2024 IEEE NSS MIC RTSD
26 OCT – 2 NOV 2024
TAMPA , FLORIDA, USA

NPL National Physical Laboratory UCL

Fast regularised PET reconstruction with stochastic averaging and an Armijo line search

Sam Porter[1,2], Edoardo Pasca[3], Margaret Duff[3]

1 Institute of Nuclear Medicine, University College London
2 National Physical Laboratory
3 Science and Technology Research Council, UKRI

Overview

Motivation: demonstrate the performance of out-the-box CIL stochastic algorithms with some small additions to account for differences in data

Two algorithms investigated:

- Stochastic Variance Reduced Gradient (SVRG)
- Stochastic averaged gradient augmentation (SAGA)

SAGA

- Warm started with 5 full gradient steps using preconditioned Armijo step sizes
- Initial step size chose to be minimum of first 5 step sizes
- Step size reduced with $\frac{\gamma}{1+\delta k}$, δ chosen to ensure step size decay of 10% per epoch
- BSREM-type preconditioner
- Subsets selected according to empirical rule wrt. counts per subset. Max subsets equal to a quarter of the total views
- Gradients initialised with a full gradient update

SVRG

- Warm started with 5 full gradient steps using preconditioned Armijo step sizes
- Initial step size chose to be minimum of first 5 step sizes
- Step size reduced with $\frac{\gamma}{1+\delta k}$, δ chosen to ensure step size decay of 10% per epoch
- BSREM-type preconditioner
- Subsets selected according to empirical rule wrt. counts per subset. Max subsets equal to a quarter of the total views
- Full gradient update chosen to be every epoch (mistake!)

Stochastic Optimisation

$$\mathcal{L}(u) = \mathcal{D}(u, f) + \mathcal{R}(u) \qquad \leftarrow \text{minimise}$$

$$u^{(k+1)} = u^{(k)} - \nabla \mathcal{L}(u^{(k)}) \qquad \leftarrow \text{slow}$$

$$\mathcal{L}(u) = \sum_n \mathcal{L}_n(u)$$

$$u^{(k+1)} = u^{(k)} - \nabla \mathcal{L}_{n(k)}(u^{(k)})$$

Variance Reduction

SAGA update

$$g(u^{(k)}) = N\left(\nabla\mathcal{L}_{n(k)}(u^{(k)}) - G_{n(k)}^{(k)}\right) + \sum_n G_n^{(k)}$$

where G_n are stored sub-gradients refreshed with the most recent sub-gradient

SVRG update

$$g(u^{(k)}) = N\left(\nabla\mathcal{L}_{n(k)}(u^{(k)}) - \nabla\mathcal{L}_{n(k)}(\tilde{u})\right) + \sum_n \nabla\mathcal{L}_n(\tilde{u})$$

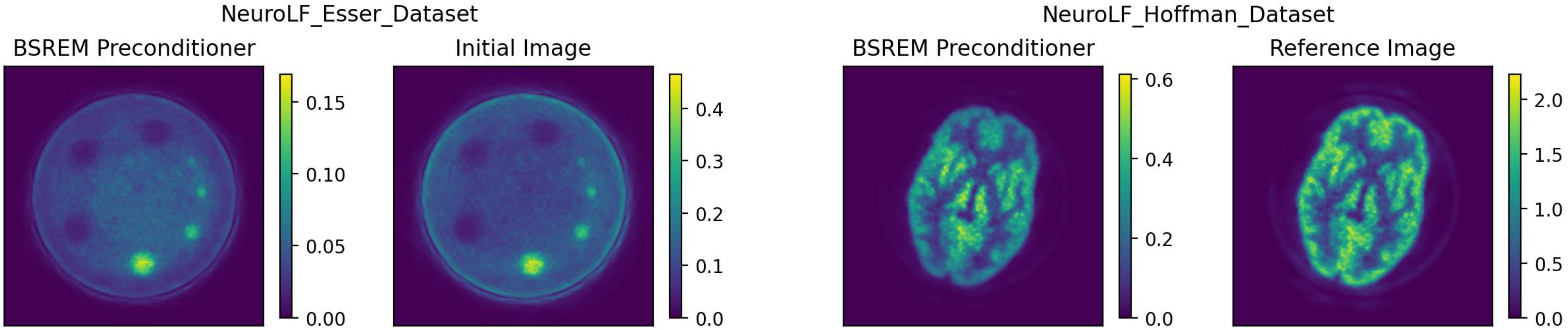
where $\tilde{u}$ is an anchor image with gradients updated periodically

