Marshall Plan Scholarship Report

Incorporation of the Superiorization Methodology into Biomedical Imaging Software

submitted by:

Oliver Langthaler, BSc

AUSTRIAN MARSHALL PLAN FOUNDATION VIENNA | AUSTRIA

Fachhochschule Salzburg University of Applied Sciences

The City University of New York

Supervisor FHS:

FH-Prof. Univ.-Doz. Mag. Dr. Stefan Wegenkittl

Supervisor CUNY:

Distinguished Professor Gabor T. Herman, Ph.D.

Salzburg, September 2014

Acknowledgement

I would like to express my sincere gratitude to my supervisor at the Graduate Center of the City University of New York, Distinguished Professor Gabor T. Herman, Ph.D., for providing his expertise and invaluable advice, as well as a pleasant working environment. Furthermore, I would like to thank my supervisor at the Salzburg University of Applied Sciences, FH-Prof. Univ.-Doz. Mag. Dr. Stefan Wegenkittl, for providing his expertise and invaluable advice. Lastly, I would like to thank the Austrian Marshall Plan Foundation and its members for their kind cooperation and generous funding, without which it would not have been possible for me to conduct this research in New York City.

Abstract

Imaging software often relies on iterative algorithms for the reconstruction of images from projections, such as ART[1], SART[2] or MLEM[3]. However, many of these algorithms only optimize a single criterion and some of them exhibit certain drawbacks, such as MLEM, which is prone to noise overfitting at higher iteration numbers. In order to improve results, it is thus often advantageous to introduce an additional optimization criterion, such as the extent to which certain properties of a reconstruction match expected values or how well it matches data from an additional data source. Developing such a modified algorithm can be a mathematically challenging task which may take considerable time to develop.

Superiorization is a newly developed heuristic for constrained optimization problems and provides a generic answer to these challenges. While, unlike their exact counterparts, heuristic approaches may not always produce an output which tends toward the optimum of a given criterion, they provide a nearly immediate solution and are nonetheless capable of producing feasible results. Additionally, they tend to require considerably fewer computational resources, which is why they have often been found useful in optimization applications.

The aim of the underlying work is the incorporation of Superiorization into SNARK09, resulting in a new version, SNARK14. SNARK is a programming system for the reconstruction of images from projections (such as CT and PET scans) and is intended to help researchers interested in developing and evaluating reconstruction algorithms. Upon a thorough evaluation of its code structure, a new module, named Superiorization, is introduced into the SNARK package. The main focus is to design this module to be as flexible as possible for various iterative algorithms and optimization criteria.

[1]Algebraic Reconstruction Technique
[2]Simultaneous Algebraic Reconstruction Technique
[3]Maximum Likelihood Expectation Maximization

Contents

Nomenclature iii

1. Introduction 1

2. Superiorization 3

2.1. The Idea of Superiorization 3
2.2. Constrained Optimization 4
2.3. Constrained Optimization vs. Superiorization 5
2.4. Problem Sets, Proximity Function and ε-compatibility 5
2.5. Algorithms and Output 6
2.6. Bounded Perturbation Resilience 7
2.7. Secondary Optimization Criteria 8
2.8. Non-ascending Vectors 9
2.9. Superiorized Version of an Algorithm 9
2.10. Non-ascending Vector Length Variation 11

3. Target Software / Environment 15

3.1. SNARK 15
3.1.1. Purpose and History of SNARK 16
3.1.2. The SNARK Framework 16
3.1.3. Information Flow 16
3.1.4. Commands 17
3.1.5. Input and Output 18
3.1.5.1. snarkInput 18
3.1.5.2. snarkDisplay 19
3.1.6. Reconstruction Algorithms 20
3.1.7. User-Defined Extensions 20
3.1.8. The SNARK Experimenter 21
3.2. Development Environment 21
3.2.1. Version Control / Collaboration 22
3.2.2. Integrated Development Environment 22
3.2.3. CentOS VirtualBox Virtual Machine 23

4. Implementation 24

4.1. Variable and Non-Variable Superiorization Constituents 24

4.2. Adaptations for Implementation 25
4.3. Ideal Approach 27
4.4. Implementation in SNARK 30
4.4.1. Command SUPERIORIZE 30
4.4.2. Secondary Optimization Criteria 31
4.4.2.1. Total Variation (Option TVAR) 31
4.4.2.2. Smoothness (Option SMOO) 32
4.4.2.3. User-Defined Criteria (Options SCR3, SCR4, SCR5) 32
4.4.3. Non-Ascending Vectors 33
4.4.3.1. Gradient for Total Variation 33
4.4.3.2. Gradient for Smoothness 34
4.4.4. Event Space for Superiorization 35
4.4.5. Alternative Handling of the Variable l 35
4.4.6. New Termination Criteria 36
4.4.6.1. Kullback-Leibler Distance (Option KLDS) 37
4.4.6.2. Residual (Option RESI) 37
4.4.6.3. Weighted Squared Distance (Option WSQD) 38
4.4.6.4. MLEM-STOP (Option MLST) 38
4.4.7. New Figures of Merit 38
4.4.7.1. Kullback-Leibler Distance (Option KLDS) 39
4.4.7.2. Weighted Squared Distance (Option WSQD) 39
4.4.8. The Class Superior 40
4.4.9. Verbosity Options 41

5. Verification and Application 42
5.1. Recreating Previous Results 42
5.2. Superiorization of ART and SART 45

6. Conclusion and Outlook 49

Bibliography vi

A. Appendix vii
A.1. SUPERIORIZE Command in EBNF vii
A.2. STOP Command in EBNF vii
A.3. SNARK14 Superiorization Input File viii
A.4. SNARK14 Superiorization Output File x
A.5. SNARK14 Superiorization Reporting Output File xxiii
A.6. SNARK14 KLDS Reporting Output File xxiii
A.7. SNARK14 MLEM-STOP Reporting Output File xxiv

Nomenclature

ART	Algebraic Reconstruction Technique
CentOS	Community Enterprise Operating System
CT	(X-ray) Computed Tomography
CVS	Concurrent Versions System
DIG	Discrete Imaging and Graphics Group
EBNF	Extended Backus-Naur Form
FOM	Figure of Merit
FORTRAN	derived from Formula Translating System
FOSS	Free and Open-Source Software
GUI	Graphical User Interface
IDE	Integrated Development Environment
IROI	Image-wise Region of Interest
KL	Kullback-Leibler
LOR	line of response
MLEM	Maximum Likelihood Expectation Maximization
PET	Positron Emission Tomography
RHEL	Red Hat Enterprise Linux
SART	Simultaneous Algebraic Reconstruction Technique
SIRT	Simultaneous Iterative Reconstruction Technique
SVN	(Apache) Subversion
tarball	files distributed as a tar archive
TV	Total Variation
VM	Virtual Machine

Chapter 1

1 Introduction

Superiorization is the term for a newly developed and innovative heuristic for constrained optimization problems. As numerous other discoveries throughout history, it was made by chance, when then-postgraduate Ran Davidi worked on the recreation of an algorithm described in [7]. His results were encouraging, however, when he discussed his implementation with Combettes and Luo, its original authors, the discovery was made that he had in fact implemented a variant of the algorithm which introduced a higher amount of perturbations than intended by the authors. A corresponding adjustment was made but the output of the corrected algorithm turned out to produce inferior results when compared to the implementation which introduced the higher amount of perturbations.

Subsequently, research was conducted to determine the reasons for this unexpected behavior. It was discovered that the cause for the superior output was the fact that directed and advantageous bounded perturbations had been introduced into an iterative algorithm which was resilient to these perturbations. These findings culminated in the publication of [4], in which bounded perturbations were used to steer an ART-based image reconstruction process towards producing a more desirable output. In the years following, research on the topic continued and it became obvious that the basic scheme was applicable in a much broader sense, coining the term Superiorization to describe the basic methodology and resulting in several further publications such as [5, 6, 10, 12, 15], which aimed at generalizing and refining the process, at exploring the limits of the mathematical conditions which are required for successful application as well as at comparing Superiorization to similar methods.

Even though considerable research has been done on the subject, the methodology is still in a relatively early stage of dissemination and as of early 2014, no software package was available yet which allowed Superiorization to be applied in an automated fashion to a series of algorithms. The availability of such software would however serve to significantly facilitate the process of finding applications for which Superiorization can provide benefits, which is why the incorporation of the methodology into SNARK09 has been chosen as a topic. Furthermore, as Superiorization has only recently been discovered, its mathematical exploration is an ongoing process and new discoveries regarding its prerequisites, or variations of the methodology, may still be made.

Chapter 2 provides insight into the mathematical aspects of the Superiorization methodology, while Chapter 3 deals with the target software and the environment which was established in order to ensure efficient development. Chapter 4 contains considerations regarding an optimal solution which were made prior to the implementation as well as detailed information about the implementation process itself. In Chapter 5, the correctness of the resulting updated SNARK package is verified by recreating previous results published in [12] and by conducting a comparison of superiorized and unsuperiorized versions of ART and SART.

Chapter 2

2 Superiorization

The Superiorization methodology represents a heuristic and generic approach towards constrained optimization problems and is applicable for perturbation resilient iterative algorithms. Due to its generic nature, it may in principle be applied to any iterative algorithm which meets the requirement of being resilient to bounded perturbations. It is capable of automatically converting the selected base algorithm into a superiorized version, providing improved results regarding a secondary optimization criterion which may be freely defined while retaining the original constraint compatibility. This chapter is intended to cover the relevant terminology and to provide information about the nature, scope and purpose of Superiorization. With the exception of Section 2.10, its contents are derived from [15].

2.1. The Idea of Superiorization

The main concept behind Superiorization is to utilize the fact that certain iterative algorithms are resilient to bounded perturbations. Directed disturbances are introduced between each algorithmic step in order to steer the optimization process towards a solution which is superior regarding a secondary optimization criterion. The original constraint compatibility is maintained by ensuring that the last step in each iteration is the application of the original algorithm. In [10], an experiment has also been conducted using a dual perturbation scheme.

The idea of designing algorithms that use interlacing steps of two different kinds (in this case, one kind of steps aim at constraints-compatibility and the other kind of steps aim at improvement of the optimization criterion) is well-established and, in fact, made use of in many approaches that have been proposed with exact constrained optimization in mind. See, for example, the works of Helou Neto and De Pierro [20, 21], of Nurminski [22], of Combettes and coworkers [7, 8], of Sidky and Pan and coworkers [28, 27, 2] and of Defrise and coworkers [11]. However, none of these approaches are capable of doing what can be done by the innovative approach of Superiorization, namely the automatic production of a heuristic constrained optimization algorithm from an iterative algorithm for constraints-compatibility. For example, in [20] it is assumed (just as in the theory presented in [5]) that all the constraints can be satisfied simultaneously.

2.2. Constrained Optimization

Constrained optimization is used by numerous applications and describes the process of optimizing an objective function with respect to some of its variables while adhering to certain limitations regarding the values of one or more of those variables. Depending on the specific application, the function may either need to be minimized or maximized and the constraints may either exist in the form of a penalty function whose value should be minimal or as fixed numerical bounds.

Specifically, Superiorization deals with minimization problems of the form

$$\begin{cases} \text{minimize } \phi(x) \\ \text{subject to } x \in \Omega \end{cases} \tag{2.1}$$

where $\phi(x)$ is the objective function and $\Omega \subseteq \mathbb{R}^n$ is a non-empty, convex and closed constraint set.

In the case of image reconstruction from projections obtained through CT, the constraints are derived from the measurements provided by the CT scanner. They may further be derived from known physical properties of the object of interest, such as piece-wise constant attenuation in machined parts and tissues or from any other additional information such as known outer dimensions or data obtained through a secondary method of measurement. The latter has, for instance, been the case in [25].

Due to factors such as noisy data, the exact nature of the constraints may be uncertain, resulting in a large number of solutions which may be considered acceptable regarding constraints-compatibility. Hence, an optimization criterion in the form of a proximity function is introduced to provide a means of evaluating the quality of a reconstruction. This proximity function may, for example, favor images that are piecewise homogeneous.

2.3. Constrained Optimization vs. Superiorization

The approach taken in Superiorization significantly differs from that of classical constrained optimization. Both in Superiorization and in classical constrained optimization, the existence of a domain Ω and an optimization criterion that is specified by a function ϕ that maps Ω into $\mathbb{R}$ is assumed. In classical optimization it is further assumed that there is a constraints set C and the task is to find an $x \in C$ for which $\phi(x)$ is minimal.

There are, however, some problems with this approach: Firstly, the constraints may not be consistent, resulting in an empty C. In this case, the optimization task as stated would not have a solution. Secondly, iterative methods of classical constrained optimization typically converge to a solution only in the limit. In practice, some stopping rule needs to be applied to terminate the process. The actual output at that time may, however, not be in C and, even if it is in C, it is not likely to be a minimizer of ϕ over C.

Both problems are handled in the Superiorization approach by replacing the constraints set C with a nonnegative real-valued function Pr that serves as an indicator of how incompatible a given $x \in \Omega$ is with the constraints. Subsequently, the merit of an actual output of an algorithm is given by the size of the two numbers $Pr(x)$ and $\phi(x)$. If an iterative algorithm produces an output x, then its superiorized version will produce an output x' for which $Pr(x')$ is not larger than $Pr(x)$, but $\phi(x')$ is, typically, much smaller than $\phi(x)$[15].

2.4. Problem Sets, Proximity Function and ε-compatibility

In medical physics, optimization is usually performed in Euclidian space $\mathbb{R}^J$. In practice, the solution space is further restricted to be an nonempty subset Ω of $\mathbb{R}^J$. In the field of image reconstruction, the result of an optimization is the image vector x. As images usually only contain positive values, Ω may be further restricted to $\mathbb{R}^J_+$.

Definition 1. The **problem set** $\mathbb{T}$ is the set of all problems $T \in \mathbb{T}$ where each T is the description of the constraints of one particular problem.

In CT, each T represents the problem of reconstructing the picture that is associated with the measurements obtained frome one particular scan. $\mathbb{T}$ represents the set of all possible measurements.

Definition 2. The **proximity function** $\mathcal{P}r$ on the problem set $\mathbb{T}$ is defined as function from the solution space into the positive real numbers $\mathcal{P}r_T(x) : \Omega \mapsto \mathbb{R}_+$. It indicates how incompatible the solution x is with the given constraints of $T \in \mathbb{T}$. If $\mathcal{P}r_T(x) = 0$, x is said to be perfectly constraint compatible with the problem T.

When reconstructing CT images, $\mathcal{P}r_T(x)$ could be the norm-distance $\|y - Rx\|$ of the reconstruction problem $y = Rx + e$, where y is the measurement vector of the CT scan, R is the projection matrix, x is the image vector and e is the error. Other candidates for the proximity function are the Kullback-Leibler distance which is described in Chapter 4.4.6.1 or the weighted squared distance which is described in Chapter 4.4.6.3.

Definition 3. A **problem structure** $\langle \mathbb{T}, \mathcal{P}r \rangle$ is defined as the combination of a nonempty problem set $\mathbb{T}$ and a proximity function $\mathcal{P}r$.

Definition 4. The solution $x \in \Omega$ for a problem $T \in \mathbb{T}$ is **ε-compatible** regarding a problem structure $\langle \mathbb{T}, \mathcal{P}r \rangle$ when $\mathcal{P}r_T(x) < \varepsilon$, where ε is a non-negative number.

The ε-compatibility is needed, as in practical applications all measurements are noisy. It is therefore unlikely that a perfect reconstruction x for the problem $T \in \mathbb{T}$ exists. ε is the threshold for the proximity function $\mathcal{P}r$ below which a reconstructed image x is acceptable ($\mathcal{P}r_T(x) < \varepsilon$). The proximity function can thus be used as a stopping criterion of an iterative algorithm.

2.5. Algorithms and Output

The concept of algorithms is defined in the general context of problem structures. For technical reasons which are covered in Section 2.8, an additional set Δ is introduced such that $\Omega \subseteq \Delta \subseteq \mathbb{R}^J$. Both Ω and Δ are assumed to be known and fixed for any particular problem structure $\langle \mathbb{T}, \mathcal{P}r \rangle$.

Definition 5. An **algorithm** P assigns to each problem $T \in \mathbb{T}$ an operator $P_T : \Delta \mapsto \Omega$.

Definition 6. An **iterative algorithm** produces an infinite sequence

$$\left((P_T)^k (x) \right)_{k=0}^{\infty} = x,\ P_T(x),\ P_T(P_T(x)), \ldots \tag{2.2}$$

where x is any initial point $x \in \Omega$ and P_T is an algorithm as described in Definition 5.

The two reconstruction algorithms ART and SART, which are relevant to the comparison of superiorized and unsuperiorized algorithms in Chapter 5.2, fall within the classification of iterative processes as defined above. A single iteration corresponds to P and its repeated execution defines the iterative process.

Definition 7. For a problem structure $\langle \mathbb{T}, \mathcal{P}r \rangle$, a $T \in \mathbb{T}$, an $\varepsilon \in \mathbb{R}_+$ and a sequence $RS = \left(x^k\right)_{k=0}^{\infty}$, the **output** $O\left(T, \varepsilon, RS\right)$ is used to denote the $x \in \Omega$ that has the following properties:

i. $\mathcal{P}r_T\left(x\right) \leq \varepsilon$

ii. there is a non-negative integer K such that $x = x^K$, $x^K \in RS$ and

iii. for all non-negative integers $k < K$, $\mathcal{P}r_T\left(x\right) > \varepsilon$

If there exists such an x, then $O\left(T, \varepsilon, RS\right)$ is defined, otherwise it is undefined.

If RS is a sequence of points produced by an algorithm that solves the problem T without termination criterion, then $O\left(T, \varepsilon, RS\right)$ is the output produced by that algorithm when the termination criterion $\mathcal{P}r_T\left(x\right) \leq \varepsilon$ is added to it. The point x denotes the first ε-compatible point within the sequence RS, which was reached after K iterations [15].

2.6. Bounded Perturbation Resilience

An iterative algorithm P is resilient to bounded perturbations for a problem structure $\langle \mathbb{T}, \mathcal{P}r \rangle$ if, irrespective of the starting point, the sequence $\left(\left(P_T\right)^k\right)_{k=0}^{\infty}$ generated by it still converges even if the result of every step is perturbed. For real applications, bounded perturbation resilience is not sufficient as it can only be used for problems $T \in \mathbb{T}$ for which a perfectly constraint compatible solution x where $\mathcal{P}r_T\left(x\right) = 0$ exists. Since the data of most practical applications contains noise, this is usually not the case. Therefore, the concept of bounded perturbation resilience needs to be extended to strong perturbation resilience.

