Stochastic Optimisation Framework using CIL and SIRF for PET Reconstruction

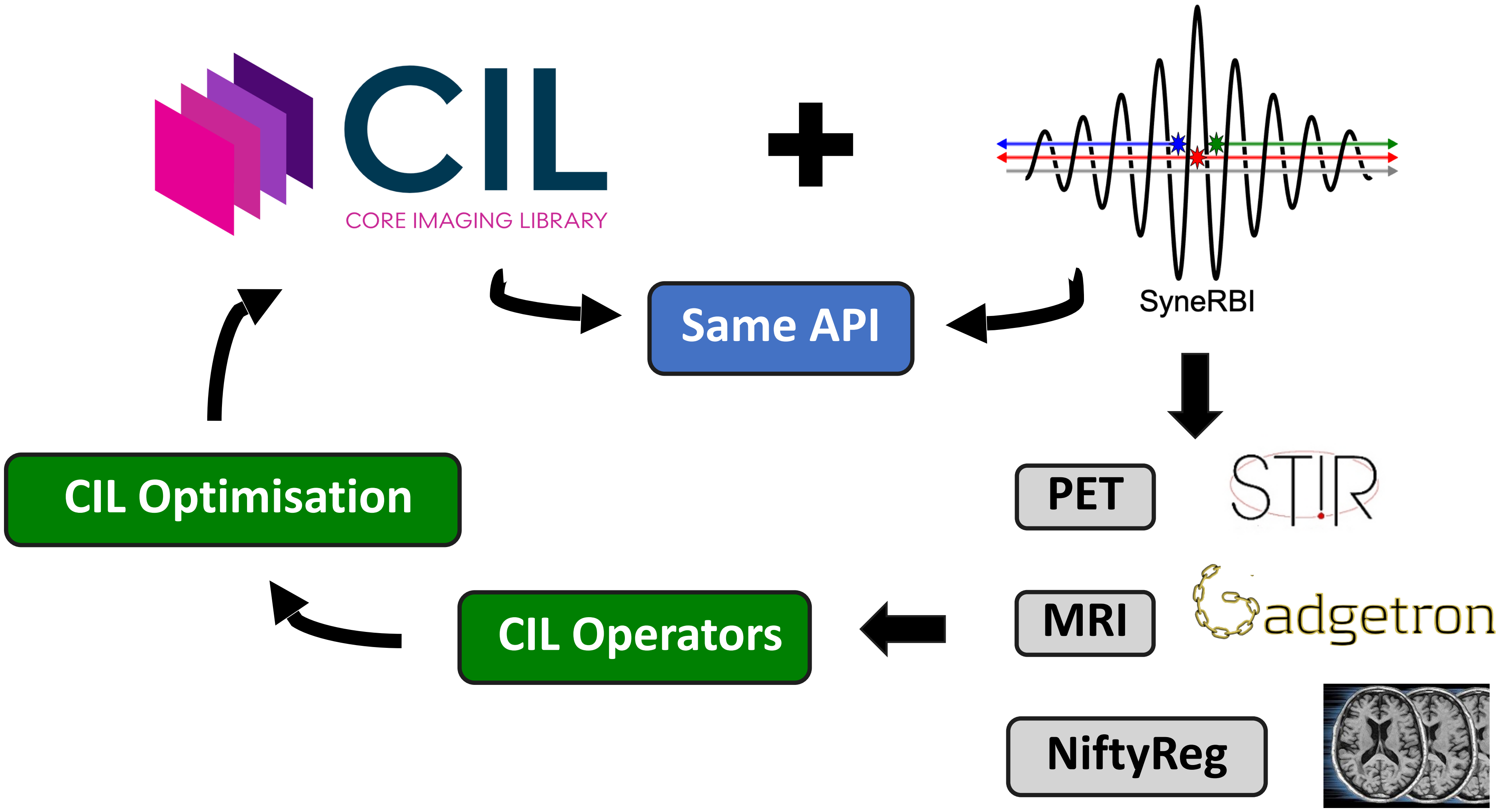
10th Conference on PET, SPECT, and MR Multimodal Technologies
Total Body and Fast Timing in Medical Imaging

Evangelos Papoutsellis - Finden Ltd, University of Manchester

Finden
Innovate UK KTN
SyneRBI
CCPi TOMOGRAPHIC IMAGING
CoSeC Computational Science Centre for Research Communities
MANCHESTER 1824 The University of Manchester
NPL National Physical Laboratory
UKRI Science and Technology Facilities Council Scientific Computing

epapoutsellis@gmail.com

Overview: CIL and SIRF



Jørgensen. et al. 2021, Core Imaging Library - Part I: a versatile Python framework for tomographic imaging
Papoutsellis et al. 2021, Core Imaging Library - Part II: multichannel reconstruction for dynamic and spectral tomography
Ovtchinnikov et al. 2017, SIRF: Synergistic Image Reconstruction Framework

Optimisation in CIL

CIL Optimisation

↓

Functions

+

Operators

=

Algorithms

name	description
CGLS	conjugate gradient least squares
SIRT	simultaneous iterative reconstruction technique
GD	gradient descent
FISTA	fast iterative shrinkage-thresholding algorithm
LADMM	linearized alternating direction method of multipliers
PDHG	primal dual hybrid gradient
SPDHG	stochastic primal dual hybrid gradient

Chambolle et al 2017, Stochastic Primal-Dual Hybrid Gradient Algorithm with Arbitrary Sampling and Imaging Applications
Chambolle et al 2010, A First-Order Primal-Dual Algorithm for Convex Problems with Applications to Imaging
Beck et al 2009, A Fast Iterative Shrinkage-Thresholding Algorithm for Linear Inverse Problems

Optimisation in CIL

$$\min_x f(x)\ , \quad f\ :\ \text{L-smooth}$$

$$\min_x f(x) + g(x)\ , \quad f\ :\ \text{L-smooth}\ ,\ g\ :\ \text{proximable}$$

$$\min_x f(Kx) + g(x)\ , \quad f\ :\ \text{proximable}\ ,\ g\ :\ \text{proximable}\ ,\ K \text{ linear operator}$$

$f, g \quad \text{convex}$

Optimisation in CIL

$$\min_x f(x) \ , \quad f : \text{L-smooth}$$

$$\min_x f(x) + g(x) \ , \quad f : \text{L-smooth} \ , \ g : \text{proximable}$$

$$\min_x f(Kx) + g(x) \ , \quad f : \text{proximable} \ , \ g : \text{proximable} \ , \ K \text{ linear operator}$$

$f, g \quad \text{convex}$

PET reconstruction with Relative Difference Prior

$$\min_u \sum Au - b\log(Au + \eta) + \mathrm{RDP}(u) + \mathbb{I}_{\{u>0\}}(u)$$

Optimisation in CIL

$$\min_x f(x) \ , \quad f: \text{L-smooth}$$

$$\min_x f(x) + g(x) \ , \quad f: \text{L-smooth} \ , \ g: \text{proximable}$$

$$\min_x f(Kx) + g(x) \ , \quad f: \text{proximable} \ , \ g: \text{proximable} \ , \ K \text{ linear operator}$$

$f, g \quad \text{convex}$

PET reconstruction with Relative Difference Prior

$$\min_u \sum Au - b\log(Au + \eta) + \text{RDP}(u) + \mathbb{I}_{\{u>0\}}(u)$$

CT reconstruction with TV regularisation

$$\min_u \frac{1}{2}||Au - b||^2 + \alpha||\nabla u||_{2,1} + \mathbb{I}_{\{u>0\}}(u)$$

Optimisation in CIL

$$\min_x f(x) \ , \quad f : \text{L-smooth}$$

$$\min_x f(x) + g(x) \ , \quad f : \text{L-smooth} \ , \ g : \text{proximable}$$

$$\min_x f(Kx) + g(x) \ , \quad f : \text{proximable} \ , \ g : \text{proximable} \ , \ K \text{ linear operator}$$

$f, g \quad \text{convex}$

PET reconstruction with Relative Difference Prior

$$\min_u \sum Au - b\log(Au + \eta) + \mathrm{RDP}(u) + \mathbb{I}_{\{u>0\}}(u)$$

CT reconstruction with TV regularisation

$$\min_u \frac{1}{2}\|Au - b\|^2 + \alpha\|\nabla u\|_{2,1} + \mathbb{I}_{\{u>0\}}(u)$$