The preconditioner

$$P_k = \frac{x^k}{\sum_n A_n^\top \mathbf{1} + \eta} + \eta$$

Frozen after 10 epochs

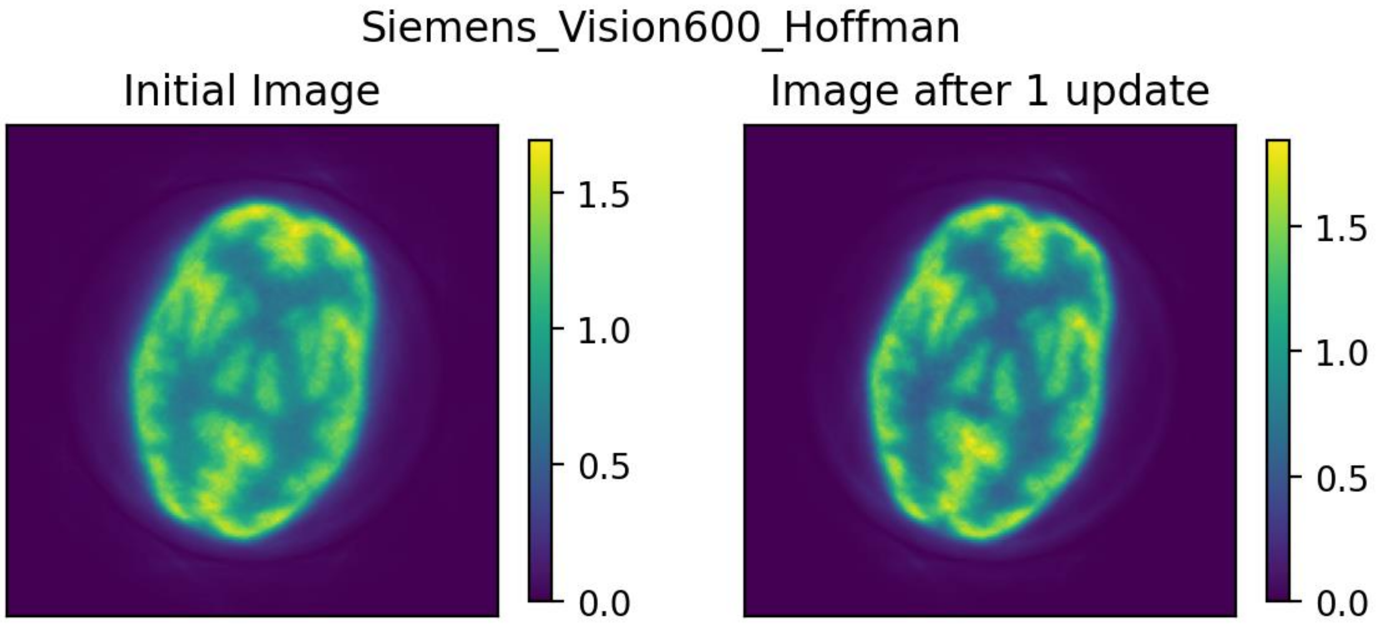
$$\eta = \max(\text{initial image})/10^6$$



Warm Start

Each algorithm was warm started with 5 iterations of gradient descent using an Armijo step size

It was observed that the first few updates could be large, and the optimal step size often decreased dramatically



Armijo step size = 0.4

Armijo step size = 0.1

Subsets

The number of subsets was chosen according to the following:

1. Minimum counts must be 2^{20}
2. The number of subsets cannot be more than a quarter of the number of views
3. The number of subsets cannot be more than 32
4. The subsets must be of equal size

This resulted in subset numbers ranging from 10 to 32

Armijo step size selection

Step 1: Set $u^{(k+1)} = u^{(k)} - \gamma P_k g(u^{(k)})$

Step 2: While the Armijo condition is not satisfied:

$$\mathcal{L}_{n(k)}(u^{(k+1)}) > \mathcal{L}_{n(k)}(u^{(k)}) - \lambda\gamma \left\langle P_k g(u^{(k)}), g(u^{(k)}) \right\rangle$$