Definition 8. An algorithm P for a problem structure $\langle \mathbb{T}, \mathcal{P}r \rangle$ is considered **strongly perturbation resilient** if, for all $T \in \mathbb{T}$,

i. there exists an $\varepsilon \in \mathbb{R}_+$ such that $O\left(T, \varepsilon, \left(\left(P_T\right)^k x\right)\right)_{k=0}^{\infty}$ is defined for every $x \in \Omega$ and

ii. for all $\varepsilon \in \mathbb{R}_+$, such that $O\left(T, \varepsilon, \left((P_T)^k x\right)\right)_{k=0}^{\infty}$ is defined for every $x \in \Omega$, we also have that $O(T, \varepsilon', R)$ is defined for every $\varepsilon' > \varepsilon$ and for every sequence $RS = \left(x^k\right)_{k=0}^{\infty}$ of points in Ω generated by

$$x^{k+1} = P_T\left(x^k + \beta_k v^k\right) \text{ for all } k \geq 0 \tag{2.3}$$

where $\beta_k v^k$ are *bounded perturbations*, meaning that the sequence $(\beta_k)_{k=0}^{\infty}$ of non-negative real numbers is summable (that is, $\sum_{k=0}^{\infty} \beta_k < \infty$), the sequence $\left(v^k\right)_{k=0}^{\infty}$ of vectors in $\mathbb{R}^J$ is bounded and, for all $k > 0$, $x^k + \beta_k v^k \in \Delta$.

The properties of this definition state that all perturbed sequences contain an ε'-compatible point if for every problem T and any non-negative number ε, there exists an ε-compatible solution for every initial point $x \in \Omega$ (and $\varepsilon' > \varepsilon$). This means that the perturbed version of the algorithm produces an ε'-compatible output $O(T, \varepsilon', RS)$. [5, 15]

2.7. Secondary Optimization Criteria

In addition to the constrained optimization previously discussed, it is in some cases advantageous to introduce an additional optimization criterion, denoted as ϕ, towards which to improve the reconstruction output. This criterion may be based on properties which are expected to produce superior results, or on additional information obtained through a secondary measurement device.

Definition 9. A **secondary optimization criterion** ϕ is a function $\phi : \Delta \mapsto \mathbb{R}$ which indicates how good a point $x \in \Delta$ fulfills the desired properties. A point $x_1 \in \Delta$ is considered superior to another point $x_2 \in \Delta$ if $\phi(x_1) < \phi(x_2)$.

Two examples of secondary optimization criteria are given in Chapter 4.4.2.

The main idea behind Superiorization is to utilize the perturbations in (2.3) to steer a strongly perturbation resilient algorithm which produces constraints-compatible solutions towards generating output which exhibits a lower value regarding the secondary optimization criterion, essentially converting it into a new, superiorized algorithm. This new algorithm is subsequently not only as constraints-compatible as its original version, but it is also superior regarding the secondary optimization criterion. This is achieved by adding the bounded perturbations $\beta_k v^k$ to the solution vector x^k between each iteration.

2.8. Non-ascending Vectors

In order to achieve the desired behavior of $\phi\left(x+\beta_k v^k\right) \leq \phi(x)$, v^k must be a non-ascending vector.

Definition 10. A vector $d \in \mathbb{R}^J$ for a given function $\phi : \Delta \mapsto \mathbb{R}$ and a point $x \in \Delta$ is called a **non-ascending vector** for ϕ at x if $||d|| \leq 1$ and there exists a $\delta > 0$ such that, for all $\lambda \in [0, \delta]$, $(x + \lambda d) \in \Delta$ and $\phi(x + \lambda d) \leq \phi(x)$.

As $(x + \lambda d) \in \Delta$ may be outside of Ω, it is important that the algorithm P is defined as $P : \Delta \mapsto \Omega$. This ensures that the final output of the superiorized version of the algorithm is still within the defined solution space Ω.

Note that irrespective of the secondary optimization criterion ϕ and the point x, the zero-vector is always a non-ascending vector. While this circumstance is useful for proving the convergence of the algorithm, in practice, the vector d should have the property $\phi(x + \lambda d) < \phi(x)$ rather than $\phi(x + \lambda d) \leq \phi(x)$ in order to result in an improvement of the secondary optimization criterion [15].

2.9. Superiorized Version of an Algorithm

This section illustrates how an iterative algorithm P can automatically be converted into its superiorized version.

Theorem 11. *Let Ω and Δ be the underlying sets for a problem structure $\langle \mathbb{T}, \mathcal{P}r \rangle$, where $\Omega \subseteq \Delta \subseteq \mathbb{R}^J$, $P : \Delta \mapsto \Omega$ is an iterative, strongly perturbation resilient algorithm for $\langle \mathbb{T}, \mathcal{P}r \rangle$ and $\phi : \Omega \mapsto \mathbb{R}$. Algorithm 2.1 represents the superiorized version of the algorithm P. It produces for any problem $T \in \mathbb{T}$ and any point $x \in \Omega$ a sequence $RS = \left(x^k\right)_{k=0}^{\infty}$. The resulting sequence RS of the superiorized algorithm is ε-compatible and expected to be superior to the original algorithm P with regard to the secondary optimization criterion ϕ.*

The proof of this theorem can be found in [15].

The superiorized algorithm depicted in Listing 2.1 depends on a specified initial point $\overline{x} \in \Omega$, a positive integer N and requires a summable sequence $(\gamma_l)_{l=0}^{\infty}$ of positive real numbers. One example of such a sequence is $\gamma_l = a^l$ where $0 < a < 1$.

In lines 1-3, the required variables are initialized. k represents the number of the current iteration, l is the integer sequence used to generate values for $(\gamma_l)_{l=0}^{\infty}$, and x^0 is the initial

Algorithm 2.1 Superiorized Version of Algorithm P [15].

```
1: set $k = 0$
2: set $x^k = \overline{x}$
3: set $l = -1$
4: while $true$ do
5:    set $n = 0$
6:    set $x^{k,n} = x^k$
7:    while $n < N$ do
8:       set $v^{k,n}$ to be a non-ascending vector for $\phi$ at $x^{k,n}$
9:       set $loop = true$
10:      while loop do
11:         set $l = l + 1$
12:         set $\beta_{k,n} = \gamma_l$
13:         set $z = x^{k,n} + \beta_{k,n} v^{k,n}$
14:         if $z \in \Delta$ and $\phi(z) \leq \phi(x^k)$ then
15:            set $n = n + 1$
16:            set $x^{k,n} = z$
17:            set $loop = false$
18:         end if
19:      end while
20:   end while
21:   set $x^{k+1} = P_T x^{k,N}$
22:   set $k = k + 1$
23: end while
```

value of the output vector. Every execution of the loop starting in line 4 corresponds to one iteration of the superiorized algorithm. During each iterative step, there are N executions of the inner loop (lines 7-20) where Superiorization is performed by calculating and applying a non-ascending vector after which the original algorithm P_T is applied once.

n represents the counter variable for the inner Superiorization loop (lines 10-19), whereas $x^{k,n}$ is the current solution, $v^{k,n}$ is the nonascending vector and $\beta_{k,n}$ is a positive real number picked from $(\gamma_l)_{l=0}^{\infty}$ (l is incremented during every iteration of the Superiorization loop). In this innermost loop, progressively smaller perturbations are added to the previous solution until $z \in \Delta$ and $\phi(z) \leq \phi(x^k)$, where x^k is the output of the previous iteration.

After N Superiorization steps have been performed, the original algorithm P_T is applied to $x^{k,N}$ in order to produce an output that is constraints-compatible and, typically, superior regarding the secondary optimization criterion $\big(\phi(x^{k,N}) \leq \phi(x^k)\big)$.

2.10. Non-ascending Vector Length Variation

Algorithm 2.1 is a general approach to improve any iterative algorithm by introducing directed perturbations in the form of $x^{k,n+1} = x^{k,n} + \beta_{k,n}v^{k,n}$ between each iteration. As long as the perturbation term $\beta_{k,n}v^{k,n}$ does not lower the value of the secondary optimization criterion, the inner loop (10-19) is repeated and the variable l, which is used to generate the values in $\beta_{k,n}$, will be increased. As l increases, β_k decreases. Subsequently, the magnitude of the perturbations $\beta_{k,n}v^{k,n}$ also decreases, until they either lower the value of the secondary optimization criterion or until the length of the non-ascending vector becomes so short that it does not have an impact on the reconstruction anymore.

As tests have shown, the value of l may sometimes increase greatly during single iterations, which may result in an undesirably short non-ascending vector. In order to avoid this behavior, it would be advantageous to reset the value of l after each iteration. However, for the mathematical proof, Superiorization requires that $(\beta_k)_{k=0}^{\infty}$ is a summable sequence, which in [15] is only the case if $(\beta_k)_{k=0}^{\infty}$ is a subsequence of $(\gamma_l)_{l=0}^{\infty}$. It is thus not mathematically justifiable to simply reset l.

The variation of the Superiorization algorithm given in Algorithm 2.2 offers a solution to this issue, as it is able to lower the value of l between iterations while maintaining all mathematical conditions for its convergence.

Line 5 represents the major change: the integer l for calculating the next element from the sequence $(\gamma_l)_{l=0}^{\infty}$ is set to k, where k is the index of the current iteration. Furthermore, the incrementation of l is moved to line 18 in order to improve comprehensibility and performance (otherwise, l would need to be set to $k-1$).

Algorithm 2.2 Variation of Superiorization

```
 1: set $k = 0$
 2: set $x^k = \overline{x}$
 3: while $true$ do
 4:     set $n = 0$
 5:     set $l = k$
 6:     set $x^{k,n} = x^k$
 7:     while $n < N$ do
 8:         set $v^{k,n}$ to be a non-ascending vector for $\phi$ at $x^{k,n}$
 9:         set $loop = true$
10:         while loop do
11:             set $\beta_{k,n} = \gamma_l$
12:             set $z = x^{k,n} + \beta_{k,n} v^{k,n}$
13:             if $z \in \Delta$ and $\phi(z) \leq \phi(x^k)$ then
14:                 set $n = n + 1$
15:                 set $x^{k,n} = z$
16:                 set $loop = false$
17:             end if
18:             set $l = l + 1$
19:         end while
20:     end while
21:     set $x^{k+1} = P_T x^{k,N}$
22:     set $k = k + 1$
23: end while
```

According to [15], in order to prove that Algorithm 2.2 fulfills Theorem 11 and still converges, it is sufficient to show that x^{k+1} of every iteration can be written in the form $x^{k+1} = x^k + \beta_k v^k$ where $(\beta_k)_{k=0}^{\infty}$ is a summable sequence of positive real numbers and $\left(v^k\right)_{k=0}^{\infty}$ is bounded.

Proof. In addition to the proof of the existing algorithm in [15], the proposed algorithm requires a sequence $(\gamma_l)_{l=0}^{\infty}$ of positive real numbers which is not only summable but also monotonously decreasing. Although this may be considered a more severe restriction, it does not have any practical impact as the most commonly used and originally proposed sequence $\gamma_l = a^l$ with $0 < a < 1$ already fulfills this condition.

The results of the Superiorization steps of each iteration (loop in line 7-20) can be written as

$$\begin{array}{ccccc} x^{k,1} & = & x^{k} & + & \beta_{k,0}\upsilon^{k,0} \\ x^{k,2} & = & x^{k,1} & + & \beta_{k,1}\upsilon^{k,1} \\ \vdots & & \vdots & & \vdots \\ x^{k,N} & = & x^{k,N-1} & + & \beta_{k,N-1}\upsilon^{k,N-1} \end{array} \tag{2.4}$$

which can be formulated to

$$x^{k,N} = x^k + \sum_{n=0}^{N-1} \beta_{k,n}\upsilon^{k,n}\,. \tag{2.5}$$

If β_k is defined as

$$\beta_k = \beta_{k,0} = \gamma_k \tag{2.6}$$

this results in the required monotonously decreasing summable sequence $(\beta_k)_{k=0}^{\infty}$ of positive real numbers by the definition of $(\gamma_l)_{l=0}^{\infty}$.

By using β_k as defined in (2.6), the iterative step of (2.5) can be reformulated to:

$$\begin{aligned} x^{k+1} = x^{k,N} &= x^k + \sum_{n=0}^{N-1} \beta_{k,n}\upsilon^{k,n} \\ &= x^k + \sum_{n=0}^{N-1} \frac{\beta_k}{\beta_k}\beta_{k,n}\upsilon^{k,n} \\ &= x^k + \beta_k \sum_{n=0}^{N-1} \frac{\beta_{k,n}}{\beta_k}\upsilon^{k,n} \end{aligned} \tag{2.7}$$

As a result,

$$\upsilon^k = \sum_{n=0}^{N-1} \frac{\beta_{k,n}}{\beta_k}\upsilon^{k,n}\,. \tag{2.8}$$

As l is increased whenever the algorithm reaches line 18 it is obvious, that $\beta_{k,n} \leq \beta_k$ for every $0 < n < N$ and therefore, $\frac{\beta_{k,n}}{\beta_k} \leq 1$. By the definition of the nonascending vector, each $\left\|\upsilon^{k,n}\right\| \leq 1$. As a result of these two limitations, $\left\|\upsilon^k\right\| \leq N$ and therefore, the sequence $\left(\upsilon^k\right)_{k=0}^{\infty}$ is bounded. $\square$

It is important to note that the variation of Superiorization presented in Algorithm 2.2 may not necessarily be considered an improvement of the existing version, but should rather be considered an additional option. As Superiorization is a general approach which may be applied to any P and ϕ, it is not possible to make a general statement as to which of the two algorithms is superior. The proposed variation merely emphasizes the directed disturbances - which might be advantageous in certain circumstances - and prevents the non-ascending vector from becoming undesirably short. In turn, it tends to require more processing time, which is why the choice of the more suitable variation ultimately depends on the specific application.

As a compromise, l may be set to a random number ranging from the current iteration number k to the value of l of the previous iteration. This results in reduced execution time compared to the proposed variation while retaining the intended vector shortness avoidance and maintaining the effect of Superiorization for a higher amount of iterations compared to Algorithm 2.1. The previously given proof is also applicable to this variation.

3 Chapter 3

Target Software / Environment

This chapter provides a description of the software into which Superiorization has been incorporated and outlines the development environment which was established prior to the implementation. It also discusses the benefits for the Discrete Imaging and Graphics Group (DIG)[1] and the users of SNARK which resulted from the new development environment.

In order to ensure efficient software development and unobstructed collaboration between all members and visitors of the DIG, prior to any actual software development, an appropriate environment needed to be created. Previous work on SNARK had been performed predominantly in a sequential manner, although there had been periods of parallel development during which CVS[2] was used as a version control system.

However, in addition to the latest version of CVS being more than 6 years old and CVS currently not being actively developed, the corresponding server setup and repository at the DIG had been lost during a server migration, after which the previous system of manually exchanging tarballs had been reestablished. Furthermore, SNARK development was traditionally only conducted using plain text editors, foregoing the numerous advantages of modern IDEs.

3.1. SNARK

SNARK represents the flagship application of the DIG and lies at the center of its efforts to improve the reconstruction of images from projections. It has been developed for several decades and continues to be utilized in the creation of numerous publications. The resulting degree of complexity is considerable, which is also reflected in the SNARK manual spanning more than 250 pages. This section will hence only provide a general overview and focus on the aspects of SNARK which are relevant to the underlying work.

[1]http://dig.cs.gc.cuny.edu
[2]http://savannah.nongnu.org/projects/cvs

3.1.1. Purpose and History of SNARK

SNARK is a programming system for algorithms which are aimed at solving reconstruction problems, specifically the reconstruction of images from projections such as CT and PET scans. It has been created to help researchers interested in developing and evaluating such reconstruction algorithms. The first version of SNARK was created by Richard Gordon in 1970 and written in FORTRAN. Between 1970 and 2014, 7 major releases (SNARK, SNARK77, SNARK89, SNARK93, SNARK95, SNARK05 and SNARK09) have been made available, gradually increasing its functionality and migrating to C++ in 2005. The recent additions, which are described in detail in Chapter 4 have been deemed sufficiently significant to warrant an incrementation of the version number to SNARK14.

3.1.2. The SNARK Framework

SNARK is based on a multi-tier architecture which comprises four main stages: phantom generation, simulation, reconstruction and analysis. During the first two stages, a digital phantom is produced and a measurement process (specifically, a CT or PET scan) is simulated. The resulting data is saved both in the form of the properties of the phantom and in the form of projections and raysums. This measurement process is highly configurable: It allows not only a significant amount of freedom when generating the phantom (e.g. geometric shapes, attenuation coefficients and inhomogeneity), but also regarding the properties and capabilities of the measurement device (e.g. detector geometry, number of projections/rays, noise levels and mono- or polychromatic beams).

In the reconstruction stage, which is the stage into which Superiorization is to be introduced, various built-in reconstruction algorithms such has EMAP or ART, which are described in Chapter 3.1.6 may be executed.

During the analysis stage, various metrics regarding the phantom and/or the reconstructions are calculated based on the information in the reconstruction file. Statistical analysis of the results as well as comparisons of the reconstructions to the phantom may also be carried out [9].

3.1.3. Information Flow

The generation and flow of information within SNARK can roughly be described as follows: Input is provided via a text file which contains a list of commands. These commands trigger

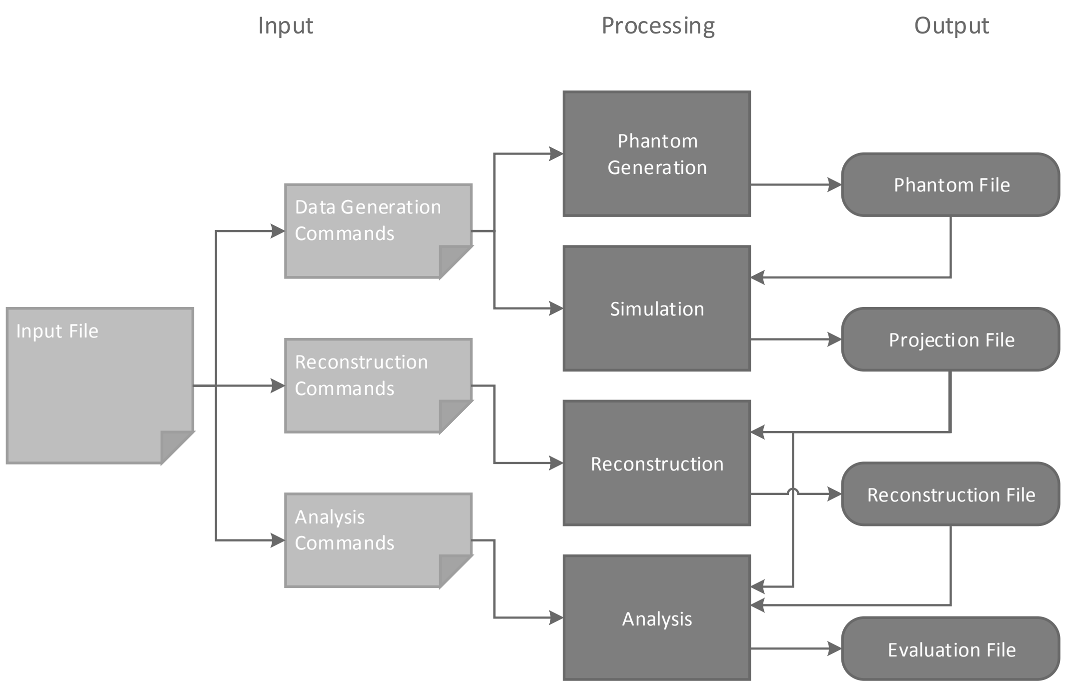


Figure 3.1.: SNARK Information Flow

the various functions within SNARK and need to be provided in a sequence which corresponds to the different stages mentioned in Chapter 3.1.2. Each stage produces according output files, which are required by subsequent stages. Output from previous SNARK executions may however be re-used. It is, for example, possible to access previously generated projection data, skipping the data generation stage. It is also possible to perform multiple reconstructions of the same projection using different reconstruction algorithms within a single SNARK execution. Figure 3.1 illustrates the generation and flow of information within SNARK, as well as the required input and the generated output files.

