PyPeT: A Python Tool for Automated Quantitative Brain CT Perfusion Analysis and Visualization

Marijn Borghouts[1] and Ruisheng Su[1]

[1]Department of Biomedical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
Correspondence: `m.m.borghouts@tue.nl`

Abstract. Computed tomography perfusion (CTP) is widely used in acute ischemic stroke assessment and other cerebrovascular conditions to generate quantitative maps of cerebral hemodynamics. While commercial CTP analysis software exists, it is often costly, closed source, and lacks customizability. This work introduces a publicly available Python tool for head CTP processing, capable of producing cerebral blood flow (CBF), cerebral blood volume (CBV), time-to-peak (TTP), and time-to-maximum (Tmax) maps from raw four-dimensional CTP data. This tool is termed **PyPeT** for **Py**thon **Pe**rfusion **T**ool. The pipeline includes image preprocessing, brain masking, arterial input function (AIF) detection, and delay-insensitive singular value decomposition (bSVD)–based deconvolution. Validation is performed via interactive step-by-step visualization and visual comparison with maps from the FDA-approved Icobrain CVA software using the ISLES2024 dataset. Since the reference software uses proprietary post-processing, direct quantitative comparison is limited. Nevertheless, our tool offers a transparent and extensible platform for CTP research. The code is openly available at our GitHub `https://github.com/Marijn311/CT-Perfusion-Tool`

Keywords: CT Perfusion · Acute Ischemic Stroke · Perfusion Analysis · Publicly Available

1 Introduction

Head computed tomography perfusion (CTP) is a valuable imaging technique in which a contrast agent is injected into the cerebral circulation, and the brain is continuously imaged using CT as the agent passes through. By applying deconvolution algorithms within a dedicated processing pipeline, perfusion maps can be generated to visualize cerebral hemodynamics. CTP is most commonly employed in the acute management of ischemic stroke, but it also has applications in other cerebrovascular conditions, including vasospasms, brain tumors, and head trauma.

Several FDA-approved commercial software solutions are available for CTP analysis (e.g., Icobrain CVA, Viz CTP, iSchemaView RAPID [2]). While these tools are effective, they are often expensive, closed-source, and limited in flexibility, making them less suitable for research. Consequently, there is a need for open alternatives that allow for customization and reproducibility.

One such open, deconvolution-based codebase for head CTP is available at `https://github.com/turtleizzy/ctp_csvd`. Another open implementation can be found in [3], which focuses on deconvolution-based perfusion analysis of DSC-MRI data in animal models. Alternative open approaches to CTP processing have also been explored: [1] introduced a fast nonlinear regression method to improve over block-circulant singular value decomposition (bSVD) for deconvolution, while recent machine learning approaches employ physics-informed neural networks [6] or U-Net architectures [4] to generate perfusion maps directly from CTP data.

This work presents an open deconvolution-based Python tool for CTP analysis built upon the work by `https://github.com/turtleizzy/ctp_csvd`. This short report serves as both a technical overview of the CTP processing pipeline and a guide for researchers wishing to reproduce or validate its output.

2 Methodology

2.1 CTP Processing

We based our implementation on the publicly available codebase at `https://github.com/turtleizzy/ctp_csvd`, which provided a foundation for head CT perfusion processing. The code was refactored to improve organization, readability, and maintainability, with extensive inline comments and detailed docstrings added to facilitate adaptation by other researchers. The visualization component was extended through improved interactive viewers accessible via a debug mode, enabling stepwise evaluation of intermediate outputs. Methodological improvements included an automated procedure for determining optimal thresholds in arterial input function detection and functionality for both visual and quantitative comparison of generated perfusion maps with user-defined reference maps. Lastly, this report was added to document the tool, presents initial validation results, and outlines current limitations.

The main CTP processing pipeline consists of the following steps. First, the raw four-dimensional CTP images are loaded from `.nii.gz` files and reoriented to a standard RAS+ orientation. The images are then smoothed using the `sitk.CurvatureFlow` function, which reduces noise while preserving edges. This smoothing step helps stabilize the analysis by mitigating noise in temporal measurements. Such noise can arise from small residual patient movements that may remain after motion correction. Next, a brain mask is generated using a combination of intensity thresholding and a fast-marching algorithm to grow the mask from the largest connected components, which should be the skull. Alternatively, users may supply a pre-computed brain mask (with tools such as `https://github.com/aqqush/CT_BET` for example).

