

GPU-accelerated superiorization on constrained physical problems with SupPy

Tobias Becher^{a,b,c}, Yair Censor^d, Kay Barshad^d, Niklas Wahl^{a,c}

^a*Department of Medical Physics in Radiation Oncology, German Cancer Research Center (DKFZ), Heidelberg, Germany*

^b*Department of Physics and Astronomy, Heidelberg University, Heidelberg, Germany*

^c*Heidelberg Institute of Radiation Oncology (HIRO) and National Center for Radiation Research in Oncology (NCRO), Heidelberg, Germany*

^d*Department of Mathematics, University of Haifa, Haifa, Israel*

June 25, 2026

Abstract

The superiorization method (SM) is situated between feasibility-seeking and constrained optimization. Instead of aiming at the minimum of a given objective function over a constraint set, it seeks a feasible point at which the objective function value is reduced — though not necessarily minimal — compared to that reached by the feasibility-seeking algorithm alone. This can be advantageous for problems in which the constraints may be inconsistent, in which secondary goals such as noise reduction or regularization describe soft preferences rather than hard targets, or in which a mathematically optimal solution is not strictly required. While the method has been investigated for several applications in physics, its broader use has been limited, in part due to the lack of openly available software for researchers wishing to explore it.

In this work we apply superiorization to three problems from applied physics: seismic image reconstruction, low-dose CT reconstruction and intensity-modulated radiotherapy treatment planning. These experiments are conducted with SupPy, an open-source modularized Python toolbox developed for this work, which supports execution of feasibility-seeking algorithms and their superiorized version on both the CPU and the GPU. In all three cases the superiorized algorithms achieve favorable results compared to feasibility-seeking alone, with reduced noise in the imaging examples and lowered body dose in the radiotherapy plans. For the radiotherapy case we further observe that superiorization produces clinically viable plans on infeasible constraint sets.

1. Introduction

Many problems in physics require solving high-dimensional systems constrained by lower and upper bound conditions. A common approach is to formulate them as constrained optimization problems and apply standard optimization techniques. In some practical applications, however, the optimal solution might not be always required or the consistency of the constraints might not be known in advance, which might lead standard optimization techniques to not converge or fail entirely.

The Superiorization method (SM) uses the same input as a constrained optimization problem, namely, an objective function and a system of constraints, but does not aim at finding an optimal solution. In its most general formulation it can be seen as lying in between feasibility-seeking methods and constrained optimization [1]. At its base is the use of a projection based feasibility-seeking algorithm (called the basic algorithm) that converges towards a point in the intersection of the (convex) constraints or, in infeasible cases, towards a point of closest proximity to the constraints sets (for appropriate algorithms). The superiorization methodology uses these algorithms and aims to find a point of reduced (i.e., smaller or equal but not necessarily minimal) objective function value compared to a solution of the feasibility-seeking algorithm on its own. For this purpose, a perturbation based on the objective function is performed without disturbing the convergence of the perturbed basic algorithm towards a feasible point.

The method has been investigated for many applications in physics and beyond, see e.g., [2]. These include reconstruction of CT images [3, 4, 5, 6, 7], radiotherapy treatment planning [8, 9, 10, 11], magnetoacoustic tomography [12] or floor planning [13], to name a few. However, many of these approaches are proof-of-principle implementations with specific parameters and few computational effectiveness efforts. To enable a simpler entry point to the superiorization method for researchers, we propose a modern pythonic approach that flexibly modularizes projection algorithms and perturbation strategies while leveraging both CPU and GPU architectures.

This implementation is tested on three physics application scenarios: The reconstruction of seismic images, CT reconstruction and treatment planning for radiotherapy based on dosimetric constraints. For the reconstruction examples we evaluate different configurations based on runtime, convergence rate and differences to the ground truth. Similarly, the radiotherapy example is evaluated based on runtime, convergence and clinical viability based on dosimetric measures.

We first give a broad overview of the mathematical background of the superiorization method, explain the implementation of the used strategies and then elaborate on the practical physics problems.

2. Materials and Methods

2.1. Mathematical background

2.1.1. Projections and feasibility-seeking algorithms

At the heart of the superiorization methodology are feasibility-seeking algorithms that aim to find a point in a feasible set $C := \cap_{i=1}^m C_i$, the intersection of a family of m given constraint sets $\{C_i\}_{i=1}^m$ in the n -dimensional Euclidean space \mathbb{R}^n . Such feasibility-seeking algorithms often use projections onto the sets, where a projection of a point \mathbf{x} onto a set, say, C_i is defined as the point in C_i which is closest to \mathbf{x} and denoted by $P_{C_i}(\mathbf{x})$. When the set is closed and convex then the projection onto the set is guaranteed to exist and is unique. Projections onto a single set C_i can be calculated efficiently when an analytical formula exists; e.g., for a linear constraint $\langle \mathbf{a}^i, \mathbf{x} \rangle \leq b_i$, the set C_i and the formula for projecting onto it are given by

$$C_i = \{\mathbf{x} \in \mathbb{R}^n \mid \langle \mathbf{a}^i, \mathbf{x} \rangle \leq b_i\} \quad (1)$$

$$P_{C_i}(\mathbf{x}) = \begin{cases} \mathbf{x} - \frac{\langle \mathbf{a}^i, \mathbf{x} \rangle - b_i}{\|\mathbf{a}^i\|^2} \mathbf{a}^i, & \text{if } \langle \mathbf{a}^i, \mathbf{x} \rangle > b_i, \\ \mathbf{x}, & \text{if } \langle \mathbf{a}^i, \mathbf{x} \rangle \leq b_i. \end{cases} \quad (2)$$

Adaptations/Relaxations of such projections can be achieved via a relaxation parameter. For any set $\Omega \subseteq \mathbb{R}^n$ and any real number $\lambda \in [0, 2]$ the endpoint of the projection operator can be modified by letting the outcome, denoted by $P_{\Omega_\lambda}(\mathbf{x})$, be a point on the line connecting the point \mathbf{x} with its projection, i.e.,

$$P_{\Omega_\lambda}(\mathbf{x}) = \mathbf{x} + \lambda(P_\Omega(\mathbf{x}) - \mathbf{x}). \quad (3)$$

Additionally, a step-size function $\sigma(\mathbf{x})$ can be added in order to obtain a more general formula:

$$P_{\Omega_{\lambda,\sigma}}(\mathbf{x}) = \mathbf{x} + \lambda\sigma(\mathbf{x})(P_\Omega(\mathbf{x}) - \mathbf{x}) \quad (4)$$

with $P_{\Omega_{\lambda,\sigma}}$ being called an extrapolation of P_{Ω_λ} when $\sigma(\mathbf{x}) > 1$, see [14, Definition 2.4.1].

Using these ideas of projecting onto a single set C_i , feasibility-seeking projection methods aim to find a point in the intersection of multiple sets by iteratively projecting onto the individual sets, taking advantage of the frequently occurring situation in which projections onto the individual sets are computationally more manageable. To do so, different methods are used that can broadly be put into four categories: Sequential algorithms, simultaneous algorithms, block-iterative projection (BIP) algorithms or string-averaging projection (SAP) methods. Several algorithms for feasibility-seeking have been published that build and extend on these principles ([15, 16, 17, 18, 19, 20, 21] and many more).

A broad introduction to projection methods and their mathematical properties is beyond the scope of this work, so we refer the reader to the vast literature on the topic (e.g., [14, 22]); in the sequel we only highlight some of the most important aspects.

When the intersection of the constraints is nonempty, the above mentioned methods generate sequences of iterations that converge under reasonable mathematical conditions, see, e.g., [23, 24, 25], and [26, 18] for some of their extrapolations. For the simultaneous projection method (and some of its variants) convergence is even ensured in the case of an empty intersection. In this case, any iterative sequence generated by the algorithm will approach a point of closest proximity to the constraints [27, 18, 14]. Proximity can be measured by

$$\mathcal{P}(\mathbf{x}) = 1/2 \sum_{i=1}^m w_i \|P_{C_i}(\mathbf{x}) - \mathbf{x}\|^2. \quad (5)$$

where w_i are weights assigned to the individual constraints, with $w_i > 0$ and $\sum_{i=1}^m w_i = 1$. This function is therefore the "half weighted sum of the squared distances" of a point to the members of the family $\{C_i\}_{i=1}^m$.

Another class of problems suitable for feasibility-seeking algorithms are split problems. These problems consist of an input vector $\mathbf{x} \in C \subseteq \mathbb{R}^n$, an output vector $\mathbf{y} \in Q \subseteq \mathbb{R}^m$ and a matrix $\mathbf{A} \in \mathbb{R}^{m \times n}$ connecting the two spaces. The goal of the "split-feasibility problem" then is to find a vector \mathbf{x}^* that fulfills

$$\mathbf{x}^* \in C = \bigcap_{i=1}^s C_i \text{ such that } \mathbf{y}^* = \mathbf{A}\mathbf{x}^* \in Q = \bigcap_{j=1}^t Q_j, \quad (6)$$

see, e.g., [28]. Among popular approaches to this problem [28, 29, 30], is the CQ -Algorithm of Byrne [30] which uses separate projection methods onto C and Q . Starting from an arbitrary initialization point $x^0 \in \mathbb{R}^n$ it uses the iterative process:

$$x^{k+1} = P_C(\mathbf{x}^k + \gamma \mathbf{A}^T (P_Q(\mathbf{A}\mathbf{x}^k) - \mathbf{A}\mathbf{x}^k)), \quad (7)$$

where $\gamma \in (0, 2/L)$ with L being the largest eigenvalue of $\mathbf{A}^T \mathbf{A}$. While this value is computationally expensive to calculate, in practice an upper bound $L^* \geq L$, using the squared Frobenius norm $\|\mathbf{A}\|_F^2$ of the matrix \mathbf{A} , can be used instead: [31]

$$L^* := \|\mathbf{A}\|_F^2 = \sum_{i=1}^m \sum_{j=1}^n |a_j^i|^2, \quad (8)$$

where the a_j^i 's are the elements of the matrix.

2.1.2. The superiorization method

By defining an objective function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ in addition to the family of m given constraints sets $\{C_i\}_{i=1}^m$, we have in hand the necessary input for a constrained minimization problem. However, instead of applying constrained minimization algorithms we resort to the “superiorization method”. Extending feasibility-seeking algorithms, the superiorization method (SM) aims to achieve a reduced (not necessarily minimal, thus called “superior”) objective function value when compared to the solution that the feasibility-seeking algorithm alone reaches. I.e., if the feasibility-seeking algorithm converges to point x^* , the superiorization method aims to generate an iterative sequence $\{x^k\}_{k=0}^\infty$ that will converge to a solution point x^{**} of the feasibility-seeking algorithm for which $f(x^{**}) \leq f(x^*)$.

This is achieved by perturbing the feasibility-seeking iterates with negative gradients (or subgradients) of the given objective function f . For a feasibility-seeking algorithm governed by an operator $T : \mathbb{R}^n \rightarrow \mathbb{R}^n$ and generating an iterative feasibility-seeking sequence by the process

$$x^0 \in \mathbb{R}^n, \quad \mathbf{x}^{k+1} = \mathbf{T}(\mathbf{x}^k), \quad \text{for all } k \geq 0, \quad (9)$$

the perturbed iterative step of the SM can be formulated as

$$\mathbf{x}^{k+1} = \mathbf{T}(\mathbf{x}^k + \beta_k \mathbf{v}^k), \quad (10)$$

where β_k is a step-size and \mathbf{v}^k is a vector in a direction that leads to a reduced objective function value.

The algorithm is given by the pseudo-code in Algorithm 1. The perturbation phase in lines 5 to 18 is followed by the feasibility-seeking phase.

Feasibility-seeking algorithms whose convergence to a feasible point is retained in spite of interlacing into them perturbations $\beta_k \mathbf{v}^k$ which are bounded are called *bounded perturbation resilient*. Bounded perturbations require that the series of step-sizes does not diverge $\sum_{k=0}^\infty \beta_k < \infty$ and that the sequence $(\mathbf{v}^k)_{k=0}^\infty$ is bounded [1].

While derivative-free superiorization is possible [32], the common way in the literature to calculate the direction vector \mathbf{v}^k is using a normalized nonzero subgradient, whenever available. For an objective function f and its subgradient $\mathbf{g}_f(\mathbf{x}^k)$ at point \mathbf{x}^k it can be expressed as $\mathbf{v}^k := \frac{\mathbf{g}_f(\mathbf{x}^k)}{\|\mathbf{g}_f(\mathbf{x}^k)\|}$, where the division by the norm is used to keep the sequence bounded. To choose the step-size β_k , a power law $\beta_k := \gamma \alpha^\ell$ can be used. Here, γ is a step-size kernel, ℓ is an integer that is related to the iteration index k , and is increasing as iterations proceed, and $\alpha \in (0, 1)$ is a step-size parameter to ensure that the step-sizes decrease over time.

