From variational exit wave reconstruction to deep unfolding

Benjamin Berkels

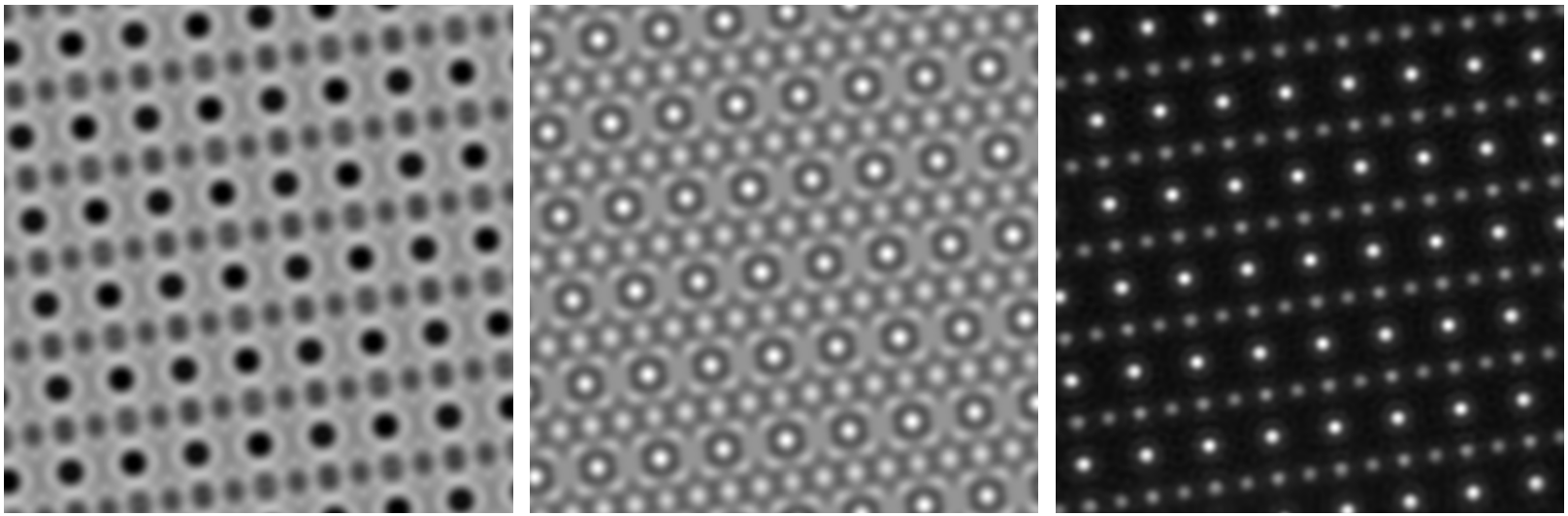
Aachen Institute for Advanced Study in Computational Engineering Science