3.1.4. Commands

This subsection is intended to give a brief overview of the most important commands of each of the different SNARK stages.

During the phantom generation stage, the command CREATE is used to generate phantoms constructed out of various geometric shapes such as ellipses, rectangles or triangles.

In the simulation stage, the command PROJECTION is used to generate a binary projection file which contains information needed during the subsequent reconstruction and analysis stages.

During the reconstruction stage, three commands are of specific interest: STOP, EXECUTE and the newly added SUPERIORIZE. STOP, which is explained in greater detail in Chapter 4.4.6, is required to define the stopping criterion for the reconstruction algorithm while EXECUTE is used to select the desired algorithm as well as to trigger its execution. Lastly, the newly introduced command SUPERIORIZE, which is described in Chapter 4.4.1, may be issued prior to the execution of an algorithm in order to automatically superiorize said algorithm.

The most important command during the analysis stage is EVALUATE. It is used to generate several quantitative measures regarding the difference of the phantom image and the various reconstructions.

3.1.5. Input and Output

As mentioned in Chapter 3.1.3, Input is basically provided via a single text file containing a sequence of commands. This file needs to be passed as an argument when calling SNARK from the command line. The generated output essentially consists of four files:

i. "file11", which contains the geometry and properties of the phantom that has been generated during the phantom generation stage

ii. "prjfil", which contains the physical measurements that have been obtained during the simulation stage

iii. "recfil", which contains the images that have been reconstructed by the various algorithms during the reconstruction stage

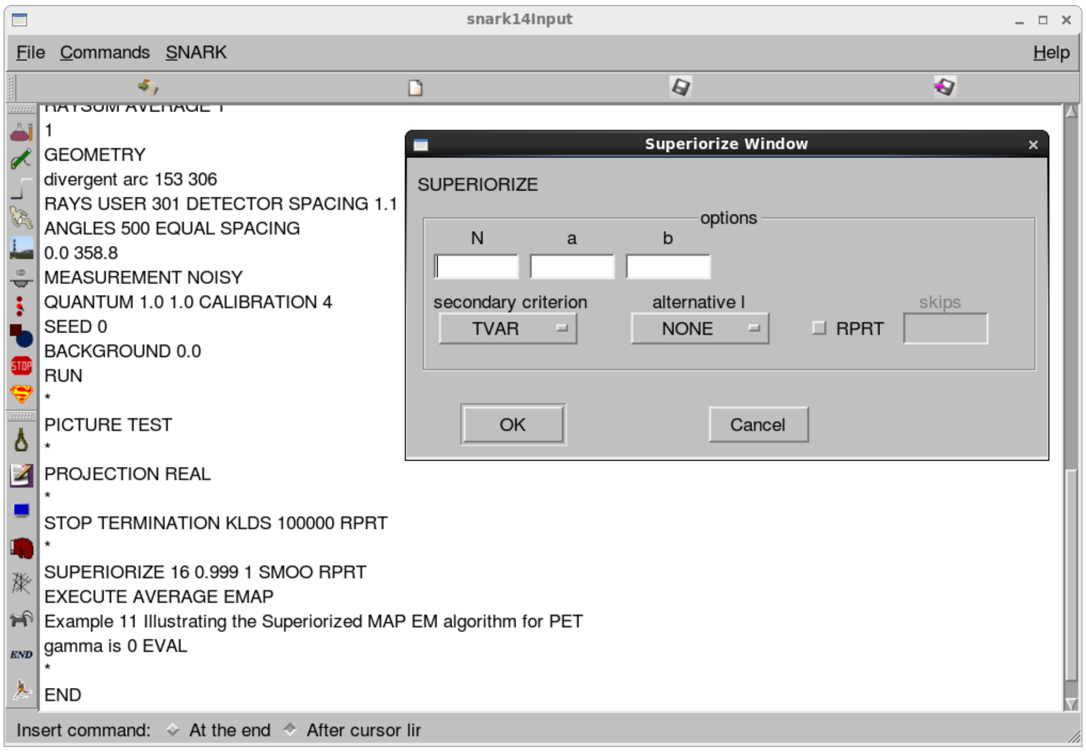
iv. "eval", which contains various metrics that have been calculated to quantify the quality of the reconstruction

In an effort to increase user friendliness, two additional programs have been developed to aid users generate an appropriate input file as well as to visualize the data contained in the output files: snarkInput and snarkDisplay.

3.1.5.1. snarkInput

snarkInput is a simple Qt-based[3] GUI which allows its user to generate or edit an input file for a SNARK execution. Each command can be built using a separate window which aids in

[3]http://qt-project.org



```
RAYSUM AVERAGE 1
1
GEOMETRY
divergent arc 153 306
RAYS USER 301 DETECTOR SPACING 1.1
ANGLES 500 EQUAL SPACING
0.0 358.8
MEASUREMENT NOISY
QUANTUM 1.0 1.0 CALIBRATION 4
SEED 0
BACKGROUND 0.0
RUN
*
PICTURE TEST
*
PROJECTION REAL
*
STOP TERMINATION KLDS 100000 RPRT
*
SUPERIORIZE 16 0.999 1 SMOO RPRT
EXECUTE AVERAGE EMAP
Example 11 Illustrating the Superiorized MAP EM algorithm for PET
gamma is 0 EVAL
*
END
```

SUPERIORIZE

options: N, a, b

secondary criterion: TVAR

alternative l: NONE

☐ RPRT

skips

Figure 3.2.: snark14Input

the creation process and performs basic checks regarding the validity of the selected options and values. Upon completion of the process, it offers the possibility of triggering a SNARK execution using the created file as input. Figure 3.2 shows an updated version of snarkInput which already contains the new command SUPERIORIZE.

3.1.5.2. snarkDisplay

snarkDisplay represents the second Qt-based GUI of the SNARK package. Its main purpose is to visualize output which has been generated by SNARK, for which it provides a rich feature set. Besides displaying the reconstructed image and enabling comparisons between different reconstructions, it also offers various functions to create graphs based not only on the reconstructed images, but also on the data contained in the evaluation file, making it an indispensable tool in the assessment of the quality of the output of the various reconstruction algorithms. Figure 3.3 shows snarkDisplay being used to perform thresholding and contrast stretching on an image of a brain phantom containing a series of simulated tumors.

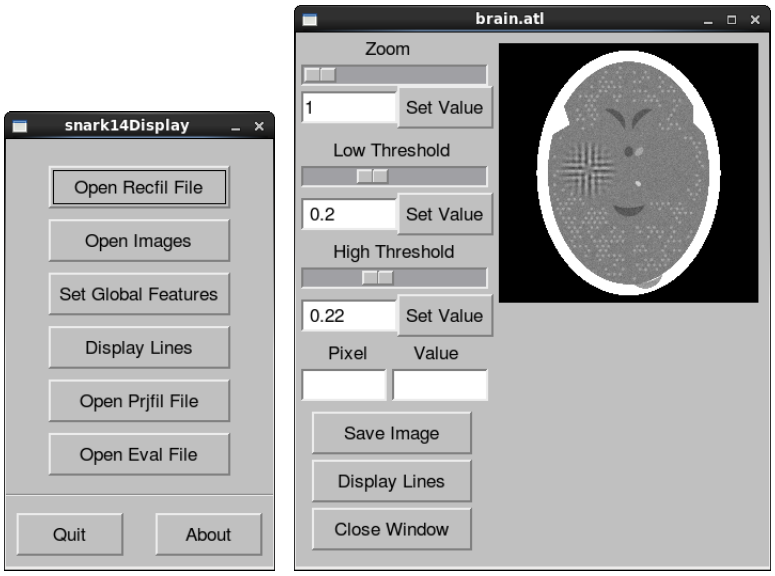


Figure 3.3.: snark14Display

3.1.6. Reconstruction Algorithms

Since the main purpose of SNARK is to aid in the development and evaluation of reconstruction algorithms, a considerable number of such algorithms has already been integrated into the SNARK package. They include several backprojection variants, the rho filtered layergram which is described in [24], the Fourier method, several variants of ART, the quadratic optimization techniques described in [16, 1], the simultaneous iterative reconstruction technique (SIRT) of [13], the linogram method as well as EMAP. EMAP and the ART-variant SART are of specific interest to the underlying work, as they are utilized during the verification process of the implementation of Superiorization.

EMAP is a general implementation of a maximum a posteriori probability (MAP) algorithm for PET based on a modified expectation-maximization (EM) algorithm [26, 17], whereas ART is a general implementation of the additive Algebraic Reconstruction Techniques [9]. SART is an abbreviation for Simultaneous Algebraic Reconstruction Technique and represents the most recent algorithmic addition to the SNARK feature set.

3.1.7. User-Defined Extensions

To facilitate the process of integrating experimental algorithms, the SNARK package supports user-defined builds which allow the introduction not only of new algorithms, but also of additional stopping criteria and figures of merit (FOM). The addition of Superiorization expands this functionality to include user-defined secondary optimization criteria as well as the according non-ascending vectors.

Technically, user-defined builds are implemented as a series of placeholder files which are addressable within SNARK, into which user-defined code can be integrated. While more modern solutions may have become available since the introduction of this functionality into SNARK, it still represents a very simple solution which may quickly be adopted even by developers with little or no prior knowledge of the SNARK framework.

3.1.8. The SNARK Experimenter

The experimenter is a sub-function of SNARK which serves to automate its execution process in order to compare the output of different reconstruction algorithms by means of statistical hypothesis testing. It is capable of automatically generating a series of randomized phantoms, performing the according simulated measurements, executing the desired reconstruction algorithms and obtaining the specified figures of merit from the reconstructions.

At the end of the experimentation process, the collected figures of merit are compared statistically, resulting in a p-value. In conjunction with a predetermined significance level, this p-value may be used to either accept or reject the null hypothesis that the output of one algorithm is superior to the other with respect to the selected FOM.

The SNARK experimenter is utilized in the process of verifying the functionality of the implementation of Superiorization by performing a comparison of superiorized and unsuperiorized ART and SART which is described in Chapter 5.2.

3.2. Development Environment

Due to the fact that during the time of the integration of Superiorization into SNARK, up to 4 people were scheduled to work on the SNARK package simultaneously, the re-introduction of a version control system was imperative. Especially in combination with the introduction of an appropriate IDE, the likelihood that the initial effort required for the improvement of the development environment would be offset by subsequent time savings was deemed sufficiently high to warrant an endeavor to that effect. Time savings were primarily expected as a result of an accelerated development process, reduced communication overhead and a more efficient way of code distribution. In order to avoid costs, only free and open-source software (FOSS) was considered in the selection process.

3.2.1. Version Control / Collaboration

The first step towards setting up an efficient working environment at the DIG was the reintroduction of a version control system. Since 2 out of 4 DIG members were already familiar with Apache Subversion[4] (SVN) and the remaining 2 members had not previously used any version control system, SVN was determined to be the system of choice. Its selection made possible a reduction of the time required for familiarization and provided a well-known list of features, enabling the DIG members which were familiar with SVN to provide advice as well as allowing the automatization of certain tasks which had previously been performed manually, e.g. updating source code file header information.

The installation of the server component was performed on the DIG's existing web server, running Red Hat Enterprise Linux (RHEL) 6.3. Considerable effort was made to install not the default version of SVN for this system, which is 1.6.21, but to install version 1.8.9 instead, which supports custom keyword substitution, enabling the aforementioned source file header to be generated in an entirely automated manner. On Windows clients, TortoiseSVN[5] 1.8.7 was used while on Linux clients, RapidSVN[6] 0.12.1 and Apache's command line client 1.8.9 was used.

3.2.2. Integrated Development Environment

In order to further streamline the development process, Eclipse[7] Kepler SR2 was introduced a the DIG as the standard IDE to be used for SNARK development. It was chosen not only because it is Open Source, but also because the same circumstances as with SVN applied, with 2 out of 4 DIG members already being familiar with it. Additionally, it is widely used and well-known among software developers, is available on all major operating systems, supports all three programming languages currently used at the DIG and is able to directly connect to an SVN server, further simplifying the collaboration aspect. Owing to its complex structure and dependencies, the process of importing the SNARK source code into an Eclipse project proved to be more challenging and time consuming than expected, but the time required was easily outweighed by the resulting advantages such as debugging, code completion and syntax highlighting.

[4]http://subversion.apache.org
[5]http://tortoisesvn.net
[6]http://www.rapidsvn.org
[7]http://www.eclipse.org

3.2.3. CentOS VirtualBox Virtual Machine

In addition to the introduction of SVN and Eclipse, a virtual machine (VM) running the RHEL-derived Linux distribution CentOS[8] 6.5 was created using Oracle VM VirtualBox[9] 4.3.10 as virtualization software. This virtualized environment, into which SNARK, SVN and Eclipse were subsequently installed, offers several benefits: Firstly, enhanced cross-platform capabilities, allowing SNARK to be invoked on any system capable of running VirtualBox. Secondly, a unified environment, which can easily be distributed to all DIG members participating in SNARK development. Thirdly, a new, additional means of distributing SNARK to end-users in a considerably more user-friendly, pre-packaged manner which includes read-only SVN access for SNARK updates as well as Eclipse for the creation of user-defined builds.

[8]http://www.centos.org
[9]http://www.virtualbox.org

4

Chapter 4

Implementation

This chapter outlines the implementation process as well as findings and advancements which have been made during implementaion. Following theoretical considerations regarding an ideal realization, the concrete implementation in SNARK, including secondary aspects which were required to ensure usefulness of the new functions, is covered.

As previously stated, the main focus of the underlying work is the integration of Superiorization into SNARK in a manner which is as flexible as possible to both pre-existing and future algorithms and secondary optimization criteria. This necessitates a breakdown of the methodology into its variable and non-variable components as well as considerations regarding possible prerequisites.

A distinction needs to be made between an ideal approach, which may be pursued when developing entirely new software, and approaches which can be applied to pre-existing code, taking into consideration given limitations. These limitations may stem from the fact that the underlying software and its data flow was not initially designed to be adaptive to changes of this nature and that, for the sake of consistency, any additions to the code should follow pre-established patterns. In SNARK09, for which the addition of Superiorization represents the transition to SNARK14 as discussed in Chapter 3.1.1, only the latter is the case.

4.1. Variable and Non-Variable Superiorization Constituents

The first step towards an implementation is a structural analysis of the algorithm and its input parameters, breaking it down into its variable and non-variable parts. The following constituents to the Superiorization methodology represent elements which may differ according to the specific application and thus need to be implemented in a way so that they may be freely defined by the user. The variable components are comprised of:

i. Variables N and γ: N defines how many times a non-ascending vector is to be applied while γ defines how quickly the step-size is reduced.

ii. ϕ function: The ϕ function defines the secondary optimization criterion.

iii. Non-ascending vector: The non-ascending vector is closely tied to the ϕ function.

iv. Δ: In all currently considered applications, Δ may either be defined as $\mathbb{R}^J$ or $\mathbb{R}^J_+$

v. Base algorithm: The initial, unsuperiorized iterative algorithm.

While some of the remaining constituents to the Superiorization methodology are variable in the sense that they change their value during the process of Superiorization, they follow a pre-defined pattern and do not need to be read externally. They can thus be hard-coded without adversely effecting the resulting program's flexibility and are comprised of:

i. Variables k, l, n. These are merely control variables which are incremented by one during various loop iterations.

ii. Inner, middle and outer loop. The loops do not vary depending on any conditions but always follow the exact same pattern. Multi-stage perturbation schemes as described in [10] are beyond the scope of this document.

4.2. Adaptations for Implementation

In Chapter 2, the Superiorization algorithm has been described strictly from a mathematical point of view. Note that Algorithm 2.1 produces an endless sequence of reconstructed images and thus requires the addition of a stopping criterion. Additionally, in order to achieve optimal results, considerations regarding code simplicity, programming efficiency and resource consumption may be taken into account prior to its actual implementation.

In practice, this amounts to several changes. Variables only need to store information regarding the current iteration, minimizing the amount of memory required. Furthermore, the variables k, $loop$, x^k and $\beta_{k,n}$ may be omitted and their values instead be assigned directly, resulting in minor memory and processing time savings. Lastly, an additional variable b is introduced. b is multiplied with a in order to increase flexibility regarding the step size of the non-ascending vector. Otherwise, the first multiplying factor of the non-ascending vector would always be 1, which may not be desirable in certain cases. Algorithm 4.1 represents the resulting adapted Superiorization algorithm.

Algorithm 4.1 Superiorization, adapted for implementation

```
1: set $l = -1$
2: while $Pr(\overline{x}) > \overline{\varepsilon}$ do
3:    set $n = 0$
4:    set $x = \overline{x}$
5:    while $n < N$ do
6:       set $v$ to be a non-ascending vector for $\phi$ at $x$
7:       while $true$ do
8:          set $l = l + 1$
9:          set $z = x + b \cdot a^l \cdot v$
10:         if $z \in \Delta$ and $\phi(z) \leq \phi(\overline{x})$ then
11:            set $n = n + 1$
12:            set $x = z$
               break
13:         end if
14:      end while
15:   end while
16:   set $\overline{x} = P_T x$
17: end while
```

4.3. Ideal Approach

Following the aforementioned notion of an ideal, state of the art implementation of Superiorization into new software using an object-oriented programming language, there is a series of conditions which need to be met. Firstly, all variable constituents listed in Chapter 4.1 need to be freely configurable. Secondly, the dynamic and unlimited addition of non-ascending vectors, ϕ functions and base algorithms needs to be possible. Thirdly, in the case of the execution of multiple algorithms in succession, it must be possible to define specific Superiorization parameters for each algorithm. The the first and third objectives are matters of software design, while the means by which the second condition is met may vary depending on the underlying programming language. For example, in Java, *reflection* may be used to dynamically load an unlimited number of new classes which implement ϕ functions based on an Interface. In languages which do not offer this functionality, a more static approach based on inheritance may be employed.