- Update $\gamma \leftarrow \sigma\gamma$

Step 3: Update $\gamma \leftarrow \gamma/\sigma$

Step 4: Set $\gamma_k = \gamma$ for the current iteration

End Repeat

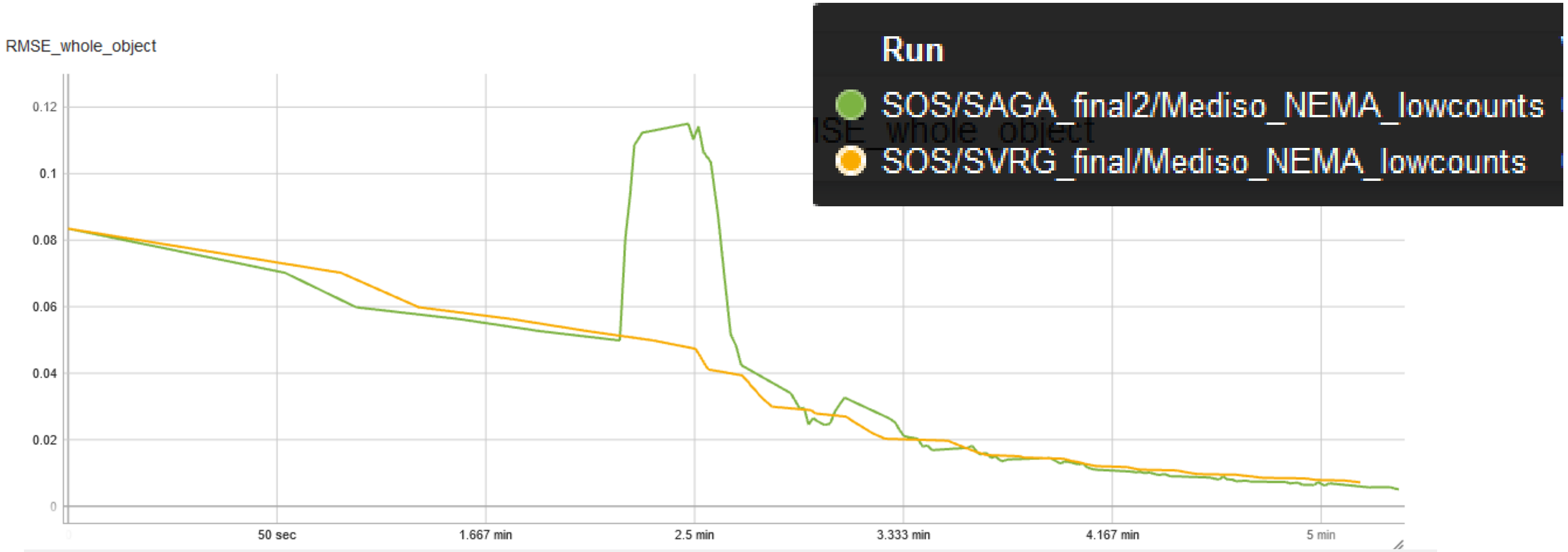
For this work, $\sigma = 0.5$

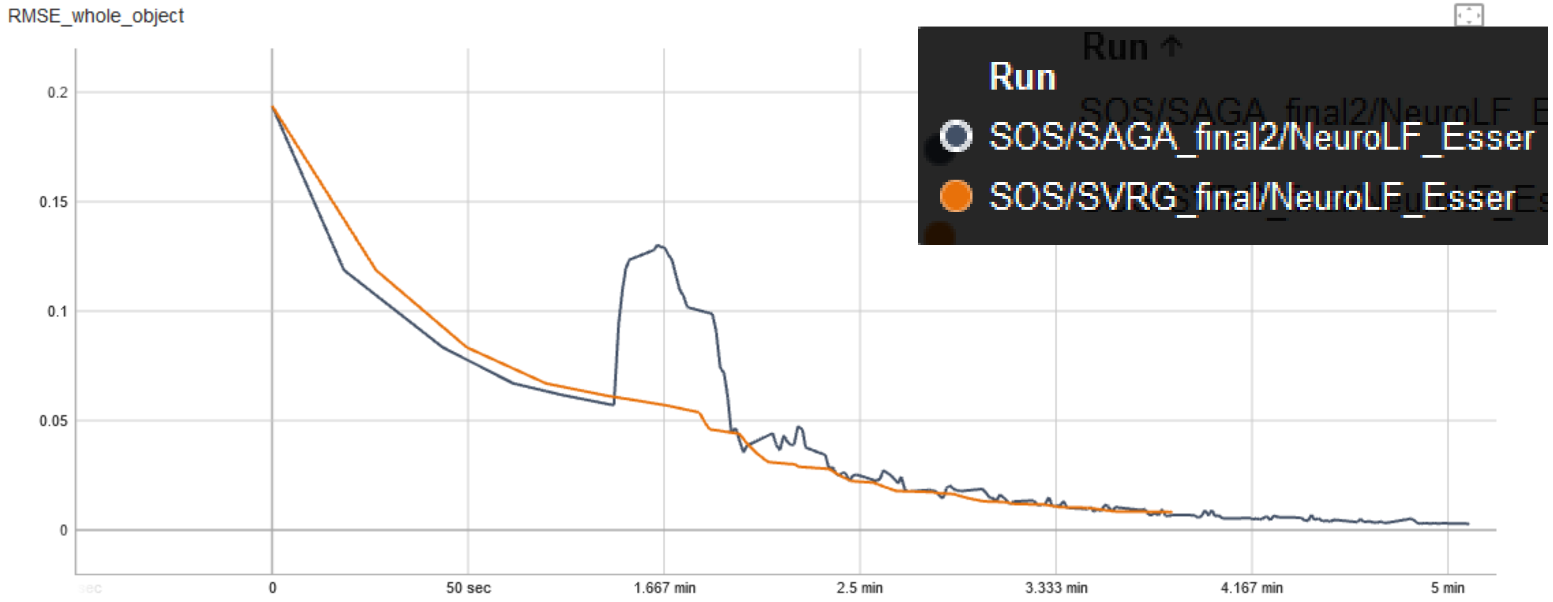
Results – SAGA vs. SVRG

SAGA noisier curves but generally faster

SAGA able to take more steps

Steps taken	SAGA	SVRG
NEMA	97	47
Esser	161	71