While this can substantially reduce the influence of the perturbation phase in later iterations, Aragón-Artacho et al. [29] recently showed that it is possible to “restart” the perturbations to counteract their diminishing effectiveness without interfering with the bounded resilience of the feasibility-seeking underlying algorithm. To keep the perturbations bounded, ℓ has to increase after each restart (e.g., by setting $\ell = 1$ after the first restart, $\ell = 2$ after the second restart and so on).

Besides using a power law, Nikazad et al. [7] introduced a subgradient-projection based method to automatically choose a step-size compatible with bounded perturbation resilience:

$$\beta_k := \begin{cases} 0, & \text{if } f(\mathbf{x}^k) \leq f_{\text{ref}} \\ \frac{f(\mathbf{x}^k) - f_{\text{ref}}}{\|g_f(\mathbf{x}^k)\|}, & \text{if } f(\mathbf{x}^k) > f_{\text{ref}}, \end{cases} \quad (11)$$

with f_{ref} being a real number that serves as a reference value, that is automatically updated throughout the iterations and $g_f(\mathbf{x}^k)$ being the (sub)gradient of the objective function at the current iteration index.

Beyond these perturbation strategies for superiorization, different approaches like the *Heavy Ball Perturbation* or *Nesterov perturbations* [33, 10] have been explored, but are beyond the scope of this work.

2.2. Implementation

Each iteration of a superiorization algorithm comprises a perturbation phase and a feasibility-seeking phase, that modify the current iterate mostly independently of each other. For general implementations of the superiorization method it is, therefore, reasonable to decouple them and allow to set up feasibility-seeking and perturbation strategies separately. This allows to, e.g., easily explore the effect of different perturbation strategies by keeping the corresponding feasibility-seeking algorithm fixed. We followed this approach while constructing SupPy [34], an open-source *Python* native toolbox for superiorization that we built for this work.¹ The modular structure that we adopted allows different feasibility-seeking algorithms and different perturbation strategies to be implemented as classes in respective modules and can be combined through classes from the overarching superiorization module.

A more in-depth explanation of these main components are further explained in the following.

¹The SupPy toolbox is available on GitHub with a *BSD 3-Clause License*. Scripts used to generate and run the algorithms are available on a dedicated branch based on *v0.4.0*.

Algorithm 1 Superiorization: The Superiorized version of a feasibility-seeking algorithm

Require: \mathbf{x}^0 : Initial point
Require: $\mathbf{T}(\mathbf{x}^k)$: Feasibility-seeking algorithmic operator
Require: f : Objective function
Require: n_{red} : number of function reduction steps in each perturbation phase

- 1: $k \leftarrow 0$
- 2: $\mathbf{x}^k \leftarrow \mathbf{x}^0$
- 3: **while** Stopping criteria not met **do**
- 4: $n \leftarrow 0$ ▷ Perturbation phase
- 5: $\mathbf{x}^{k,n} \leftarrow \mathbf{x}^k$
- 6: **while** $n < n_{\text{red}}$ **do** ▷ Apply n_{red} function reduction steps
- 7: $loop \leftarrow True$
- 8: $j \leftarrow 0$
- 9: **while** $loop$ **do**
- 10: $\mathbf{z} = \mathbf{x}^{k,n} + \beta_{k,j} \mathbf{v}^k$
- 11: $j \leftarrow j + 1$ ▷ function reduction step
- 12: **if** $f(\mathbf{z}) \leq f(\mathbf{x}^{k,n})$ **then**
- 13: $n \leftarrow n + 1$
- 14: $\mathbf{x}^{k,n} = \mathbf{z}$
- 15: $loop \leftarrow False$
- 16: **end if**
- 17: **end while**
- 18: **end while**
- 19: $\mathbf{x}^{k+1} = \mathbf{T}(\mathbf{x}^{k,n})$ ▷ Feasibility-seeking algorithm iteration
- 20: $k \leftarrow k + 1$
- 21: **end while**

2.2.1. Feasibility-seeking

Feasibility-seeking algorithms can cover many types of problems, from small scale examples intended for the study and verification of algorithms, to large scale linear problems that thrive with dedicated implementations. Both of these are possible within SupPy.

Smaller problems can be set up by instantiating individual constraints and treating them with an appropriate projection method (e.g., a sequential or a simultaneous projections method).

A simple example of how such a problem can be set up in SupPy is presented in

Listing 1.

```

1 import numpy as np
2 from suppy.projections import BallProjection,
   SequentialProjection
3 center_1, center_2 = np.array([1,1]), np.array([-0.5, -1])
4 radius_1, radius_2 = 2, 1
5 ball_1 = BallProjection(center_1, radius_1)
6 ball_2 = BallProjection(center_2, radius_2)
7 joined_projection = SequentialProjection([ball_1, ball_2])

```

Listing 1: Example of how to set up a small scale feasibility-seeking example in SupPy. The goal of the problem here is to find the intersection of two balls via a sequential projection.

While this approach could also be used for larger problems, it becomes computationally inefficient and faster runtimes can be achieved by employing matrices, as done in several dedicated implementations in SupPy.

Apart from these common problem formulations, projections can be fully customized by deriving them from a *CustomProjection* interface of SupPy.

Projection based feasibility-seeking algorithms all aim to find a point inside a feasible set or generate an iterative sequence that converges towards a point of closest proximity to all individual sets, see Subsection 2.1.1. To provide a measure of distance from the feasible set or check whether a point is feasible, each algorithm needs to evaluate the proximity of a current iterate to the constraints sets.

To account for all of this, each algorithm requires a method to calculate the updated iteration of the current iteration and a proximity measure. Projection algorithms further require a “control sequence” that dictates in which order update steps are performed until feasibility is reached or a suitable stopping criterion is met. In SupPy this is realized through a *BaseProjection* class that provides an interface for the *project* and *proximity* functions, which all projections and projection algorithms are using.

Proximity for individual constraints can be calculated through analytical formulas, while algorithms combining several constraints evaluate a weighted sum of the distances to the comprising constraints, see Equation 5.

For proximity, SupPy implements an extension of the weighted sum of squared distances. I.e., a p -norm like general power sum with the power p , chosen by the user (with default $p = 2$). Denoting the distance to the i -th constraint by \mathcal{P}_i we use,

$$\mathcal{P} := \sum_{i=1}^m \mathbf{w}_i (\mathcal{P}_i)^p. \quad (12)$$

A max norm measure is also available,

$$\mathcal{P}_{\max} := \max_{1 \leq i \leq m} \mathcal{P}_i. \quad (13)$$

For split-feasibility problems, proximity measures are evaluated independently for the input and the target space. Based on these notions, the standard stopping criterion for each algorithm checks the following three conditions:

1. Whether the proximity value \mathcal{P} drops below a pre-defined threshold \mathcal{P}^* (which is by default 10^{-6}): $\mathcal{P} \leq \mathcal{P}^*$.
2. Whether the relative change in proximity value $\Delta\mathcal{P} := \left| \frac{\mathcal{P}^{k+1} - \mathcal{P}^k}{\max\{1, \mathcal{P}^k\}} \right|$, where \mathcal{P}^k is the value of \mathcal{P} at the k -th iteration, drops below a threshold $\Delta\mathcal{P}^*$ for N consecutive iterations (by default $\Delta\mathcal{P}^* = 10^{-8}$, $N = 5$).
3. Whether a maximum number of iterations N_{\max} is reached (which is by default $N_{\max} = 500$).

These criteria ensure that the algorithm terminates either when the current iterate is feasible or very close to feasibility, or when only small changes of the proximity value can be achieved by additional iterations, or when a maximal number of iterations is reached. For split-feasibility-seeking algorithms these criteria are only evaluated for the “target space” \mathbb{R}^m , consult (6). Individual user-defined stopping criteria can be added through a callback interface.

While feasibility-seeking algorithms can be evaluated on their own they can also be combined with a perturbation strategy for a superiorized version.

2.2.2. Perturbations

Perturbation strategies present an option to improve upon the current iterates of a feasibility-seeking algorithm with respect to some objective function. Commonly, this is achieved through a gradient step, although derivative free updates are also possible, see, e.g., [35]. To choose the step-size for gradient based strategies, a common approach in the literature is to use a decreasing power law, although adaptive step-sizes based on subgradient directions [7] have been shown to work, too. Both of these methods are implemented as individual classes in `SupPy`.

To initialize the perturbations, function handles for the objective function f and its (sub-)gradient \mathbf{g}_f are required, while additional parameters like the kernel γ for the power law, the parameter α , or the number of function reduction steps are treated as optional parameters.

While derivative free or higher order perturbation strategies are currently not incorporated in `SupPy`, the current class based modular structure would allow easy future extensions.

2.2.3. Superiorization

Combination of a feasibility-seeking algorithm with a perturbation strategy is achieved in SupPy through the “superiorization module”. Creation of a superiorization algorithm requires an instance of a feasibility-seeking algorithm and a compatible perturbation strategy. In case of applicable split-feasibility-seeking problems, perturbation strategies can be applied separately to the input \mathbb{R}^n and the output \mathbb{R}^m spaces (see [29]). The main *solve* function then runs the algorithm as defined in Algorithm 1 until a suitable stopping criterion is met. Listing 2 shows an example of how the setup and execution of an algorithm may look like.

Considering the numbered list of stopping conditions in Subsection 2.2.1, for the default termination criterion, individual checks for the feasibility-seeking algorithm and the objective function have to be met.

For the former, either criterion (1) or criterion (2) for the feasibility-seeking algorithms introduced in the previous section need to be met.

For the latter, a criterion similar to criterion (2) is employed, requiring that the relative change in objective function value $\Delta f := \left| \frac{f^{k+1} - f^k}{\max\{1, f^k\}} \right|$, where $f^k := f(x^k)$, falls below a threshold $\Delta f^* \leq 10^{-6}$. If the objective function criterion is met, as well as one of the criteria on the feasibility-seeking algorithm, the procedure stops. Otherwise, it continues until a maximum number of iterations is reached.

2.2.4. Interoperability

Many feasibility-seeking algorithms and perturbation strategies rely on large scale matrix operations that can be executed faster on a GPU than on a CPU. In other cases however, e.g., large scale sequential algorithms, a CPU evaluation might be faster.

To enable both options, all implementations in SupPy allow the usage of the CPU as well as the GPU. This is achieved by using the NumPy and SciPy.sparse libraries for CPU and the CuPy library for GPU operations. SupPy will, for the most part, automatically detect whether an algorithm should be executed on the CPU or on the GPU based on the input types and then act accordingly.

2.3. Applications

To verify the usability of the toolbox we present and analyze different applications from the fields of seismic wave tomography, CT reconstruction and treatment planning for radiotherapy. All scripts were run on a Lenovo ThinkPad P16 Gen2 notebook with a 13th Gen Intel(R) Core(TM) i9-13980HX 2.20 GHz, 64 GB RAM CPU and an NVIDIA RTX 3500 Ada Generation Laptop GPU. In the next subsec-

```

1 import numpy as np
2 from suppy.projections import BallProjection,
   SequentialProjection
3 from suppy.perturbations import
   PowerSeriesGradientPerturbation
4 from suppy.superiorization import Superiorization
5
6 func_1 = lambda x: x@x
7 grad_1 = lambda x: 2*x
8
9 #set up model
10 center_1, center_2 = np.array([1.2,0]), np.array([0,1.4])
11 radius = 1
12 ball_1 = BallProjection(center_1, radius)
13 ball_2 = BallProjection(center_2, radius)
14 x0 = np.array([2.5,1.5])
15
16 proj = SequentialProjection([ball_1,ball_2])
17
18 pert = PowerSeriesGradientPerturbation(func_1,grad_1)
19
20 sup = Superiorization(proj,pert)
21
22 #solve the problem
23 x_proj = proj.solve(np.array([2.5,1.5]),storage =True)
24
25 xF = sup.solve(np.array([2.5,1.5]),10,storage = True)

```

Listing 2: Example of how to set up a small scale superiorization example in SupPy. The goal of the problem here is to find the intersection of two balls via a sequential projection with an additional objective to reduce the distance to the origin.

tions, different algorithms that are available in the toolbox are highlighted through applications in the realm of physics.