TGV denoising Salt and Pepper Noise

$$\min_u \frac{1}{2}\|u - b\|_1 + \alpha\|\nabla u - w\|_{2,1} + \beta\|\mathcal{E}w\|_{2,1}$$

Stochastic Project

Extend CIL Optimisation (Deterministic) framework to Stochastic Optimisation

Organised: CCP SyneRBI , CCPi , PET++

- **1st Hackathon:** November 23-26, 2021
- **2nd Hackathon**: April 4-7, 2022
- **Finden Ltd - Analysis for Innovators (A4i):** Denoising of chemical imaging and tomography data. In collaboration with National Physical Laboratory (05/2023 - 09/2023)

Joint work: Kris Thielemans, Gillman Ashley, Tang Junqi, Zeljko Kereta, Imraj Singh, Gemma Fardell, Evgueni Ovtchinnikov, Matthias Ehrhardt, Laura Murgatroyd, Robert Twyman, Edoardo Pasca, Claire Delplancke, Georg Schramm, Daniel Deidda, Jakob Jørgensen, Sam Porter, Margaret Duff, Antony Vamvakeros, Simon Jacques, Casper da Costa-Luis

Stochastic Project

<u>Extend CIL Optimisation (Deterministic) framework to Stochastic Optimisation</u>

- **<u>Implement splitting for CIL/SIRF DataContainers: CT, PET, SPECT, MRI data</u>**
- **<u>Implement randomized algorithms in CIL, e.g., SGD, SAG, SAGA, SVRG and more</u>**
- **<u>Improve CIL optimisation functionality, e.g., step size, preconditioning</u>**

$$\min_x f(x) + g(x) \quad , \quad \min_x f(Kx) + g(x)$$

$$\min_x \sum_{i=1}^{n} f_i(x) + g(x) \quad , \quad \min_x \sum_{i=1}^{n} f_i(K_i x) + g(x)$$

Defazio et al. 2014, SAGA: A Fast Incremental Gradient Method With Support for Non-Strongly Convex Composite Objectives
Schmidt et al. 2017, Minimizing finite sums with the stochastic average gradient
Johnson et al., 2013, Accelerating Stochastic Gradient Descent using Predictive Variance Reduction

Stochastic Optimisation in CIL

$$\min_x f(x) + g(x)$$

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f(x_k)$	$x_{k+1} = \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f(x_k))$	$x_{k+1} = \text{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f(y_k))$ $\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$

Stochastic Optimisation in CIL

$$\min_x \sum_{i=1}^{n} f_i(x) + g(x)$$

- Avoid computing the full gradient per iteration, i.e., gradient for all n.

- Select a random index $i_k \in \{1, \ldots, n\}$ ***and compute*** ∇f_{i_k} ***per iteration.***

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$

Stochastic Optimisation in CIL

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1 + \sqrt{1 + 4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$
SGD	***Prox-SGD***	***Acc-Prox-SGD***

Stochastic Optimisation in CIL

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$
SGD	***Prox-SGD***	***Acc-Prox-SGD***

Stochastic Optimisation Design

- No direct implementation of Stochastic algorithms, e.g., SGD.
- Use a deterministic algorithm, e.g., GD, already available in CIL.
- Implement a Stochastic Gradient "Functions" or Variance-Reduced "Functions"
- Example SGD := GD (CIL Algorithm) + SGFunction (CIL Function)

Stochastic Optimisation in CIL

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \text{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \text{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$
$\underbrace{\qquad}_{\text{SGD}}$	$\underbrace{\qquad}_{\text{Prox-SGD}}$	$\underbrace{\qquad}_{\text{Acc-Prox-SGD}}$

Stochastic Optimisation in CIL

GD	PGA/ISTA	APGA/FISTA
$x_{k+1} = x_k - \gamma_k \nabla f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \nabla f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \nabla f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1+\sqrt{1+4\alpha_k^2}}{2}$ $y_k = x_k + \frac{\alpha_k - 1}{a_{k+1}}(x_k - x_{k-1})$
SGD	***Prox-SGD***	***Acc-Prox-SGD***

`ApproximateGradientSumFunction`

$$\sum_{i=1}^{n} f_i(x)$$

- **CIL Class Function**
- **Selects (randomly)** $i_k \in \{1, \ldots, n\}$
- **Computes** ∇f_{i_k}

Stochastic Optimisation in CIL

GD	ISTA	FISTA
$x_{k+1} = x_k - \gamma_k \tilde{\nabla} f_{i_k}(x_k)$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k \tilde{\nabla} f_{i_k}(x_k))$	$x_{k+1} = \mathrm{prox}_{\gamma_k g}(y_k - \gamma_k \tilde{\nabla} f_{i_k}(y_k))$ $\alpha_{k+1} = \frac{1 + \sqrt{1 + \alpha_k^2}}{2}$ $y_{k+1} = x_k + \frac{\alpha_k - 1}{\alpha_{k+1}}(x_k - x_{k-1})$

`ApproximateGradientSumFunction`

$$\sum_{i=1}^{n} f_i(x)$$

Stochastic Gradient and Variance-Reduced CIL "Functions"

$$\left\{ \right.$$

- `SGFunction`
- `SAGFunction`
- `SVRGFunction`
- `SAGAFunction`
- `LSVRGFunction`
- `and more ...`

Stochastic Optimisation in CIL

Plug and Play Framework - Different Stochastic Gradient Functions

Algorithms / Stochastic Function	GD	ISTA	FISTA
SGFunction	SGD	Prox-SGD	Acc-Prox-SGD
SAGFunction	SAG	Prox-SAG	Acc-Prox-SAG
SAGAFunction	SAGA	Prox-SAGA	Acc-Prox-SAGA
SVRGFunction	SVRG	Prox-SVRG	Acc-Prox-SVRG
LSVRGFunction	LSVRG	Prox-LSVRG	Acc-Prox-LSVRG

```
gd = GD(initial=initial, objective_function=f, step_size=step_size,
        update_objective_interval=1, max_iteration=10)
gd.run(verbose=1)

pgd = ISTA(initial=initial, f=f, g=g, step_size=step_size,
        update_objective_interval=1, max_iteration=10)
pgd.run(verbose=1)
```

Stochastic Optimisation in CIL

Plug and Play Framework - Different Stochastic Gradient Functions

Algorithms Stochastic Function	GD	ISTA	FISTA
SGFunction	SGD	Prox-SGD	Acc-Prox-SGD
SAGFunction	SAG	Prox-SAG	Acc-Prox-SAG
SAGAFunction	SAGA	Prox-SAGA	Acc-Prox-SAGA
SVRGFunction	SVRG	Prox-SVRG	Acc-Prox-SVRG
LSVRGFunction	LSVRG	Prox-LSVRG	Acc-Prox-LSVRG

```
sgd = GD(initial=initial, objective_function=SGFunction(fi), step_size=step_size,
         update_objective_interval=1, max_iteration=10)
sgd.run(verbose=1)

prox_sgd = ISTA(initial=initial, f=SGFunction(fi), g=g, step_size=step_size,
         update_objective_interval=1, max_iteration=10)
prox_sgd.run(verbose=1)
```

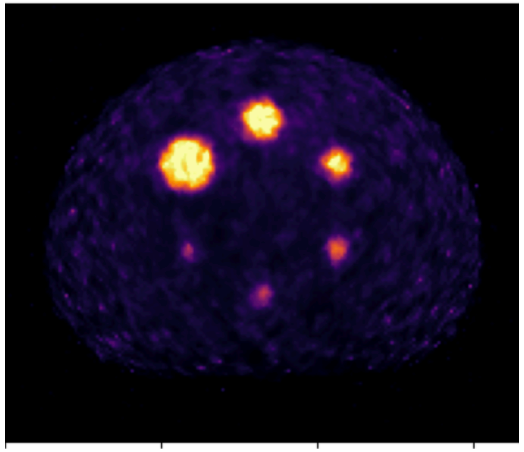
Stochastic Utilities/Improvements

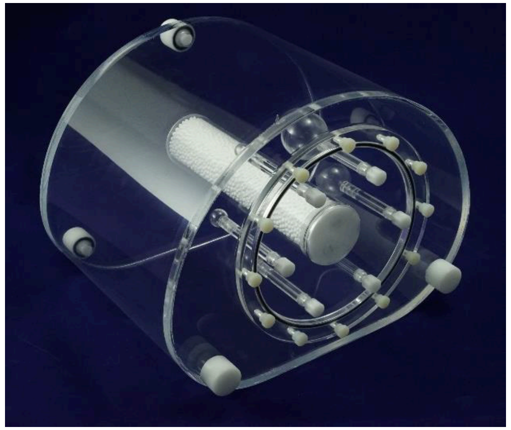
- ✓ <u>Data splitting methods for AcquisitionData (CIL + SIRF).</u>
- ✓ <u>Sampling methods used by Stochastic Functions, i.e., select the next function</u> f_{i_k}
- ✓ <u>Callable Classes to improve functionality of CIL Algorithms</u>
- ✓ <u>Proximal Gradient Algorithm (PGA) : Base class for Proximal Gradient Algorithms, e.g., GD, ISTA, FISTA</u>
- ✓ <u>CIL + SIRF fully compatible --> SIRF Functions can be used with CIL Algorithms</u>