Figure 4.1 shows an example of how software which supports Superiorization as an option may be designed. A class `Main` may be used to repeat executing iterations the selected reconstruction algorithm and terminate once the specified termination criterion is met. The pseudocode depicted in Algorithm 4.1 would be implemented in a class `Superiorization`, which is also called by the class `Main` between each iteration of the original algorithm in case Superiorization is enabled. This `Superiorization` class would in turn rely on separate classes to perform the calculations necessary to generate the appropriate non-ascending vectors and the values related to the secondary optimization criteria. In order to provide the aforementioned flexibility regarding new optimization criteria and non-ascending vectors, the corresponding classes would need to be derived from an interface class, the same of which is the case for reconstruction algorithms and termination criteria.

The basic sequence of events is shown in Figures 4.2 and 4.3, with Figure 4.2 depicting the Superiorization-related logic within the `Main` class and Figure 4.3 illustrating one Superiorization step within the class `Superiorization`. Apart from minor adjustments which were necessary due to the given SNARK code structure, these figures are an accurate representation of the logic which has been incorporated into the files `exalg.cpp` and `superior.cpp` of the SNARK package.

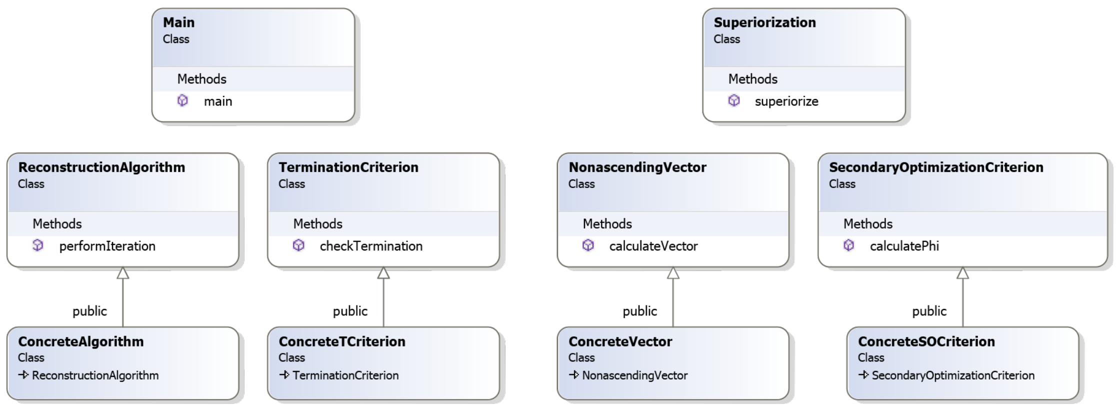


Figure 4.1.: Sample Class Diagram

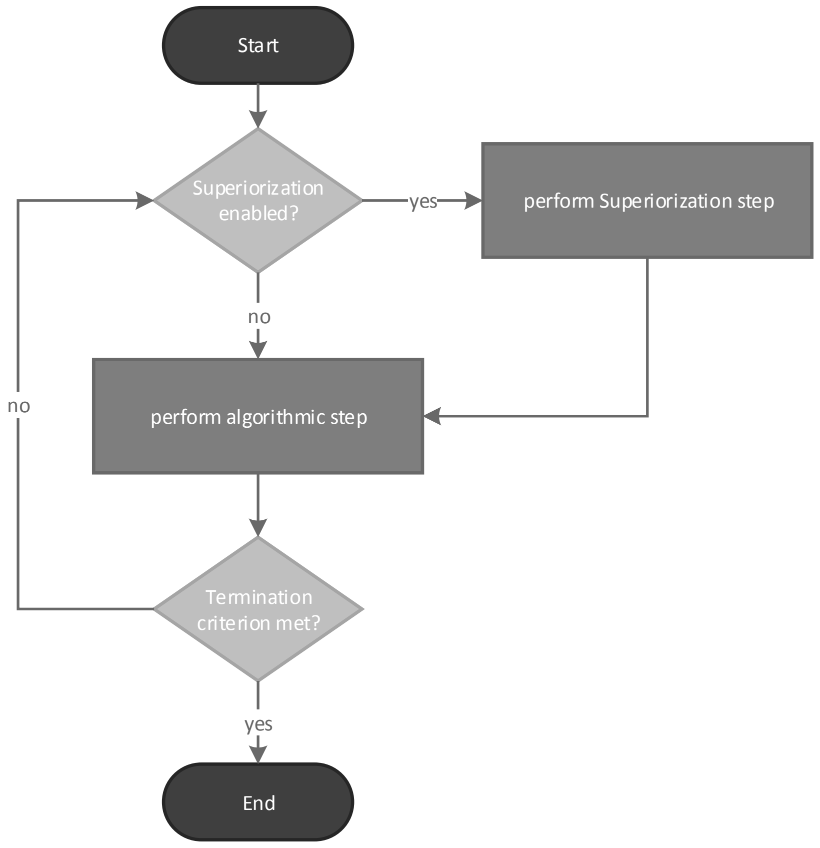


Figure 4.2.: Main Class Flow Diagram

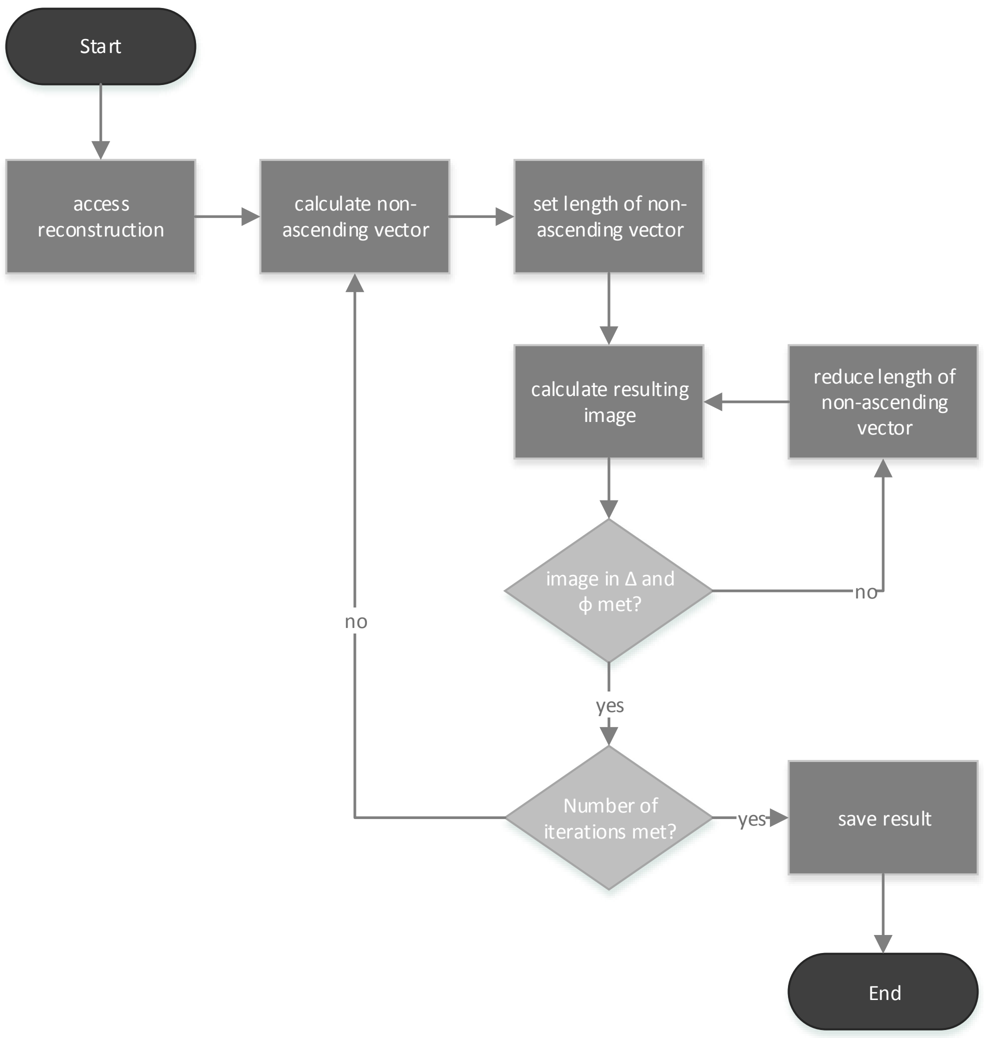

Start
access reconstruction
calculate non-ascending vector
set length of non-ascending vector
calculate resulting image
reduce length of non-ascending vector
image in Δ and ϕ met?
no
yes
no
Number of iterations met?
yes
save result
End

Figure 4.3.: Superiorization Class Flow Diagram

4.4. Implementation in SNARK

Despite the fact that SNARK09 is a descendant of earlier releases of SNARK and as a result, its code architecture dates back several decades, the implementation of Superiorization could be performed in a manner which not only closely resembles the ideal implementation outlined in Chapter 4.3 and provides a high amount of flexibility but also adheres to the pre-exissting code structure. Barring some remnant architectural elements which are representing concessions to its FORTRAN heritage, SNARK essentially already provides a suitable environment for the implementation of Superiorization. The point of entry is the function `exalg()`, which repeatedly executes the various reconstruction algorithms and may be upgraded to perform Superiorization between each algorithmic step. It also makes use of interface classes for algorithms (`alg_class`) and termination criteria (`termtest_class`).

Due to the fact that SNARK was created with limited extensibility in terms of reconstruction algorithms, termination criteria and figures of merit in mind, it already provides functionality for the addition of algorithms which are not part of the original software package. This ability of creating user-defined builds merely needs to be extended to include user-defined secondary optimization criteria and non-ascending vectors. Furthermore, the pre-existing structure of input commands allows Superiorization to be integrated simply as a new command which may be issued prior to the execution of a reconstruction algorithm, indicating the desire to superiorize the subsequent algorithm. This was achieved by an addition to `snark.cpp` (which passes the various commands to the respective software parts) and the incorporation of the command parsing logic into the Superiorization class.

4.4.1. Command SUPERIORIZE

Upon close inspection of the underlying code and the way the execution of an algorithm is triggered, the introduction of a new command named SUPERIORIZE was found to be an ideal approach because it could not only be implemented with relative ease, but it also provides a maximum amount of flexibility. It can be issued prior to the execution of an algorithm and, in the case of the execution of multiple algorithms in succession, may be issued repeatedly to define specific settings for each algorithm. This way of handling Superiorization input integrates seamlessly and without adverse effects into the pre-existing SNARK architecture, as other commands operate in exactly the same way and remain valid until re-issued with different settings.

The initial version of the command included the separate specification of the secondary optimization criterion and the method of calculating the non-ascending vector. However, due to the fact that in practice, secondary optimization criteria are not expected to have more than one suitable non-ascending vector, the specification of the non-ascending vector has been dropped in favor of simplicity and has instead been pre-set for each built-in secondary optimization criterion. Since the opposite might not always be the case though and one method of calculating a non-ascending vector might theoretically be determined to be suitable for additional or modified secondary optimization criteria, non-ascending vectors are still handled separately within the source code and can be reused and reassigned with ease.

The final version of the command with all input variables and options is given below. For a detailed description of the syntax in Extended Backus-Naur Form (EBNF), please refer to Appendix A.1.

$$\text{SUPERIORIZE N a b} \left\{ \begin{array}{c} \text{TVAR} \\ \text{SMOO} \\ \text{SCR3} \\ \text{SCR4} \\ \text{SCR5} \end{array} \right\} \text{[POSI]} \left\{ \begin{array}{c} \text{ATL1} \\ \text{ATL2} \end{array} \right\} \text{[RPRT [n]]}$$

Regarding the input variables, N, a and b have been explained in Chapter 4.1 and Chapter 4.2. The remaining options and variables are described in detail below.

4.4.2. Secondary Optimization Criteria

In the scope of the underlying research, two secondary optimization criteria have been integrated into SNARK: total variation and smoothness, both of which are based on [12], where they have previously been successfully used for Superiorization.

4.4.2.1. Total Variation (Option TVAR)

Total variation (TV) has been chosen as a secondary optimization criterion due to the fact that it has recently been popular in medical imaging. For a reconstructed image vector x, consisting of gray values, the TV is defined as:

$$TV(x) = \sum_{c \in C} \sqrt{\left(x_c - x_{\rho(c)}\right)^2 + \left(x_c - x_{\xi(c)}\right)^2}, \quad TV(x) \in \mathbb{R} \tag{4.1}$$

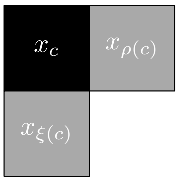


Figure 4.4.: Total Variation Function Kernel

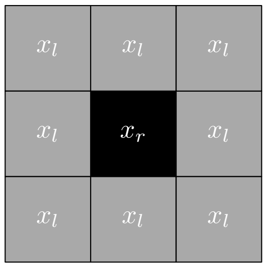


Figure 4.5.: Smoothness Function Kernel

where C is the set of all indices of pixels (numbered serially and line-by-line) that are not in the rightmost column or the bottom row of the pixel array. For any single pixel x with index c in C, $\rho(c)$ and $\xi(c)$ are the indices of the pixels to its right and below it. A graphic representation of the function's kernel is given in Figure 4.4.

4.4.2.2. Smoothness (Option SMOO)

Smoothness is an implementation of the penalty function ψ recommended in [19] and is defined as:

$$\psi(x) = \sum_{r \in M} \left(x_r - \frac{1}{8} \sum_{l \in M_r} x_l \right)^2, \quad \psi(x) \in \mathbb{R} \tag{4.2}$$

where again, x is a reconstructed image, M is the set of indices r such that the rth pixel is not on the border of the image region and M_r, for $r \in M$, is the set of indices associated with the eight pixels neighboring the rth pixel. Figure 4.5 depicts the kernel of the function.

4.4.2.3. User-Defined Criteria (Options SCR3, SCR4, SCR5)

The options SCR3, SCR4 and SCR5 represent placeholders for the addition of user-defined secondary optimization criteria. As with other user-defined additions (such as algorithms or figures of merit), SNARK already provides the necessary infrastructure, meaning that the required classes already exist and the command option is available. The user merely needs to add the own code to the corresponding method of the corresponding class.

4.4.3. Non-Ascending Vectors

As non-ascending vector for a given secondary optimization criterion (see Chapters 2.8 and 2.7, respectively), the inverse gradient of the objective function, whose length is modified by the scaling factor γ_l is used. The results of the calculations which are required to determine each gradient are given below.

4.4.3.1. Gradient for Total Variation

In the case of TV, up to three pixels are involved in the calculation of each element of the gradient, resulting in up to three partial derivatives which need to be summed up:

$$g_j = \frac{\partial \tau}{\partial x_j}(x) = \sum_{t=0}^{2} \frac{\partial \tau_t}{\partial x_j}(x) \quad [12] \tag{4.3}$$

As not all three terms are applicable for every pixel, two additonal sets of indices need to be introduced: C_1, which is a set of indices of pixels that are not on the left border of the reconstruction region and C_2, which is a set of indices of pixels that are not on the bottom border of the reconstruction region. This allows the partial derivatives to be expressed as follows:

$$\tau_0(x_c) = \begin{cases} \sqrt{\left(x_c - x_{\rho(c)}\right)^2 + \left(x_c - x_{\xi(c)}\right)^2} & \text{if } c \in C \\ 0 & \text{otherwise} \end{cases} \tag{4.4}$$

$$\tau_1(x_c) = \begin{cases} \sqrt{\left(x_{\theta(c)} - x_c\right)^2 + \left(x_{\theta(c)} - x_{\xi(\theta(c))}\right)^2} & \text{if } c \in C_1 \\ 0 & \text{otherwise} \end{cases} \tag{4.5}$$

$$\tau_2(x_c) = \begin{cases} \sqrt{\left(x_{\alpha(c)} - x_{\rho(\alpha(c))}\right)^2 + \left(x_{\alpha(c)} - x_c\right)^2} & \text{if } c \in C_2 \\ 0 & \text{otherwise} \end{cases} \tag{4.6}$$

where $\theta(c)$ is the index of the pixel to the left and $\alpha(c)$ is the index of the pixel above. The resulting derivatives, which are required for producing the corresponding source code, are:

$$\tau_0'(x_c) = \frac{2x_c - x_{\rho(c)} - x_{\xi(c)}}{\sqrt{\left(x_c - x_{\rho(c)}\right)^2 + \left(x_c - x_{\xi(c)}\right)^2}} \tag{4.7}$$

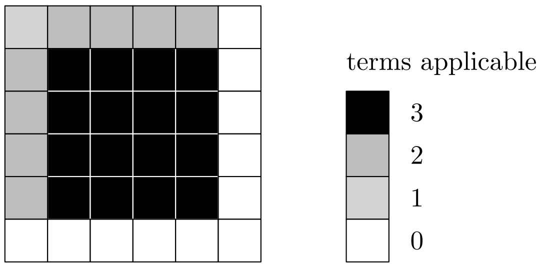


Figure 4.6.: Total Variation Gradient Calculation Scheme

$$\tau_1'(x_c) = \frac{x_c - x_{\theta(c)}}{\sqrt{\left(x_{\theta(c)} - x_c\right)^2 + \left(x_{\theta(c)} - x_{\xi(\theta(c))}\right)^2}} \tag{4.8}$$

$$\tau_2'(x_c) = \frac{x_c - x_{\alpha(c)}}{\sqrt{\left(x_{\alpha(c)} - x_{\rho(\alpha(c))}\right)^2 + \left(x_{\alpha(c)} - x_c\right)^2}} \tag{4.9}$$

In less formal terms, due to the layout of the 2x2 kernel depicted in Figure 4.4, an element of the gradient may either be considered top left, top right or bottom left. For elements inside M, all three terms are applicable, whereas the border region needs to be handled separately. In the case of the top left element, only the first term is applicable. For the remainder of the top border, terms 1 and 2 are applicable and for the remainder of the left border, terms 1 and 3 are applicable. For the right and bottom borders, none of the terms apply, resulting in zero values. Figure 4.6 illustrates the calculation scheme.