2.3.1. Seismic Tomography

The first problem we showcase is in the field of seismographic tomography. Here, spatial information about the medium traversed by seismic waves is reconstructed from propagation time measured between source-receiver pairs. The reconstruction problem can be modeled as an approximate system of linear equations,

$$\mathbf{Ax} \approx \mathbf{b} \tag{14}$$

where \mathbf{b} is the measured (noisy) data, \mathbf{x} is the (discretized) $N \times N$ structure, reformulated as a single column vector, and \mathbf{A} is a matrix encoding the physics of the reconstruction problem. A solution of these equations can be achieved through least-squares minimization problem ($\min \|\mathbf{Ax} - \mathbf{b}\|_2^2$), with possible additional regularization to counteract the noise and steer the reconstruction towards acceptable physical solutions [36, 37].

We explored this problem with `SupPy` based on the reconstruction of 2-dimensional (2D) tectonic plates. In contrast to the least-squares minimization approach, a projection based feasibility-seeking algorithm was used to solve the system of linear equations while different perturbation strategies were applied, to perform a superiorization.

The underlying problem was generated using the *seismicwavetomo* setup of the `AIR Tools II Toolbox` [38], which uses a simplified model that assumes that waves travel only in the first Fresnel zone. 64 sources and 128 receivers were used to simulate the measurements on an underlying structure of 256×256 pixels, leading to a matrix \mathbf{A} of size 8192×65536 . Since the problem $\mathbf{Ax} = \mathbf{b}$ has an exact solution, we also investigated a modified version with 5% Gaussian noise added to the “measurements” vector \mathbf{b} in order to simulate a more realistic setting.

For feasibility-seeking we used the Diagonally Relaxed Orthogonal Projection (DROP) algorithm [17] which is also present in the CPU-based `AIR Tools II Toolbox` which implements several sequential and simultaneous reconstruction techniques in Matlab. The DROP algorithm is a variation of the simultaneous projection that takes the sparsity pattern of matrix \mathbf{A} into account (see the Appendix at the end of the paper).

Besides the feasibility-seeking on its own, we also investigated the superiorized algorithm by evaluating three different perturbation approaches on this problem. In particular, we looked at different regularization techniques ($L1$, $L2$ and Total

Variation (TV)-regularization) and performed perturbation strategies inspired by them.

Total Variation for a 2D $N \times N$ image \mathbf{X} can be computed as

$$TV(\mathbf{X}) = \sum_{i=1}^{N-1} \sum_{j=1}^{N-1} \sqrt{(\mathbf{X}_{i+1,j} - \mathbf{X}_{i,j})^2 + (\mathbf{X}_{i,j+1} - \mathbf{X}_{i,j})^2} \quad (15)$$

where $\mathbf{X}_{i,j}$ is the image brightness at the (i, j) -th pixel, see, e.g., [4], and relies on the assumption that real world images usually do not contain random noise, but contain gradual changes. To calculate the subgradient of the TV , required for the perturbations, we used in this work the methodology explained by Combettes et al. [39, 4].

All perturbations were performed using the standard perturbation strategy described in Subsection 2.1.2. In particular, the step-sizes β_k were updated here according to a power law $\beta_k = \gamma \alpha^\ell$ with $\gamma = 1$ and $\alpha = 0.975$, where $\ell = \ell(k)$ is an integer that depends on the iteration index k . Four reduction steps were performed in each perturbation phase and ℓ was restarted after every 50 iterations, see [29].

The perturbations were only performed in SupPy as AIR Tools II is not intended for such modifications.

To quantify how good an iterate \mathbf{x}^k approximates the true structure \mathbf{x}_T , the relative error was used, calculated via

$$\epsilon = \frac{\|\mathbf{x}^k - \mathbf{x}_T\|}{\|\mathbf{x}_T\|}. \quad (16)$$

2.3.2. CT image reconstruction

Reconstruction of Computed Tomography (CT) images has been at the forefront of applications of superiorization [3, 4, 5, 6, 7]. It is a computationally similar, but physically different, concept to that of seismic image reconstruction discussed above. For CT reconstruction, x-ray induced line integrals are taken from various angles, with each direction encoding different information of the patient’s anatomy. Well-known methods to retrieve the anatomical information from these line integrals are the filtered back-projection (FBP), the class of iterative reconstruction techniques, as well as more recent deep-learning based approaches [40, 41]. The problem to solve is a linear system of equations, similar to Equation 14, with \mathbf{A} in this case encoding the information on how much each pixel is contributing to the measurements b_i of the i -th line integral.

Superiorization for CT reconstruction with SupPy is demonstrated on the Benchmark Dataset for Low-Dose CT (LoDoPaB-CT) dataset [42]. It consists of 362×362

CT images and corresponding sinograms based on simulated low dose-measurements acquired with scans taken over 1000 equidistant angles between 0° and 180° , each measurement being recorded by 513 evenly-spaced detector bins, resulting in a $(513,000) \times (131,044)$ sparse reconstruction matrix \mathbf{A} . While the dataset does not offer the reconstruction matrix \mathbf{A} directly, it can be extracted from the ASTRA-toolbox [43, 44] used to generate the dataset.

We present the reconstruction process using different feasibility-seeking algorithms and different perturbation strategies of the superiorization algorithm that are available in `SupPy`. The first experiment looked at different feasibility-seeking algorithms, evaluating their runtime and convergence behavior on their own, as well as in combination with a fixed perturbation strategy of the superiorization algorithm. A second experiment compared the effect of different perturbation strategies of the superiorization algorithm evaluated on the same feasibility-seeking algorithm.

In the first experimental setup, we compared the following sequential and simultaneous feasibility-seeking algorithms.

1. The sequential projection method of Kaczmarz algorithm [19], also known in the field of image reconstruction as the Algebraic Reconstruction Technique (ART) see, e.g. [45].
2. The Extrapolated Landweber (EL) method [14, 26].
3. The Conjugate Gradient (CG) method [27].
4. A modification of the Landweber algorithm with built-in error minimization relaxation (EMR) [18].

For the simultaneous algorithms runtimes were evaluated for the CPU and GPU, while the sequential algorithm was only run on the CPU. An overview of the different update steps used can be found in the Appendix below.

For the perturbation in the first experiment, a kernel $\gamma = 5$, step size parameter $\alpha = 0.99$ and $n_{\text{red}} = 4$ function reduction steps in each perturbation phase were used. In addition, restarts [29] were performed every $n_{\text{restart}} = 50$ iterations.

In the second experimental setup different perturbation strategies in combination with the EMR algorithm were tested. In particular, variations of parameters for the power series perturbation were explored, as well as an adaptive perturbation strategy. A summary of the parameters used for the power series is found in Table 1. For the adaptive perturbation strategy no step-size parameters had to be chosen and the initialization was performed as described in [7].

Besides running for a fixed number of iterations and analyzing the performance of the algorithms, we also looked at a custom stopping criterion based on a variance measure of consecutive iterates [7]. In each iteration the variance of the relative

	α	γ	n_{red}	n_{restart}
Strategy 1	0.99	5	4	50
Strategy 2	0.99	5	4	/
Strategy 3	0.9	5	4	50
Strategy 4	0.9	5	4	/
Strategy 5	0.9	1	1	/
Strategy 6	Adaptive			

Table 1: Parameters used for the power series perturbation strategies. For the initialization of the adaptive strategy see [7].

change between iterates was evaluated

$$\text{Var}(\boldsymbol{\omega}^k) = \frac{1}{k-1} \sum_{i=1}^k (\omega_i^k - \bar{\omega}^k)^2 \quad (17)$$

$$\text{with } \boldsymbol{\omega}^k = \left(\frac{\|\mathbf{x}^1 - \mathbf{x}^0\|}{\|\mathbf{x}^1\|}, \dots, \frac{\|\mathbf{x}^{i+1} - \mathbf{x}^i\|}{\|\mathbf{x}^{i+1}\|}, \dots, \frac{\|\mathbf{x}^k - \mathbf{x}^{k-1}\|}{\|\mathbf{x}^k\|} \right), \quad (18)$$

where $\bar{\omega}^k$ denotes the mean of the vector at iteration k . Once $\text{Var}(\boldsymbol{\omega}^k)$ falls below a predetermined threshold (here 0.01) the algorithm stops. This can be used to stop the reconstruction process early and therefore counteracts the overfitting of the noise in later iterations which leads to an increase in relative error (see Equation 16).

2.3.3. Radiotherapy treatment planning

The next application we highlight is in the field of treatment planning for radiotherapy. Radiotherapy is a modality used for treating cancer, with around 50% of all cancer patients receiving it during their treatment [46]. For each individual patient a suitable setup of the radiation field delivered by the radiation machine has to be found based on their anatomy and on the medical prescription. Arriving at such a treatment plan involves several steps, starting with the acquisition of a CT of the patient. Based on this image, structures of the patient, like the tumor volume and surrounding organs, are segmented. The next step then deals with finding a suitable irradiation setup through modeling the situation as a feasibility-seeking problem or as an optimization problem.

While several parameters such as the beam angles can be varied, the main mathematical feasibility or optimization problem in treatment planning deals with finding suitable intensities of the beams \mathbf{w} that allow delivery of a specific dose distribution \mathbf{d} .

For this a discretized model is used: The 3D CT image of the patient is divided into small cubic volumes called *voxels* (which are analogous to pixels in a 2D picture) while the external radiation beams are separated into smaller subelements called beamlets with variable intensity w_j of each. The dose delivered to the voxels can be expressed as a vector $\mathbf{d} = (d_i)_{i=1}^m \in \mathbb{R}^m$ in the "dose-space", while the intensities can be modeled as the vector $\mathbf{w} = (w_j)_{j=1}^n \in \mathbb{R}^n$ in the "intensity-space". The physics of the model is condensed into the $m \times n$ (pre-computed) dose influence matrix $\mathbf{A} = (a_{ij})_{i=1,j=1}^{m,n}$. Each element a_{ij} in \mathbf{A} is the dose absorbed in voxel i due to a unit of intensity along the j -th beamlet. This means that

$$d_i = \sum_{j=1}^n a_{ij} w_j \quad (19)$$

is the total dose absorbed in voxel i due to an intensity vector \mathbf{w} .

To find a suitable dose distribution coverage of the tumor has to be weighed against sparing of surrounding organs at risk. For this, goals on the different organs, as defined by a clinician, are translated into mathematical objective functions and constraints. For example the dose in the tumor volume may have to be kept between lower and upper bounds, d_{\min} and d_{\max} , respectively. This is modeled as a constraint

$$d_{\min} \leq d_i \leq d_{\max} \quad \forall i \in S, \quad (20)$$

for all $i \in S$, where S is the set of voxels that belong to the structure of interest.

A different goal of reducing the mean dose in an organ at risk surrounding the tumor, could be modeled as an objective function that should be minimized

$$f_{Mean}(\mathbf{d}) = \frac{1}{N_S} \sum_{i \in S} d_i \quad (21)$$

where N_S is the number of voxels in the structure S .

Further options are dose-volume constraints (DVC) that come in two different variants:

$$D_{V\%} \leq d_{ref} \quad (22)$$

$$D_{V\%} \geq d_{ref} \quad (23)$$

Here $D_{V\%}$ stands for the dose absorbed in voxels that constitute a fraction of $V\%$ of all voxels included in a certain organ, without regard to which specific voxels are the ones that absorb this. The first means that not more than $V\%$ of the voxels of

an organ should absorb a dose higher than d_{ref} . The second requires the opposite - that a specific dose d_{ref} needs to be delivered to at least $V\%$ of the voxels of an organ.

Combining all such objectives and constraints can be formulated as the following constrained optimization problem.

$$\begin{aligned}
& \underset{\mathbf{w}}{\text{minimize}} && \mathbf{f}(\mathbf{d}) = \sum_l p_l f_l(\mathbf{d}) \\
& \text{subject to} && \mathbf{d} \in C = \cap_k C_k, \\
& && \mathbf{d} = \mathbf{A}\mathbf{w}, \\
& && \mathbf{w} \geq 0
\end{aligned} \tag{24}$$

Here f_ℓ are objective functions (with associated penalties p_ℓ), C_k are constraints sets and $\mathbf{w} \geq 0$ a physical constraint since beam intensities cannot be negative.

If there are no objective functions f_ℓ involved then solving to find a vector \mathbf{w} that will obey the constraints is a feasibility-seeking problem.

An alternative approach to solving the full-fledged constrained optimization problem (Equation 24) is to use these data in a superiorization approach.

This is explored here in the context of conflicting constraints, that lead to infeasibility, with an underlying split-feasibility-seeking algorithm (see Equation 6). Of particular interest is it here, since if $C = \emptyset$ is the empty set then a constrained optimization algorithm cannot work at all.

We demonstrate this on a simple structure with a horseshoe-shaped tumor wrapped around a circular organ at risk, as well as by a clinical patient case. Both cases are part of the CORT dataset [47] and openly available.