Concentration–time curves (CTC) are extracted by tracking voxel intensity over time. For each time point, the total contrast agent signal in the volume is calculated by first segmenting contrast voxels. A voxel is classified as a contrast voxel if its intensity exceeds a predefined threshold in Hounsfield units. The values of all contrast voxels are then summed across the volume to obtain a contrast measure for that time point. These measures are normalized to the first time point to effectively remove the baseline signal. The bolus arrival time is defined as the first time point at which the normalized contrast measure increases by more than 5% relative to the baseline. Using the extracted concentration–time curves, time-to-peak (TTP) maps are generated.

The arterial input function (AIF) was extracted using an adaptive thresholding and model-fitting approach. First, the distribution of the TTP and area under the concentration–time curves (AUC) within the brain mask was computed. A series of progressively relaxed percentile-based thresholds for TTP and AUC were then applied to identify candidate voxels likely to represent the arterial signal. Voxels exceeding the AUC threshold and exhibiting an early TTP were considered initial AIF candidates. Connected component analysis was performed to isolate distinct clusters. For each candidate cluster, the mean concentration–time curve was extracted, and clusters with volumes exceeding a predefined threshold were discarded. The mean curve of each remaining candidate was then fitted with a gamma variate function, and the quality of the fit was evaluated in terms of fitting error, curve shape, and peak-to-end contrast difference. A scoring function combining cluster volume, curve peak characteristics, and mean fitting error was computed, and the candidate with the highest score was selected as the final AIF.

Finally, cerebral blood flow (CBF), cerebral blood volume (CBV), mean-transit-time (MTT), and time-to-maximum (Tmax) maps are computed using a deconvolution with the AIF according to the definitions provided in Figure 1. PyPeT has implemented both standard SVD and Block-circulant SVD based on the work of [7]. Depending on the desired application, additional post-processing, such as smoothing or normalization, can be applied to the produced perfusion maps.

$$\mathbf{R(t) = (1/CBF) \cdot (CTC(t) \oslash AIF(t))}$$

$$\mathbf{TTP = argmax_t(AIF(t))}$$

$$\mathbf{CBF = max_t R(t)}$$

$$\mathbf{CBV = \int R(t)\, dt}$$

$$\mathbf{Tmax = argmax_t R(t)}$$

$$\mathbf{MTT = CBV / CBF}$$

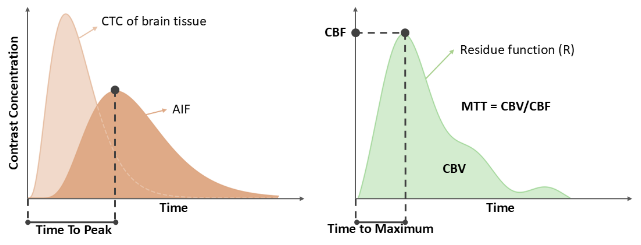

Fig. 1. Time concentration curves (CTC), residue function, and the derived perfusion parameters.

2.2 Interactive Visualizations for Debugging

An interactive debugging mode is available by setting the `DEBUG` flag in `main.py` to `True`. When enabled, the pipeline generates interactive visualizations at each processing stage to help verify that intermediate steps behave as expected. This facilitates early detection of errors before they propagate and affect the final results. The first viewer displays the raw four-dimensional CTP images, both in their original form and as a maximum intensity projection of the central third of slices. Users can inspect the middle brain slice and scroll through the temporal frames (Figure 2). An equivalent visualization is provided after application of the temporal smoothing step. Next, the generated three-dimensional brain mask and corresponding segmentation are presented in an interactive viewer, allowing slice-by-slice inspection (Figure 3). The extracted concentration–time curves of the contrast signal are then displayed for quality control (Figure 4). Finally, the slice containing the largest number of AIF cluster voxels is shown, with AIF voxels highlighted in purple. This viewer also presents the corresponding time–concentration image, the averaged AIF curve, and its fitted gamma variate model. Temporal scrolling is supported for detailed inspection of curve dynamics (Figure 5).