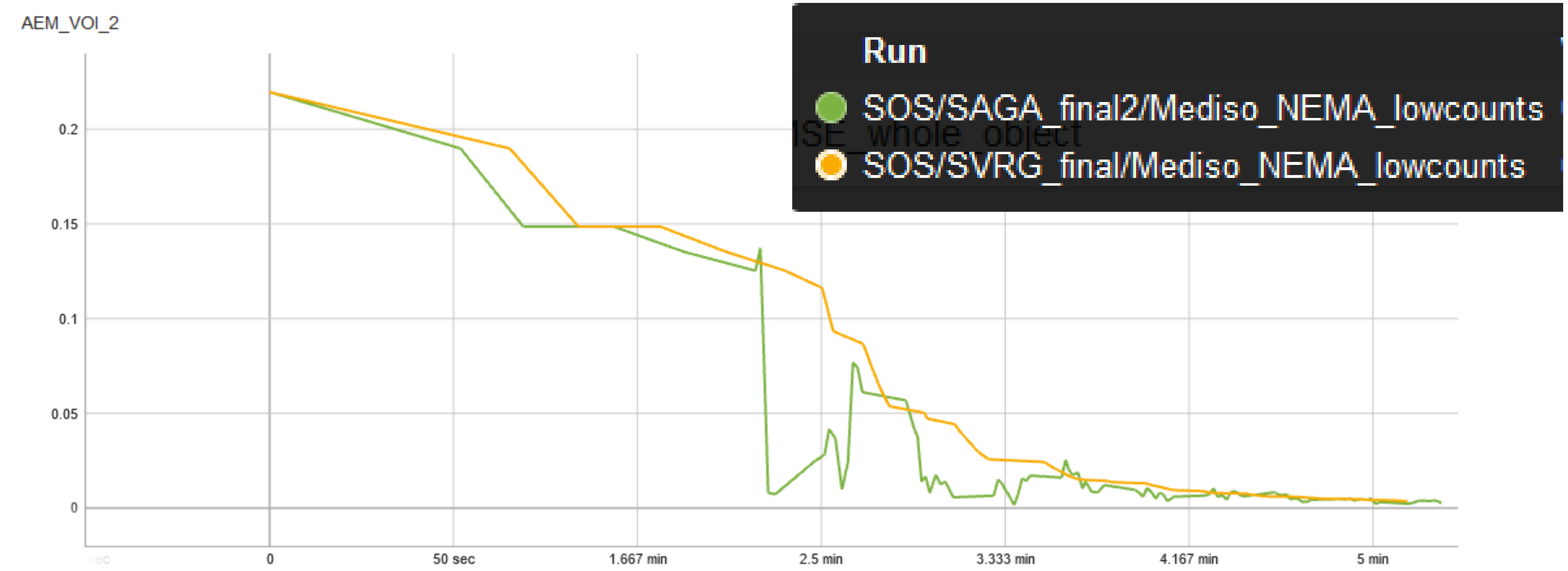
Results – SAGA vs. SVRG

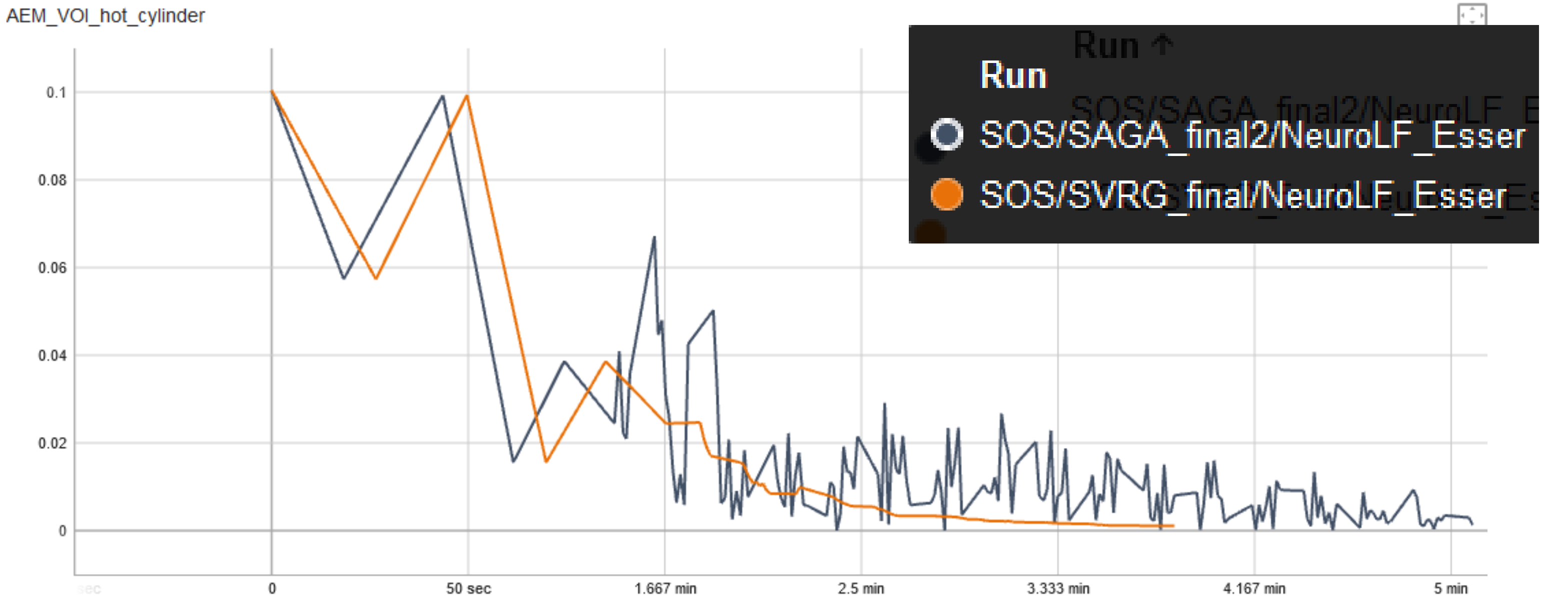
SAGA noisier curves but generally faster

SAGA able to take more steps

Steps taken	SAGA	SVRG
NEMA	97	47
Esser	161	71

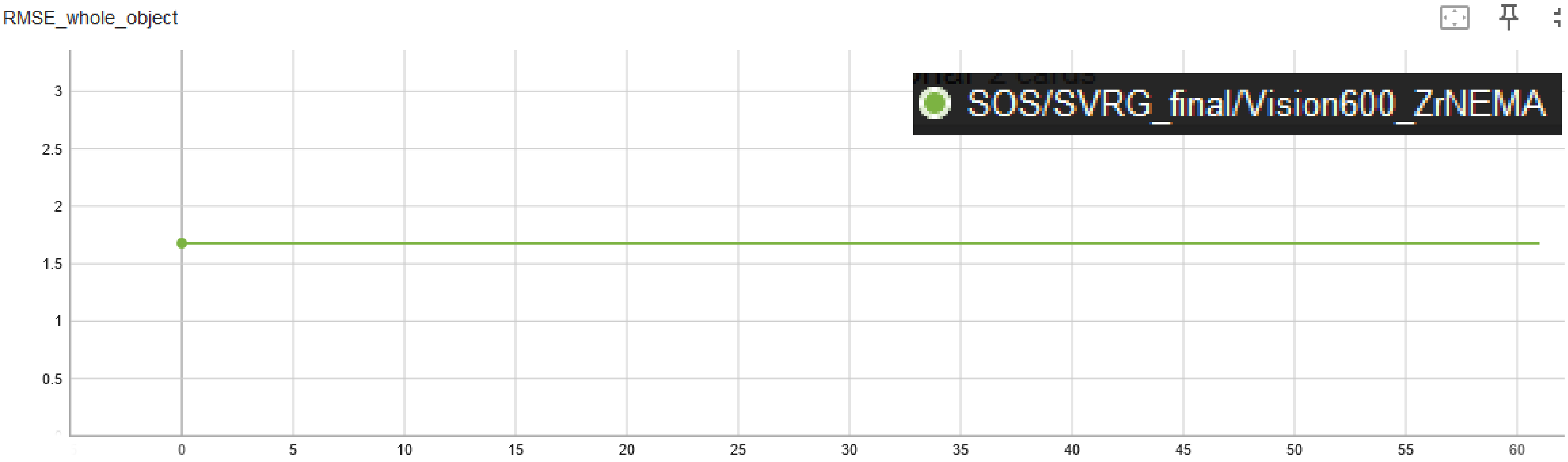
SVRG's stability sometimes beneficial in smaller VOIs





Results – step size selection

Even with Armijo, step size selection still an issue.



