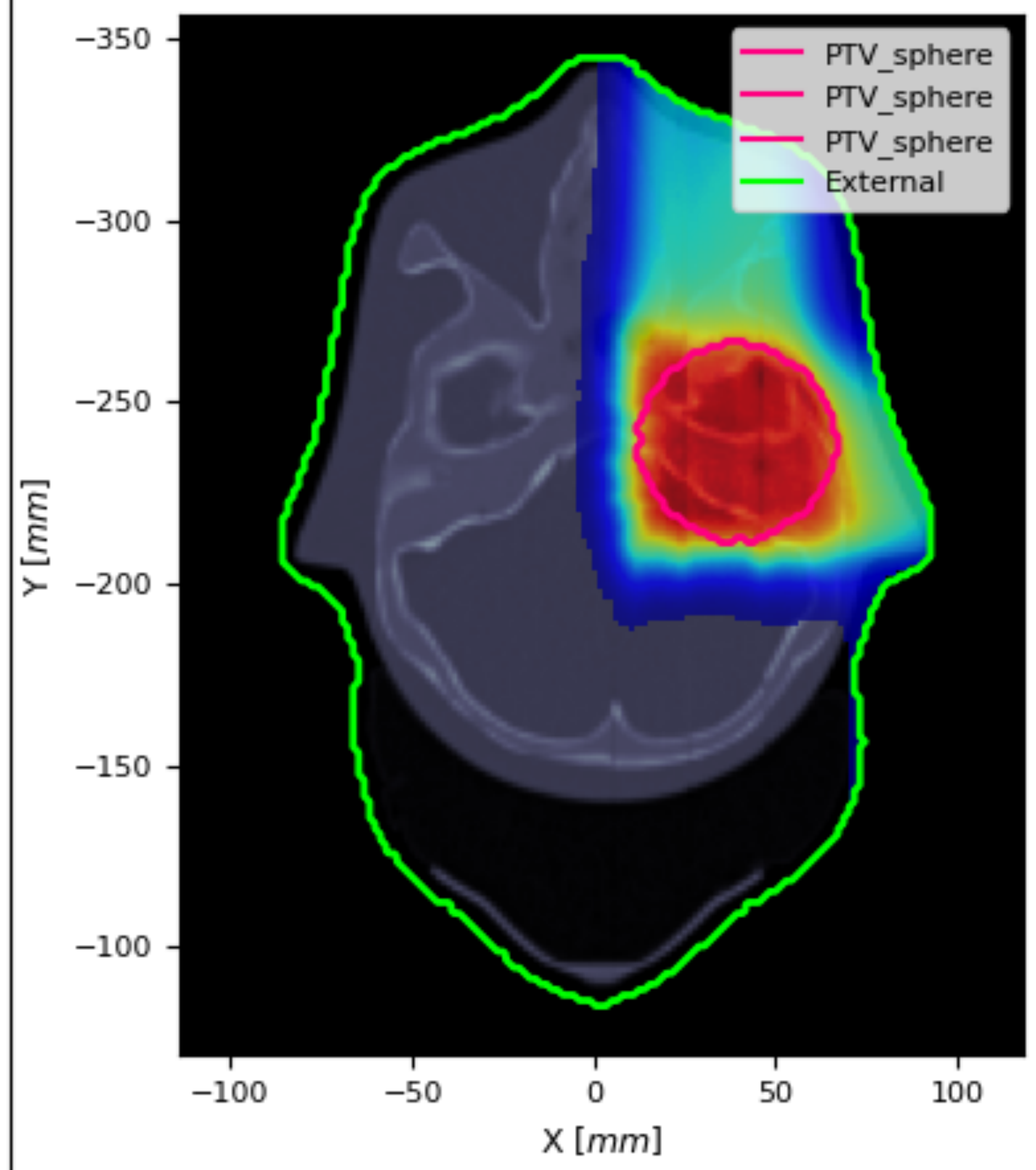


DVH FREDopt



red DHV - FREDopt
blue DVH - FRED MC

FRED MC recalculation of the TPS-optimized TP



damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

FREDopt - next steps

- Speed up the code
    - Using GPU accelerators (numba Python modules ?)
    - Problems: the size of Dij matrix -> Applying the sparse matrix
- Optimization algorithm: ipopt or in house implementation?
- Extending the cost Function (LETd, …)

damian.borys@polsl.pl
antoni.rucinski@ifj.edu.pl

Thank you!