Monte Carlo Simulations in Ion Beam Therapy at MedAustron

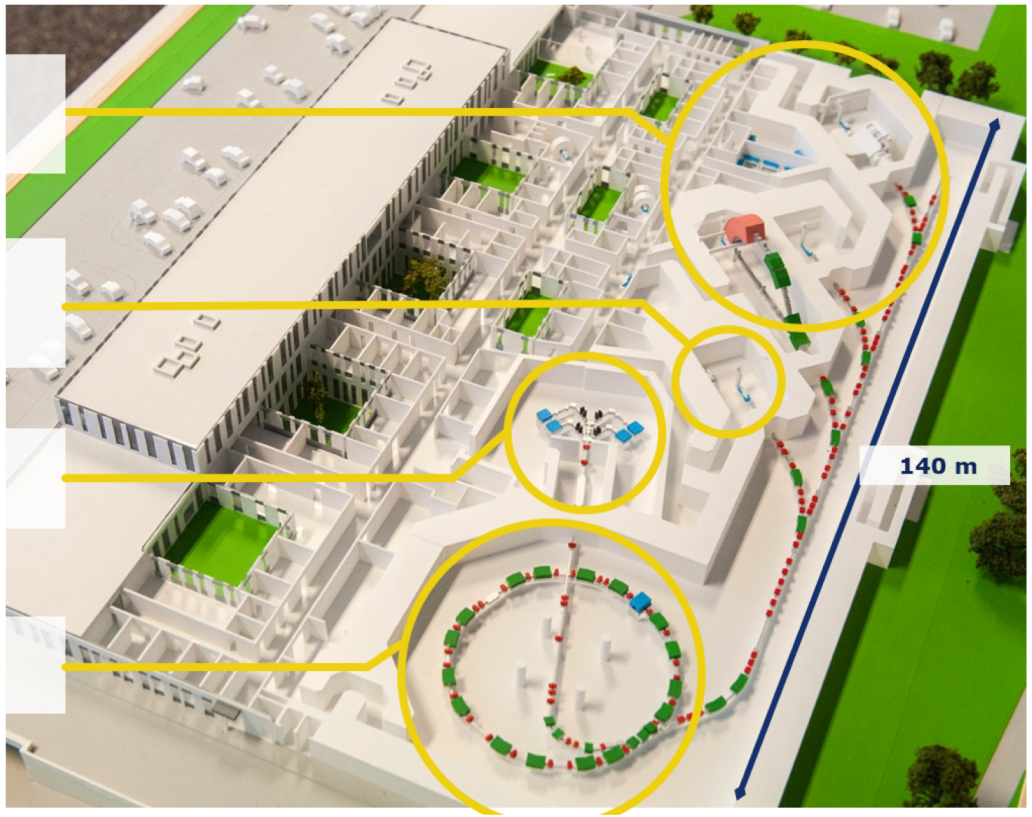
MedAustron Ion Therapy Center Overview

Irradiation Rooms
Three rooms for patient treatments

Research
Irradiation room for non-clinical use

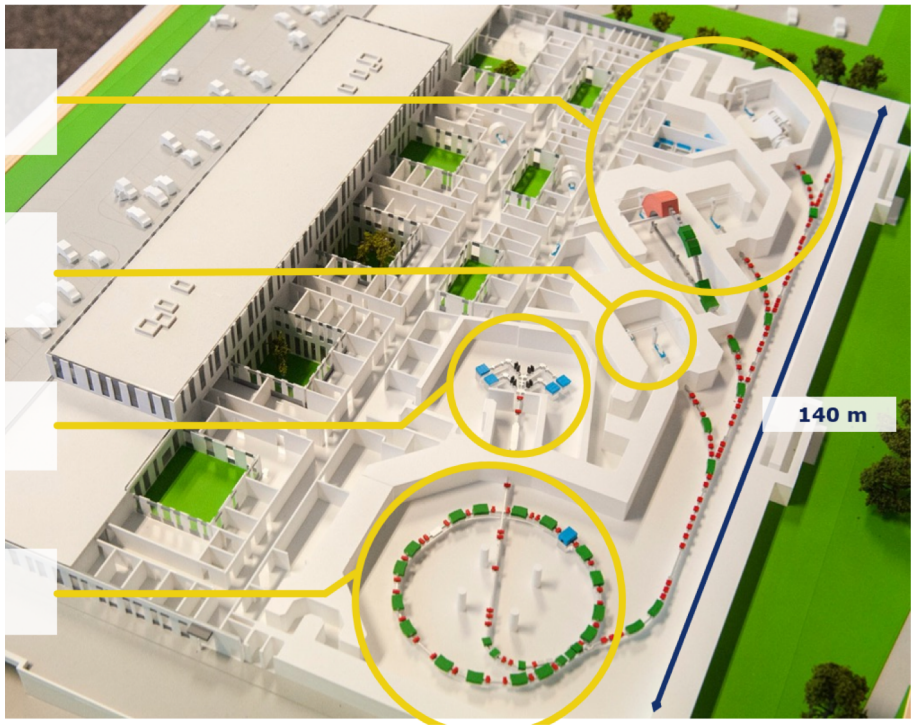
Ion Sources
and linear accelerator

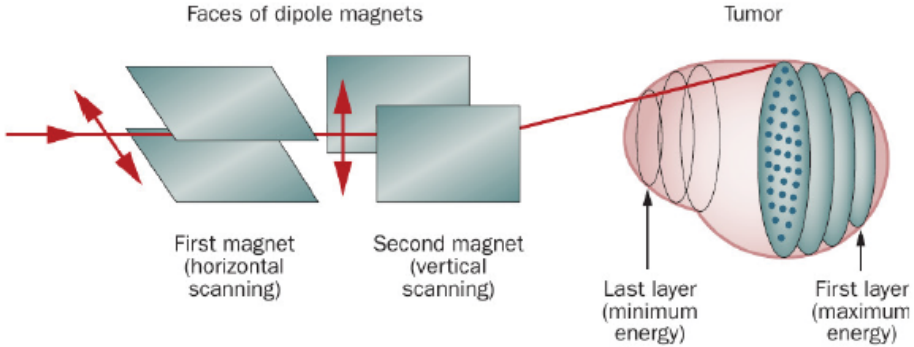
Synchrotron
proton/carbon/helium accelerator



- Wiener Neustadt, Austria
- Synchrotron based