4.4.3.2. Gradient for Smoothness

Due to the 3x3 kernel of the smoothness penalty function which is shown in Figure 4.5, up to 9 pixels need to be considered in the calculation of each element of the gradient: a center pixel and up to eight surrounding pixels. The term to calculate gradient elements which are not on the border is:

$$g_j = \frac{\partial \psi}{\partial x_j}(x) = 2\left(x_j - \frac{1}{8}\sum_{l \in M_j} x_l\right) - \frac{1}{4}\sum_{r \in M_j \cap M}\left(x_r - \frac{1}{8}\sum_{l \in M_r} x_l\right) \quad [12] \tag{4.10}$$

In this term, j is the index for all pixels in the reconstruction and M_j and M_r indicate the sets of the indices l of the eight neighboring pixels.

As is the case for TV, the border region of the gradient needs to be handled separately, accounting for the reduced number of surrounding pixels. A distinction needs to be made specifically between the four corner pixels, the eight pixels adjoining the corner pixels and

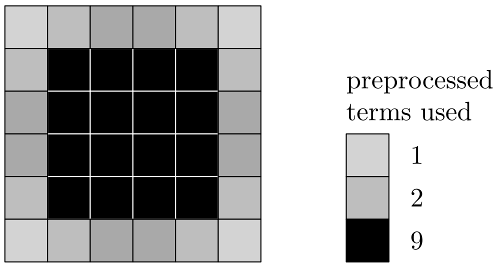


Figure 4.7.: Smoothness Gradient Calculation Scheme

the remainder of the border region.

The symmetry of this kernel, combined with the fact that SNARK exclusively handles rectangular images, allows the resulting code to be optimized considerably: the bracketed expressions are algorithmically equivalent and can be pre-processed for each pixel, after which the pre-processed values merely need to be summated accordingly to determine the gradient, resulting in an approximately ninefold reduction in processing time compared to a per-element calculation. Furthermore, if the pre-processed values are stored in an array which is similar in layout and size to the original reconstruction and its border region is initialized to zero, all gradient elements which are not on the border may be calculated using the same logic, reducing code complexity. The calculation scheme is graphically depicted in Figure 4.7.

4.4.4. Event Space for Superiorization

The valid event space Δ for Superiorization is defined via the option POSI. If it is specified, $z \in \mathbb{R}_+^J$ is required for the inner loop to be terminated, otherwise this check is omitted and $z \in \mathbb{R}^J$ is accepted.

4.4.5. Alternative Handling of the Variable l

During early testing of the implementation of Superiorization, it has been observed by the author that in some cases, the value of the variable l was incremented greatly during a single Superiorization step. This results in a severely shortened non-ascending vector in all subsequent steps and attenuates the effect of Superiorization to a degree where it essentially becomes ineffective, especially in cases where this phenomenon occurs at an early stage.

It has thus been proposed to reset l to the value of the iteration number k for each iteration, which has been discussed in Chapter 2.10 and would provide a solution to this undesired behavior while retaining the requirement that the sequence β_k is summable, which is necessary for the introduced perturbations to remain bounded. This alternative handling of the variable

l is referred to as ATL1 in SNARK. Further considerations resulted in an additional variant of handling l, specifically, to randomize its value to range from k to the value at the end of the previous iteration, which has also been implemented and is referred to as ATL2 in SNARK. As mentioned in Chapter 2.10, the latter basically represents a hybrid of the original handling and the first proposed variant, providing faster computation and reducing the likelihood of applying an inappropriately long non-ascending vector while essentially retaining the intended phenomenon avoidance capability. A thorough mathematical analysis of this strategy is currently outstanding but first experimental evidence, as is given in Chapter 5.1 appears to confirm that ATL1 and ATL2 achieve their intended effect.

4.4.6. New Termination Criteria

The integration of Superiorization also necessitated the integration of several new stopping criteria. The resulting new stopping command including new and existing options is given below as well as in EBNF in Appendix A.2. It is followed by a detailed explanation of only the newly introduced criteria (ITERATION simply stops after a given number of iterations while VARIANCE has previously been introduced into SNARK and TRM1/TRM2 represent placeholders for user-defined stopping criteria).

$$\text{STOP}\left\{\begin{array}{l}\text{ITERATION iter}\\ \text{TERMINATION}\left\{\begin{array}{l}\text{VARIANCE } \varepsilon\\ \text{KLDS } \varepsilon \text{ [RPRT [n]]}\\ \text{RESI } \varepsilon \text{ [RPRT [n]]}\\ \text{WSQD } \varepsilon \text{ [RPRT [n]]}\\ \text{MLST [RPRT [n]]}\\ \text{TRM1}\\ \text{TRM2}\end{array}\right\}\end{array}\right\}$$

For the following termination criteria and figures of merit, it is necessary to first perform a series of definitions:

i. a *line of response* (LOR) is the line between detectors, along which events have been measured

ii. x denotes a reconstructed image vector

iii. y denotes the corresponding phantom image vector

iv. I denotes number of lines of response of the underlying set of emission tomography measurements

v. J denotes the number of pixels for which the activity needs to be determined by the reconstruction process

vi. $a_{ij}$is defined as the length of intersection of the ith line with jth pixel, which is equivalent to the probability of counting an event generated in the jth pixel for the ith LOR

vii. b_i is defined as the count of events in the ith LOR

4.4.6.1. Kullback-Leibler Distance (Option KLDS)

The Kullback-Leibler (KL) distance is a proximity value defined as:

$$Pr\left(x,y\right)=\sum_{i=1}^{\mathrm{I}}\left(b_i\ln\frac{b_i}{\sum_{j=1}^{J}a_{ij}x_j}+\sum_{j=1}^{J}a_{ij}x_j-b_i\right),\quad Pr\left(x,y\right)\in\mathbb{R}\quad [12] \tag{4.11}$$

It is a measure, how well a given reconstruction matches the readings obtained during the simulation process. If KLDS is specified, the iterative process is terminated as soon as the value of the Kullback-Leibler distance of the reconstructed image is less than or equal to the specified value of ε.

4.4.6.2. Residual (Option RESI)

In an effort to further extend the termination capabilities of SNARK towards providing more criteria which may be suitable for superiorized algorithms, the residual - which had previously only been available as a figure of merit - was implemented as a stopping criterion. It is defined as:

$$Pr\left(x,y\right)=\sqrt{\sum_{i=1}^{\mathrm{I}}\left(b_i-\sum_{j=1}^{J}a_{ij}x_j\right)^2},\quad Pr\left(x,y\right)\in\mathbb{R} \tag{4.12}$$

Similar to (4.11), the reconstruction process is terminated once the value of this function is lower than the specified ε.

4.4.6.3. Weighted Squared Distance (Option WSQD)

Basically, the weighted squared distance represents a weighted variant of the residual. Its implementation has been performed by Bernhard Prommegger in the scope of [23]. It was required in order to provide a suitable stopping condition for the experiments involving SART since in [18] it was shown, that SART is a minimizer of this measure. Like the Kullback-Leibler distance and the residual, it is a proximity value and defined as:

$$Pr\left(x,y\right)=\sum_{i=1}^{\mathrm{I}}\left(\frac{\left(b_i-\sum_{j=1}^{J}a_{ij}x_j\right)^2}{\sum_{j=1}^{J}a_{ij}}\right),\quad Pr\left(x,y\right)\in\mathbb{R} \tag{4.13}$$

It is assumed that $\sum_{j=1}^{J}a_{ij}>0$. Similar to (4.11) and (4.12), the reconstruction process is terminated once the value of this function drops below the specified ε.

4.4.6.4. MLEM-STOP (Option MLST)

MLEM-STOP is an implementation of the indicator function $J(x)$, which is advocated to be used for MLEM in [3]. When MLST is specified, the reconstruction process is terminated after the first iteration at which the value of $J(x)$ is less than or equal to 1. It is defined as:

$$J(x,y)=\frac{\sum_{i=1}^{I}\left(b_i-\sum_{j=1}^{J}a_{ij}x_j\right)^2}{\sum_{i=1}^{I}\sum_{j=1}^{J}\left(a_{ij}x_j\right)},\quad J\left(x,y\right)\in\mathbb{R} \tag{4.14}$$

Although MLST does not constitute a viable stopping criterion for superiorized algorithms, its implementation is useful when comparing Superiorization to MLEM-STOP, as it has recently been done in [12]. It is also helpful in order to verify the correctness of the implementation by making it possible to recreate the results given in the paper.

4.4.7. New Figures of Merit

In order to provide enhanced overall functionality in parts of SNARK which are related to Superiorization, as well as for verification purposes and for the comparison of superiorized and unsuperiorized ART and SART, two new figures of merit have been added to the functionality set of SNARK: The Kullback-Leibler distance and the weighted squared distance, the mathematical aspects of both of which have already been described in Chapter 4.4.6 in the context of algorithm termination.

As explained in Chapter 3.1.8, figures of merit are calculated during the analysis stage and

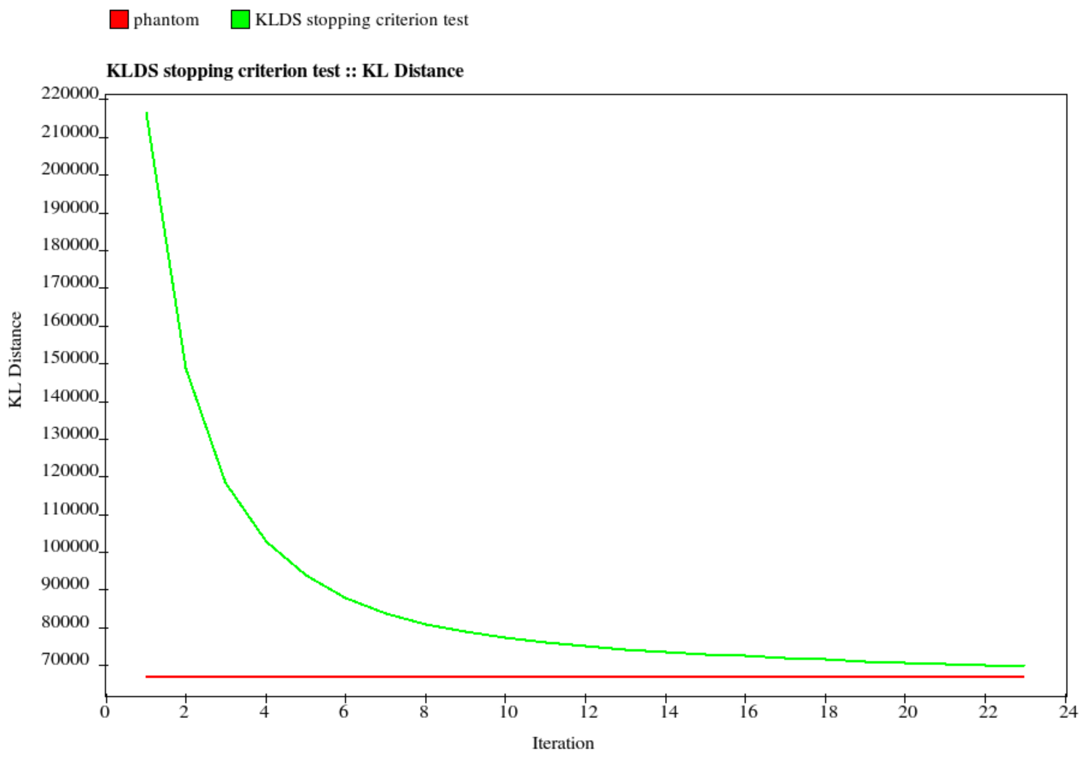


Figure 4.8.: KLDS plot generatd by snark14Display

then written to the evaluation file. They may subsequently be processed by snarkDisplay to generate graphs as well as by the SNARK experimenter in order to perform statistical hypothesis testing. In case of a complete integration, the addition of a FOM therefore involves considerable effort, as it necessitates the modification of several parts of the SNARK package.

4.4.7.1. Kullback-Leibler Distance (Option KLDS)

Besides the application as a termination criterion, the Kullback-Leibler distance may also be considered a figure of merit. While it was not directly used as a FOM in the context of the underlying work, it is expected to be useful in the near future, which is why it has been fully included in SNARK as well as in snarkDisplay. Figure 4.8 shows a plot of the new FOM which was generated by the updated version of snarkDisplay, snark14Display.

4.4.7.2. Weighted Squared Distance (Option WSQD)

As is the case with the corresponding termination criterion, the weighted squared distance has been implemented to be included in the evaluation file by Bernhard Prommegger in the scope of [23].

4.4.8. The Class Superior

Adhering to both the prior considerations and to the methodology with which other commands have been implemented in SNARK, the implementation has been carried out by introducing a new class `Superior`. It essentially contains 2 methods: `initSuperiorization()`, which is invoked by `snark.cpp` to parse and store the SUPERIORIZE command and `executeSuperiorization()`, which is repeatedly called from within `exalg.cpp` and houses the core of the Superiorization algorithm. The code which corresponds to one iteration of the adapted version of Superiorization shown in Algorithm 4.1 is depicted in Listing 4.1.

```
int l;
if (count==1) SuperSet.l = l = -1;
else if (SuperSet.altL==0) l = SuperSet.l;
else if (SuperSet.altL==1) l = count-2;
else if (SuperSet.altL==2) l = (int)round(Rand()*(SuperSet.l-count)) + count - 1;
int n = 0;
memcpy(xkn, recon, area * sizeof(double));
double phixk=secCrit->Run(recon);
while (n < N) {
  naV->Run(xkn, vkn);
  while (true) {
    l++;
    REAL betakn = b*pow(a, l);
    if (betakn<min) betakn=0;
    for (int i = 0; i < area; i++) z[i] = xkn[i] + (betakn * vkn[i]);
    bool inDelta = true;
    if (SuperSet.posZ) for (int i = 0; i < area; i++) if (z[i] < 0) inDelta=false;
    if (inDelta && secCrit->Run(z) <= phixk) {
      n++;
      memcpy(xkn, z, area * sizeof(double));
      break;
    }
  }
}
memcpy(recon, xkn, area * sizeof(double));
SuperSet.l = l;
return algorithm->Run(recon, list, weight, count);
```

Listing 4.1: Superiorization Routine in SNARK

4.4.9. Verbosity Options

In order to improve analysis and troubleshooting functionality, all new commands and options support the output of their most important internal variable values during each iteration by specifying the option RPRT flag at the end. Specifying an additional value for n will cause them to only report on the first, last and n-th iteration.

Output is provided both on the command line as well as in tabular form in separate text files. In the case of SUPERIORIZE, this functionality is especially useful when observing the "conflict of interest" between the original algorithm and the secondary optimization criterion, which typically converge towards different points. Note that Superiorization may theoretically also be used to accelerate a reconstruction process if the secondary optimiztion criterion converges towards the same point. It was through this additional output that the phenomenon of the high incrementation of the variable l during some itreations was observed. In the case of proximity function based stopping criteria, the verbosity option may further be used to estimate the remaining amount of iterations or to assess whether a SNARK run is likely to terminate at all, given a certain ε.

Chapter 5
Verification and Application

This chapter covers the verification process which followed the implementation. Confirmation of the correctness of the implementation was achieved in two ways: Firstly, by an extended re-creation of the experiments performed in [12] and secondly by using the SNARK experimenter to conduct a comprehensive comparison of unsuperiorized and superiorized ART and SART by means of statistical hypothesis testing. The latter was performed by Bernhard Prommegger in the scope of [23].

5.1. Recreating Previous Results

Since [12] represents the basis of some of the new functionality which has been integrated into SNARK14, a logical approach towards verifying the correctness of the implementation was to perform similar experiments and to compare the results. First, using the same brain phantom - which is part of the SNARK package - baseline values for the Kullback-Leibler distance as well as the ϕ values for total variation and smoothness were generated using MLEM as base algorithm and MLEM-STOP as a stopping criterion. The resuls are given in Table 5.1. Subsequently, the calculated value of the Kullback-Leibler distance was used as stopping criterion for the superiorized versions of the MLEM algorithm. As in the original paper, TVAR and SMOO were used as secondary optimization criteria and the values of N and a were set to 8/16/32 and 0.99/0.995/0.999, respectively. The results, which are listed in the columns "Standard" of Table 5.2 differ less than 2‰ from those given in [12], which may be considered limited proof for the correctness of the implementation.

The reason for the slightly different results compared to the original experiment is the fact that the experiments were performed on different operating systems. SNARK is known to produce slightly different results in different environments, a circumstance which is attributed to different random number sequences, different precision levels and/or different third-party

KLDS	22317.9
ϕ (TVAR)	21268.5
ϕ (SMOO)	946.3

Table 5.1.: Baseline Values Generated Using MLEM-STOP

			ϕ					
			TVAR			SMOO		
N	a	b	Standard	ATL1	ATL2	Standard	ATL1	ATL2
8	0.99	1	6179.64	4679.49	4890.5	19.5956	16.3508	15.5324
16	0.99	1	5085.82	3735.94	3786.78	14.2321	14.9397	12.1796
32	0.99	1	6116.71	3487.09	3550.03	35.0211	14.3115	13.0677
8	0.995	1	5154.46	4565.17	4657.59	15.6031	15.9682	16.6274
16	0.995	1	4001.11	3712.95	3742.29	12.6077	15.1203	15.2137
32	0.995	1	3823.17	3481.67	3505.17	13.2456	14.4345	14.3887
8	0.999	1	4574.98	4481.56	4498.39	16.5771	16.3362	16.1981
16	0.999	1	3741.21	3698.98	3703.54	14.9158	15.0372	15.1018
32	0.999	1	3517.54	3484.77	3487.68	13.2815	14.3964	14.437

Table 5.2.: Numerical Output of the Superiorized MLEM Algorithm

libraries.

Although the given results are merely exemplary and a statistical analysis would need to be performed in order to make a well-founded statement, it may be pointed out that at least for the given experiment, ATL1 (and to almost the same extent, ATL2) appears to produce considerably better and more consistent results for TV. For the most noteworthy case of TV Superiorization with N=32 and a=0.99, a plot of the ϕ values across iterations is given in Figure 5.4. Furthermore, both ATL versions appear to eliminate the outliers present in the smoothness results. It may thus be concluded that in some cases, the alternative l handling does indeed appear to provide advantages beyond the prevention of an undesirably short non-ascending vector.

To provide a visual example of the output, one set of images generated in the experiment is shown in Figures 5.1, 5.2 and 5.3. Figure 5.1 shows the phantom image, while Figure 5.2 shows the algorithm output after the last iteration of MLEM-STOP and superiorized MLEM (N=32, a=0.995, ATL1), respectively. Figure 5.3 shows the same images as in Figure 5.2 after they have been thresholded and contrast stretched to display values below 0.55 as black and values above 0.95 as white.