Each setup consisted of 9 photon beams arranged equally around the patient. The dose-influence matrix \mathbf{A} was calculated using the open-source radiotherapy treatment planning software `matRad` [48, 49]. Each patient had several constraints assigned to various structures and a further objective function to reduce the dose in the surrounding body (see Table 2 and Table 3, respectively). The constraints in both cases were constructed in such a way, that they cannot be fulfilled simultaneously, leading to the problem being infeasible, which makes it non applicable for many common constrained optimization solvers. To demonstrate this for the horseshoe phantom, the same set of objectives and constraints was passed to the Interior Point OPTimizer IPOPT solver [50]. For the head-and-neck plan a reference plan was calculated on optimizable objectives and constraints to assess the clinical relevance.

The constraints employed in the two setups comprised both linear constraints and dose-volume constraints (DVCs). These were modeled as a split-feasibility-

seeking problem (Equation 6) with the linear constraints modeled as the constraints C_k and the DVC constraints modeled as the constraints Q_j . For projections onto the C_k constraints a modification of the EMR iterate suitable for linear inequalities was applied, while for the DVC constraints the methodology described by Penfold et al.[51] was used. Lastly, negative weights were set to zero after each iteration to maintain the non-negativity requirement $\mathbf{w} \geq 0$, which was performed through a *BoxProjection* onto the nonnegative orthant \mathbb{R}_+^n in SupPy.

Perturbations for superiorization were performed with the step sizes chosen by a power law $\beta_k = \gamma\alpha^k$ with $\gamma = 1, \alpha = 0.7$ for the horseshoe-shaped phantom and $\gamma = 1, \alpha = 0.99$ for the head and neck patient. Both algorithms performed restarts after each consecutive 500 iterations.

VOI/Structure	Constraints	Objective
Target	$D_{95\%} \geq 2 Gy$	f_{Mean}
	$D_{5\%} \leq 2.17 Gy$	
	$1.93 Gy \leq d \leq 2.27 Gy$	
Core	$D_{5\%} \leq 0.33 Gy$	
Body		

Table 2: Constraints and objectives used for the horseshoe-shaped tumor plan calculation. Superiorization uses all components, while the feasibility-seeking algorithm considers only the constraints.

VOI/Structure	Superiorization		Optimization	
	Constr.	Obj.	Constr.	Obj.
Skin	$\mathbf{d} \leq 2.17 Gy$	f_{Mean}	$\mathbf{d} \leq 2.17 Gy$	f_{Mean}
PTV _{2.1}	$2.03 Gy \leq \mathbf{d} \leq 2.17 Gy$			$f_{SqDev}(2.1 Gy)$
PTV _{2.33}	$2.27 Gy \leq \mathbf{d} \leq 2.40 Gy$			$f_{SqDev}(2.33 Gy)$
Brain stem PRV	$D_{5\%} \leq 0.33 Gy$ $\mathbf{d} \leq 1 Gy$			$f_{SqOD}(0.33 Gy)$
Spinal cord PRV	$D_{5\%} \leq 0.33 Gy$ $\mathbf{d} \leq 1.33 Gy$			$f_{SqOD}(0.33 Gy)$
Parotid LT	$D_{5\%} \leq 0.33 Gy$ $\mathbf{d} \leq 1.33 Gy$			$f_{SqOD}(0.33 Gy)$
Parotid RT	$D_{5\%} \leq 0.33 Gy$ $\mathbf{d} \leq 1.33 Gy$			$f_{SqOD}(0.33 Gy)$
Larynx	$D_{5\%} \leq 0.33 Gy$ $\mathbf{d} \leq 1.33 Gy$			$f_{SqOD}(0.33 Gy)$

Table 3: Constraints and objectives used for the head and neck plan calculation. Superiorization uses all components, while the feasibility-seeking algorithm considers only the constraints. For the optimized plan a *squared deviation* objective function is used instead of the min-max dose constraints on the target and for most organs at risk a *squared overdosing* objective function is used. A further explanation with formulas for the objective functions can be found in the appendix (Table A.6).

3. Results

3.1. Application 1: Seismic image reconstruction

Reconstructions with DROP in SupPy and *AIR Tools II* on the same machine resulted in numerically identical solutions. However, running 1000 iterations of DROP took roughly 100 s with *AIR Tools II*, whereas the GPU-accelerated evaluation in SupPy took less than 10 s.

Figure 1 shows the evolution of relative error against the number of iterations. In case of the noise-free model ($\mathbf{Ax} = \mathbf{b}$) a continuous decrease of the relative error is visible. While differences between feasibility-seeking and the *L1* and *TV* perturbations are small, the perturbed versions achieve marginally lower relative errors.

For the noisy data, this improvement for the two perturbation strategies is even more pronounced.

The *L2* perturbation on the other hand is not able to achieve this, as the feasibility-seeking on its own outperforms the strategy for both noisy and noise-free data.

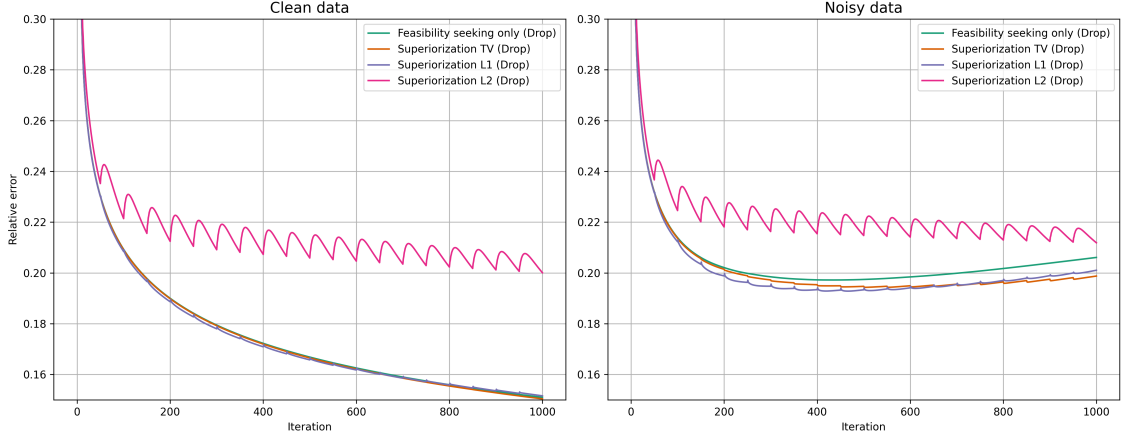


Figure 1: Evolution of the relative errors for the different perturbation strategies

However, as shown in Figure 2 and Figure 3, all models reconstructed the underlying structures relatively well, although some variations in the right tectonic plates are visible (especially for the noisy data).

3.2. Application 2: CT Image reconstruction

Table 4 shows the runtime for the different feasibility-seeking algorithms and their superiorized versions. The expected increase in runtime was observed when moving from feasibility-seeking to superiorization, due to the added perturbation phase’s operations. For the simultaneous variants (EMR, EL, CG), both perturbed and unperturbed models took only a few seconds to run on the GPU and the increase due to the perturbations is on the order of 1.5-2. When running the same algorithms on the CPU, the runtime increased significantly with the feasibility-seeking algorithms taking about 20 times as long and their superiorized version about 13-20 times as long. A similar increase could be seen when running on the GPU, albeit the magnitude was lower by a factor of $\sim 1.2-1.5$. In contrast, the sequential method’s runtime on the CPU heavily exceeded that of the simultaneous variants. Contrary to the simultaneous variants, the runtime increase because of the perturbations can be neglected due to the long runtime of a single iteration of the feasibility-seeking algorithm.

The table also shows the achieved relative errors with and without the early stopping criterion with only minor deviations when stopping based on the variance measure.

Furthermore, Figure 4 shows the convergence behavior of the tested algorithms. The sequential algorithm was not able to achieve reasonable CT reconstruction

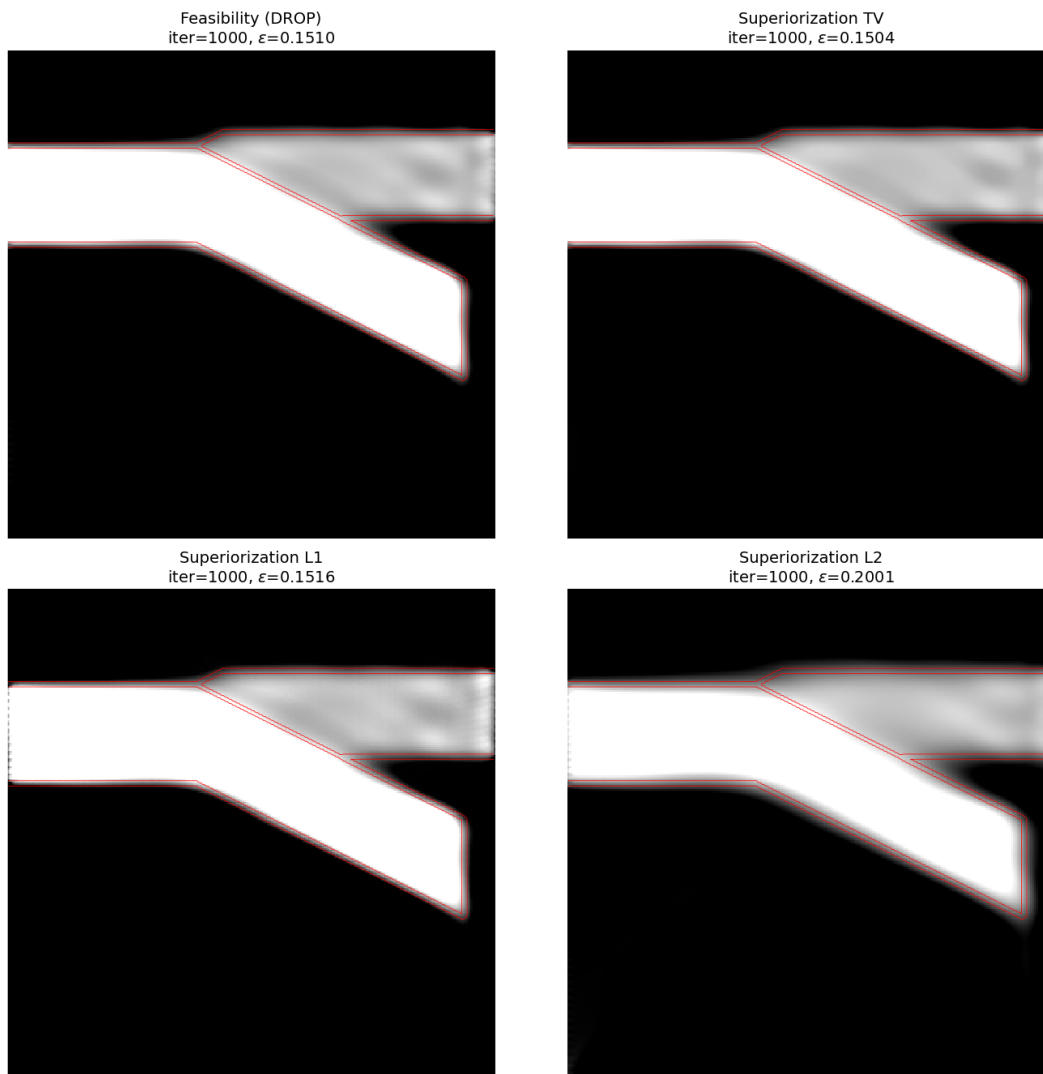


Figure 2: Seismic reconstructions for clean data. Contours of the original tectonic plates are shown in red.