- **Protons** and **carbon ions** clinical, **helium** commissioned for research
- Pencil beam scanning

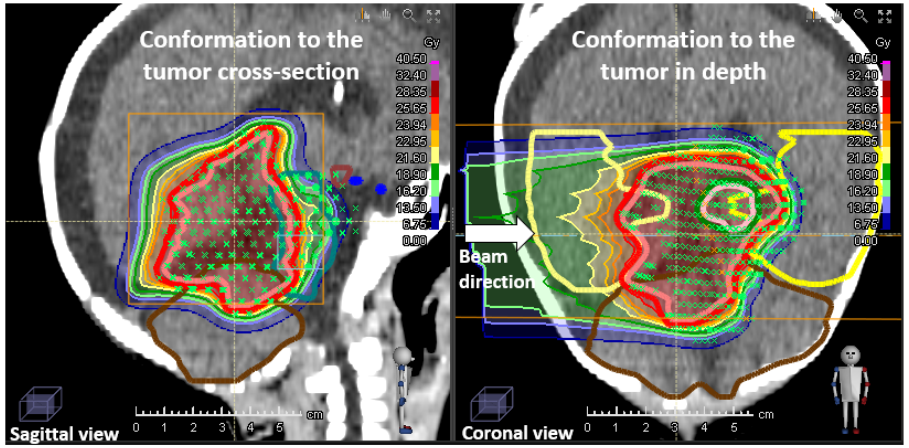
Pencil Beam Scanning





[1]

Pencil beam scanning



[2]

Treatment plan example

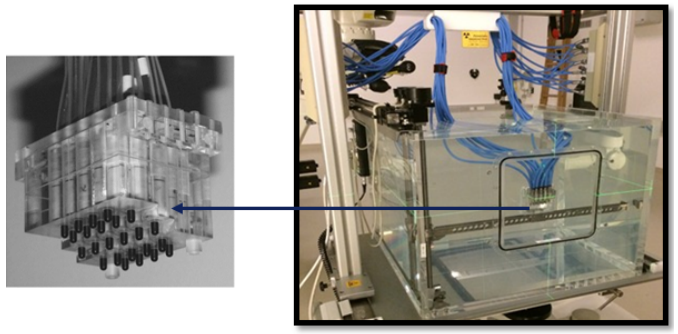
[1] M. Durante and J. S. Loeffler. "Charged particles in radiation oncology"

[2] Grevillot 2021, Monte Carlo Modelling of Scanned Ion Beams in Radiotherapy, eBook ISBN9781003211846

Patient Specific Quality Assurance (PSQA)

Up to 55 fractions treated every day!