Stochastic Utilities

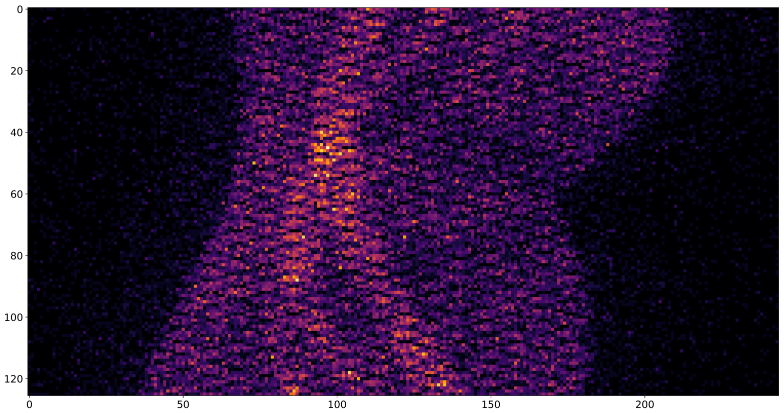
✓ Example: Data splitting methods for AcquisitionData (CIL + SIRF)

NEMA dataset



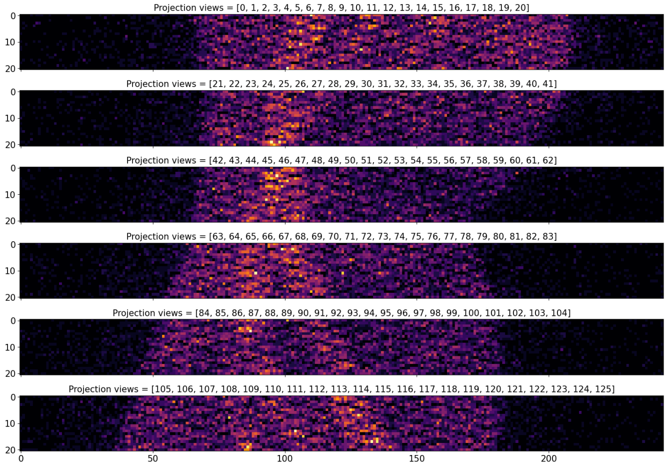


126 projection views

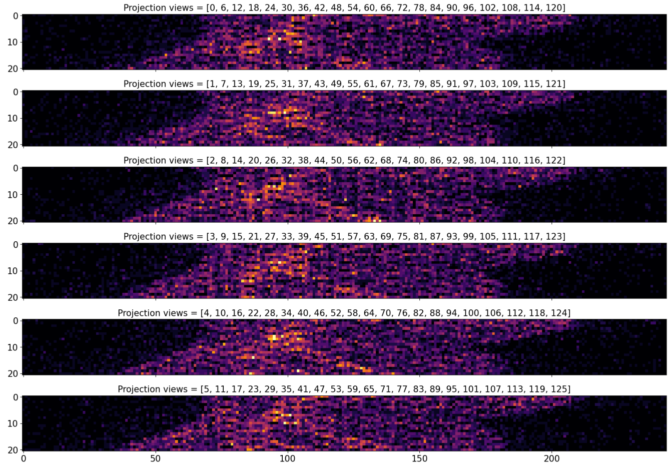


Split to 6 subsets

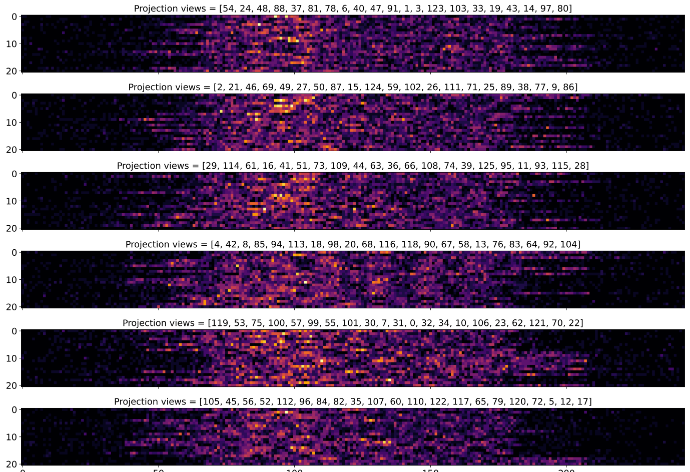
Ordered (Step_size=1)



Ordered (Step_size= NumSubsets)



Random



Stochastic Utilities

✓ **Example: Sampling methods used from Stochastic Functions**

Uniform Sampling (With replacement) n=6

Iteration	0	1	2	3	4	5	6	7	8	9	10	11
Function	f_3	f_4	f_0	f_5	f_1	f_4	f_3	f_1	f_4	f_0	f_0	f_1

RandomShuffle (Without replacement)

Iteration	0	1	2	3	4	5	6	7	8	9	10	11
Function	f_3	f_1	f_0	f_2	f_4	f_5	f_4	f_1	f_0	f_2	f_5	f_3

SingleShuffle (Without replacement)

Iteration	0	1	2	3	4	5	6	7	8	9	10	11
Function	f_3	f_2	f_4	f_0	f_1	f_5	f_3	f_2	f_4	f_1	f_5	f_3

HermanMeyer

Iteration	0	1	2	3	4	5	6	7	8	9	10	11
Function	f_0	f_3	f_1	f_4	f_2	f_5	f_0	f_3	f_1	f_4	f_2	f_5

CIL Improvements

✓ **Example: Proximal Gradient Algorithm (PGA) Base Class**

$$x_{k+1} = \mathrm{prox}_{\gamma_k g}(x_k - \gamma_k D(x_k)\nabla f(x_k))$$

$$x_{k+1} = \mathrm{prox}_{\gamma g}(x_k - \gamma \nabla f(x_k))$$

```python
class Preconditioner(ABC)
class StepSizeMethod(ABC)
```

Nocedal and Wright "Numerical Optimisation"

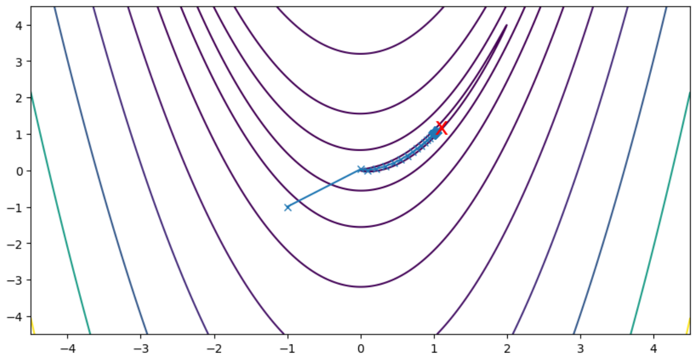
Choose $\bar{\alpha} > 0$, $\rho \in (0, 1)$, $c \in (0, 1)$; Set $\alpha \leftarrow \bar{\alpha}$;
repeat until $f(x_k + \alpha p_k) \le f(x_k) + c\alpha \nabla f_k^T p_k$
$\alpha \leftarrow \rho\alpha$;
end (repeat)
Terminate with $\alpha_k = \alpha$.