RWTH AACHEN UNIVERSITY

SFB Seminar, January 25th, 2023

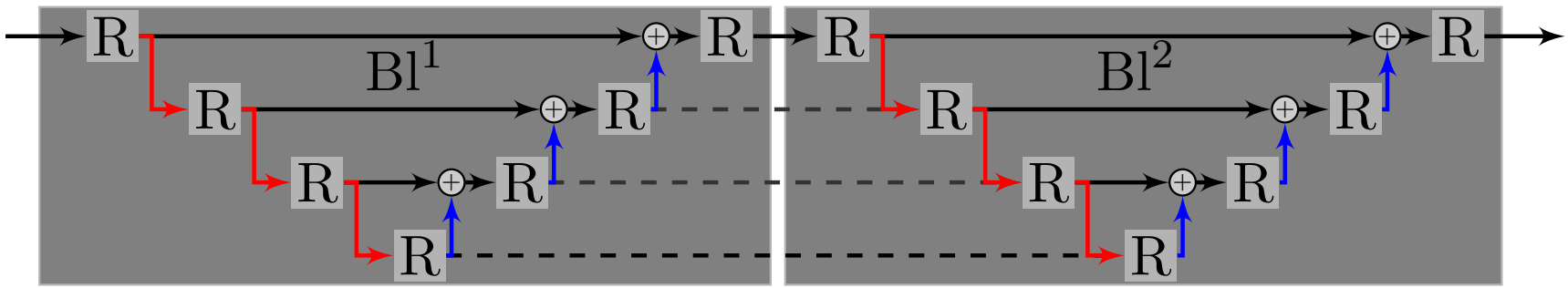
Outline

- Variational exit wave reconstruction



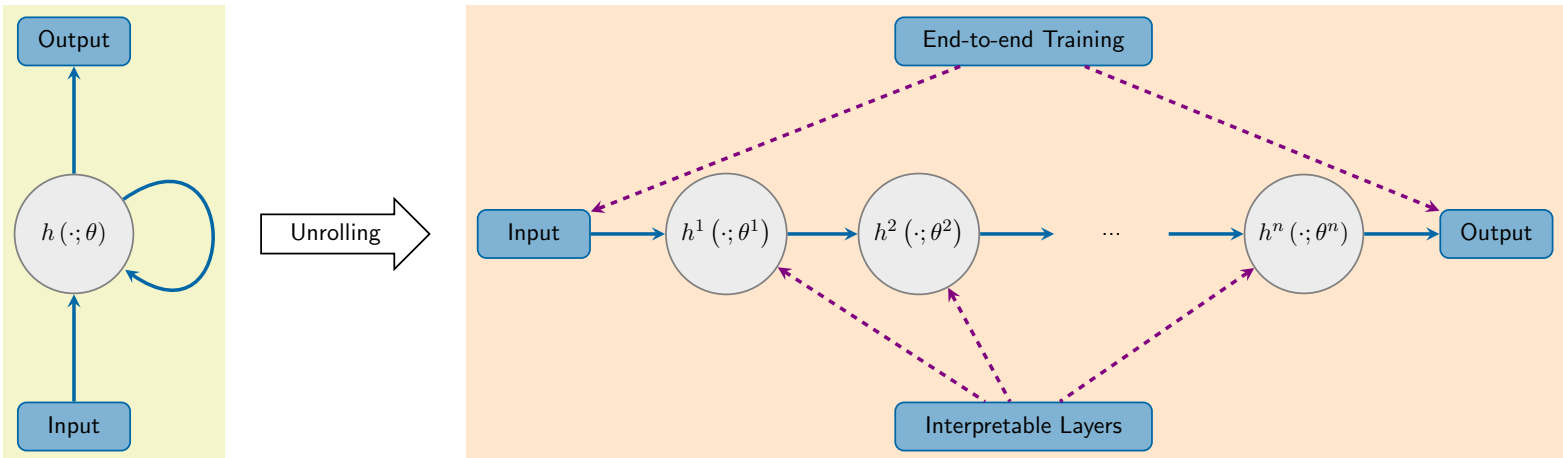
Outline

- Variational exit wave reconstruction
- Total Deep Variation for noisy exit wave reconstruction



Outline

- Variational exit wave reconstruction
- Total Deep Variation for noisy exit wave reconstruction
- Deep unfolding