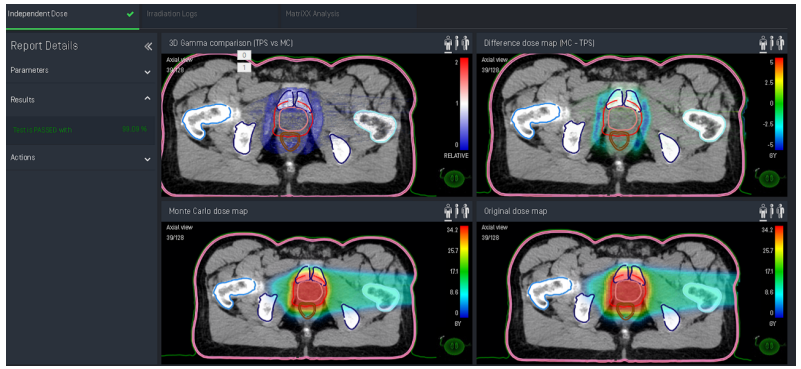
Measurement based PSQA



Measurement set-up using 3D-block

- Treatment plan delievered in water and dose measured at the plan position
- 24 pin-point measurements "only"
- Limited to low gradient dose regions
- Beam time required

Independent Dose Calculation (IDC)



IDC using myQAiON (IBA-dosimetry)

- Treatment plan is simulated with an independent dose engine in the CT of the patient
- Full 3D geometry
- Account for high gradient dose regions
- Save beam time and man power

Patient QA at MedAustron:

- **Proton:** ~ 100% IDC with **myQAiON-MCsquare** since 2021 – CE marked IDC product
- **Carbon:** 100% plans measured, soon replaced with **myQAiON-IDEAL** – CE marked IDC product (2025)

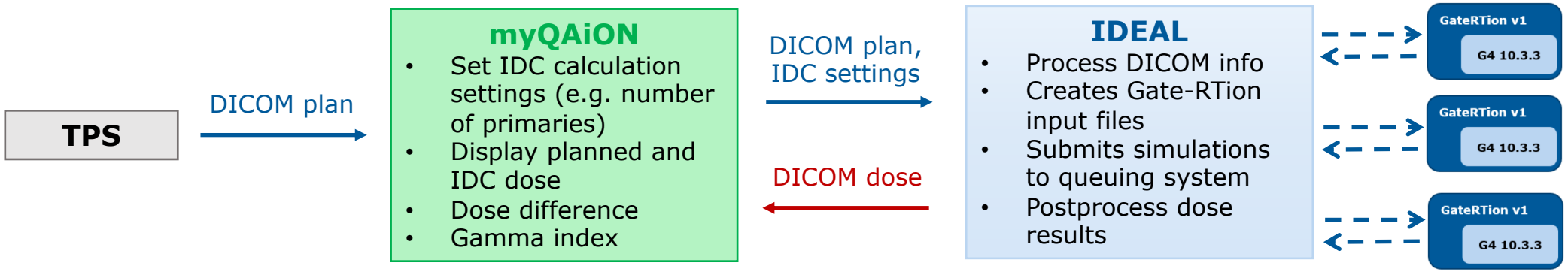
The IDEAL Project

IDEAL: **I**ndependent **D**os**E** c**A**lculation for **L**ight ion beam therapy using Geant4/GATE

- Based on **Gate-RTion** (Gate with G4 10.3.3) -> validated Gate version for ion beam therapy
- Gate wrapper, with a DICOM in – DICOM out framework
- Open source, part of Opengate collaboration, developed at MedAustron
- Project Start: 2018