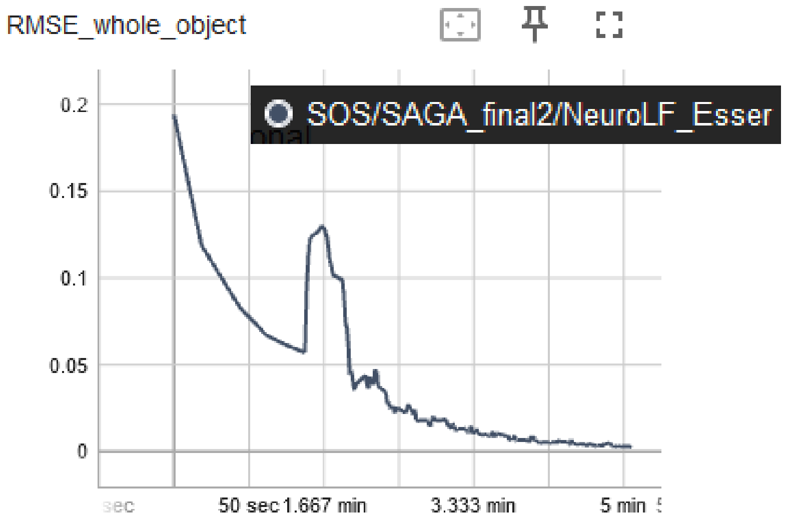
Results – The wrong preconditioner

BSREM is a preconditioner designed solely for the data term

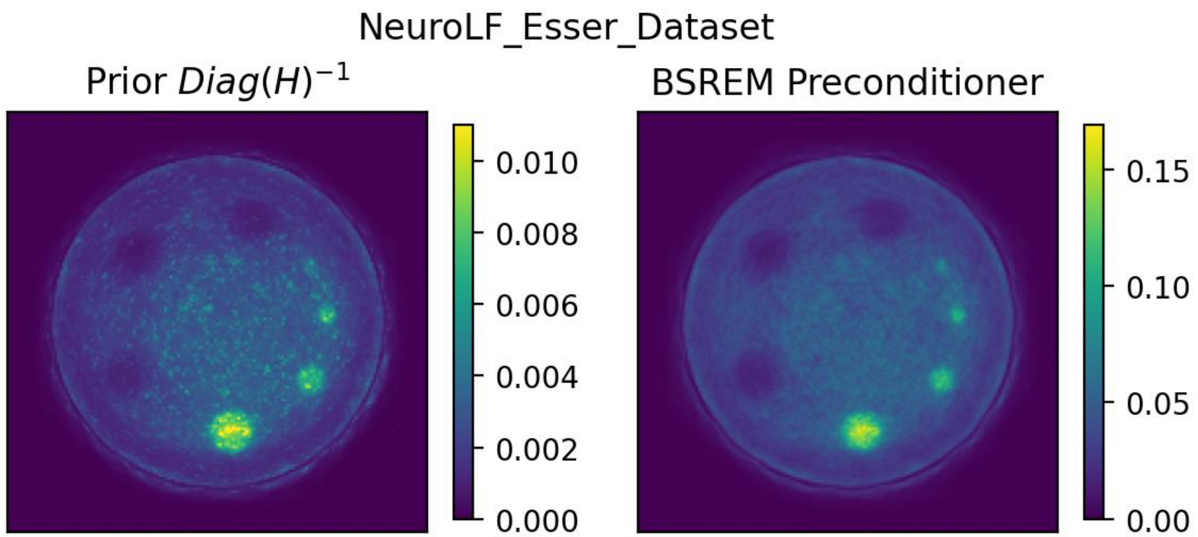
Results – The wrong preconditioner

BSREM is a preconditioner designed solely for the data term