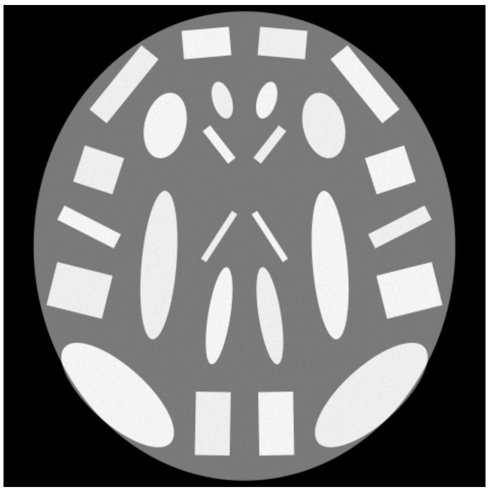


Figure 5.1.: Brain Phantom

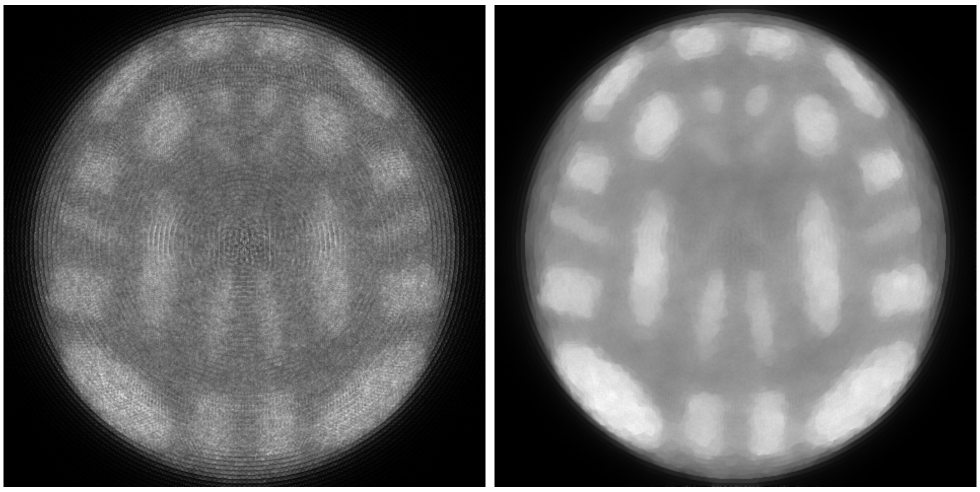


Figure 5.2.: MLEM-STOP vs. Superiorized MLEM

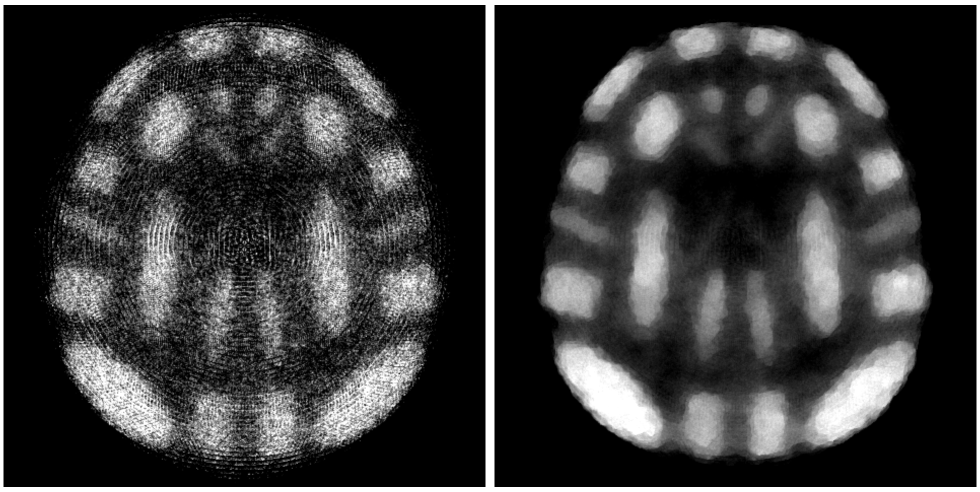


Figure 5.3.: MLEM-STOP vs. Superiorized MLEM, thresholded

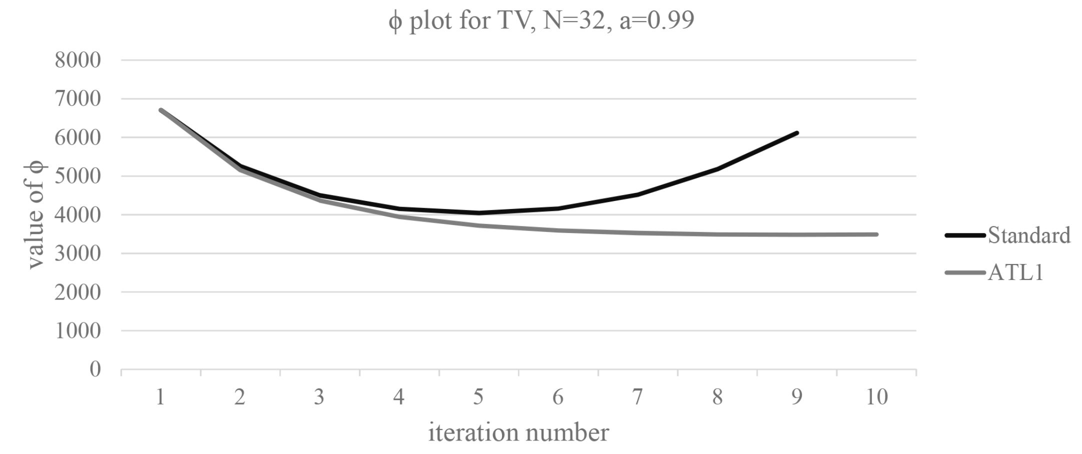


Figure 5.4.: Comparison of ϕ Values Across Iterations

The SNARK input file which was used to perform the corresponding SNARK run is given in Appendix A.3 and the command line output is given in Appendix A.4. The reporting output files are given in Appendix A.5, A.6 and A.7.

5.2. Superiorization of ART and SART

In [23], the new command SUPERIORIZE has been used in conjunction with the SNARK experimenter to conduct a comprehensive comparison of unsuperiorized and TV superiorized ART and SART by means of statistical hypothesis testing, using the FOM *image-wise region of interest* (IROI). In short, statistical hypothesis testing, which is described in greater detail in [14], calculates a significance level based on which the null hypothesis, which is that the observed difference in average values of two reconstruction methods across a series of experiments (in the given case, 30) is a random occurrence, can be accepted or rejected. In this context, the FOM IROI, which is defined in [14] is used to automatically assess the detectability of small, low-contrast tumors in a reconstruction of a cross-section of the human brain, which have been placed randomly across the 30 experiments. Figure 5.5 shows one of the phantoms which have been generated during the experiments.

Prior to a discussion of the results, it is important to point out that the layout of this experiment is fundamentally different from the one described in 5.1. For example, the features in this experiment are much smaller in size (4-9 pixels vs. 350-6000 pixels), which is relevant when applying TV Superiorization, as it results in a smoothing of the reconstruction. Furthermore, the experiment has been performed in a way which aimed at maximizing the

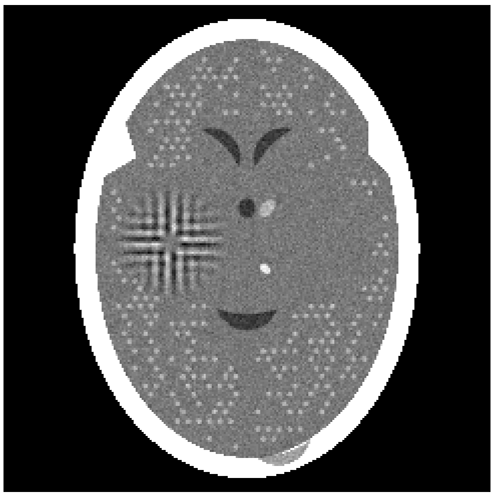


Figure 5.5.: Brain Phantom with Randomly Placed Tumors [23]

IROI rather than minimizing TV, although it is generally based on the assumption that - up to a certain point - lower TV will result in an improved IROI.

Table 5.3 shows the the average values obtained from the 30 experiments, comparing the output of the original ART and SART algorithms on the left side with their 3 superiorized variants on the right side. The last column contains the significance level which indicates whether the observed differences are a random occurrence.

Base Algorithm			**Superiorized Version**			**Significance**
Name	**IROI**	**ϕ (TV)**	**Variant**	**IROI**	**ϕ (TV)**	**Level**
ART	0.15406	497.495	Standard	0.15532	486.841	8.872e-06
			ATL1	0.15362	451.211	0.16163
			ATL2	0.15615	467.743	1.728e-05
SART	0.17542	468.527	Standard	0.17509	466.536	0.02714
			ATL1	0.17449	462.711	3.941e-05
			ATL2	0.17464	463.357	0.00023

Table 5.3.: Comparison of Superiorized and Unsuperiorized ART and SART [23]

The values for N, a and b have been set to 3, 0.4 and 1 for ART and 2, 0.6 and 1 for SART. These relatively low values have been determined to produce the best results for the given setup during preliminary testing, with higher values resulting excessive smoothing, effectively eliminating the low-contrast tumors due to their small size. Figures 5.6 and 5.7 show the output of the original ART algorithm as well as its three superiorized counterparts. Due to the chosen values for N and a, the effect of Superiorization is highly limited, the most noticeable visual difference exists between the unsuperiorized and the ATL1 superiorized image.

To provide an additional method of visualizing the results, a plot of the intensity values

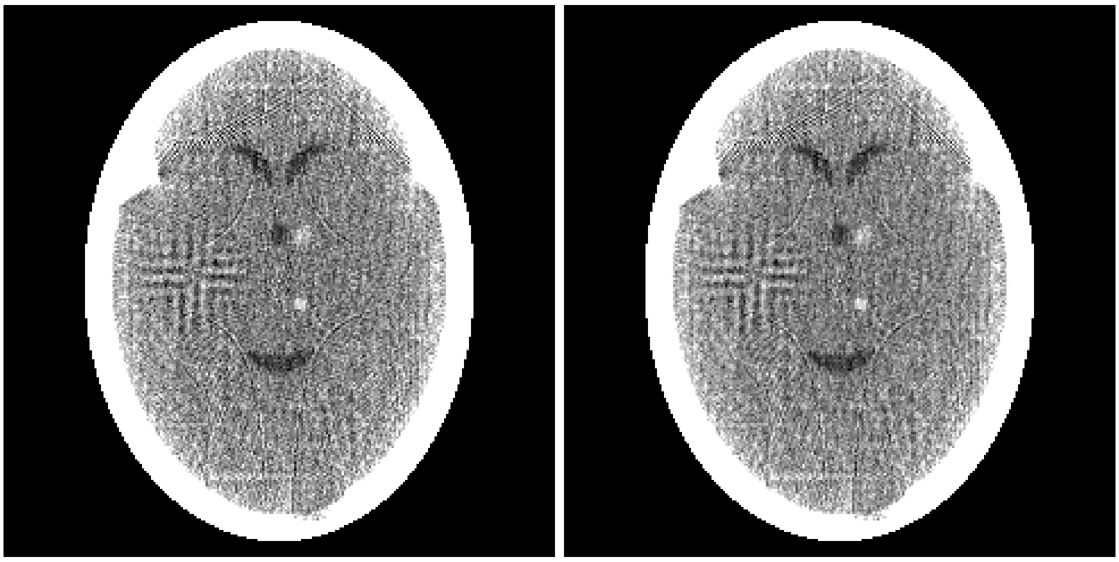


Figure 5.6.: ART vs. Superiorized ART [23]

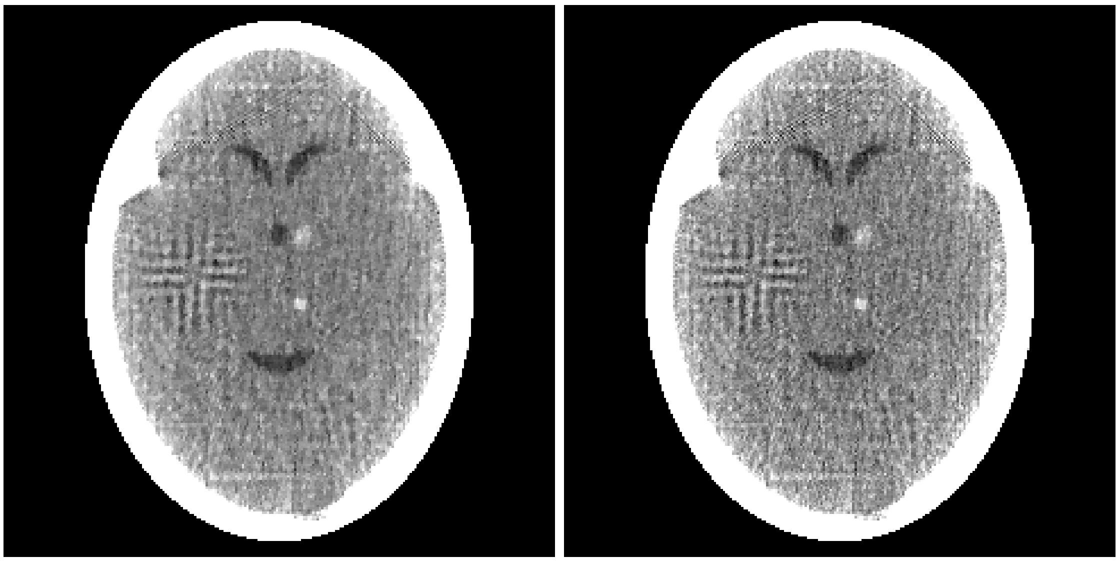


Figure 5.7.: ATL1 vs. ATL2 Superiorized ART [23]

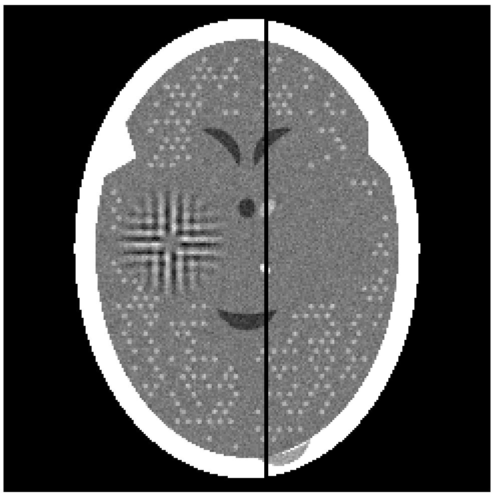


Figure 5.8.: Evaluated Column [23]

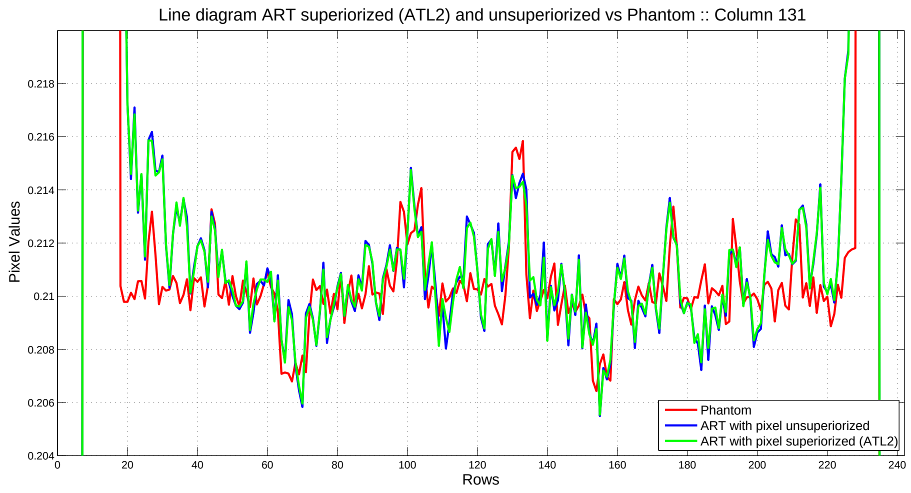


Figure 5.9.: Graph of Column 131 [23]

along column 131 of the reconstructed image is given in Figure 5.9. Figure 5.8 shows its location within the reconstruction as well as the features with which it intersects.

Although TV Superiorization does appear to remove the disturbance patterns which are a result of the geometry of the simulated CT scan, it also tends to decrease the visibility of the desired features. It may thus be concluded that the implementation of the command SUPERIORIZE appears to be functional for ART and SART, as it generally results in smoother output with lower TV. However, for the given setup, with only 2 out of 6 superiorized algorithm variants showing an improved IROI (higher is better) and the best result being produced by the unsuperiorized SART algorithm, TV Superiorization does not appear to contribute towards an improvement in overall visual reconstruction quality.

6 Chapter 6 Conclusion and Outlook

In the scope of this document, it has been shown how the new and innovative methodology of Superiorization has successfully been incorporated into the SNARK software package, which is a testbed for medical image reconstruction algorithms. Following a discussion of the mathematical concept of Superiorization, the underlying software as well as the established development environment has been outlined. Thereafter, the process of adaptation and implementation, including the implications regarding secondary aspects such as stopping criteria and figures of merit has been presented. Lastly, the correctness of the implementation was demonstrated by recreating a previous experiment as well as by discussing the results of a comparison of superiorized and unsuperiorized ART and SART.

The description of the implementation process was written with the intention to make it possible for the reader to gain an impression of how the initially declared goal of implementing Superiorization in a flexible way which can accomodate new algorithms and secondary optimization criteria could be achieved to full extent. In addition, the implementation could be performed in a manner which is consistent with the remainder of the SNARK package, maintaining simplicity and facilitating the use of the new functionality. The intent of encouraging and promoting further research involving Superiorization was also met with success: the usefulness and relevance of the conducted work could already be demonstrated by its subsequent application in performing a comparison between unsuperiorized and superiorized algorithms, the results of which not only provide the basis for a master's thesis, but may further result in the publication of a paper on this topic.

As a side benefit of a careful and observant approach towards the process of implementation, it was possible to refine the underlying mathematical aspect by introducing additional variants of defining the length of the non-ascending vector. Although a general statement towards the usefulness of these variants beyond the avoidance of an undesirably short non-ascending vector may not be made at this point, first results suggest that they may also serve to improve results in certain applications. This functionality may therefore be considered an additional option when gearing Superiorization towards a specific application.

Further work involving Superiorization may be performed in numerous areas. To name just two examples, research is currently being conducted towards a possible application of Superiorization in 3D electron microscopy and which advantages Superiorization may offer for applications which use shearlets as basis functions. Regarding further software packages into which Superiorization may be incorporated, jSNARK[1] and XMIPP[2] are possible candidates.