2.3 Technical Details

PyPeT was developed using Python 3.10 and relies on widely used packages such as NumPy, SciPy, scikit, and matplotlib (see requirements.txt on GitHub). The code was developed on Windows 11 but should run on most systems without requiring a GPU or high-performance CPU. Its modular design organizes the main pipeline into self-contained functions, such as preprocessing, brain masking, arterial input function detection, and deconvolution, allowing individual components to be reused or adapted. The GitHub repository includes one demo CTP image with dimensions $220 \times 220 \times 16 \times 55$ (in-plane x, in-plane y, number of slices, number of time points). Processing this image takes about 20 seconds on an Intel i5 processor, of which 15 seconds are spent on smoothing the 55 three-dimensional volumes. This makes smoothing the primary target for potential speed improvements.

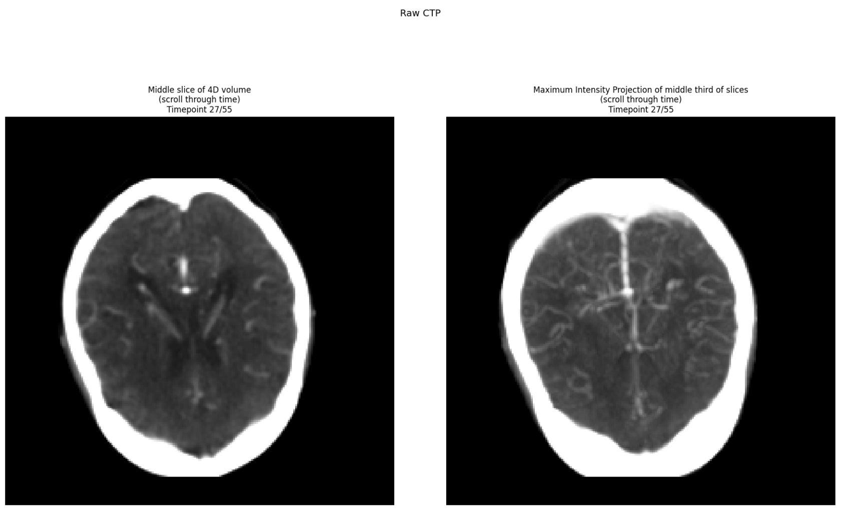
Raw CTP
Middle slice of 4D volume
(scroll through time)
Timepoint 27/55
Maximum Intensity Projection of middle third of slices
(scroll through time)
Timepoint 27/55

Fig. 2. Snapshot of the interactive viewer showing the raw 4D input CTP image with temporal scrolling. Left: Middle slice. Right: maximum intensity projection of the middle third of slices.

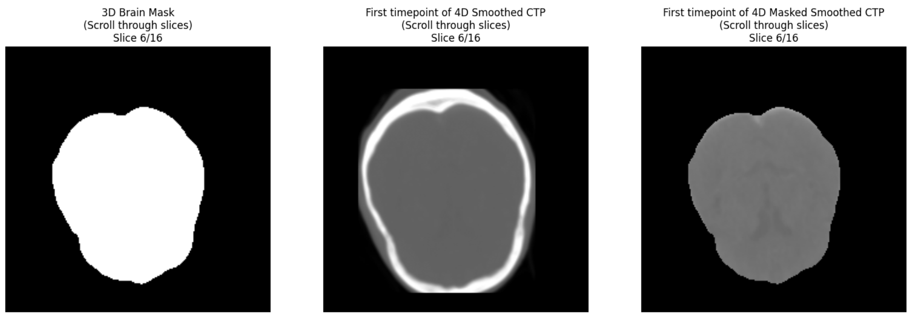

Fig. 3. Snapshot of the interactive viewer showing the results of mask generation with spatial scrolling. Left: A slice of the binary mask. Middle: Corresponding slice in the unmasked images. Right: Corresponding slice in the masked image.