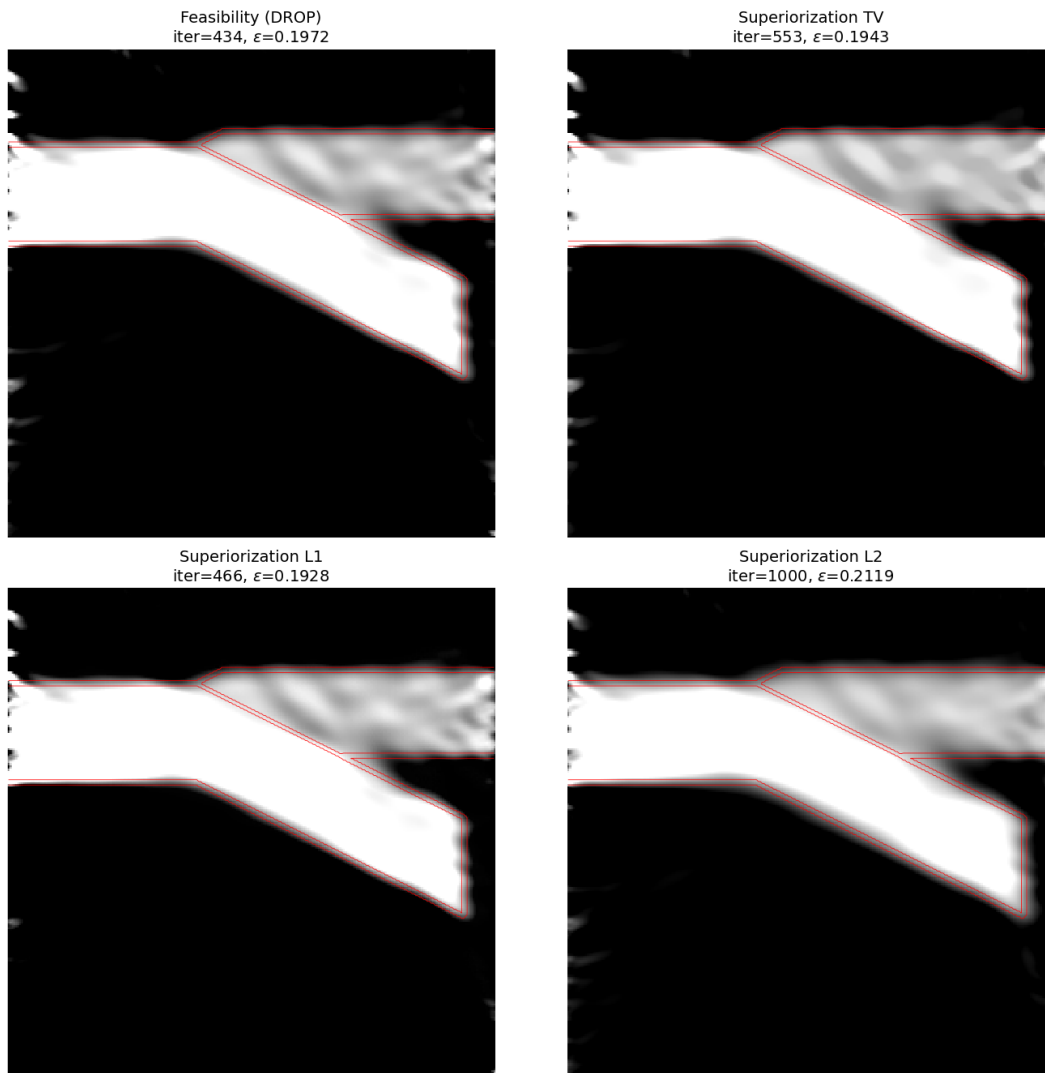


Figure 3: Seismic reconstructions for noisy data. Contours of the original image are shown in red.

Alg.		300 iterations				Variance stopping criterion					
		t_{feas} [s]	ϵ_{min}	t_{sup} [s]	ϵ_{min}	n_{feas}	t_{feas} [s]	ϵ_{stop}	n_{sup}	t_{sup} [s]	ϵ_{stop}
ART	CPU	1447	1.0	1578	1.0	147	697	14.569	144	753	11.392
EMR	GPU	10.0	0.093	17.3	0.066	109	3.7	0.110	109	6.7	0.075
	CPU	235		348			86			135	
EL	GPU	7.2	0.321	15.1	0.316	/	7.6	0.689	/	15.0	0.661
	CPU	182		277		/	202		/	244	
CG	GPU	9.4	0.090	15.5	0.068	115	3.7	0.861	114	6.3	0.069
	CPU	229		286			83			111	

Table 4: Results of the CT reconstruction. For each feasibility-seeking algorithm and its superiorized version, the table reports runtimes and relative errors under two settings. The left block terminates after 300 iterations, with ϵ_{min} the minimal relative error attained over those iterations. The right block uses the variance-based stopping criterion: n_{feas} and n_{sup} are the iterations at which the criterion was met for feasibility-seeking and superiorization, respectively, and ϵ_{stop} is the relative error at that iteration. t_{feas} and t_{sup} denote the runtimes (in seconds) of feasibility-seeking alone and of superiorization.

quality and led to an increase in relative error. Similarly, the EL algorithm initially decreased in relative error but then started fluctuating for both the perturbed and unperturbed case. This is partly expected, as neither EL nor sequential Kaczmarz is guaranteed to converge on inconsistent feasibility problems ($C = \cap_i C_i = \emptyset$). CG and EMR on the other hand were able to achieve the lowest relative errors, which are further decreased when using their superiorized versions.

Qualitative analysis of the reconstructed image slices in Figure 5 and Figure 6 reveals further details. Comparing the superiorized solution of EMR to the solution of the feasibility-seeking algorithm on its own, both variants reconstructed the anatomical features clearly, with only minor quality loss when the variance-based early-stopping criterion was applied (see Figure 5). However the superiorized version reduced the noise in the reconstructed images, underlined by the lower relative errors than its unsuperiorized version. This is highlighted by Figure 6 which shows difference plots for the unperturbed and perturbed solution. The superiorized version achieved visually better reconstruction than feasibility-seeking alone.

Figure 7 shows the results for the second experiment. The strategies without restart show an L-shape in the error curve, which is not present for the restarted strategies. Furthermore, the adaptive strategy resulted in a lower relative error than the power law strategies without restart, but was outperformed when using a restart.

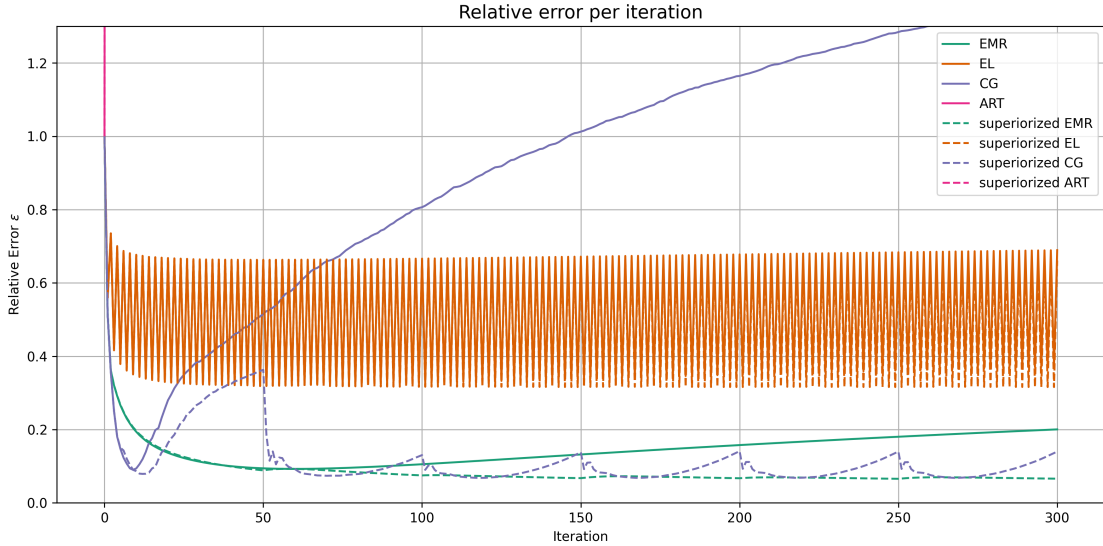


Figure 4: Evolution of relative error for different perturbation strategies. For each algorithm curves for unperturbed and perturbed algorithms are shown. The iterates of the sequential algorithm immediately reach relative errors out of the bounds of the plot and are therefore only visible in the first iteration.

3.3. Application 3: Radiotherapy treatment planning

Figure 8 and Figure 9 show a dose volume histogram (DVH) and dose distributions for the horseshoe phantom plan, respectively. Both superiorization and feasibility-seeking were started from the same initial point. In contrast to the standard optimizer which struggles with the infeasible constraints employed, feasibility-seeking and superiorization both reach plan configurations that take this into account as well as possible.

Comparing the superiorized version to the plan using feasibility-seeking alone, a reduction of the dose delivered to the surrounding body can be achieved through the perturbation, as seen in the right plot in Figure 9.

The feasibility-seeking algorithm took 105 seconds to run, while the superiorized version runs for ~ 300 seconds, both running on the GPU. Both algorithms were terminated after running 10,000 iterations.

Similar results were achieved on the more demanding head and neck patient. Figure 11 shows dose slices for a head and neck patient. As the two plots in the top row and Figure 10 show, the superiorization algorithm was able to reduce the dose in the body significantly and achieves similar dosimetric quality as the optimized plan.

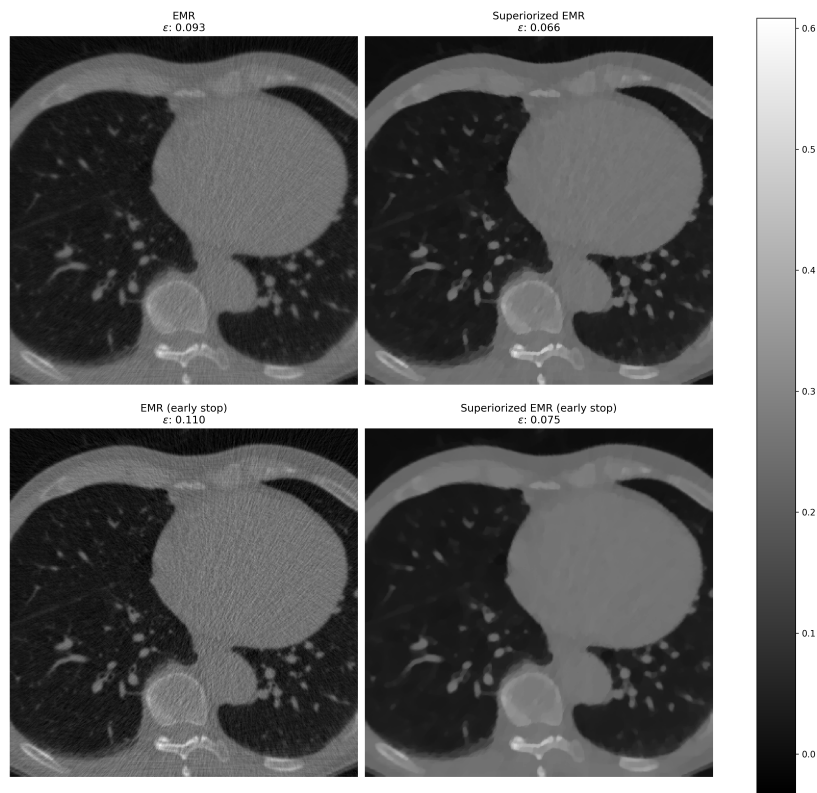


Figure 5: The ground truth CT, the best reconstruction via feasibility-seeking only and its superiorized version for the error minimizing Landweber method (EMR) are depicted.

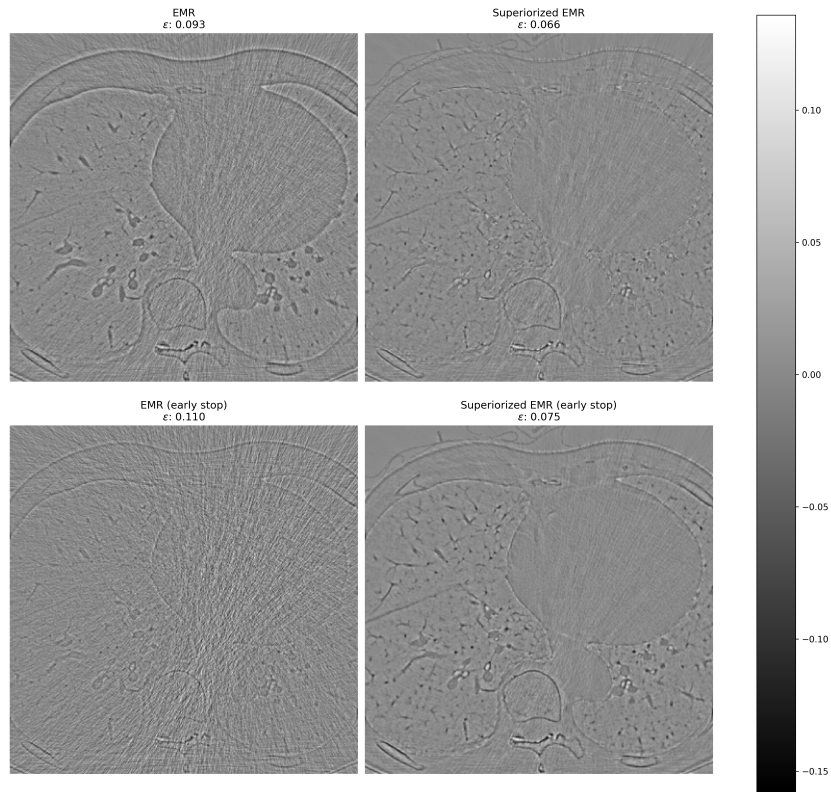


Figure 6: Difference to the true solution for the reconstructions by feasibility-seeking and by superiorization, respectively.