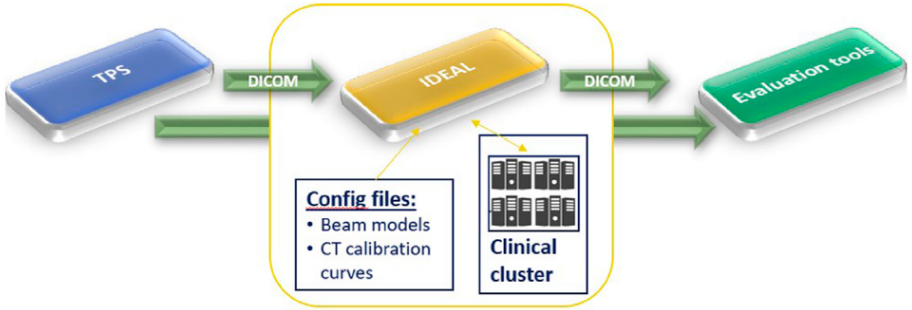
myQAiON-IDEAL:

- CE medical product, licensed



IDEAL: let's get into details

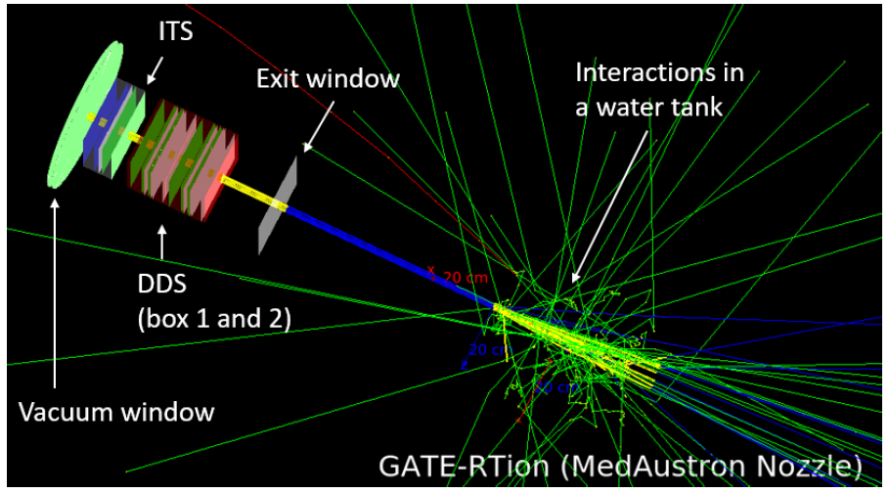
DICOM info:
- **CT** -> patient geometry, HU map, CT scan protocol, scoring grid
- **Structures** -> External structure used to crop CT
- **RT plan**
  - Energies, beam positions and intensities -> treatment plan source
  - Isocenter position and couch angle -> patient positioning in the simulation
  - Beamline -> correlates with beam model and nozzle geometry
  - Gantry angle, Passive elements
- **RT doses** -> dose is simulated in the CT grid and re-sampled on the plan dose grid



Configuration files:
- CT calibration curves: density and composition curves, for each CT protocol
- Beam models: energy-dependent source parameters (beam optics, energy spread…)
- Nozzle geometry

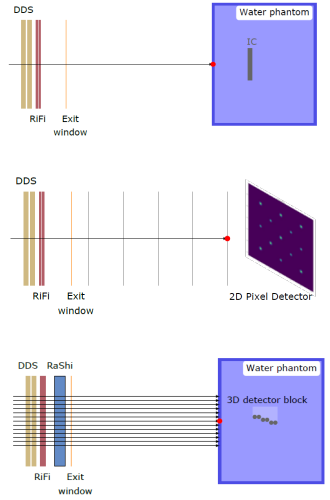
Parallelization of simulation splits -> HT Condor queuing system

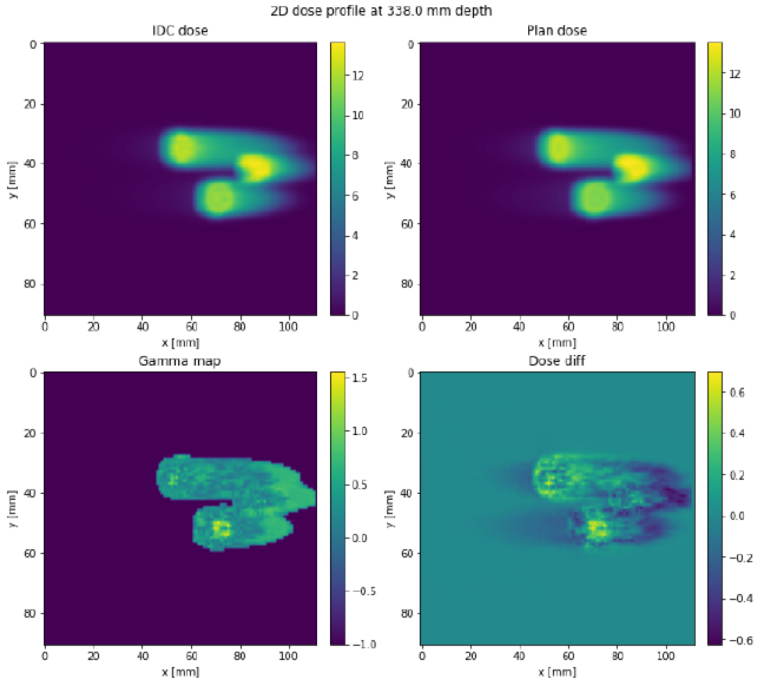
User interfaces:
- Command line
- Python script -> can be integrated your python program!
- API -> HTTP/HTTPS frameworks