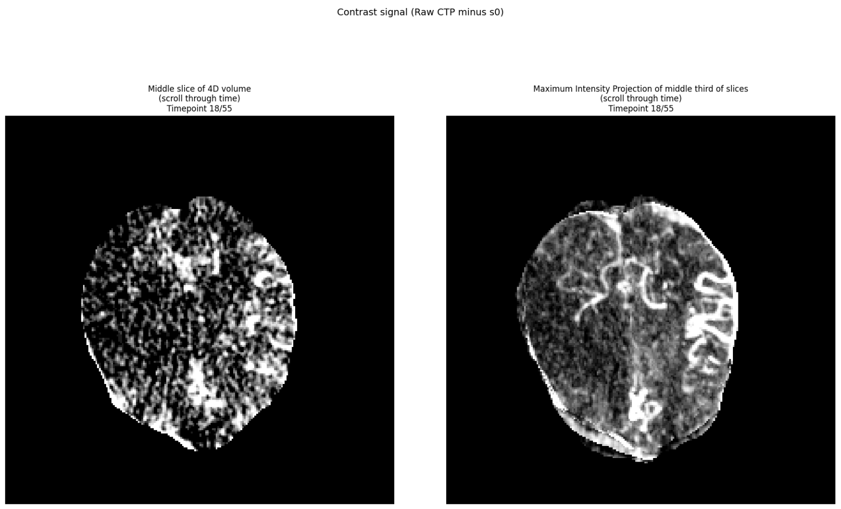
Contrast signal (Raw CTP minus s0)
Middle slice of 4D volume
(scroll through time)
Timepoint 18/55
Maximum Intensity Projection of middle third of slices
(scroll through time)
Timepoint 18/55

Fig. 4. Snapshot of the interactive viewer showing the concentration–time curve images, with temporal scrolling.

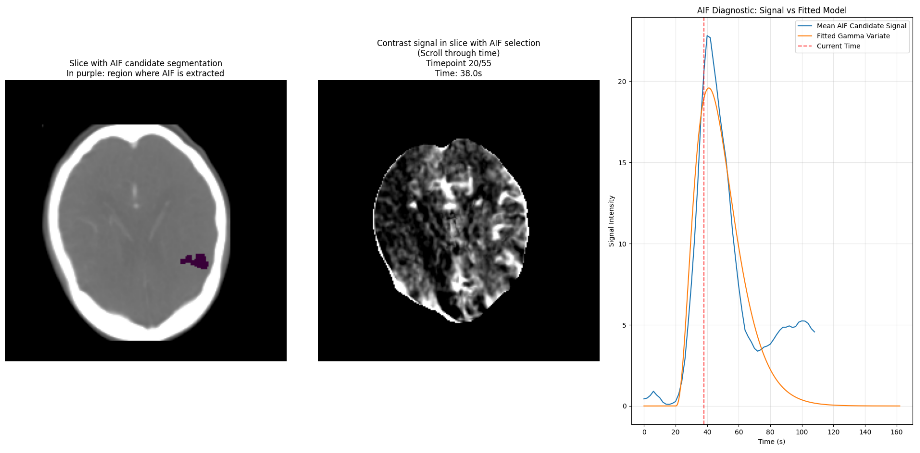
Slice with AIF candidate segmentation
In purple: region where AIF is extracted
Contrast signal in slice with AIF selection
(Scroll through time)
Timepoint 20/55
Time: 38.0s
AIF Diagnostic: Signal vs Fitted Model
Mean AIF Candidate Signal
Fitted Gamma Variate
Current Time
Signal Intensity
Time (s)
0
5
10
15
20
0
20
40
60
80
100
120
140
160

Fig. 5. Slice with the highest number of arterial input function (AIF) voxels highlighted in purple. The corresponding slice of the CTC image, and the averaged AIF signal, and its gamma-variate fit.

3 Validation Experiment

To evaluate the performance of PyPeT, perfusion maps generated by our pipeline were compared with those produced by the commercially available, FDA-approved software *Icobrain CVA*. For this purpose, we utilized the publicly available ISLES2024 dataset [5], originally created for the ischemic stroke lesion segmentation challenge at MICCAI 2024. The dataset provides both raw four-dimensional CTP images and the corresponding perfusion maps generated by *Icobrain CVA* for 149 training and 96 test patients. The developed tool allows both direct visual comparison of the resulting perfusion maps and the computation of quantitative similarity metrics (normalized cross-correlation, structural similarity index, mean absolute error, mean squared error, and root mean squared error). Visual and quantitative comparisons for the demo CTP image are shown in Figure 6 and Table 1 respectively.

4 Results

Table 1. Quantitative comparison of perfusion maps from the provided CTP demo image.