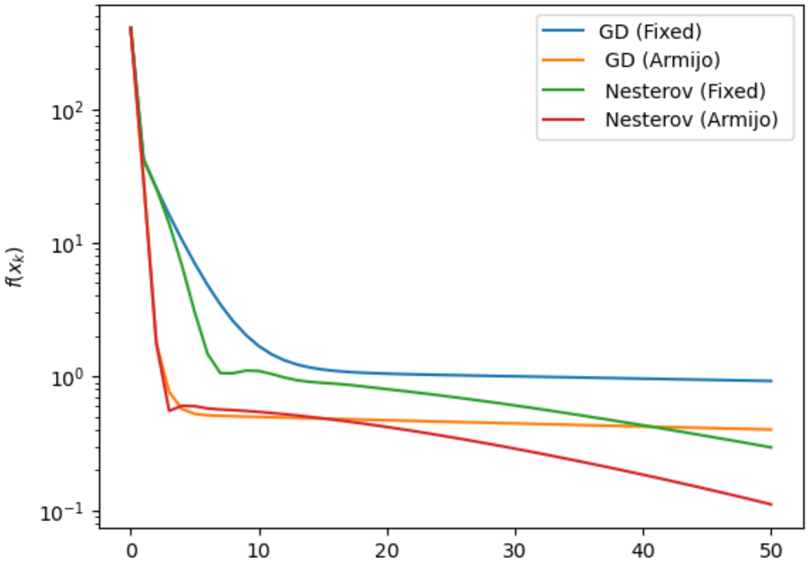
Armijo condition

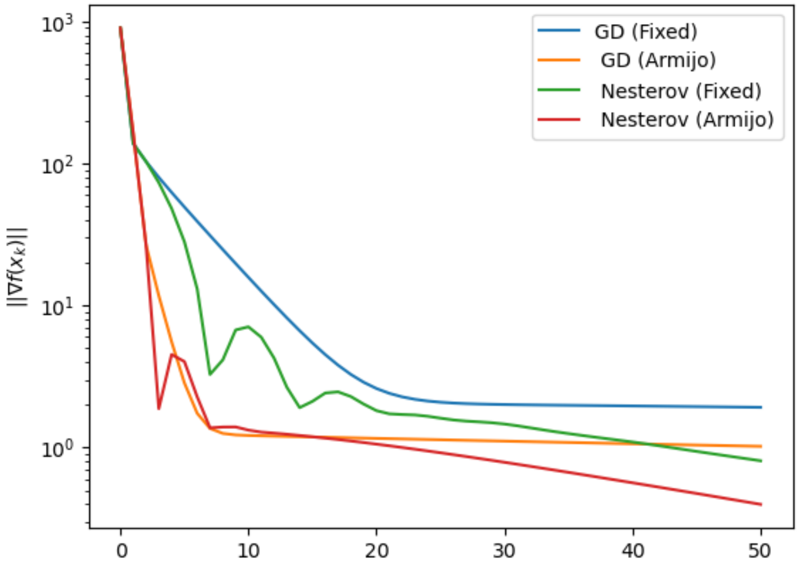
```python
armijo = ArmijoStepSize(initial = 1., rho = 0.5, c = 1e-4, iterations=50)
gd = GD(initial = initial, objective_function = f, step_size = armijo,
            max_iteration = 100, update_objective_interval =  1)
gd.run(verbose=1)
```

Rosenbrock Function: minimum at (x,y) = (1,1)

$$f(x, y) = (1 - x)^2 + 100(y - x^2)^2$$







CIL Improvements

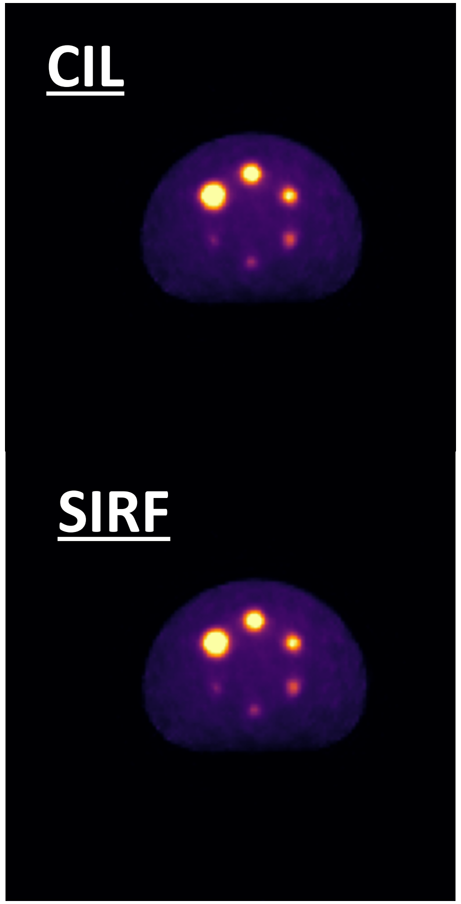
✓ **CIL + SIRF fully compatible** ➡

```
class ObjectiveFunction(object)
class Prior(object)
```

$$f(x) = \mathrm{KL}(b, Ax) + \alpha \mathrm{RDP}(x) \quad \rightarrow \quad x_{k+1} = \mathbb{P}_{\geq 0}(x_k - \gamma_k D(x_k) \nabla f(x_k))$$

$$\alpha = 0, \gamma_k = 1$$

$$D(x_k) = \frac{x_k}{A^T \mathbf{1}} \quad \rightarrow \quad x_{k+1} = \frac{x_k}{A^T \mathbf{1}} A^T \left(\frac{d}{Ax_k}\right) \quad (MLEM)$$



SIRF

```
objective = make_Poisson_loglikelihood(d)
objective.set_acquisition_model(A)
objective.set_prior(RelativeDifference)

recon = OSMAPOSLReconstructor()
recon.set_objective_function(objective)
recon.set_num_subsets(1)
recon.set_num_subiterations(50)
recon.set_up(initial)
recon.set_current_estimate(initial)
recon.process()
```

CIL

```
D = AdaptiveSensitivity(A)
pgd = ISTA(initial = initial, f = objective, g = IndicatorBox(lower=0.),
           preconditioner = D, step_size = -1.,
           update_objective_interval=1, max_iteration=50)
pgd.run(verbose=1)
```

CIL Improvements

✓ **Example: Callable Classes to improve functionality of CIL Algorithms**

```
from cil.optimisation.utilities import MetricsDiagnostics,StatisticsDiagnostics, RSE
from skimage.metrics import structural_similarity as SSIM
from skimage.metrics import normalized_root_mse as NRMSE

cb1 = MetricsDiagnostics(reference_image = fista_1000.solution, verbose=1,
                         metrics_dict={'mae': MAE, 'ssim': SSIM, 'nrmse': NRMSE})
cb2 = StatisticsDiagnostics(verbose=1, statistics_dict={'mean': (lambda x: x.mean())})

fista = FISTA(initial = initial, f = fidelity, g=G, update_objective_interval = 1,
            max_iteration = 5)
fista.run(verbose=1, callback=[cb1, cb2])
```

- ***Access to all attributes of the Algorithm***
- ***Define custom metrics (images and/or ROI)***
- ***Define new stopping criteria***
- ***Integrate with Weights and Biases***

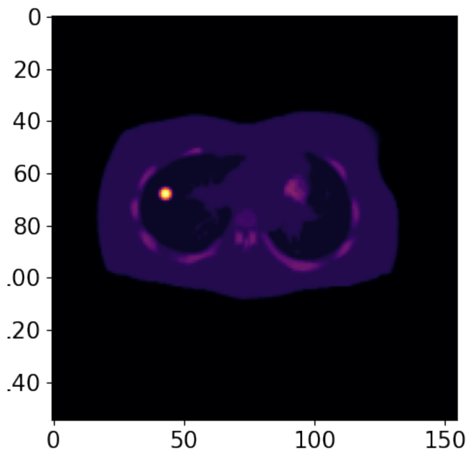
Iter	Max Iter	Time(s)/Iter	Objective	mae	ssim	nrmse	mean
0	5	0.000	8.17946e+02	6.51097e-05	5.05983e-01	1.00000e+00	0.00000e+00
1	5	0.542	9.93983e+01	7.07093e-05	2.84027e-01	7.29477e-01	0.00000e+00
2	5	0.391	7.81449e+01	6.34461e-05	3.14256e-01	6.82127e-01	7.00874e-05
3	5	0.329	6.22991e+01	5.61340e-05	3.50318e-01	6.37522e-01	6.89979e-05
4	5	0.297	5.16572e+01	4.95575e-05	3.92073e-01	5.98112e-01	6.79314e-05
5	5	0.330	4.49825e+01	4.41283e-05	4.37438e-01	5.63993e-01	6.69932e-05