IDEAL v1 Validation

- **Dosimetric commissioning** (MedAustron beam model validation, for carbon ions) [1]
  - Depth dose profiles in water
  - Pencil beam sizes and positions in air
  - Reference dose in water
  - Dose in SOBPs in water
- **Clinical commissioning:**
  - Simulation of patient cases in water against PSQA measurements
- **HU to material conversion**
  - rWET of calibration tissue equivalent material
  - rWET of animal tissue
- **Functional tests**
  - Alignment (CT, beam, dose grid)
  - Selection of correct settings (HU to material, beamline, …)





Grid resolution: [1, 3, 2] mm
Couch angle: 15 degrees
Gamma index pass rate: 98.58%
(global gamma, 3% 3 mm, simulations with 1% statistical uncertainty)

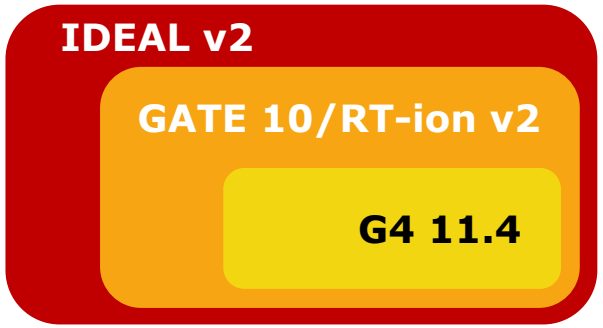
[1] Experimental validation of Geant4 nuclear interaction models in dose calculations of therapeutic carbon ion beams - Jia - 2025

IDEAL v2 – Gate-RTion v2

IDEAL v2: source code

Start simulation in **Gate 10**

- Enables **MT** -> Optimize use of computational resources
- Additional LIBT features (e.g. dose uncertainty calculation, DICOM reading)
- Calculation of **LET** and **RBE weighted dose** available -> results presented during Gate 10 scientific meeting 2025
- Replace ancient G4 version (10.3.3) with more recent one



Gate-RTion v2:

Goal: release in 2026

- Advanced validation with **G4 11.3.0** for carbon ions
  - results presented during Gate 10 scientific meeting 2025
- Current issues:
  - Calculation time overhead (under investigation)
  - Bug in transversal profiles QMD -> fixed in **G4 11.4**
- Collaboration driven by MedAustron:
  - Several LIBT centers in Europe
  - Regular update meeting
  - Define test and criteria for the validation
- Next steps at MedAustron :
  - Work on calculation time issue
  - Validation for protons and carbon ions

Gate For Ion Beam Therapy

From research tools to first medical applications at MedAustron!

- **Development of MC dose engines for IDC**
  - 2012: **GATE/Geant4** for proton IDC (Grevillot et al, *Phys. Med. Biol.* **57** (2012) 4223–4244)
  - 2016: **MCSQUARE**, a fast proton MC code (Souris et al, *Med. Phys. 43 (4)*, April 2016)
  - 2020: **GATE-RTion v1.0**: a GATE release for clinical use (Grevillot et al, *Med. Phys. 47 (8),* August 2020)

- **Development of home-made IDC systems for clinical use**
  - 2020: **AUTOMC/GATE-RTion v1.0** – protons (Aitkenhead et al, *Br J Radiol* 2020; 93: 20200228)
  - 2021: **IDEAL/GATE-RTion v1.0** – protons & carbon ions

    (Grevillot & Boersma et al, *frontiers in Physics,* August 2021 | Volume 9 | Article 704760)

- **Implementation of medical products for proton & carbon ion IDC at MedAustron**
  - 2021: **myQAiON/MCSQUARE** (IBA-dosimetry, CE marked IDC product) - for protons!