[1]http://jsnark.sourceforge.net
[2]http://xmipp.cnb.csic.es

Bibliography

[1] Artzy, E., Elfving, T., and Herman, G. T.: *Quadratic optimization for image reconstruction ii.* Computer Graphics and Image Processing, 11:242–261, 1979.

[2] Bian, J., Siewerdsen, J. H., Han, X., Sidky, E. Y., Prince, J. L, Pelizzari, C. A., and Pan, X.: *Evaluation of sparse-view reconstruction from flat-panel-detector cone-beam CT.* Physics in Medicine and Biology, 55:6575–6599, 2010.

[3] Bouallègue, F. B., Crouzet, J. F., and Mariano-Goulart, D.: *A heuristic statistical stopping rule for iterative reconstruction in emission tomography.* Annals of Nuclear Medicine, 27:84–95, 2013.

[4] Butnariu, D., Davidi, R., Herman, G. T., and G.Kazantsev, I.: *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, 1:540–547, 2007.

[5] Censor, Y., Davidi, R., and Herman, G. T.: *Perturbation resilience and superiorization of iterative algorithms.* Inverse Problems, 26, 065008, 2010.

[6] Censor, Y., Davidi, R., Herman, G. T., Schulte, R. W., and Tetruashvili, L.: *Projected subgradient minimization versus superiorization.* Journal of Optimization Theory and Applications, 160:730–747, 2014.

[7] Combettes, P. L. and Luo, J.: *An adaptive level set method for nondifferentiable constrained image recovery.* IEEE Transactions on Image Processing, 11:1295–1304, 2002.

[8] Combettes, P. L. and Pesquet, J. C.: *Image restoration subject to a total variation constraint.* IEEE Transactions on Image Processing, 13:1213–1222, 2004.

[9] Davidi, R., Herman, G. T., Klukowska, J., Langthaler, O., and Prommegger, B.: *SNARK14: A Programming System for the Reconstruction of 2D Images from 1D Projections*, 2014.

[10] Davidi, R., Schulte, R. W., Censor, Y., and Xing, L.: *Fast superiorization using a dual perturbation scheme for proton computed tomography.* Transactions of the American Nuclear Society, 106:73–76, 2012.

[11] Defrise, M., Vanhove, C., and Liu, X.: *An algorithm for total variation regularization in high dimensional linear problems.* Inverse Problems, 27, 065002, 2011.

[12] Garduño, E. and Herman, G. T.: *Superiorization of the ML-EM algorithm.* IEEE Transactions on Nuclear Science, 61(1):162–172, 2014.

[13] Gilbert, P.: *Iterative methods for three-dimensional reconstruction of an object from projections.* Journal of Theoretical Biology, 36:105–117, 1972.

[14] Herman, G. T.: *Fundamentals of Computerized Tomography: Image Reconstruction from Projections.* Springer, 2nd edition, 2009.

[15] Herman, G. T., Garduño, E., Davidi, R., and Censor, Y.: *Superiorization: An optimization heuristic for medical physics.* Medical Physics, 39(9):5532–5546, 2012.

[16] Herman, G. T. and Lent, A.: *Quadratic optimization for image reconstruction part i.* Computer Graphics and Image Processing, 5:319–332, 1976.

[17] Herman, G. T., Pierro, A. R. De, and Gai, N.: *On methods for maximum a posteriori image reconstruction with a normal prior.* Journal of Visual Communication and Image Representation, 3:316–324, 1992.

[18] Jiang, M. and Wang, G.: *Convergence of the simultaneous algebraic reconstruction technique (SART).* IEEE Transactions on Image Processing, 12:957–961, 2003.

[19] Levitan, E. and Herman, G. T.: *A maximum a posteriori probability expectation maximization algorithm for image reconstruction in emission tomography.* IEEE Transactions on Medical Imaging, 6:185–192, 1987.

[20] Neto, E. S. Helou and Pierro, Á. R. De: *Incremental subgradients for constrained convex optimization: A unified framework and new methods.* SIAM Journal on Optimization, 20(3):1547–1572, 2009.

[21] Neto, E. S. Helou and Pierro, Á. R. De: *On perturbed steepest descent methods with inexact line search for bilevel convex optimization.* Optimization, 60(8-9):991–1008, 2011.

[22] Nurminski, E. A.: *Envelope stepsize control for iterative algorithms based on fejer processes with attractants.* Optimization Methods and Software, 25:97–108, 2010.

[23] Prommegger, B.: *Comparison of iterative algorithms with and without superiorization for image reconstruction from projections.* Master's thesis, University of Applied Sciences Salzburg, 2014.

[24] Rowland, S. W.: *Image Reconstruction from Projections: Implementation and Applications*, volume 32 of *Topics in applied physics*, chapter Computer implementation of image reconstruction formulas, pages 9–79. Springer-Verlag, Berlin, 1979.

[25] Schrapp, M. J. and Herman, G. T.: *Data fusion in x-ray computed tomography using a superiorization approach.* Technical Report 85, 053701, Review of Scientific Instruments, 2014.

[26] Shepp, L. A. and Vardi, Y.: *Maximum likelihood reconstruction in positron emission tomography.* IEEE Transactions on Medical Imaging, 1:113–122, 1982.

[27] Sidky, E. Y., Duchin, Y., Pan, X., and Ullberg, C.: *A constrained, total-variation minimization algorithm for low-intensity x-ray CT.* Medical Physics, 38:5117–5125, 2011.

[28] Sidky, E. Y. and Pan, X.: *Image reconstruction in circular cone-beam computed tomography by constrained, total-variation minimization.* Physics in Medicine and Biology, 53:4777–4807, 2008.

A Appendix A

Appendix

A.1. SUPERIORIZE Command in EBNF

```
command = "SUPERIORIZE ", N, " ", a, " ", b , " ", sec_opt_criterion, options;

N = positive_integer;
a = positive_float_less_than_one;
b = positive_float;
sec_opt_criterion = "TVAR" | "SMOO" | "SCR3" | "SCR4" | "SCR5";
options = [" POSI"], [" ", l_handling], [" RPRT", [" ", n]];
l_handling = "ATL1" | "ATL2";
n = positive_integer;
positive_float = positive_float_less_than_one | "1" | float_greater_than_one;
float_greater_than_one = [{digit}], non-zero_digit, [{digit}], ["."], [{digit}];
positive_float_less_than_one = "0.", positive_integer;
positive_integer = [{digit}], non-zero_digit, [{digit}];
digit = "0" | non-zero_digit;
non-zero_digit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
```

Listing A.1: SUPERIORIZE Command in EBNF

A.2. STOP Command in EBNF

```
command = "STOP ", criterion;

criterion = "ITERATION ", iter | "TERMINATION ", condition;
iter = positive_integer;
condition = "VARIANCE ", epsilon |
            "KLDS ", epsilon, [" RPRT", [" ", n]] |
                        "RESI ", epsilon, [" RPRT", [" ", n]] |
            "WSQD ", epsilon, [" RPRT", [" ", n]] |
            "MLST", [" RPRT", [" ", n]] |
            "TRM1", [" ", custom_text] | "TRM2", [" ", custom_text];
epsilon = positive_float;
n = positive_integer;
positive_float = non-zero_digit, [{digit}], ["."], [{digit}] | "0.", positive_integer;
positive_integer = [{digit}], non-zero_digit, [{digit}];
custom_text = [{character}];
character = letter | digit | special_character;
digit = "0" | non-zero_digit;
non-zero_digit = "1" | "2" | "3" | "4" | "5" | "6" | "7" | "8" | "9";
letter = "A" | "B" | "C" | "D" | "E" | "F" | "G" | "H" | "I" | "J" | "K" | "L" |"M" |
         "N" | "O" | "P" | "Q" | "R" | "S" | "T" | "U" | "V" | "W" | "X" | "Y" | "Z";
special_character = " " | "-" | "_" | "." | ",";
```

Listing A.2: STOP Command in EBNF

A.3. SNARK14 Superiorization Input File

```
* SUPERIORIZED MAPEM ALGORITHM FOR PET. RECONSTRUCTION OF A
* BRAIN SIMULATING PET GEOMETRY WITH A RING OF 300 DETECTORS
* WITH EACH DETECTOR IN COINCIDENCE WITH 101 DETECTORS
*
CREATE
PET Brain Phantom
SPECTRUM MONOCHROMATIC 511
OBJECTS
 1 elip   -7.0   46.0   3.0   6.0   17.0  0.95  1.0
 2 elip    7.0   46.0   3.0   6.0  -17.0  1.0   0.95
 3 rect  -12.0   64.0   7.5   4.5    5.0  1.0   1.0
 4 rect   12.0   64.0   7.5   4.5   -5.0  0.95  0.95
 5 rect  -38.0   51.0   3.5  13.0  -39.0  0.95  1.0
 6 rect   38.0   51.0   3.5  13.0   39.0  1.0   0.95
 7 rect  -46.0   24.0   6.5   6.0  -18.0  0.95  1.0
 8 rect   46.0   24.0   6.5   6.0   18.0  1.0   0.95
 9 rect  -49.0    6.0   2.5  10.0   63.0  1.0   1.0
10 rect   49.0    6.0   2.5  10.0  -63.0  0.95  0.95
11 rect  -52.0  -14.0   9.0   7.0  -14.0  0.95  1.0
12 rect   52.0  -14.0   9.0   7.0   14.0  1.0   0.95
13 rect  -10.0  -56.0   5.5  10.0   -1.0  0.95  1.0
14 rect   10.0  -56.0   5.5  10.0    1.0  1.0   0.95
15 elip  -40.0  -47.0   9.0  22.5   48.0  1.0   1.0
```

```
16 elip   40.0  -47.0   9.0  22.5  -48.0  0.95  0.95
17 elip   -8.0  -22.0   3.5  15.5   -9.0  1.0   1.0
18 elip    8.0  -22.0   3.5  15.5    9.0  0.95  0.95
19 elip  -27.0   -6.0   5.5  23.5   -5.0  0.95  1.0
20 elip   27.0   -6.0   5.5  23.5    5.0  1.0   0.95
21 elip  -25.0   38.0   6.5  10.5  -14.0  1.0   1.0
22 elip   25.0   38.0   6.5  10.5   14.0  0.95  0.95
23 rect   -8.0   32.0   1.5   6.5   38.0  1.0   1.0
24 rect    8.0   32.0   1.5   6.5  -38.0  0.95  0.95
25 rect   -8.0    3.0   1.0   9.0  -33.0  0.95  1.0
26 rect    8.0    3.0   1.0   9.0   33.0  1.0   0.95
27 elip    0.0    0.0  66.5  74.0    0.0  1.0   1.0
LAST .51    1   0.01
PHANTOM AVERAGE 11
475 PIXELS OF SIZE 0.32
RAYSUM AVERAGE 11
1 1 1 1 1 1 1 1 1 1 1
GEOMETRY
divergent arc 153 306
RAYS USER 101 DETECTOR SPACING 3.2
ANGLES 300 EQUAL SPACING
0.0 358.8
MEASUREMENT NOISY
QUANTUM 1.0 1.0 CALIBRATION 4
SEED 0
BACKGROUND 0.0
RUN
*
PICTURE TEST
PROJECTION REAL
*
STOP TERMINATION MLST RPRT
EXECUTE AVERAGE EMAP
MLEM-STOP
gamma is 0
*
STOP TERMINATION KLDS 22317.9 RPRT
SUPERIORIZE 32 0.995 1 TVAR ATL1 RPRT
EXECUTE AVERAGE EMAP
Superiorized MLEM
gamma is 0
```

```
END
```

Listing A.3: SNARK14 Superiorization Sample Input file

A.4. SNARK14 Superiorization Output File

Listing A.4 shows the SNARK14 output which corresponds to the input file given in Listing A.3.

```
snark14.s140730 - A PICTURE RECONSTRUCTION PROGRAM



     <*>  SUPERIORIZED MAPEM ALGORITHM FOR PET. RECONSTRUCTION OF A

     <*>  BRAIN SIMULATING PET GEOMETRY WITH A RING OF 300 DETECTORS

     <*>  WITH EACH DETECTOR IN COINCIDENCE WITH 101 DETECTORS

     <*>

     <#> CREATE

         PET Brain Phantom


     <#> SPECTRUM MONOCHROMATIC 511
         energy spectrum is monochromatic at energy level   511


     <#> OBJECTS
         description of objects
                                                                density at levels
 numb type  x-coord  y-coord x-length y-length    angle  av dens       511

    1 elip  -7.0000  46.0000   3.0000   6.0000  17.0000   0.9500    0.9500
    2 elip   7.0000  46.0000   3.0000   6.0000 -17.0000   1.0000    1.0000
    3 rect -12.0000  64.0000   7.5000   4.5000   5.0000   1.0000    1.0000
    4 rect  12.0000  64.0000   7.5000   4.5000  -5.0000   0.9500    0.9500
    5 rect -38.0000  51.0000   3.5000  13.0000 -39.0000   0.9500    0.9500
    6 rect  38.0000  51.0000   3.5000  13.0000  39.0000   1.0000    1.0000
    7 rect -46.0000  24.0000   6.5000   6.0000 -18.0000   0.9500    0.9500
 8 rect  46.0000  24.0000   6.5000   6.0000  18.0000   1.0000    1.0000
 9 rect -49.0000   6.0000   2.5000  10.0000  63.0000   1.0000    1.0000
10 rect  49.0000   6.0000   2.5000  10.0000 -63.0000   0.9500    0.9500
11 rect -52.0000 -14.0000   9.0000   7.0000 -14.0000   0.9500    0.9500
12 rect  52.0000 -14.0000   9.0000   7.0000  14.0000   1.0000    1.0000
13 rect -10.0000 -56.0000   5.5000  10.0000  -1.0000   0.9500    0.9500
14 rect  10.0000 -56.0000   5.5000  10.0000   1.0000   1.0000    1.0000
15 elip -40.0000 -47.0000   9.0000  22.5000  48.0000   1.0000    1.0000
16 elip  40.0000 -47.0000   9.0000  22.5000 -48.0000   0.9500    0.9500
17 elip  -8.0000 -22.0000   3.5000  15.5000  -9.0000   1.0000    1.0000
18 elip   8.0000 -22.0000   3.5000  15.5000   9.0000   0.9500    0.9500
19 elip -27.0000  -6.0000   5.5000  23.5000  -5.0000   0.9500    0.9500
20 elip  27.0000  -6.0000   5.5000  23.5000   5.0000   1.0000    1.0000
21 elip -25.0000  38.0000   6.5000  10.5000 -14.0000   1.0000    1.0000
22 elip  25.0000  38.0000   6.5000  10.5000  14.0000   0.9500    0.9500
23 rect  -8.0000  32.0000   1.5000   6.5000  38.0000   1.0000    1.0000
24 rect   8.0000  32.0000   1.5000   6.5000 -38.0000   0.9500    0.9500
25 rect  -8.0000   3.0000   1.0000   9.0000 -33.0000   0.9500    0.9500
26 rect   8.0000   3.0000   1.0000   9.0000  33.0000   1.0000    1.0000
27 elip   0.0000   0.0000  66.5000  74.0000   0.0000   1.0000    1.0000

      scale factor multiplying object densities     0.5100

      seed set to 1
      inhomogeneity set to     0.0100

  <#> PHANTOM AVERAGE 11

      this run will generate a phantom
      density in each pixel is obtained as the average of 11 x 11 points


  <#> 475 PIXELS OF SIZE 0.32
      picture size 475 x 475,  pixel size     0.3200


  <#> RAYSUM AVERAGE 11

      this run will generate projection data
      projection data are calculated by dividing each ray interval into 11 substrips

```

```
        with aperture (substrip) weights  1  1  1  1  1  1  1  1  1  1  1


    <#> GEOMETRY


    <#> divergent arc 153 306
        rays are divergent from point sources
        source to origin distance    153.0000
        the detectors lie on an arc with source to detector distance =   306.0000


    <#> RAYS USER 101 DETECTOR SPACING 3.2
        number of rays per projection   101
        at detector spacing     3.2000


    <#> ANGLES 300 EQUAL SPACING
        total number of projections   300

                projection angles
          0.0    1.2    2.4    3.6    4.8    6.0    7.2    8.4    9.6   10.8
         12.0   13.2   14.4   15.6   16.8   18.0   19.2   20.4   21.6   22.8
         24.0   25.2   26.4   27.6   28.8   30.0   31.2   32.4   33.6   34.8
         36.0   37.2   38.4   39.6   40.8   42.0   43.2   44.4   45.6   46.8
         48.0   49.2   50.4   51.6   52.8   54.0   55.2   56.4   57.6   58.8
         60.0   61.2   62.4   63.6   64.8   66.0   67.2   68.4   69.6   70.8
         72.0   73.2   74.4   75.6   76.8   78.0   79.2   80.4   81.6   82.8
         84.0   85.2   86.4   87.6   88.8   90.0   91.2   92.4   93.6   94.8
         96.0   97.2   98.4   99.6  100.8  102.0  103.2  104.4  105.6  106.8
        108.0  109.2  110.4  111.6  112.8  114.0  115.2  116.4  117.6  118.8
        120.0  121.2  122.4  123.6  124.8  126.0  127.2  128.4  129.6  130.8
        132.0  133.2  134.4  135.6  136.8  138.0  139.2  140.4  141.6  142.8
        144.0  145.2  146.4  147.6  148.8  150.0  151.2  152.4  153.6  154.8
        156.0  157.2  158.4  159.6  160.8  162.0  163.2  164.4  165.6  166.8
        168.0  169.2  170.4  171.6  172.8  174.0  175.2  176.4  177.6  178.8
        180.0  181.2  182.4  183.6  184.8  186.0  187.2  188.4  189.6  190.8
        192.0  193.2  194.4  195.6  196.8  198.0  199.2  200.4  201.6  202.8
        204.0  205.2  206.4  207.6  208.8  210.0  211.2  212.4  213.6  214.8
        216.0  217.2  218.4  219.6  220.8  222.0  223.2  224.4  225.6  226.8
        228.0  229.2  230.4  231.6  232.8  234.0  235.2  236.4  237.6  238.8
```

```
      240.0  241.2  242.4  243.6  244.8  246.0  247.2  248.4  249.6  250.8
      252.0  253.2  254.4  255.6  256.8  258.0  259.2  260.4  261.6  262.8
      264.0  265.2  266.4  267.6  268.8  270.0  271.2  272.4  273.6  274.8
      276.0  277.2  278.4  279.6  280.8  282.0  283.2  284.4  285.6  286.8
      288.0  289.2  290.4  291.6  292.8  294.0  295.2  296.4  297.6  298.8
      300.0  301.2  302.4  303.6  304.8  306.0  307.2  308.4  309.6  310.8
      312.0  313.2  314.4  315.6  316.8  318.0  319.2  320.4  321.6  322.8
      324.0  325.2  326.4  327.6  328.8  330.0  331.2  332.4  333.6  334.8
      336.0  337.2  338.4  339.6  340.8  342.0  343.2  344.4  345.6  346.8
      348.0  349.2  350.4  351.6  352.8  354.0  355.2  356.4  357.6  358.8


    <#> MEASUREMENT NOISY
        noise characteristics of projection data follow
              nature          characteristics

    <#> QUANTUM 1.0 1.0 CALIBRATION 4
             Emission tomography

    <#> SEED 0
         seed for random number generator is              0


    <#> BACKGROUND 0.0
                              at levels
                                    511
         background absorption   0.0000


    <#> RUN
             1.374 seconds phantom creation
             2.772 seconds projection data creation
             4.145 seconds used for processing command crea


    <*>

    <#> PICTURE TEST

        PET Brain Phantom