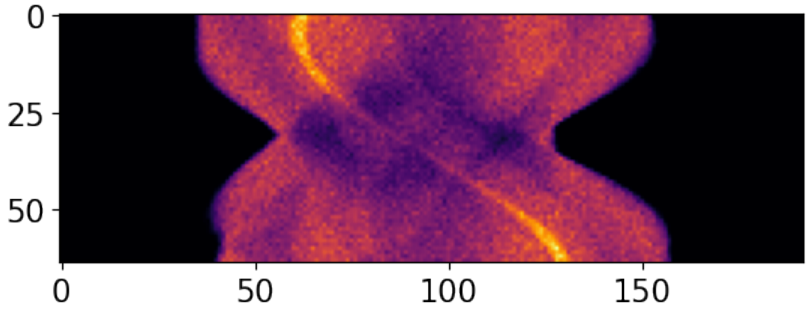
```
-------------------------------------------------------------------------------
Stop criterion has been reached.
```

Applications (PET - Non Smooth optimisation)

$$\underset{u}{\operatorname{argmin}} \sum Au - b\log(Au+\eta) + \alpha\|\nabla u\|_{2,1} + \mathbb{I}_{\{u>0\}}(u)$$

- **Splitting Method:** Ordered (64 projections), Subsets = 32
- **Sampling Method (Functions):** Random with replacement
- **Optimal Solution:** SPDHG-32 subsets, 500 epochs
- **Step size:** $\gamma_k = \gamma$

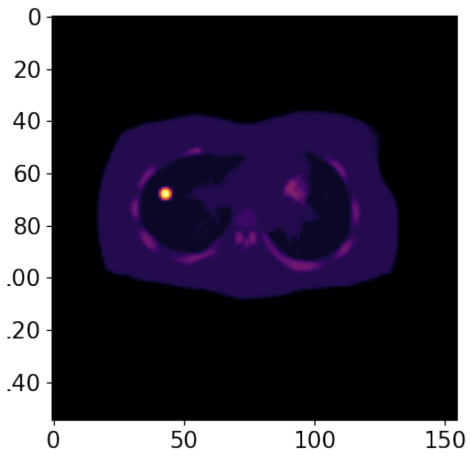


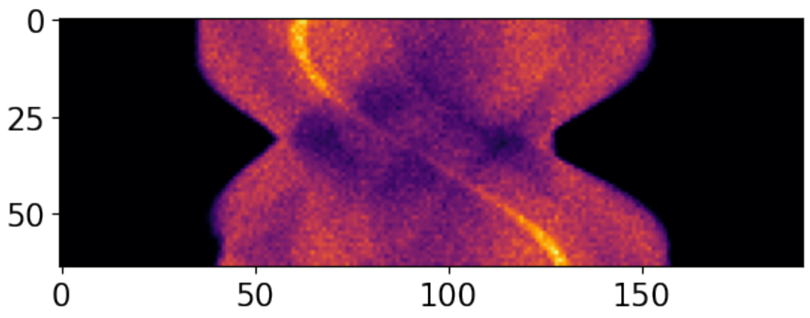


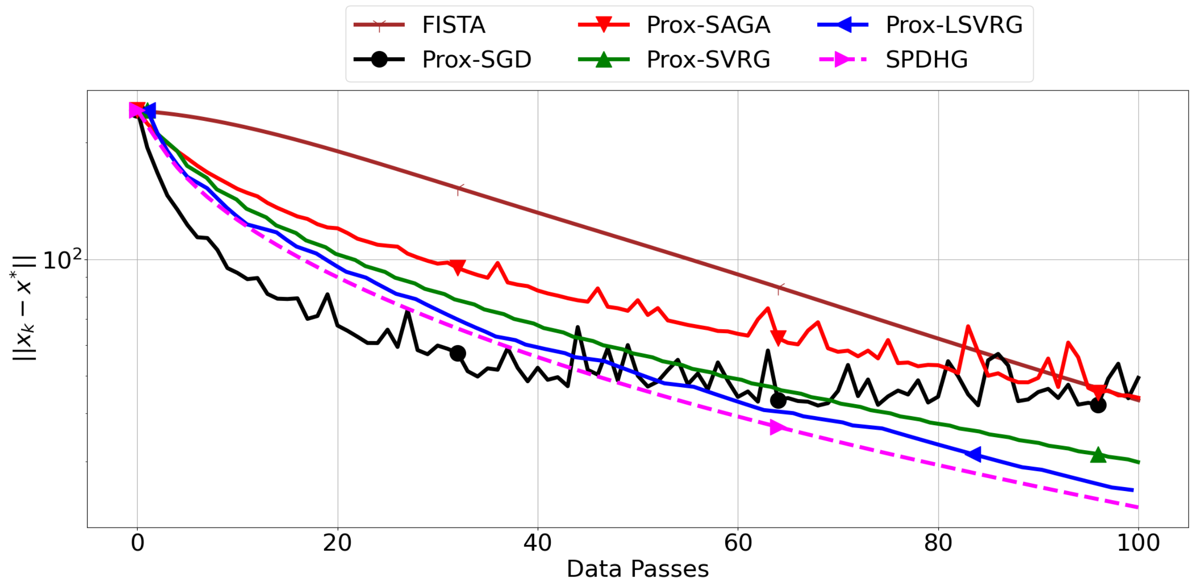
Applications (PET - Non Smooth optimisation)

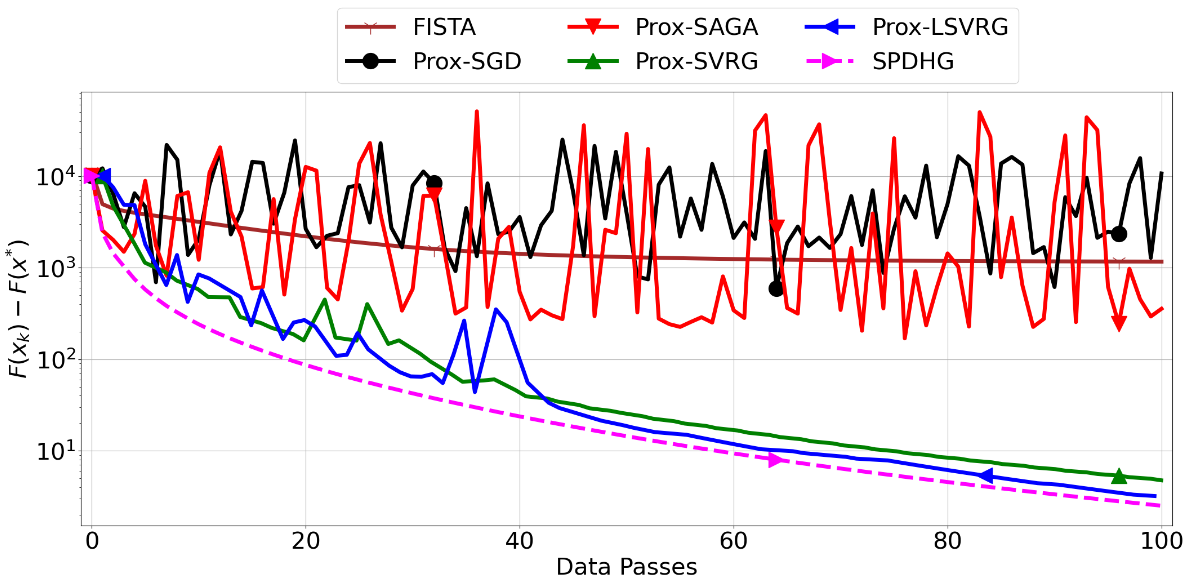
- **Splitting Method:** Ordered (64 projections), Subsets = 32
- **Sampling Method (Functions):** Random with replacement
- **Optimal Solution:** SPDHG-32 subsets, 500 epochs
- **Step size:** $\gamma_k = \gamma$

$$\underset{u}{\operatorname{argmin}} \sum Au - b\log(Au + \eta) + \alpha\|\nabla u\|_{2,1} + \mathbb{I}_{\{u>0\}}(u)$$