    (Dreindl et al, PTCOG2021, Grevillot et al, ESTRO 2021)
  - 2025: **myQAiON + IDEAL/GATE-RTion v1.0** (IBA-dosimetry, CE marked IDC product) -for carbon ions!

    (MedAustron/IBA collaboration)

- **Future perspective:**
  - **myQAiON + IDEAL v2/GATE-RTion v2.0** -> Biological dose for carbon ions is available
  - **IDEAL v2/GATE-RTion v2.0** standalone for advanced research purposes:
    - LET distribution, new RBE models…
    - MRI-guided LIBT (Hermann Fuchs)
    - Mixed beam irradiations
    - Prompt-gammas/PET imaging
    - Etc.

Thank You For Your Attention!

Hackathon Project: Beam Model From LU Tables

- **TreatmentPlanPBSource** in python sends to the Cpp side a vector with the pencil beam parameters for each spots -> how the parameters are calculated as a function of energy happens before the simulation is started and is under our control on python side
- Conversion from beam model to energy specific parameters is handled by the **BeamlineModel** object.
- Introduce the possibility to provide LU tables in the form "Energy – Parameter" for each beam model parameter, on top of the current polynomial-based beam model.
- The "get_parameter" functions in this case will calculate the parameter value for the input energy by piecewise linear interpolation, between the LU table datapoints.
- Summary: the work will mainly affect the refactoring of the BeamlineModel object.

```python
# set optics parameters
partPhSp_xV.append(
    [
        beamline.get_sigma_x(spot.energy),
        beamline.get_theta_x(spot.energy),
        beamline.get_epsilon_x(spot.energy),
        beamline.conv_x,
    ]
)
partPhSp_yV.append(
    [
        beamline.get_sigma_y(spot.energy),
        beamline.get_theta_y(spot.energy),
        beamline.get_epsilon_y(spot.energy),
        beamline.conv_y,
    ]
)
```

In TreatmentPlanPBSource

```python
class BeamlineModel:
    def __init__(self):
        self.name = None
        self.radiation_types = []
        self.rm = None  # range modulator
        self.rashi = None
        # Nozzle entrance to Isocenter distance
        self.distance_nozzle_iso = 0  # mm
        # SMX (X bending magnet) to Isocenter distance
        self.distance_stearmag_to_isocenter_x = float(
            "inf"
        )  # default infinity for parallel beams
        # SMY (Y bending magnet) to Isocenter distance
        self.distance_stearmag_to_isocenter_y = float("inf")
        # polinomial coefficients
        self.energy_mean_coeffs = [0]
        self.energy_spread_coeffs = [0]
        self.sigma_x_coeffs = [0]
        self.theta_x_coeffs = [0]
        self.epsilon_x_coeffs = [0]
        self.sigma_y_coeffs = [0]
        self.theta_y_coeffs = [0]
        self.epsilon_y_coeffs = [0]
        # convergence
        self.conv_x = 0
        self.conv_y = 0

    def _polynomial_map(self, base, coeff):
        # coeff are given with decreasing degree (coeff[0]->max degree)
        polyDegree = len(coeff)
        exp = list(range(polyDegree))
        exp.reverse()

        return sum([c * (base ** (i)) for c, i in zip(coeff, exp)])

    def get_energy(self, nominal_energy):
        return self._polynomial_map(nominal_energy, self.energy_mean_coeffs)

    def get_sigma_energy(self, nominal_energy):
        return self._polynomial_map(nominal_energy, self.energy_spread_coeffs)

    def get_sigma_x(self, energy):
        return self._polynomial_map(energy, self.sigma_x_coeffs)

    def get_theta_x(self, energy):
        return self._polynomial_map(energy, self.theta_x_coeffs)

    def get_epsilon_x(self, energy):
        return self._polynomial_map(energy, self.epsilon_x_coeffs)

    def get_sigma_y(self, energy):
        return self._polynomial_map(energy, self.sigma_y_coeffs)

    def get_theta_y(self, energy):
        return self._polynomial_map(energy, self.theta_y_coeffs)

    def get_epsilon_y(self, energy):
        return self._polynomial_map(energy, self.epsilon_y_coeffs)
```