Metric	**CBF**	**CBV**	**MTT**	**Tmax**
Normalized Cross-Correlation (NCC)	0.64	0.46	0.04	0.40
Structural Similarity Index (SSIM)	0.82	0.75	0.67	0.75
Mean Absolute Error (MAE)	0.51	0.78	0.59	0.70
Mean Squared Error (MSE)	2.13	4.00	1.47	1.91
Root Mean Squared Error (RMSE)	1.46	2.00	1.21	1.38

Our visual assessment indicates that the *Icobrain CVA* software employs proprietary post-processing steps, including adaptive smoothing and additional ventricular segmentation. Despite these differences, the generated perfusion maps largely correspond to those produced by PyPeT. The closed-source nature of Icobrain's post-processing pipeline limits the ability to achieve exact visual correspondence between the two sets of maps. Consequently, the quantitative similarity metrics reported in Table 1 are modest, and should be interpreted with caution. Interestingly, the similarity of the MTT maps is generally lower than that of the CBF and CBV maps across metrics, which is unexpected given that MTT is defined as the ratio of CBV to CBF.

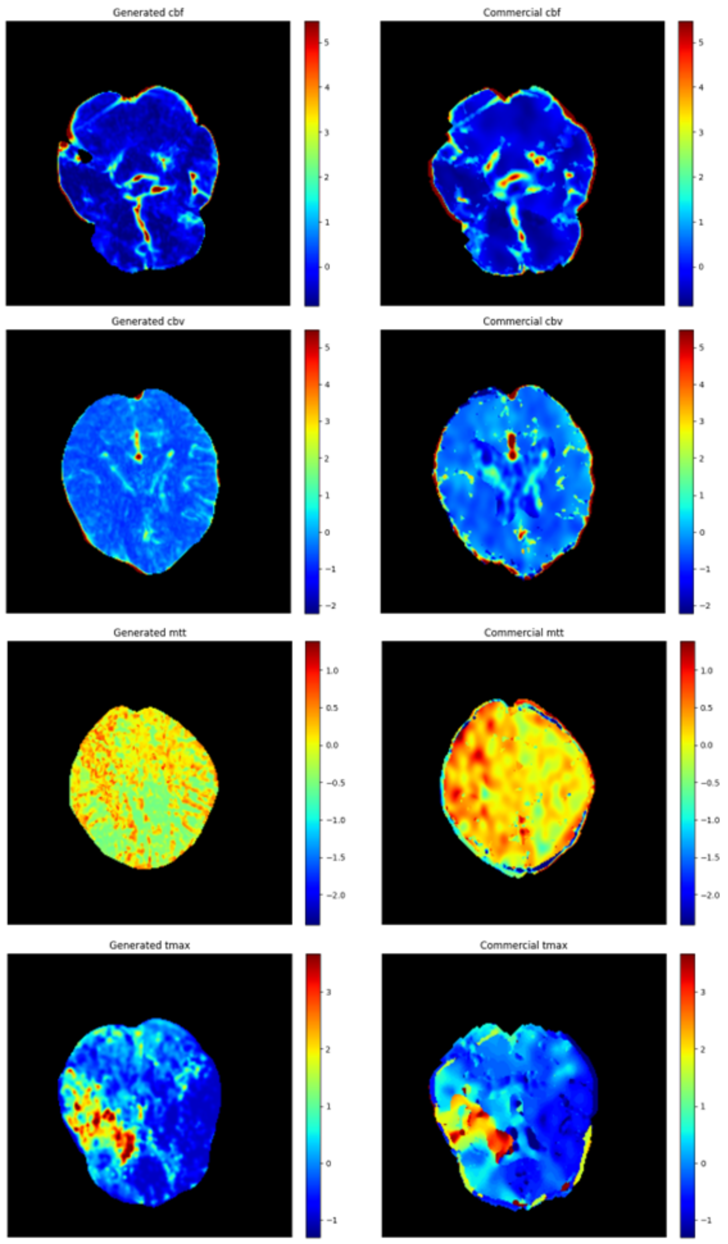

Fig. 6. Visual comparisons of the perfusion maps generated by PyPeT compared to maps generated by Icobrain CVA. Perfusion data comes from the provided CTP demo image.