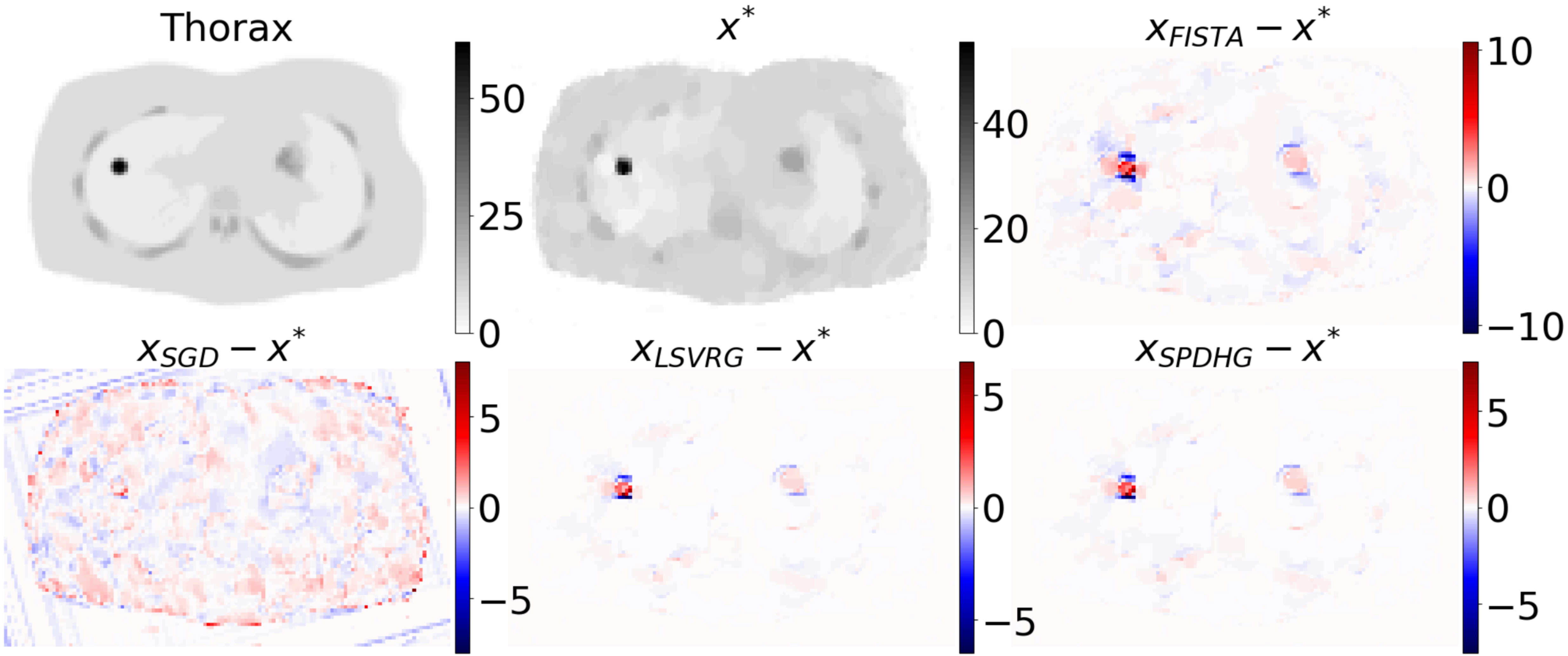








Applications (PET - Non Smooth optimisation)



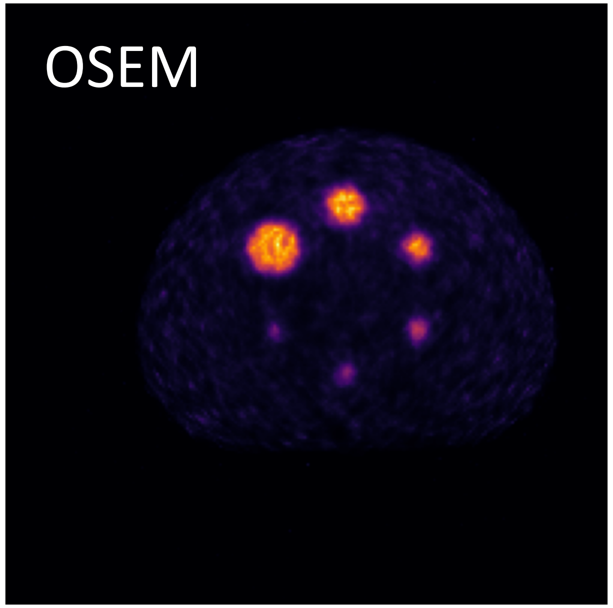
Applications (PET - Smooth optimisation)

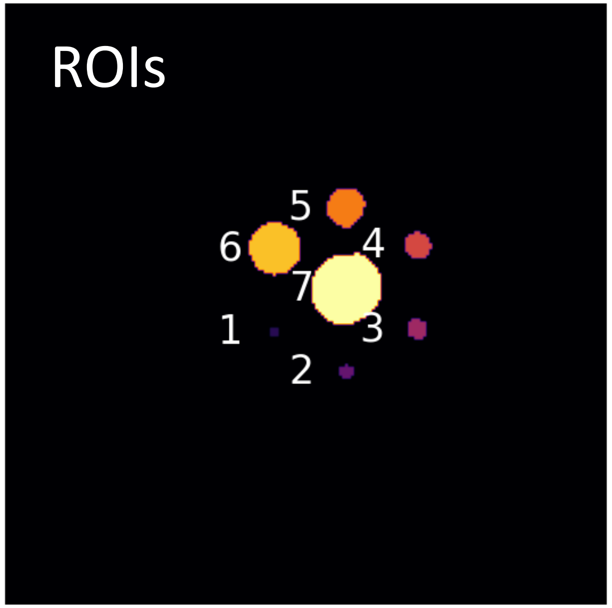
$$\min_u \sum Au - b\log(Au + \eta) + \mathrm{RDP}(u) + \mathbb{I}_{\{u>0\}}(u)$$

- **Splitting Method:** Ordered (126 projections), Subsets = 21, 42, 63
- **Sampling Method (Functions):** Random with replacement
- **Optimal Solution:** (subset) Preconditioned Gradient Descent with relaxed step size, 7 subsets (20000 iterations)
- **Preconditioning:** $D(x_k) = \dfrac{x_{OSEM} + \varepsilon}{A^T 1}$