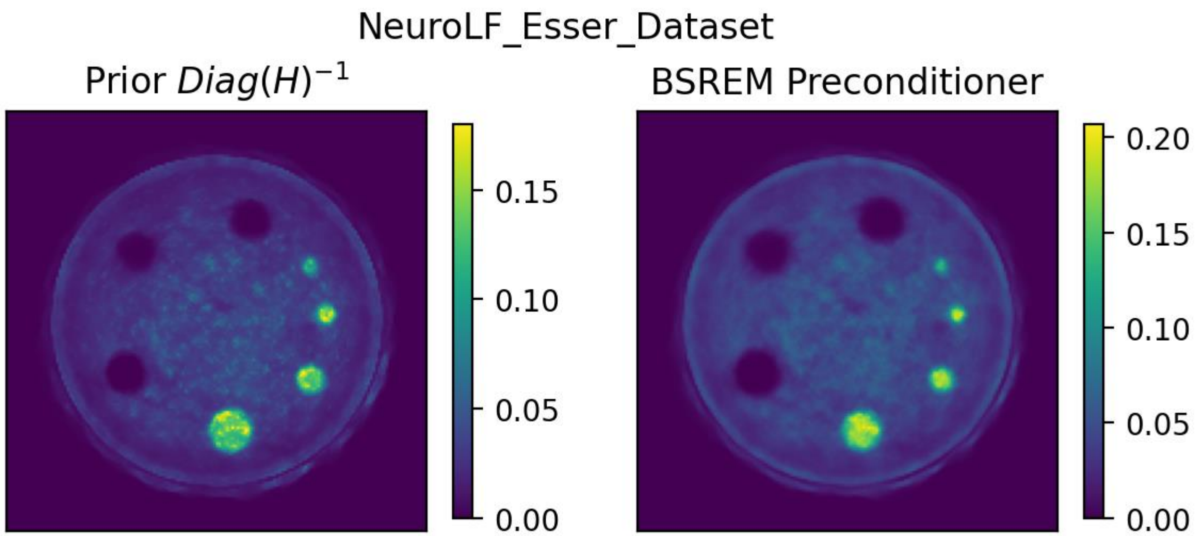
But performs well in some cases



Initial Image



Converged Image

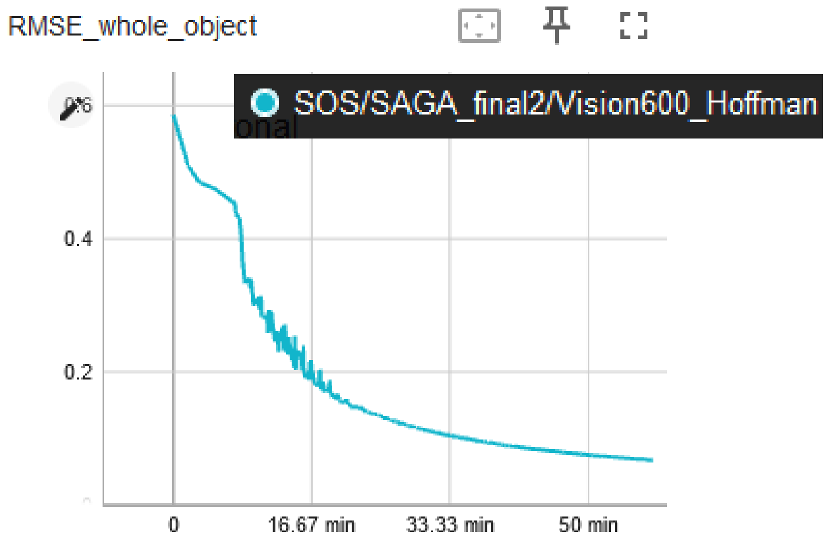


Results – The wrong preconditioner

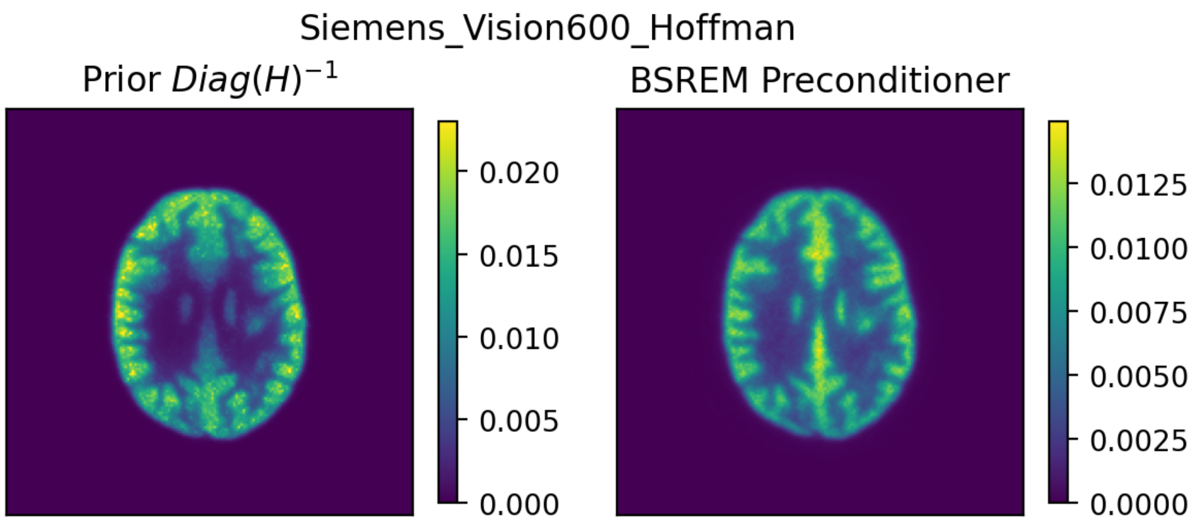
BSREM is a preconditioner designed solely for the data term