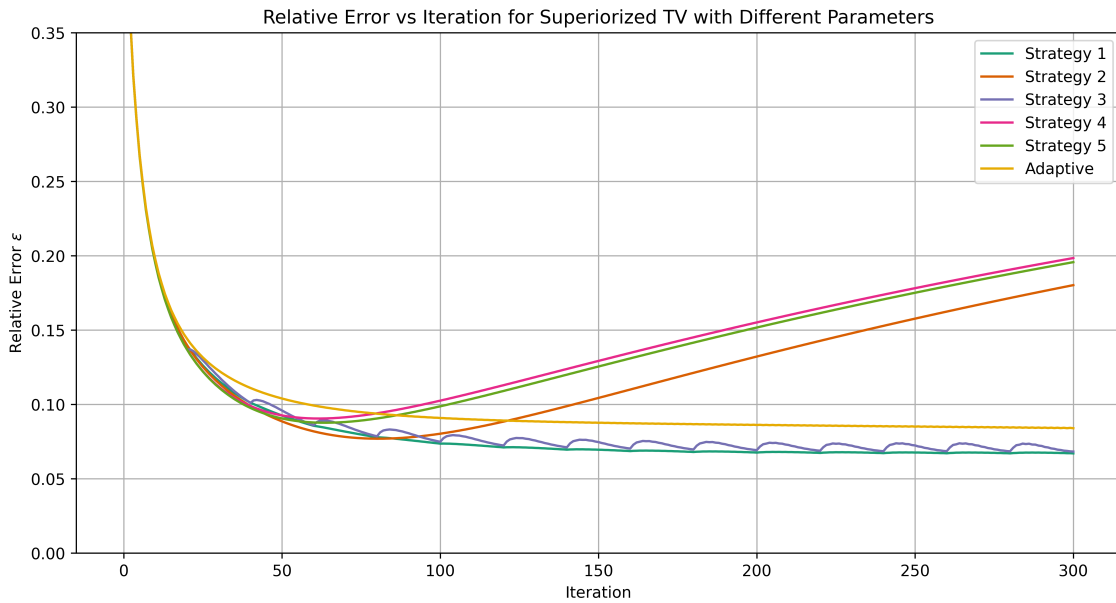


Figure 7: Evolution of the relative error for different perturbation strategies (explanation of the strategies in Table 1).

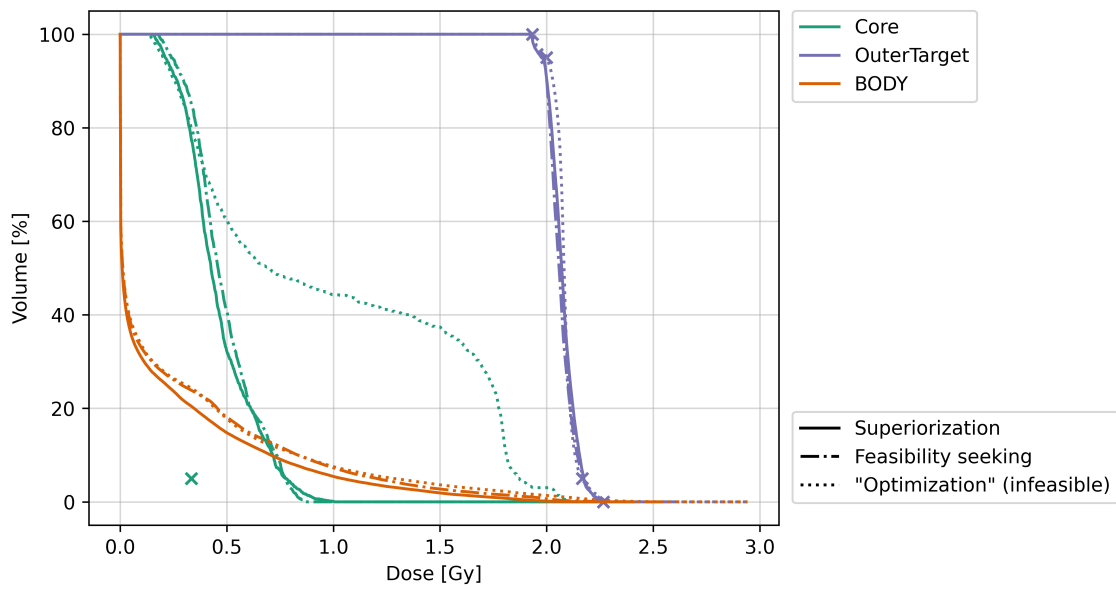


Figure 8: DVH comparison for the three different plan configurations of the horseshoe phantom. The plan label "Optimization" was obtained by the IPOPT solver.

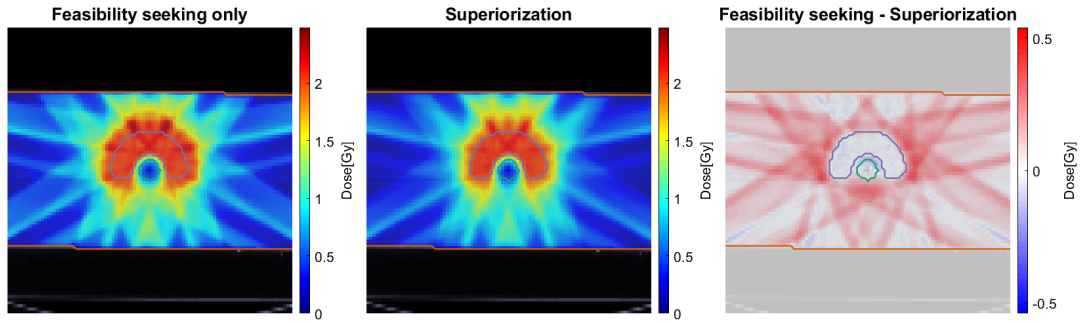


Figure 9: Dose slices for a single fraction computed with feasibility-seeking and superiorization.

Running the feasibility-seeking algorithm alone took about 1.75 minutes for 2500 iterations, while the runtime for the superiorized version increased to 4.5 minutes. The plan calculated through CPU-based optimization took about 8.5 minutes.

While the superiorized plan shows the practical applicability of DVC based constraints, their slow convergence rate reduces their efficiency. Replacing them with linear max-dose constraints achieves a comparable plan quality after 500 iterations (see also Figure 11), while requiring only 50 s of runtime.

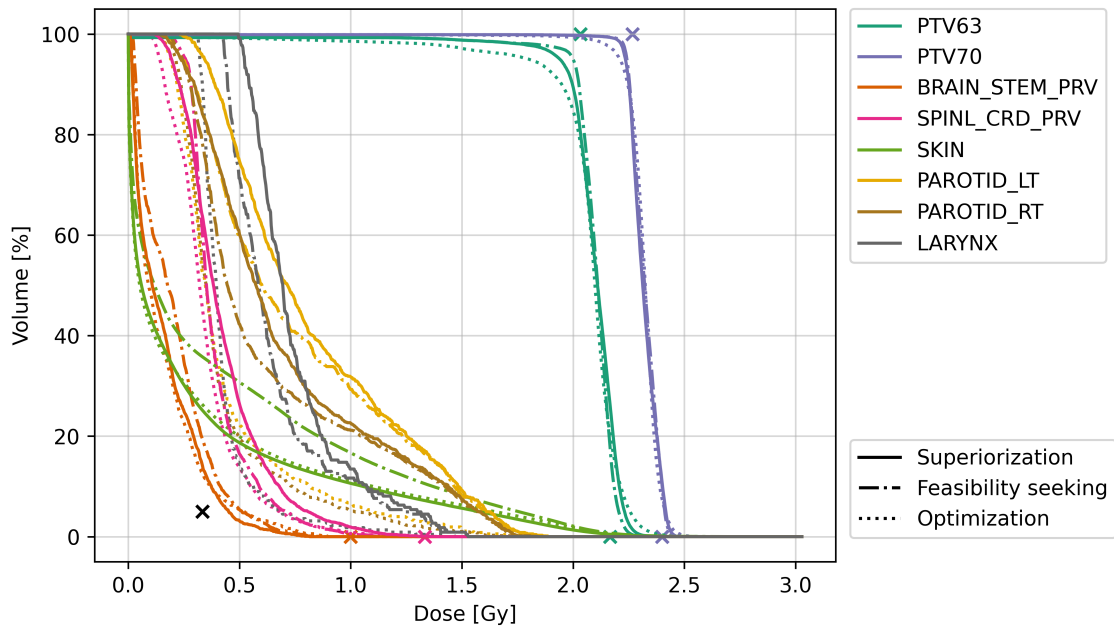


Figure 10: DVH-comparison for the three different plan configurations of the Head and neck patient. The *feasibility-seeking* and *superiorization* plan were calculated using the plan described in Table 3. Furthermore an optimized plan was calculated using only objective functions to show how a clinically relevant plan might look like. The *Superiorization* and *feasibility-seeking* plans show similar qualities, with the *Superiorization* plan additionally reducing the dose in the body, as expected.

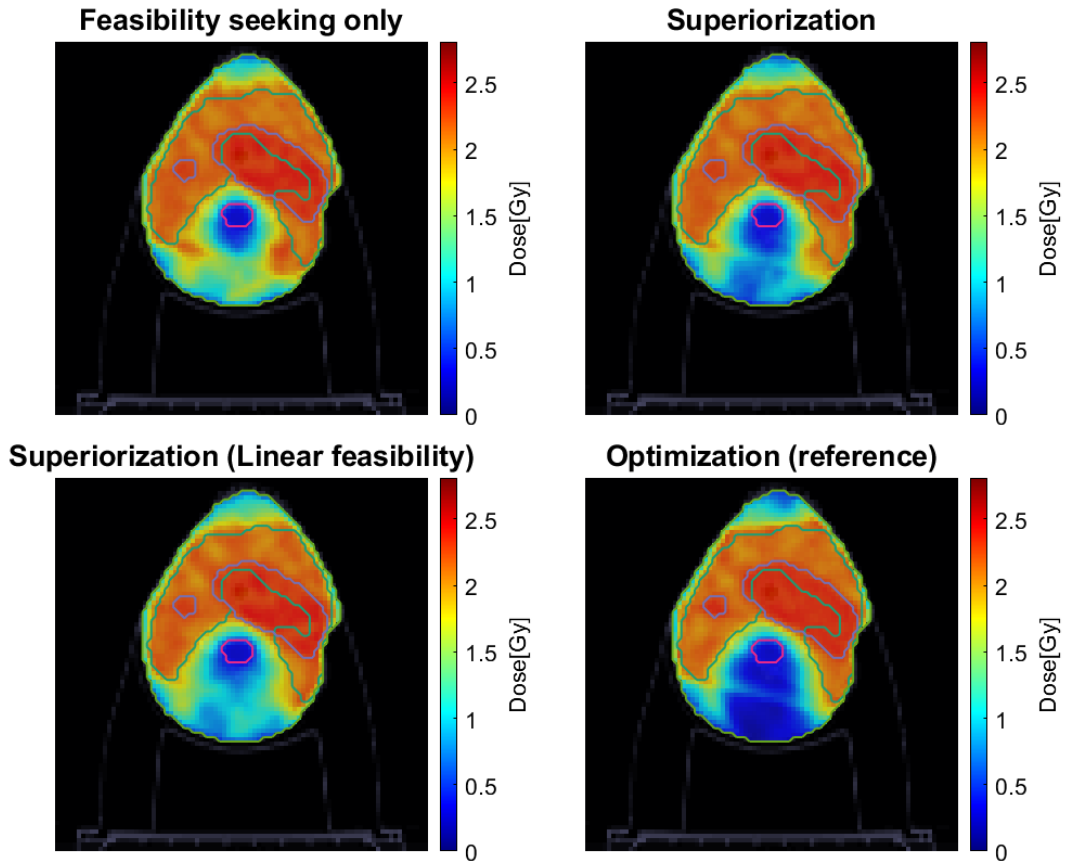


Figure 11: Comparison of different plan slices for the head and neck patient. The top row shows plans based on split-feasibility problems. As visible, the superiorized version reduces the dose in the surrounding body substantially and achieves dosimetric quality comparable to a reference plan calculated with standard optimization (bottom right). The bottom left plan additionally shows a dose distribution achieved through superiorization only employing linear constraints.