Step size: $\gamma_k = \dfrac{1}{1 + \eta\left(k//(n/2)\right)}$

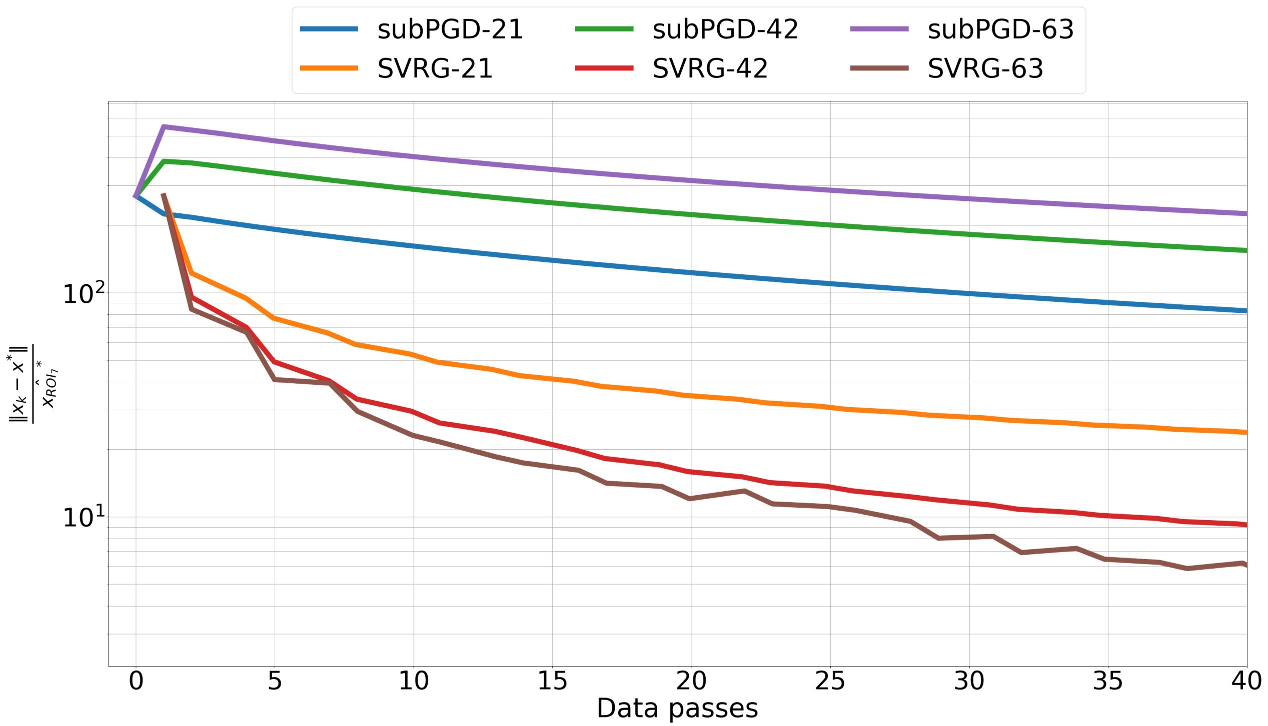
- **Initial:** OSEM

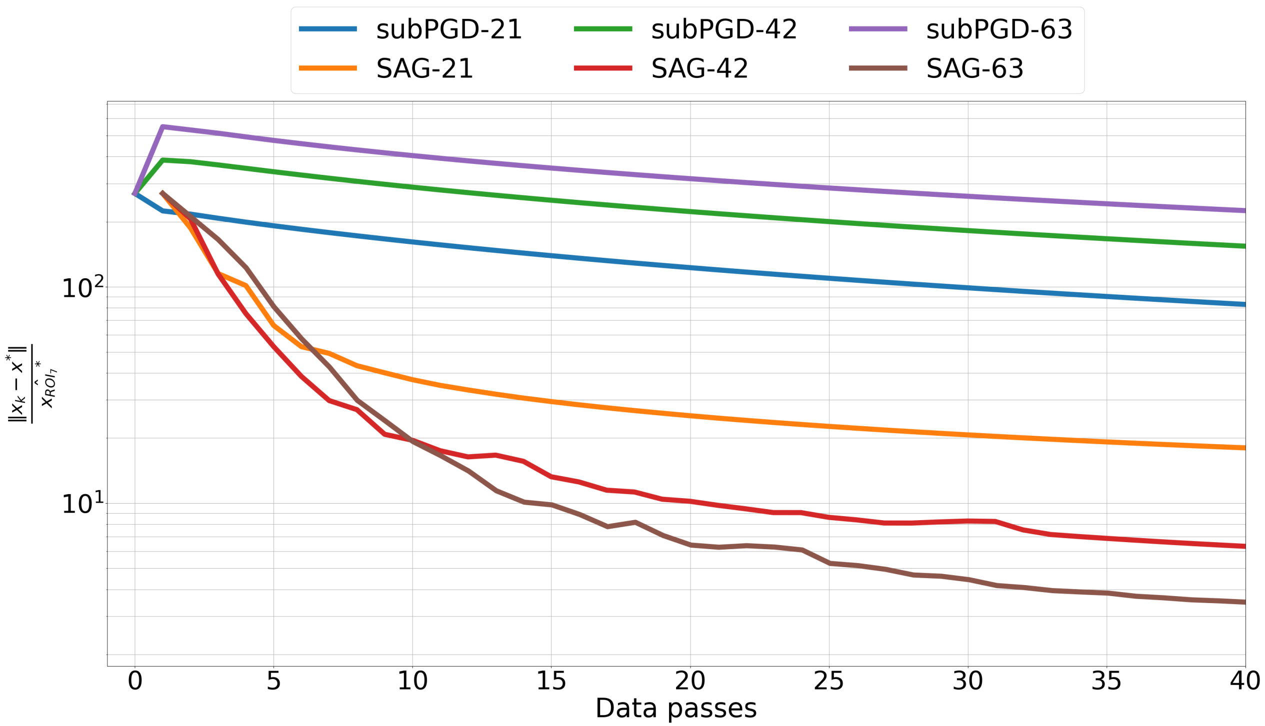




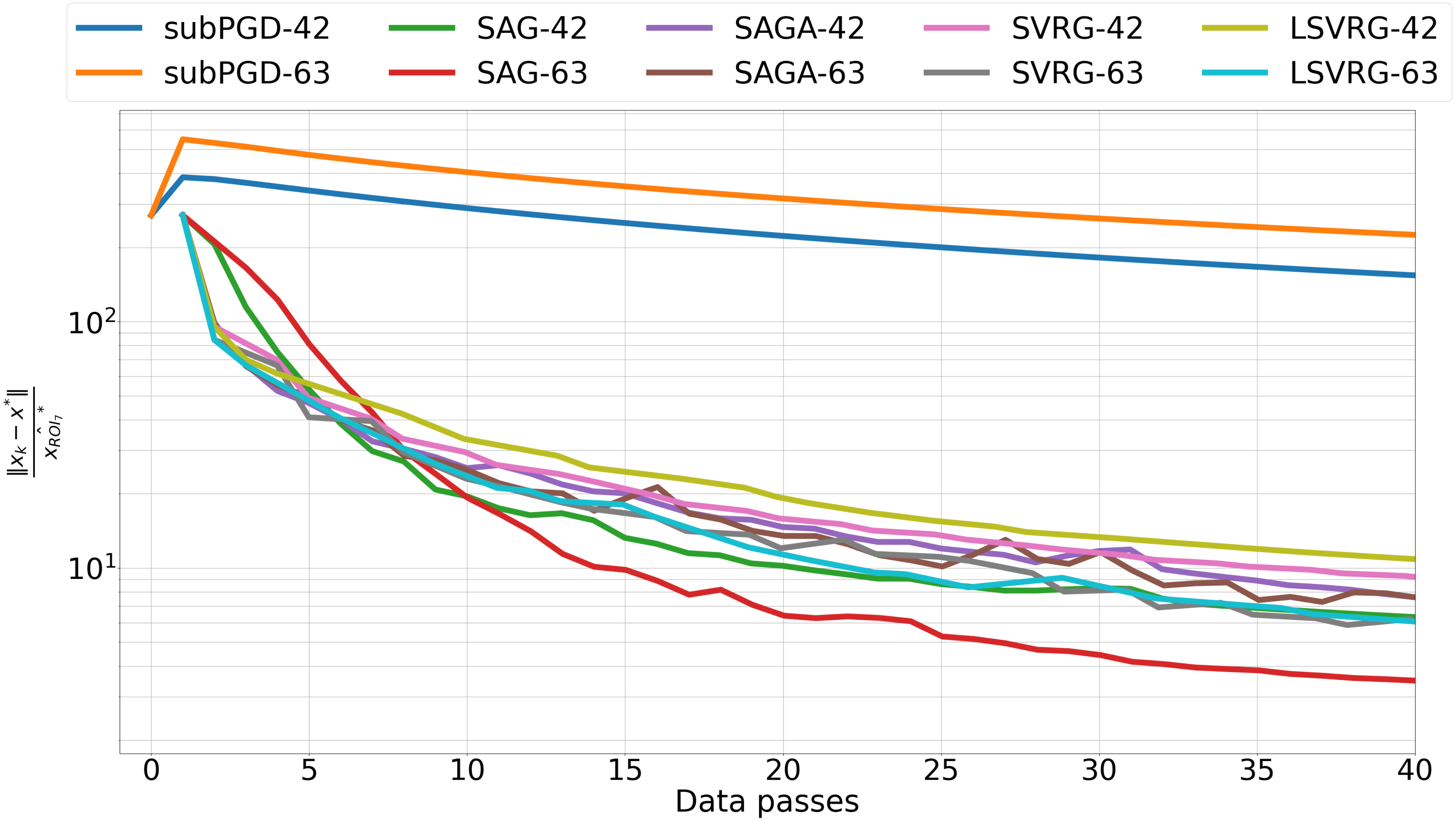
Twyman et al 2023. An Investigation of Stochastic Variance Reduction Algorithms for 3D Penalised PET Image Reconstruction

Applications (PET - Smooth optimisation)