But performs well in some cases

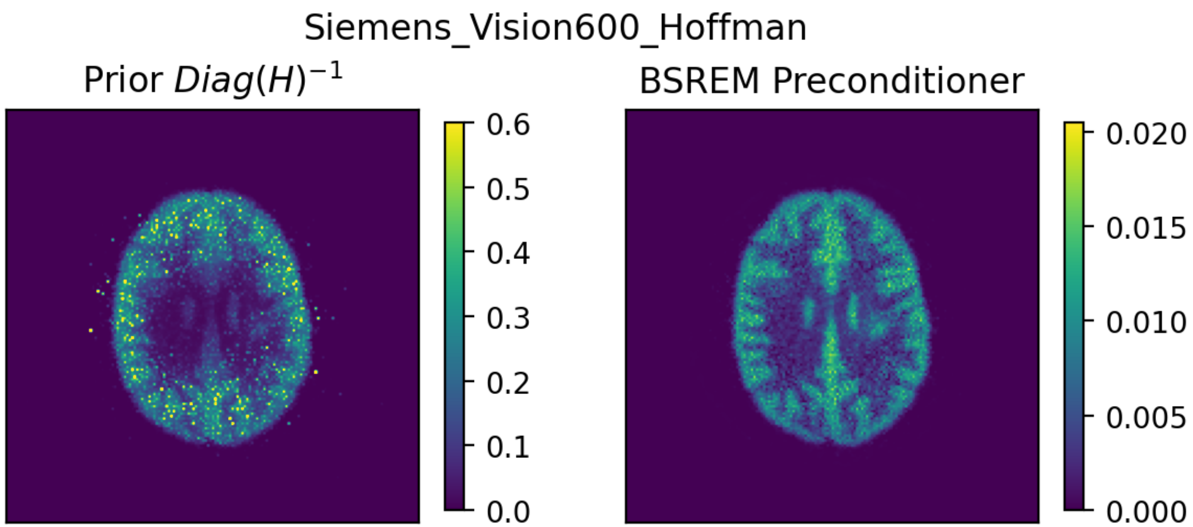
And not so well in others



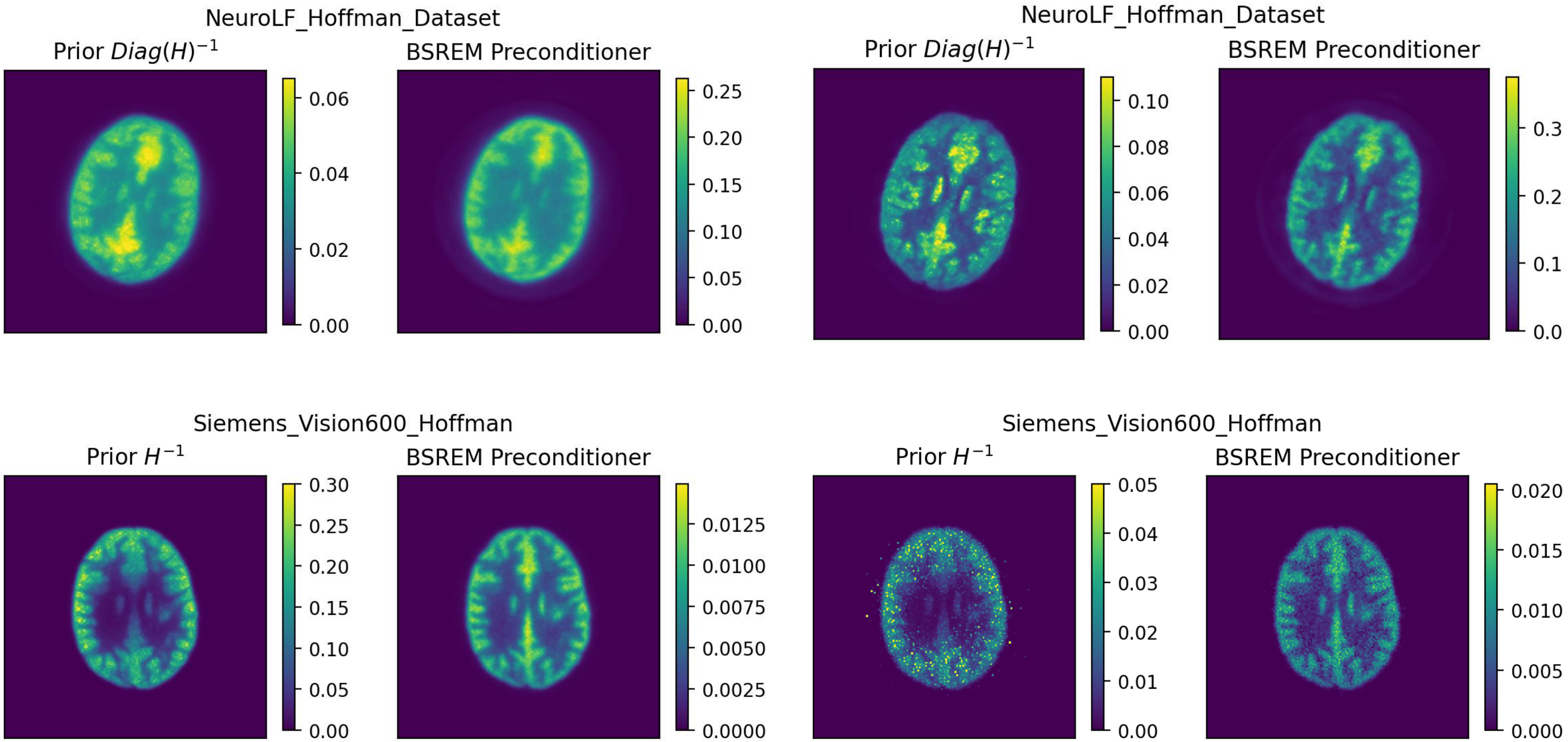
Initial Image



Converged Image



Results – The wrong preconditioner



What have I learned?

1. Include your prior in your preconditioner

2. Different data can require very different step size choices (0.001 ► 0.8+)

3. Updating every epoch with SVRG is a bad idea
   - Most literature suggests 2 epochs
   - Probably a good idea to vary the interval. Larger intervals in later stages