4. Discussion

This work applies the superiorization method to constrained problems in applied physics, which are paired with regularization strategies or secondary objective functions describing soft goals. The method is rigorously implemented in the presented open-source toolbox `SupPy`, enabling GPU-accelerated feasibility-seeking with adequate perturbation strategies to achieve competitive runtimes for the application of superiorization.

All three demonstrated problems employed superiorization, while focusing on different end goals.

The first experiment tackled seismic image reconstruction on noisy and noise-free data and explored the idea of using superiorization in a similar manner to regularization. Both the L1 and TV strategy were able to outperform their feasibility-seeking algorithm alone, albeit achieving only small improvements. The L2 perturbation on the other hand did not achieve any improvement over the feasibility-seeking strategy on its own.

The second experiment dealt with the reconstruction of CT images and explored the possibility to compare different feasibility-seeking algorithms as well as different perturbation strategies. Runtime and convergence analysis were performed for a sequential method run on the CPU as well as three simultaneous methods run on the GPU (EL, EMR and CG). Out of all four methods, EMR and CG performed best, while EL and the sequential Kaczmarz/ART strategy struggled to achieve meaningful reconstructions with and without superiorization. For EMR and CG the feasibility-seeking results were further enhanced through the TV-superiorization, enabling a reduction in reconstruction noise.

Comparing different perturbation strategies for the CT reconstruction showed several notable trends: First, an increased perturbation magnitude leads to improved reconstruction quality. Second, the incorporation of restarts (strategy 1 and strategy 3) yields an enhancement in solution quality and mitigates the L-shape of the error curve, thereby reducing susceptibility to noise overfitting. Third, although the adaptive perturbation approach requires fewer input parameters, it is capable of outperforming several power-law-based strategies; nevertheless, power-law approaches with high values of α and possible restarts can achieve superior results.

The possibility of direct comparisons with different algorithms or perturbation strategies, as shown in these experiments is supported by the implementation design of `SupPy`, providing interfaces for projection algorithms as well as perturbation strategies.

In the final experiment an application to radiotherapy was explored for inconsistent constraints. This could provide an advantage over standard optimization

algorithms, as inconsistent problems pose a challenge for optimization algorithms. Furthermore, the option to use a split-feasibility-seeking problem was demonstrated with dose volume constraints shown to be a viable, albeit slow option for volumetric goals.

Runtime of the used algorithms depends on their respective convergence rate, problem size and stopping criterion. For the CT reconstruction the same algorithms were evaluated on the CPU and GPU, with the latter providing a significant speedup due to the faster matrix operations.

Moving from feasibility-seeking to superiorization leads to an expected increase in runtime in all cases, whose amount depends on the computational complexity added by the perturbations. For the simultaneous CT reconstruction the perturbations approximately double the runtime, whereas it takes a multiple of the feasibility-seeking time for both radiotherapy treatment planning cases. This can be explained by looking at the sizes of the matrices employed. For the CT reconstruction the same underlying matrix is used for the calculation of the feasibility-seeking algorithm and perturbation steps. For the investigated radiotherapy plans on the other hand fewer organs are considered for the feasibility-seeking algorithm than for the perturbations, leading to a larger dose-influence matrix \mathbf{A} for the latter and therefore more computationally expensive operations.

This work and SupPy thereby serve as an entry point for future application of the superiorization method in physical problems, where secondary goals, like for example noise, are not required to be optimized to absolute minima.

While we report the current state of the toolbox, it is still actively improved upon and extended with new features. Of particular interest may be the addition of dedicated CUDA kernels for implementation of various algorithms, perturbation strategies that do not rely on subgradients, as well as matrix free options for linear algorithms.

5. Conclusion

This work demonstrated multiple potential applications for superiorization in applied computational physics with competitive solutions and runtimes. Rigorous implementation of feasibility-seeking algorithms and corresponding perturbation strategies in an open-source toolbox SupPy provides the research community with a platform to explore further applications without re-implementing the underlying algorithmic architecture.

Declaration of generative AI and AI-assisted technologies in the manuscript preparation process

During the preparation of this work the authors used Claude Opus 4.7 and Claude Opus 4.8 by Anthropic in order to spell check and improve the consistency and readability of the manuscript. After using this tool/service, the authors reviewed and edited the content as needed and take full responsibility for the content of the published article.

Acknowledgments

This work is supported by the Cooperation Program in Cancer Research of the German Cancer Research Center (DKFZ) and Israel's Ministry of Innovation, Science and Technology (MOST).

Conflict of Interest

The authors have declared no conflict of interest.

Appendix A. Used Methods

Method	Update Step
DROP [17]	$\mathbf{x}^{k+1} = \mathbf{x}^k - \mathbf{Z}\mathbf{A}^T\mathbf{M}(\mathbf{A}\mathbf{x}^k - \mathbf{b})$
Sequential/Kaczmarz [19]	$\mathbf{x}^{k,i} = \mathbf{x}^{k,i-1} - \frac{\langle \mathbf{a}^i, \mathbf{x}^{k,i-1} \rangle - b_i}{\ \mathbf{a}^i\ ^2} \mathbf{a}^i \forall C_i$
Extrapolated Landweber (EL) [14, 26]	$\mathbf{x}^{k+1} = \mathbf{x}^k - \frac{(\mathbf{A}\mathbf{x}^k - \mathbf{b})^T \mathbf{D}(\mathbf{A}\mathbf{x}^k - \mathbf{b})}{\ \mathbf{A}^T \mathbf{D}(\mathbf{A}\mathbf{x}^k - \mathbf{b})\ ^2} \mathbf{A}^T \mathbf{D}(\mathbf{A}\mathbf{x}^k - \mathbf{b})$
Landweber with error minimization (EMR) [18]	$\mathbf{x}^k - \frac{\ \mathbf{A}^T \mathbf{M}(\mathbf{A}\mathbf{x}^k - \mathbf{b})\ ^2}{\ \mathbf{M}^{1/2} \mathbf{A} \mathbf{A}^T \mathbf{M}(\mathbf{A}\mathbf{x}^k - \mathbf{b})\ ^2} \mathbf{A}^T \mathbf{M}(\mathbf{A}\mathbf{x}^k - \mathbf{b})$
Conjugate Gradient	see Zibetti et al., Algorithm 9 [27]

Table A.5: Algorithms used in the seismic tomography and CT reconstruction examples. In the case of the sequential algorithm, one iteration sweeps over all constraints C_i , while the other four handle the constraints simultaneously. D, M and Z are diagonal matrices: $D = \text{diag}(w_1/\|\mathbf{a}^1\|^2, \dots, w_n/\|\mathbf{a}^n\|^2)$, $M = \text{diag}(w_1, \dots, w_n)$ and $Z = \text{diag}(\text{nnz}(A_{:,1})^{-1}, \dots, \text{nnz}(A_{:,m})^{-1})$, where each entry corresponds to the number of nonzero elements in the respective column. In both cases $\sum_i w_i = 1$ holds.

Name	Mathematical expression
Mean Dose (f_{Mean})	$\frac{1}{N_S} \sum_i d_i$
Squared Deviation (f_{SqDev})	$\frac{1}{N_S} \sum_i (d_i - d_{\text{ref}})^2$
Squared Overdosing (f_{SqOD})	$\frac{1}{N_S} \sum_i (d_i - d_{\text{ref}})_+^2$

Table A.6: Different objective functions used in optimization for radiotherapy treatment planning. In all cases the sum goes over the dose d_i in all N_S voxels in the structure and d_{ref} is the reference value for the dose d_i . For the *Squared Overdosing* function only dose values above the reference dose are taken into account for the sum.

References

- [1] Y. Censor, R. Davidi, G. T. Herman, Perturbation resilience and superiorization of iterative algorithms, *Inverse Problems* 26 (2010) 065008. URL: <https://dx.doi.org/10.1088/0266-5611/26/6/065008>. doi:10.1088/0266-5611/26/6/065008.
- [2] Y. Censor, Superiorization and Perturbation Resilience of Algorithms: A Continuously Updated Bibliography, 2023. URL: <http://arxiv.org/abs/1506.04219>. doi:10.48550/arXiv.1506.04219, arXiv:1506.04219 [math].

- [3] E. Y. Sidky, X. Pan, Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization, *Physics in Medicine & Biology* 53 (2008) 4777. URL: <https://doi.org/10.1088/0031-9155/53/17/021>. doi:10.1088/0031-9155/53/17/021.
- [4] S. N. Penfold, R. W. Schulte, Y. Censor, A. B. Rosenfeld, Total variation superiorization schemes in proton computed tomography image reconstruction, *Medical Physics* 37 (2010) 5887–5895. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1118/1.3504603>. doi:10.1118/1.3504603, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1118/1.3504603>.
- [5] G. T. Herman, E. Garduño, R. Davidi, Y. Censor, Superiorization: An optimization heuristic for medical physics, *Medical Physics* 39 (2012) 5532–5546. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1118/1.4745566>. doi:10.1118/1.4745566, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1118/1.4745566>.
- [6] Y. Censor, K. E. Schubert, R. W. Schulte, Developments in Mathematical Algorithms and Computational Tools for Proton CT and Particle Therapy Treatment Planning, *IEEE Transactions on Radiation and Plasma Medical Sciences* 6 (2022) 313–324. URL: <https://ieeexplore.ieee.org/abstract/document/9521582>. doi:10.1109/TRPMS.2021.3107322.
- [7] T. Nikazad, M. Abbasi, L. Afzalipour, T. Elfving, A new step size rule for the superiorization method and its application in computerized tomography, *Numerical Algorithms* 90 (2022) 1253–1277. URL: <https://doi.org/10.1007/s11075-021-01229-z>. doi:10.1007/s11075-021-01229-z.
- [8] F. Barkmann, Y. Censor, N. Wahl, Superiorization of projection algorithms for linearly constrained inverse radiotherapy treatment planning, *Frontiers in Oncology* 13 (2023) 1238824. URL: <https://www.frontiersin.org/articles/10.3389/fonc.2023.1238824/full>. doi:10.3389/fonc.2023.1238824.
- [9] A. Gibali, K.-H. Küfer, D. Reem, P. Süß, A generalized projection-based scheme for solving convex constrained optimization problems, *Computational Optimization and Applications* 70 (2018) 737–762. URL: <https://doi.org/10.1007/s10589-018-9991-4>. doi:10.1007/s10589-018-9991-4.
- [10] E. Bonacker, *Perturbed Projection Methods in Convex Optimization - Applied to Radiotherapy Planning*, Ph.D. thesis, Fraunhofer Verlag, 2020.

URL: <https://publica.fraunhofer.de/handle/publica/282962>. doi:10.24406/publica-fhg-282962.

- [11] D. Borys, J. Gajewski, T. Becher, Y. Censor, R. Kopeć, M. Rydygier, A. Schiavi, T. Skóra, A. Spaleniak, N. Wahl, A. Wochnik, A. Ruciński, GPU-accelerated FREDopt package for simultaneous dose and LETd proton radiotherapy plan optimization via superiorization methods, *Physics in Medicine & Biology* 70 (2025) 155011. URL: <https://doi.org/10.1088/1361-6560/ade841>. doi:10.1088/1361-6560/ade841.
- [12] Y. Nie, M. Wang, Y. Wang, J. Lin, B. Liu, T. Yin, Z. Liu, S. Zhang, Superiorized model-based real-time inversion for cross-sectional magnetoacoustic tomography combined with magnetic induction, *Physics in Medicine & Biology* 71 (2026) 015040. URL: <https://doi.org/10.1088/1361-6560/ae2cdd>. doi:10.1088/1361-6560/ae2cdd.
- [13] S. Yu, Y. Censor, G. Luo, Floorplanning With I/O Assignment via Feasibility-Seeking and Superiorization Methods, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 44 (2025) 317–330. URL: <https://ieeexplore.ieee.org/abstract/document/10543062>. doi:10.1109/TCAD.2024.3408106.
- [14] A. Cegielski, Iterative Methods for Fixed Point Problems in Hilbert Spaces, volume 2057 of *Lecture Notes in Mathematics*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2013. URL: <http://link.springer.com/10.1007/978-3-642-30901-4>. doi:10.1007/978-3-642-30901-4.
- [15] Y. Censor, An automatic relaxation method for solving interval linear inequalities, *Journal of Mathematical Analysis and Applications* 106 (1985) 19–25. URL: <https://www.sciencedirect.com/science/article/pii/0022247X85901271>. doi:10.1016/0022-247X(85)90127-1.
- [16] G. T. Herman, W. Chen, A fast algorithm for solving a linear feasibility problem with application to intensity-modulated radiation therapy, *Linear Algebra and its Applications* 428 (2008) 1207–1217. URL: <https://linkinghub.elsevier.com/retrieve/pii/S0024379506005106>. doi:10.1016/j.laa.2006.11.009.
- [17] Y. Censor, T. Elfving, G. T. Herman, T. Nikazad, On Diagonally Relaxed Orthogonal Projection Methods, *SIAM Journal on Scientific Computing* 30 (2008) 473–504. URL: <http://epubs.siam.org/doi/10.1137/050639399>. doi:10.1137/050639399.