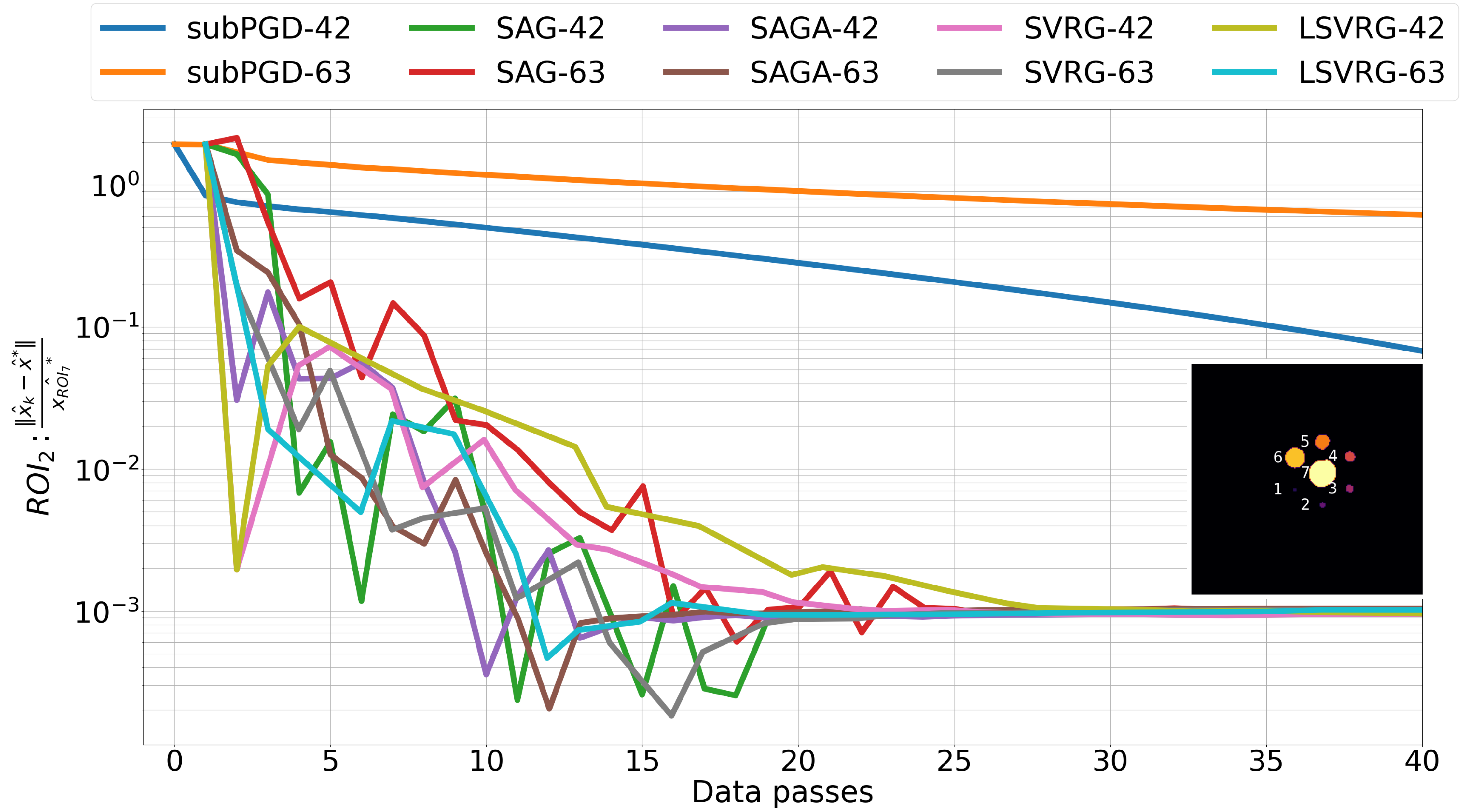




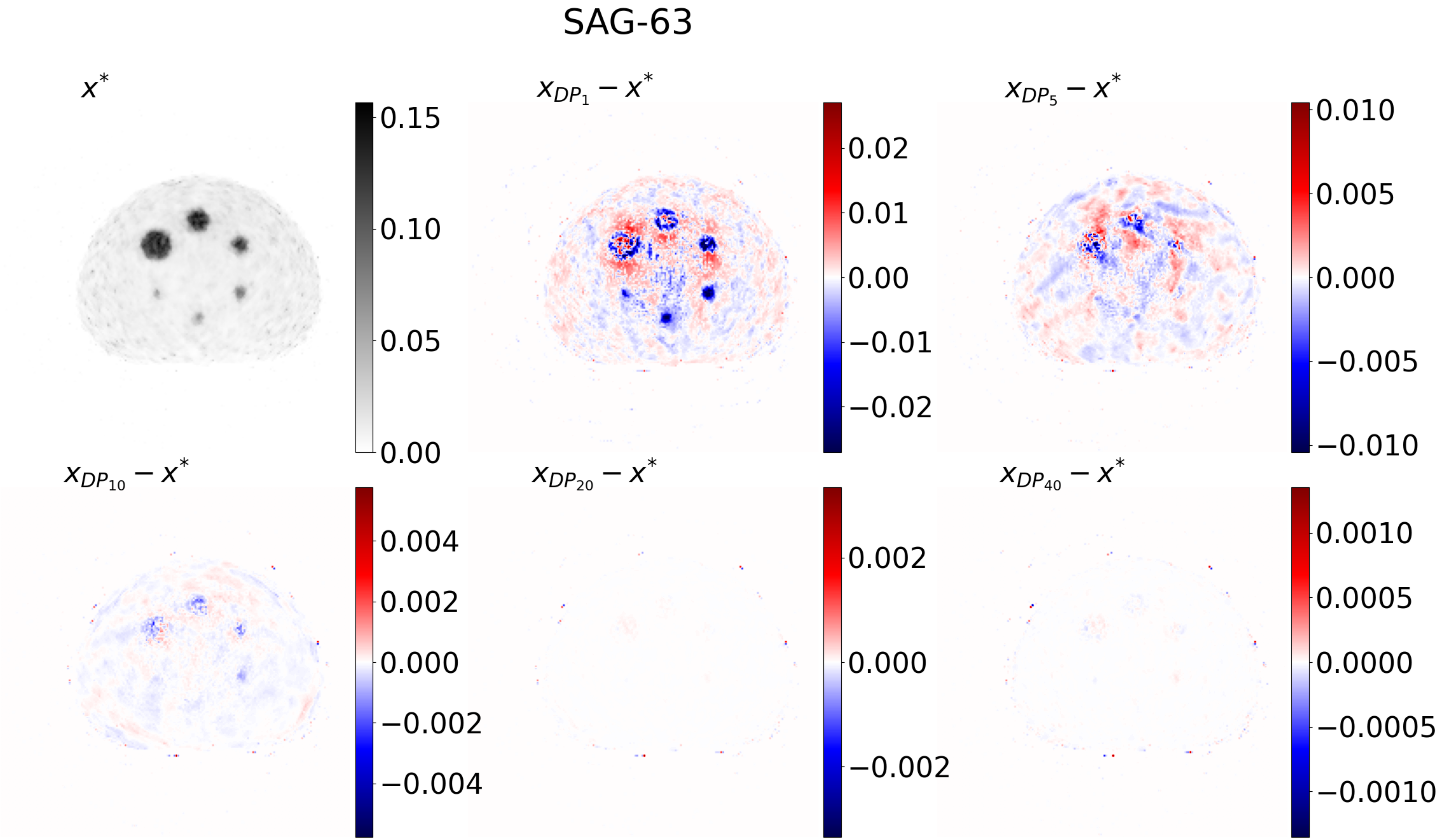
Applications (PET - Smooth optimisation)



Applications (PET - Smooth optimisation)



Applications (PET - Smooth optimisation)



Summary

- **Stochastic Optimisation Framework in CIL**
- **Flexible and user friendly design with Plug and Play Stochastic Estimators**
- **Different modalities CT, PET, SPECT, MRI**
- **Improvements for CIL Optimisation Module**

- **Website** https://www.ccpi.ac.uk/CIL
- **Docs** https://tomographicimaging.github.io/CIL/nightly/index.html
- **Discord** https://discord.gg/kmBcU2kebB

Thank you for your attention

https://epapoutsellis.github.io