4. SAGA’s instability can cause issues

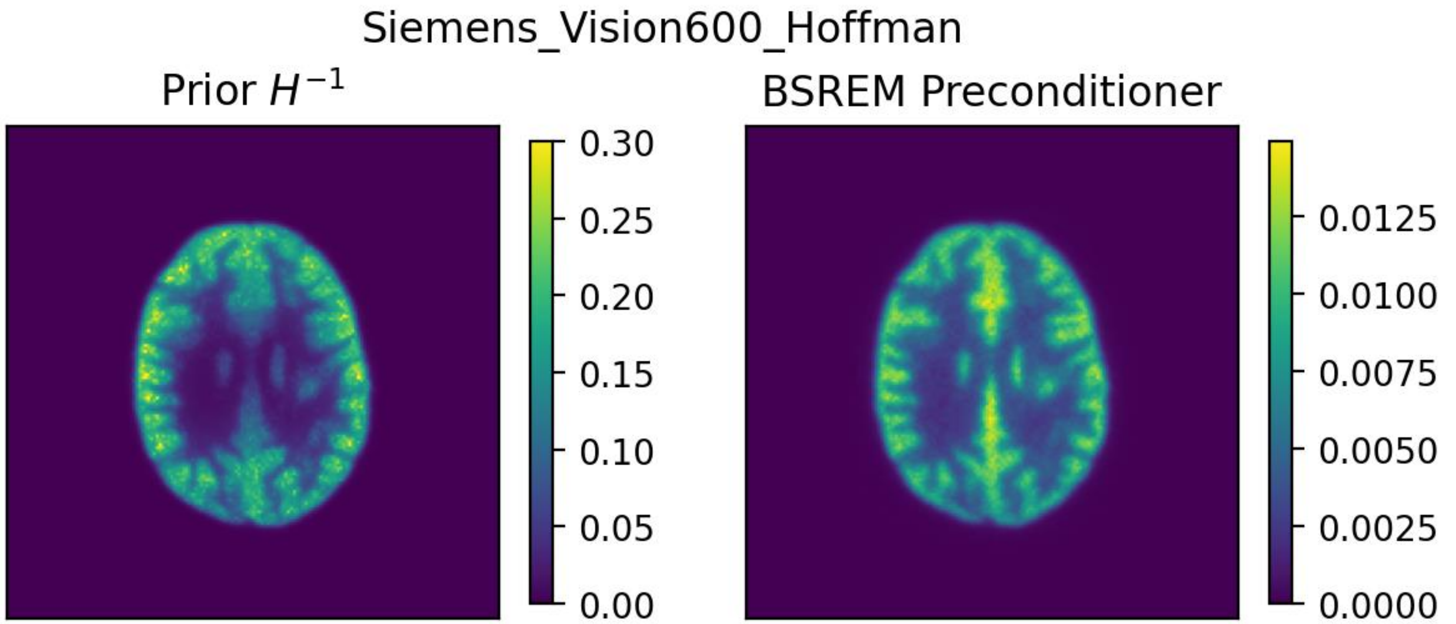
Any Questions?

sam.porter.18@ucl.ac.uk

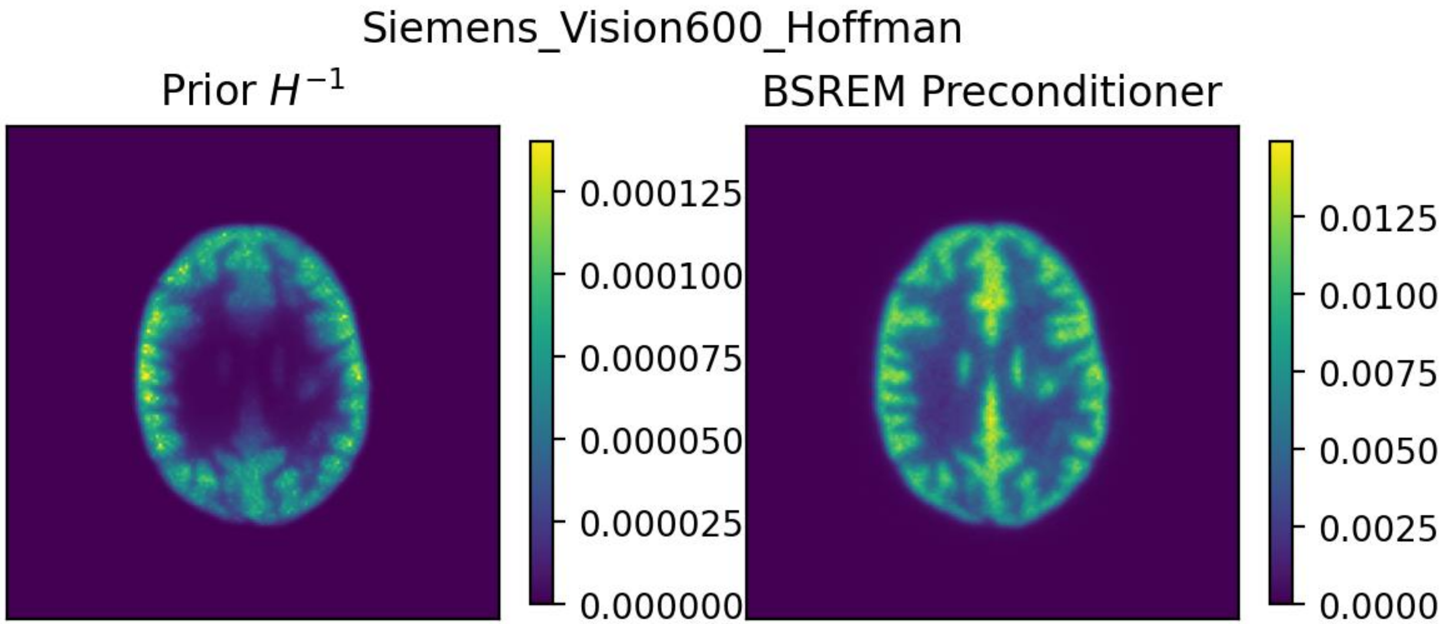
Acknowledgements:

- Thanks so much to Margaret, Edo and Casper for all their help
- Also, thanks to Kris and the rest of the PETRIC team for setting up a very fun challenge!

Additional slides – prior without the kappa



With kappa



Without kappa