- [18] T. Nikazad, M. Abbasi, T. Elfving, Error minimizing relaxation strategies in Landweber and Kaczmarz type iterations, *Journal of Inverse and Ill-posed Problems* 25 (2017) 35–56. URL: <https://www.degruyterbrill.com/document/doi/10.1515/jiip-2015-0082/html>. doi:10.1515/jiip-2015-0082.
- [19] S. Kaczmarz, Przybliżone rozwiązywanie układów równan liniowych. – Angenäherte Auflösung von Systemen linearer Gleichungen., *Bulletin International de l’Académie Polonaise des Sciences et des Lettres. Classe des Sciences Mathématiques et Naturelles.* (1937) 355–357.
- [20] S. Agmon, The Relaxation Method for Linear Inequalities, *Canadian Journal of Mathematics* 6 (1954) 382–392. URL: <https://www.cambridge.org/core/journals/canadian-journal-of-mathematics/article/relaxation-method-for-linear-inequalities/7C0393A9DDB5EBB9FA5F1E78C591D820>. doi:10.4153/CJM-1954-037-2.
- [21] T. S. Motzkin, I. J. Schoenberg, The Relaxation Method for Linear Inequalities, *Canadian Journal of Mathematics* 6 (1954) 393–404. URL: <https://www.cambridge.org/core/journals/canadian-journal-of-mathematics/article/relaxation-method-for-linear-inequalities/8EDDF8B5F08B146AC71A5CF4F304A560>. doi:10.4153/CJM-1954-038-x.
- [22] H. H. Bauschke, J. M. Borwein, On Projection Algorithms for Solving Convex Feasibility Problems, *SIAM Review* 38 (1996) 367–426. URL: <https://epubs.siam.org/doi/10.1137/S0036144593251710>. doi:10.1137/S0036144593251710.
- [23] D. Butnariu, R. Davidi, G. T. Herman, I. G. Kazantsev, Stable Convergence Behavior Under Summable Perturbations of a Class of Projection Methods for Convex Feasibility and Optimization Problems, *IEEE Journal of Selected Topics in Signal Processing* 1 (2007) 540–547. URL: <https://ieeexplore.ieee.org/abstract/document/4407758>. doi:10.1109/JSTSP.2007.910263.
- [24] G. T. Herman, R. Davidi, Image reconstruction from a small number of projections, *Inverse Problems* 24 (2008) 045011. URL: <https://iopscience.iop.org/article/10.1088/0266-5611/24/4/045011>. doi:10.1088/0266-5611/24/4/045011.
- [25] R. Davidi, G. Herman, Y. Censor, Perturbation-resilient block-iterative projection methods with application to image reconstruction from projections,

- International Transactions in Operational Research 16 (2009) 505–524. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2009.00695.x>. doi:10.1111/j.1475-3995.2009.00695.x, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2009.00695.x>.
- [26] A. Cegielski, On linear convergence of Landweber-type methods, *Numerical Algorithms* (2025). URL: <https://link.springer.com/10.1007/s11075-025-02163-0>. doi:10.1007/s11075-025-02163-0.
- [27] M. V. W. Zibetti, C. Lin, G. T. Herman, Total Variation Superiorized Conjugate Gradient Method for Image Reconstruction, *Inverse Problems* 34 (2018) 034001. URL: <http://arxiv.org/abs/1709.04912>. doi:10.1088/1361-6420/aaa49b, arXiv:1709.04912 [math].
- [28] Y. Censor, T. Elfving, N. Kopf, T. Bortfeld, The multiple-sets split feasibility problem and its applications for inverse problems, *Inverse Problems* 21 (2005) 2071–2084. URL: <https://iopscience.iop.org/article/10.1088/0266-5611/21/6/017>. doi:10.1088/0266-5611/21/6/017.
- [29] F. J. Aragón-Artacho, Y. Censor, A. Gibali, D. Torregrosa-Belén, The superiorization method with restarted perturbations for split minimization problems with an application to radiotherapy treatment planning, *Applied Mathematics and Computation* 440 (2023) 127627. URL: <https://www.sciencedirect.com/science/article/pii/S0096300322007007>. doi:10.1016/j.amc.2022.127627.
- [30] C. L. Byrne, A. Moudafi, Extensions of the CQ Algorithm for the split feasibility and split equality problems (10th draft) (2012). URL: <https://hal.univ-antilles.fr/hal-00776640/document>.
- [31] C. Byrne, Bounds on the largest singular value of a matrix and the convergence of simultaneous and block-iterative algorithms for sparse linear systems, *International Transactions in Operational Research* 16 (2009) 465–479. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2009.00692.x>. doi:10.1111/j.1475-3995.2009.00692.x, _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1475-3995.2009.00692.x>.
- [32] Y. Censor, E. Garduño, E. S. Helou, G. T. Herman, Derivative-free superiorization: principle and algorithm, *Numerical Algorithms* 88 (2021) 227–

248. URL: <https://doi.org/10.1007/s11075-020-01038-w>. doi:10.1007/s11075-020-01038-w.
- [33] E. Bonacker, A. Gibali, K.-H. Küfer, Accelerating Two Projection Methods via Perturbations with Application to Intensity-Modulated Radiation Therapy, *Applied Mathematics & Optimization* 83 (2021) 881–914. URL: <http://link.springer.com/10.1007/s00245-019-09571-4>. doi:10.1007/s00245-019-09571-4.
- [34] T. Becher, DKFZ-OpenMedPhys/SupPy: v0.4.0 (2026). URL: <https://zenodo.org/records/20085222>. doi:10.5281/zenodo.20085222.
- [35] Y. Censor, H. Heaton, R. Schulte, Derivative-free superiorization with component-wise perturbations, *Numerical Algorithms* 80 (2019) 1219–1240. URL: <http://link.springer.com/10.1007/s11075-018-0524-0>. doi:10.1007/s11075-018-0524-0.
- [36] I. Loris, H. Douma, G. Nolet, I. Daubechies, C. Regone, Nonlinear regularization techniques for seismic tomography, *Journal of Computational Physics* 229 (2010) 890–905. URL: <https://www.sciencedirect.com/science/article/pii/S0021999109005725>. doi:10.1016/j.jcp.2009.10.020.
- [37] J. Charléty, S. Voronin, G. Nolet, I. Loris, F. J. Simons, K. Sigloch, I. C. Daubechies, Global seismic tomography with sparsity constraints: Comparison with smoothing and damping regularization, *Journal of Geophysical Research: Solid Earth* 118 (2013) 4887–4899. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/jgrb.50326>. doi:10.1002/jgrb.50326, _eprint: <https://agupubs.onlinelibrary.wiley.com/doi/pdf/10.1002/jgrb.50326>.
- [38] P. C. Hansen, J. S. Jørgensen, AIR Tools II: algebraic iterative reconstruction methods, improved implementation, *Numerical Algorithms* 79 (2018) 107–137. URL: <https://doi.org/10.1007/s11075-017-0430-x>. doi:10.1007/s11075-017-0430-x.
- [39] P. Combettes, J. Luo, An adaptive level set method for nondifferentiable constrained image recovery, *IEEE Transactions on Image Processing* 11 (2002) 1295–1304. URL: <https://ieeexplore.ieee.org/document/1097765/>. doi:10.1109/TIP.2002.804527.
- [40] J. Hsieh, B. Nett, Z. Yu, K. Sauer, J.-B. Thibault, C. A. Bouman, Recent Advances in CT Image Reconstruction, *Current Radiology Reports* 1 (2013)

- 39–51. URL: <https://doi.org/10.1007/s40134-012-0003-7>. doi:10.1007/s40134-012-0003-7.
- [41] T. P. Szczykutowicz, G. V. Toia, A. Dhanantwari, B. Nett, A Review of Deep Learning CT Reconstruction: Concepts, Limitations, and Promise in Clinical Practice, *Current Radiology Reports* 10 (2022) 101–115. URL: <https://doi.org/10.1007/s40134-022-00399-5>. doi:10.1007/s40134-022-00399-5.
- [42] J. Leuschner, M. Schmidt, D. O. Bager, P. Maass, LoDoPaB-CT, a benchmark dataset for low-dose computed tomography reconstruction, *Scientific Data* 8 (2021) 109. URL: <https://www.nature.com/articles/s41597-021-00893-z>. doi:10.1038/s41597-021-00893-z.
- [43] W. van Aarle, W. J. Palenstijn, J. De Beenhouwer, T. Altantzis, S. Bals, K. J. Batenburg, J. Sijbers, The ASTRA Toolbox: A platform for advanced algorithm development in electron tomography, *Ultramicroscopy* 157 (2015) 35–47. URL: <https://www.sciencedirect.com/science/article/pii/S0304399115001060>. doi:10.1016/j.ultramic.2015.05.002.
- [44] W. v. Aarle, W. J. Palenstijn, J. Cant, E. Janssens, F. Bleichrodt, A. Dabralovski, J. D. Beenhouwer, K. J. Batenburg, J. Sijbers, Fast and flexible X-ray tomography using the ASTRA toolbox, *Optics Express* 24 (2016) 25129–25147. URL: <https://opg.optica.org/oe/abstract.cfm?uri=oe-24-22-25129>. doi:10.1364/OE.24.025129.
- [45] G. T. Herman, *Fundamentals of Computerized Tomography*, *Advances in Pattern Recognition*, Springer, London, 2009. URL: <http://link.springer.com/10.1007/978-1-84628-723-7>. doi:10.1007/978-1-84628-723-7.
- [46] Y. Lievens, J. M. Borrás, C. Grau, Provision and use of radiotherapy in Europe, *Molecular Oncology* 14 (2020) 1461–1469. doi:10.1002/1878-0261.12690.
- [47] D. Craft, M. Bangert, T. Long, D. Papp, J. Unkelbach, Shared data for intensity modulated radiation therapy (IMRT) optimization research: the CORT dataset, *GigaScience* 3 (2014) 2047–217X–3–37. URL: <https://doi.org/10.1186/2047-217X-3-37>. doi:10.1186/2047-217X-3-37.
- [48] B. Ackermann, M. Bangert, A. B. A. Bennan, L. Burigo, G. Cabal, E. Cisternas, L. Charton, E. Christiansen, E. Doerner, S. Ecker, M. Ellerbrock, H. Gabryś, J. Handrack, E. Heath, C. Hermann, N. Homolka, O. Jäkel,

- T. Klinge, A. Mairani, H. Mescher, L.-R. Müller, A. Neishabouri, M. Palkowitsch, K. Parodi, G. Pezzano, D. Ramirez, A. Stadler, S. Ulrich, N. Wahl, J. Welsch, H.-P. Wieser, J. Winter, T. Xu, *matRad*, 2020. URL: <https://zenodo.org/record/3879616>. doi:10.5281/zenodo.3879616.
- [49] H.-P. Wieser, E. Cisternas, N. Wahl, S. Ulrich, A. Stadler, H. Mescher, L.-R. Müller, T. Klinge, H. Gabrys, L. Burigo, A. Mairani, S. Ecker, B. Ackermann, M. Ellerbrock, K. Parodi, O. Jäkel, M. Bangert, Development of the open-source dose calculation and optimization toolkit *matRad*, *Medical Physics* 44 (2017) 2556–2568. URL: <https://aapm.onlinelibrary.wiley.com/doi/abs/10.1002/mp.12251>. doi:10.1002/mp.12251, tex.ids= Wieser2017b ISBN: 4955139574.
- [50] A. Wächter, L. T. Biegler, On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming, *Mathematical Programming* 106 (2006) 25–57. URL: <https://doi.org/10.1007/s10107-004-0559-y>. doi:10.1007/s10107-004-0559-y.
- [51] S. Penfold, R. Zalas, M. Casiraghi, M. Brooke, Y. Censor, R. Schulte, Sparsity constrained split feasibility for dose-volume constraints in inverse planning of intensity-modulated photon or proton therapy, *Physics in Medicine & Biology* 62 (2017) 3599. URL: <https://dx.doi.org/10.1088/1361-6560/aa602b>. doi:10.1088/1361-6560/aa602b.