5 Conclusion and Discussion

This work presents PyPeT, a tool for processing head CTP images into quantitative perfusion maps. This tool is intended solely for research purposes. It has not undergone comprehensive validation and is not designed for clinical use. A significant limitation arises from the unknown post-processing steps applied in the commercial reference maps, which prevents the use of quantitative similarity metrics for a fair and direct comparison between the two methods. Furthermore, PyPeT remains a work in progress. Within the codebase, several `TODO` tags highlight areas for potential improvement. Key avenues for future development include:

1. **Enhanced Brain Masking:** Currently, the brain tissue segmentation appears reasonably accurate; however, the lower slices occasionally include extracranial soft tissue not part of the brain.
2. **Motion Correction Module:** Implementing motion correction could stabilize perfusion map generation for patients exhibiting head motion during image acquisition, thereby improving robustness and reliability.
3. **Improving Absolute Values:** The comparisons presented were performed using normalized perfusion maps. To ensure that non-normalized perfusion maps have accurate absolute values, it is important to tune physiological constants such as the tissue density and the hematocrit correction factor. Secondly, one should revisit the quality of arterial input function selection, and the regularization of the deconvolution process.
4. **Investigation of MTT Discrepancies:** Although CBV and CBF maps closely match those produced by the commercial tool, the MTT maps show greater variation. Since MTT is theoretically the ratio of CBV to CBF, further investigation is needed to understand the post-processing strategies employed by the commercial software and to reconcile these differences.
5. **Additional Validation:** Expanding validation efforts by applying PyPeT to additional CTP datasets would strengthen confidence in its generalizability and performance.

Despite its current limitations, PyPeT lowers the barrier to entry for brain perfusion research and contributes to the growing movement toward reproducible and openly accessible medical imaging analysis.

References

1. Bennink, E., Oosterbroek, J., Kudo, K., Viergever, M.A., Velthuis, B.K., de Jong, H.W.A.M.: Fast nonlinear regression method for ct brain perfusion analysis. Journal of Medical Imaging **3**, 026003 (6 2016). `https://doi.org/10.1117/1.jmi.3.2.026003`
2. Chandrabhatla, A.S., Kuo, E.A., Sokolowski, J.D., Kellogg, R.T., Park, M., Mastorakos, P.: Artificial intelligence and machine learning in the diagnosis and management of stroke: A narrative review of united states food and drug administration-approved technologies. Journal of Clinical Medicine **12** (6 2023). `https://doi.org/10.3390/jcm12113755`

3. Fernández-Rodicio, S., Ferro-Costas, G., Sampedro-Viana, A., Bazarra-Barreiros, M., Ferreirós, A., López-Arias, E., Pérez-Mato, M., Ouro, A., Pumar, J.M., Mosqueira, A.J., Alonso-Alonso, M.L., Castillo, J., Hervella, P., Iglesias-Rey, R.: Perfusion-weighted software written in python for dsc-mri analysis. Frontiers in Neuroinformatics **17** (2023). `https://doi.org/10.3389/fninf.2023.1202156`
4. Gava, U.A., D'Agata, F., Tartaglione, E., Renzulli, R., Grangetto, M., Bertolino, F., Santonocito, A., Bennink, E., Vaudano, G., Boghi, A., Bergui, M.: Neural network-derived perfusion maps: A model-free approach to computed tomography perfusion in patients with acute ischemic stroke. Frontiers in Neuroinformatics **17** (2023). `https://doi.org/10.3389/fninf.2023.852105`
5. Riedel, A.E.O., de la Rosa, E., Baran, T.A., Petzsche, M.H., Baazaoui, H., Yang, K., Musio, F.A., Huang, H., Robben, D., Seia, J.O., Wiest, R., Reyes, M., Su, R., Zimmer, C., Boeckh-Behrens, T., Berndt, M., Menze, B., Rueckert, D., Wiestler, B., Wegener, S., Kirschke, J.S., Riedel, E.O.: Isles'24-a real-world longitudinal multimodal stroke dataset. arXiv (2024). `https://doi.org/10.48550/arXiv.2408.11142`
6. de Vries, L., van Herten, R.L., Hoving, J.W., Išgum, I., Emmer, B.J., Majoie, C.B., Marquering, H.A., Gavves, E.: Spatio-temporal physics-informed learning: A novel approach to ct perfusion analysis in acute ischemic stroke. Medical Image Analysis **90** (12 2023). `https://doi.org/10.1016/j.media.2023.102971`
7. Zanderigo, F., Bertoldo, A., Pillonetto, G., Cobelli, C.: Nonlinear stochastic regularization to characterize tissue residue function in bolus-tracking mri: Assessment and comparison with svd, block-circulant svd, and tikhonov. IEEE Transactions on Biomedical Engineering **56**, 1287–1297 (5 2009). `https://doi.org/10.1109/TBME.2009.2013820`