```

```

     <#> spec    mono  511
         energy spectrum is monochromatic at energy level   511


     <#> obje
         description of objects
                                                                        density at levels
 numb type  x-coord  y-coord x-length y-length    angle  av dens        511

    1 elip  -7.0000  46.0000   3.0000   6.0000  17.0000   0.4845    0.4845
    2 elip   7.0000  46.0000   3.0000   6.0000 -17.0000   0.5100    0.5100
    3 rect -12.0000  64.0000   7.5000   4.5000   5.0000   0.5100    0.5100
    4 rect  12.0000  64.0000   7.5000   4.5000  -5.0000   0.4845    0.4845
    5 rect -38.0000  51.0000   3.5000  13.0000 -39.0000   0.4845    0.4845
    6 rect  38.0000  51.0000   3.5000  13.0000  39.0000   0.5100    0.5100
    7 rect -46.0000  24.0000   6.5000   6.0000 -18.0000   0.4845    0.4845
    8 rect  46.0000  24.0000   6.5000   6.0000  18.0000   0.5100    0.5100
    9 rect -49.0000   6.0000   2.5000  10.0000  63.0000   0.5100    0.5100
   10 rect  49.0000   6.0000   2.5000  10.0000 -63.0000   0.4845    0.4845
   11 rect -52.0000 -14.0000   9.0000   7.0000 -14.0000   0.4845    0.4845
   12 rect  52.0000 -14.0000   9.0000   7.0000  14.0000   0.5100    0.5100
   13 rect -10.0000 -56.0000   5.5000  10.0000  -1.0000   0.4845    0.4845
   14 rect  10.0000 -56.0000   5.5000  10.0000   1.0000   0.5100    0.5100
   15 elip -40.0000 -47.0000   9.0000  22.5000  48.0000   0.5100    0.5100
   16 elip  40.0000 -47.0000   9.0000  22.5000 -48.0000   0.4845    0.4845
   17 elip  -8.0000 -22.0000   3.5000  15.5000  -9.0000   0.5100    0.5100
   18 elip   8.0000 -22.0000   3.5000  15.5000   9.0000   0.4845    0.4845
   19 elip -27.0000  -6.0000   5.5000  23.5000  -5.0000   0.4845    0.4845
   20 elip  27.0000  -6.0000   5.5000  23.5000   5.0000   0.5100    0.5100
   21 elip -25.0000  38.0000   6.5000  10.5000 -14.0000   0.5100    0.5100
   22 elip  25.0000  38.0000   6.5000  10.5000  14.0000   0.4845    0.4845
   23 rect  -8.0000  32.0000   1.5000   6.5000  38.0000   0.5100    0.5100
   24 rect   8.0000  32.0000   1.5000   6.5000 -38.0000   0.4845    0.4845
   25 rect  -8.0000   3.0000   1.0000   9.0000 -33.0000   0.4845    0.4845
   26 rect   8.0000   3.0000   1.0000   9.0000  33.0000   0.5100    0.5100
   27 elip   0.0000   0.0000  66.5000  74.0000   0.0000   0.5100    0.5100

         scale factor multiplying object densities     0.5100

         seed set to 1
```

```
          inhomogeneity set to     0.0100

      <#> phan    aver   11

          density in each pixel is obtained as the average of 11 x 11 points


      <#> pixe      475    size        0.3200
          picture size 475 x 475,  pixel size     0.3200

          test picture read
          PET Brain Phantom
               0.064 seconds used for processing command pict


      <#> PROJECTION REAL

          PET Brain Phantom


      <#> spec    mono  511
          energy spectrum is monochromatic at energy level   511


      <#> obje
          description of objects
                                                                       density at levels
 numb type  x-coord  y-coord x-length y-length    angle  av dens       511

    1 elip  -7.0000  46.0000   3.0000   6.0000  17.0000   0.4845    0.4845
    2 elip   7.0000  46.0000   3.0000   6.0000 -17.0000   0.5100    0.5100
    3 rect -12.0000  64.0000   7.5000   4.5000   5.0000   0.5100    0.5100
    4 rect  12.0000  64.0000   7.5000   4.5000  -5.0000   0.4845    0.4845
    5 rect -38.0000  51.0000   3.5000  13.0000 -39.0000   0.4845    0.4845
    6 rect  38.0000  51.0000   3.5000  13.0000  39.0000   0.5100    0.5100
    7 rect -46.0000  24.0000   6.5000   6.0000 -18.0000   0.4845    0.4845
    8 rect  46.0000  24.0000   6.5000   6.0000  18.0000   0.5100    0.5100
    9 rect -49.0000   6.0000   2.5000  10.0000  63.0000   0.5100    0.5100
   10 rect  49.0000   6.0000   2.5000  10.0000 -63.0000   0.4845    0.4845
   11 rect -52.0000 -14.0000   9.0000   7.0000 -14.0000   0.4845    0.4845
   12 rect  52.0000 -14.0000   9.0000   7.0000  14.0000   0.5100    0.5100
 13 rect -10.0000 -56.0000   5.5000  10.0000  -1.0000   0.4845    0.4845
 14 rect  10.0000 -56.0000   5.5000  10.0000   1.0000   0.5100    0.5100
 15 elip -40.0000 -47.0000   9.0000  22.5000  48.0000   0.5100    0.5100
 16 elip  40.0000 -47.0000   9.0000  22.5000 -48.0000   0.4845    0.4845
 17 elip  -8.0000 -22.0000   3.5000  15.5000  -9.0000   0.5100    0.5100
 18 elip   8.0000 -22.0000   3.5000  15.5000   9.0000   0.4845    0.4845
 19 elip -27.0000  -6.0000   5.5000  23.5000  -5.0000   0.4845    0.4845
 20 elip  27.0000  -6.0000   5.5000  23.5000   5.0000   0.5100    0.5100
 21 elip -25.0000  38.0000   6.5000  10.5000 -14.0000   0.5100    0.5100
 22 elip  25.0000  38.0000   6.5000  10.5000  14.0000   0.4845    0.4845
 23 rect  -8.0000  32.0000   1.5000   6.5000  38.0000   0.5100    0.5100
 24 rect   8.0000  32.0000   1.5000   6.5000 -38.0000   0.4845    0.4845
 25 rect  -8.0000   3.0000   1.0000   9.0000 -33.0000   0.4845    0.4845
 26 rect   8.0000   3.0000   1.0000   9.0000  33.0000   0.5100    0.5100
 27 elip   0.0000   0.0000  66.5000  74.0000   0.0000   0.5100    0.5100

       scale factor multiplying object densities     0.5100

       seed set to 1
       inhomogeneity set to     0.0100

   <#> rays    aver   11

       projection data are calculated by dividing each ray interval into 11 substrips

       with aperture (substrip) weights  1  1  1  1  1  1  1  1  1  1  1


   <#> geom


   <#> dive    arc     source at  153.0000     det dist  306.0000
       rays are divergent from point sources
       source to origin distance    153.0000
       the detectors lie on an arc with source to detector distance =   306.0000


   <#> rays    user      101    spacing        3.2000
       number of rays per projection   101
       snark computed number of rays   151
       at detector spacing     3.2000


 <#> angl      300
     total number of projections   300


     projection angles
       0.0    1.2    2.4    3.6    4.8    6.0    7.2    8.4    9.6   10.8
      12.0   13.2   14.4   15.6   16.8   18.0   19.2   20.4   21.6   22.8
      24.0   25.2   26.4   27.6   28.8   30.0   31.2   32.4   33.6   34.8
      36.0   37.2   38.4   39.6   40.8   42.0   43.2   44.4   45.6   46.8
      48.0   49.2   50.4   51.6   52.8   54.0   55.2   56.4   57.6   58.8
      60.0   61.2   62.4   63.6   64.8   66.0   67.2   68.4   69.6   70.8
      72.0   73.2   74.4   75.6   76.8   78.0   79.2   80.4   81.6   82.8
      84.0   85.2   86.4   87.6   88.8   90.0   91.2   92.4   93.6   94.8
      96.0   97.2   98.4   99.6  100.8  102.0  103.2  104.4  105.6  106.8
     108.0  109.2  110.4  111.6  112.8  114.0  115.2  116.4  117.6  118.8
     120.0  121.2  122.4  123.6  124.8  126.0  127.2  128.4  129.6  130.8
     132.0  133.2  134.4  135.6  136.8  138.0  139.2  140.4  141.6  142.8
     144.0  145.2  146.4  147.6  148.8  150.0  151.2  152.4  153.6  154.8
     156.0  157.2  158.4  159.6  160.8  162.0  163.2  164.4  165.6  166.8
     168.0  169.2  170.4  171.6  172.8  174.0  175.2  176.4  177.6  178.8
     180.0  181.2  182.4  183.6  184.8  186.0  187.2  188.4  189.6  190.8
     192.0  193.2  194.4  195.6  196.8  198.0  199.2  200.4  201.6  202.8
     204.0  205.2  206.4  207.6  208.8  210.0  211.2  212.4  213.6  214.8
     216.0  217.2  218.4  219.6  220.8  222.0  223.2  224.4  225.6  226.8
     228.0  229.2  230.4  231.6  232.8  234.0  235.2  236.4  237.6  238.8
     240.0  241.2  242.4  243.6  244.8  246.0  247.2  248.4  249.6  250.8
     252.0  253.2  254.4  255.6  256.8  258.0  259.2  260.4  261.6  262.8
     264.0  265.2  266.4  267.6  268.8  270.0  271.2  272.4  273.6  274.8
     276.0  277.2  278.4  279.6  280.8  282.0  283.2  284.4  285.6  286.8
     288.0  289.2  290.4  291.6  292.8  294.0  295.2  296.4  297.6  298.8
     300.0  301.2  302.4  303.6  304.8  306.0  307.2  308.4  309.6  310.8
     312.0  313.2  314.4  315.6  316.8  318.0  319.2  320.4  321.6  322.8
     324.0  325.2  326.4  327.6  328.8  330.0  331.2  332.4  333.6  334.8
     336.0  337.2  338.4  339.6  340.8  342.0  343.2  344.4  345.6  346.8
     348.0  349.2  350.4  351.6  352.8  354.0  355.2  356.4  357.6  358.8

 <#> meas    nois
     noise characteristics of projection data follow
           nature          characteristics
```

```

   <#> quan            1.0000            1.0000    cali  4
            Emission tomography

   <#> seed       0
        seed for random number generator is              0


   <#> back      0.0000
                             at levels
                                   511
        background absorption   0.0000

       estimate of totlen =   4263169.769963
       estimate of totden =   2022892.000000
       estimate of average density =     0.4745
       projection data read
       PET Brain Phantom
            0.012 seconds used for processing command proj


   <*>

   <#> STOP TERMINATION MLST RPRT
       termination test mlst
       reporting is enabled
       reporting file: RPRTmlst
       reporting on every iteration
            0.000 seconds used for processing command stop


   <#> EXECUTE AVERAGE EMAP

       MLEM-STOP

   <#> gamma is 0

-----------------------------------------------------------

 maximum a-posteriori probability expectation maximization

```

```
          gamma:   0.000
          evaluation flag is not set

-----------------------------------------------------------

       algorithm executed in iteration    1
            0.671 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=3.89397
       iteration    1 completed
            1.308 seconds for this iteration
       algorithm executed in iteration    2
            0.402 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=2.34063
       iteration    2 completed
            0.906 seconds for this iteration
       algorithm executed in iteration    3
            0.398 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=1.69124
       iteration    3 completed
            0.869 seconds for this iteration
       algorithm executed in iteration    4
            0.400 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=1.37882
       iteration    4 completed
            0.906 seconds for this iteration
       algorithm executed in iteration    5
            0.398 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=1.20302
       iteration    5 completed
            0.869 seconds for this iteration
       algorithm executed in iteration    6
            0.401 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=1.09042
       iteration    6 completed
            0.876 seconds for this iteration
       algorithm executed in iteration    7
            0.409 seconds for the execution of the algorithm
       MLEM-STOP indicator function I(x)=1.01143
       iteration    7 completed
            0.913 seconds for this iteration
       algorithm executed in iteration    8
```

```
                 0.394 seconds for the execution of the algorithm
            MLEM-STOP indicator function I(x)=0.952455
            reconstruction completed after iteration    8
                 0.911 seconds for this iteration
                 7.557 seconds for all iterations
                 7.615 seconds used for processing command exec


        <*>

        <#> STOP TERMINATION KLDS 22317.9 RPRT
            termination test klds
            reporting is enabled
            reporting file: RPRTklds
            reporting on every iteration
            epsilon = 22317.9
                 0.000 seconds used for processing command stop


        <#> SUPERIORIZE 32 0.995 1 TVAR ATL1 RPRT
            Superiorization is enabled
            N = 32
            a = 0.995
            b = 1
            secondary criterion: tvar
            alternative l handling 1 is enabled
            reporting is enabled
            reporting file: RPRTsuperiorization
            reporting on every iteration
                 0.000 seconds used for processing command supe


        <#> EXECUTE AVERAGE EMAP

            Superiorized MLEM

        <#> gamma is 0

    -----------------------------------------------------------

     maximum a-posteriori probability expectation maximization
```

```

             gamma:   0.000
             evaluation flag is not set

  ------------------------------------------------------------

               value of l: 31
               value of phi before algorithm operator: 0
               value of phi after algorithm operator:  6707.18
          algorithm executed in iteration    1
               1.149 seconds for the execution of the algorithm
          current epsilon (KL distance) = 114652
          iteration    1 completed
               1.710 seconds for this iteration
               value of l: 32
               value of phi before algorithm operator: 749.921
               value of phi after algorithm operator:  5151.06
          algorithm executed in iteration    2
               1.992 seconds for the execution of the algorithm
          current epsilon (KL distance) = 80776.6
          iteration    2 completed
               2.557 seconds for this iteration
               value of l: 33
               value of phi before algorithm operator: 989.063
               value of phi after algorithm operator:  4360.99
          algorithm executed in iteration    3
               2.001 seconds for the execution of the algorithm
          current epsilon (KL distance) = 60523.2
          iteration    3 completed
               2.614 seconds for this iteration
               value of l: 34
               value of phi before algorithm operator: 1187.72
               value of phi after algorithm operator:  3945.32
          algorithm executed in iteration    4
               1.994 seconds for the execution of the algorithm
          current epsilon (KL distance) = 47545.2
          iteration    4 completed
               2.551 seconds for this iteration
               value of l: 35
               value of phi before algorithm operator: 1355.11
               value of phi after algorithm operator:  3713.48
```

```
algorithm executed in iteration    5
     1.996 seconds for the execution of the algorithm
current epsilon (KL distance) = 38802.6
iteration    5 completed
     2.584 seconds for this iteration
     value of l: 36
     value of phi before algorithm operator: 1499.08
     value of phi after algorithm operator:  3590.35
algorithm executed in iteration    6
     1.993 seconds for the execution of the algorithm
current epsilon (KL distance) = 32710.3
iteration    6 completed
     2.573 seconds for this iteration
     value of l: 37
     value of phi before algorithm operator: 1619.26
     value of phi after algorithm operator:  3522.23
algorithm executed in iteration    7
     2.027 seconds for the execution of the algorithm
current epsilon (KL distance) = 28362.8
iteration    7 completed
     2.599 seconds for this iteration
     value of l: 38
     value of phi before algorithm operator: 1715.84
     value of phi after algorithm operator:  3489.84
algorithm executed in iteration    8
     2.008 seconds for the execution of the algorithm
current epsilon (KL distance) = 25202.8
iteration    8 completed
     2.567 seconds for this iteration
     value of l: 39
     value of phi before algorithm operator: 1798.18
     value of phi after algorithm operator:  3477.95
algorithm executed in iteration    9
     1.992 seconds for the execution of the algorithm
current epsilon (KL distance) = 22867.8
iteration    9 completed
     2.555 seconds for this iteration
     value of l: 40
     value of phi before algorithm operator: 1869.26
     value of phi after algorithm operator:  3481.67
algorithm executed in iteration   10
```

```
             2.009 seconds for the execution of the algorithm
        current epsilon (KL distance) = 21115.4
        reconstruction completed after iteration   10
             2.539 seconds for this iteration
            24.845 seconds for all iterations
            24.904 seconds used for processing command exec

    <#> END
```

Listing A.4: SNARK14 Superiorization Sample Output file

A.5. SNARK14 Superiorization Reporting Output File

```
Superiorization reporting output

    iter         l    phi pre-algorithm      phi post-algorithm
       1        31          0.000000000          6707.180789787
       2        32        749.921260409          5151.059263269
       3        33        989.063049185          4360.991349264
       4        34       1187.716484199          3945.318334612
       5        35       1355.105774700          3713.484551323
       6        36       1499.080846424          3590.349413698
       7        37       1619.263389894          3522.231783623
       8        38       1715.843384092          3489.841097489
       9        39       1798.175901313          3477.946030854
      10        40       1869.264190230          3481.667929164
```

Listing A.5: SNARK14 Superiorization Sample Reporting Output File

A.6. SNARK14 KLDS Reporting Output File

```
Kullback-Leibler distance stopping criterion reporting output

    iter    Kullback-Leibler distance
       1        114652.268956247
       2         80776.630516494
       3         60523.230635785
       4         47545.157451798
       5         38802.595194905
```

```
      6         32710.322011602
      7         28362.835216531
      8         25202.838403171
      9         22867.816715075
     10         21115.387590195
```

Listing A.6: SNARK14 Sample KLDS Reporting Output File

A.7. SNARK14 MLEM-STOP Reporting Output File

```
MLEM-STOP indicator function I(x) reporting output

    iter              I(x)
       1          3.893969199
       2          2.340634531
       3          1.691244617
       4          1.378815104
       5          1.203017996
       6          1.090418350
       7          1.011430281
       8          0.952454964
```

Listing A.7: SNARK14 Sample MLEM-STOP Reporting Output File