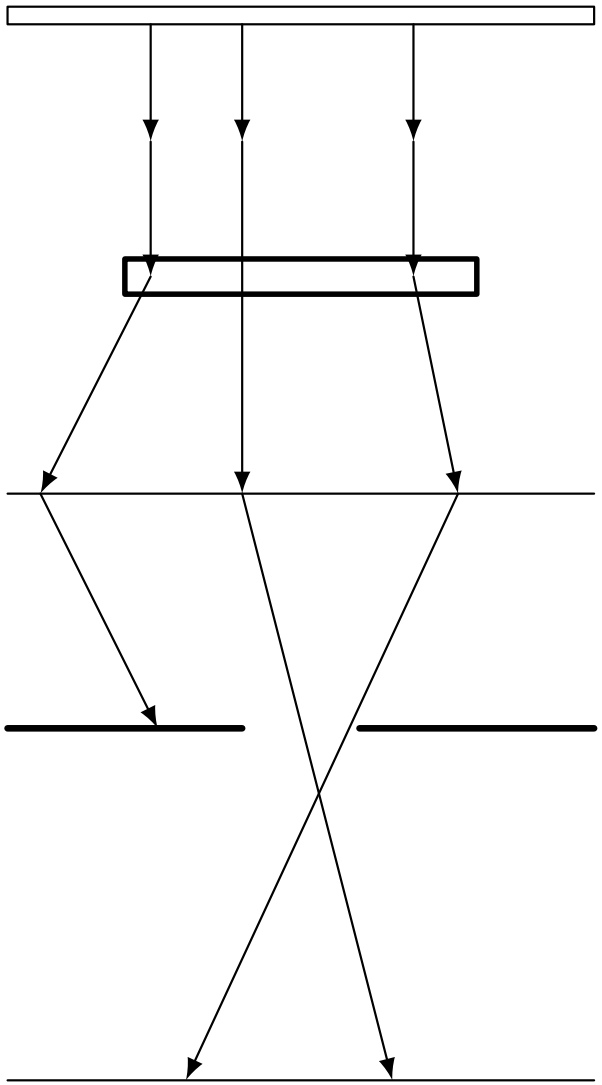


Joint exit wave reconstruction and image registration

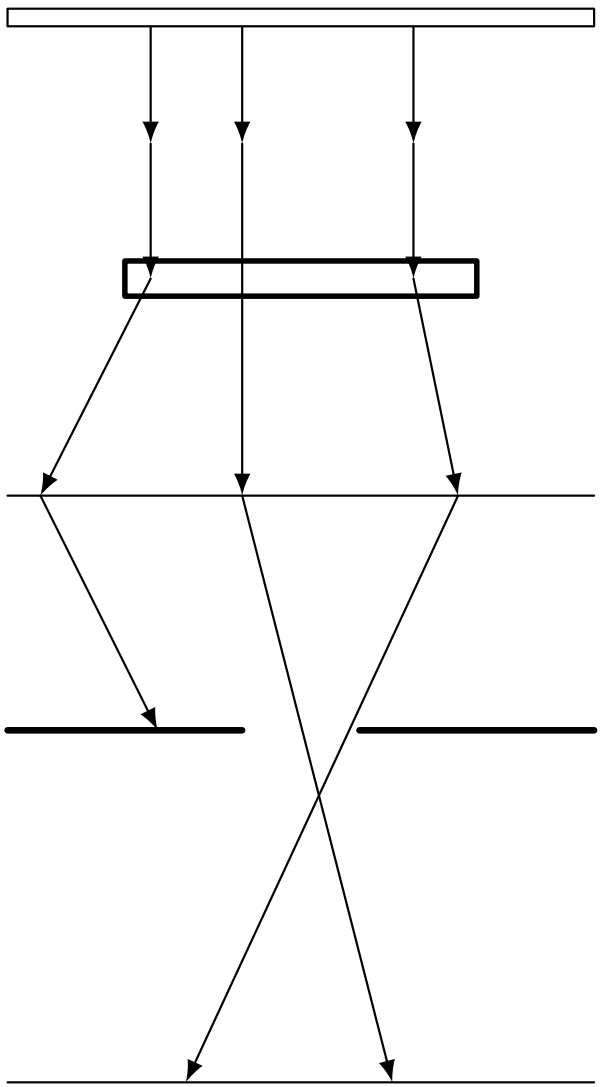
joint work with:

C. Doberstein (University of SC)

Background: transmission electron microscopy

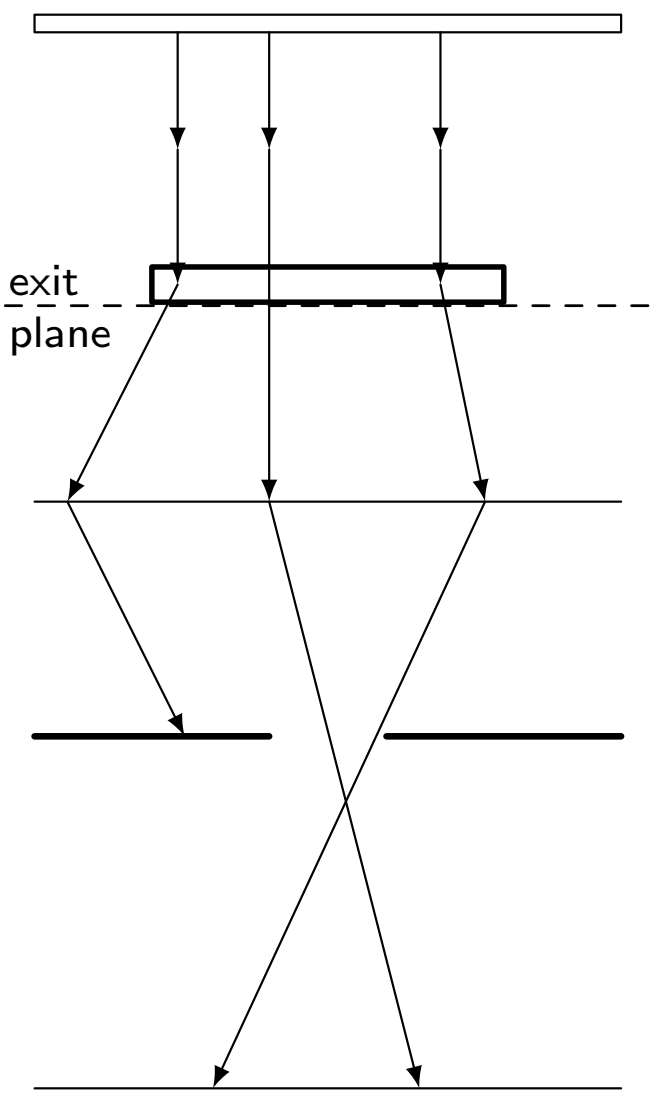


Background: transmission electron microscopy



(1) Electron gun: emits and accelerates a highly coherent beam of electrons.

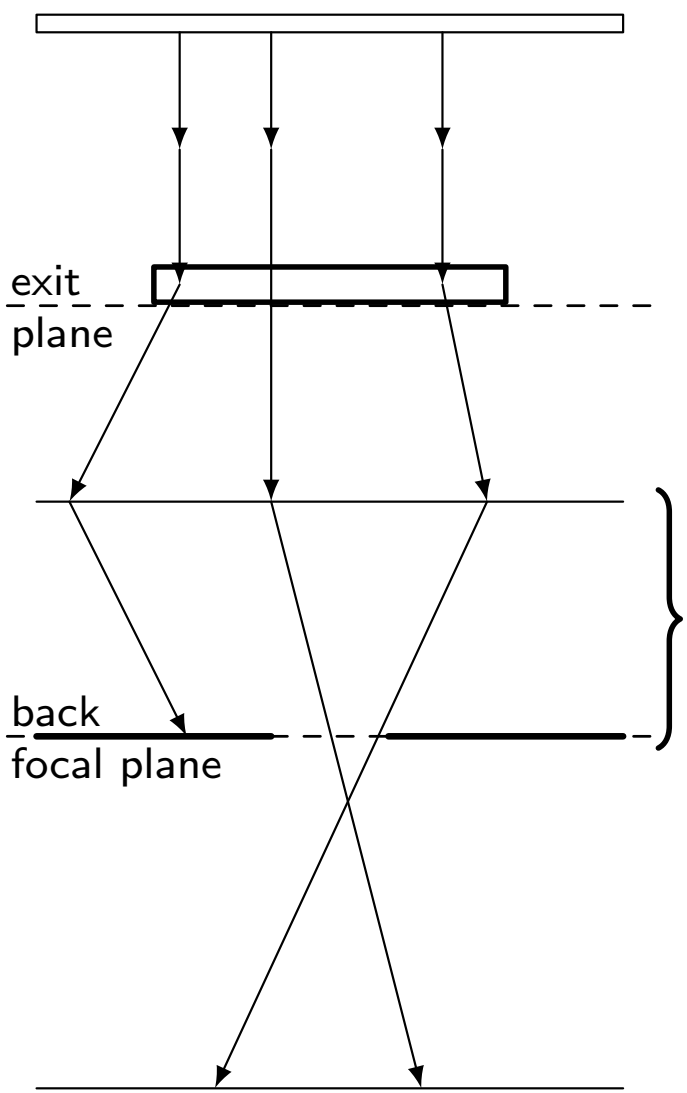
Background: transmission electron microscopy



(1) Electron gun: emits and accelerates a highly coherent beam of electrons.

(2) Specimen: transmission and scattering of the electrons.

Background: transmission electron microscopy

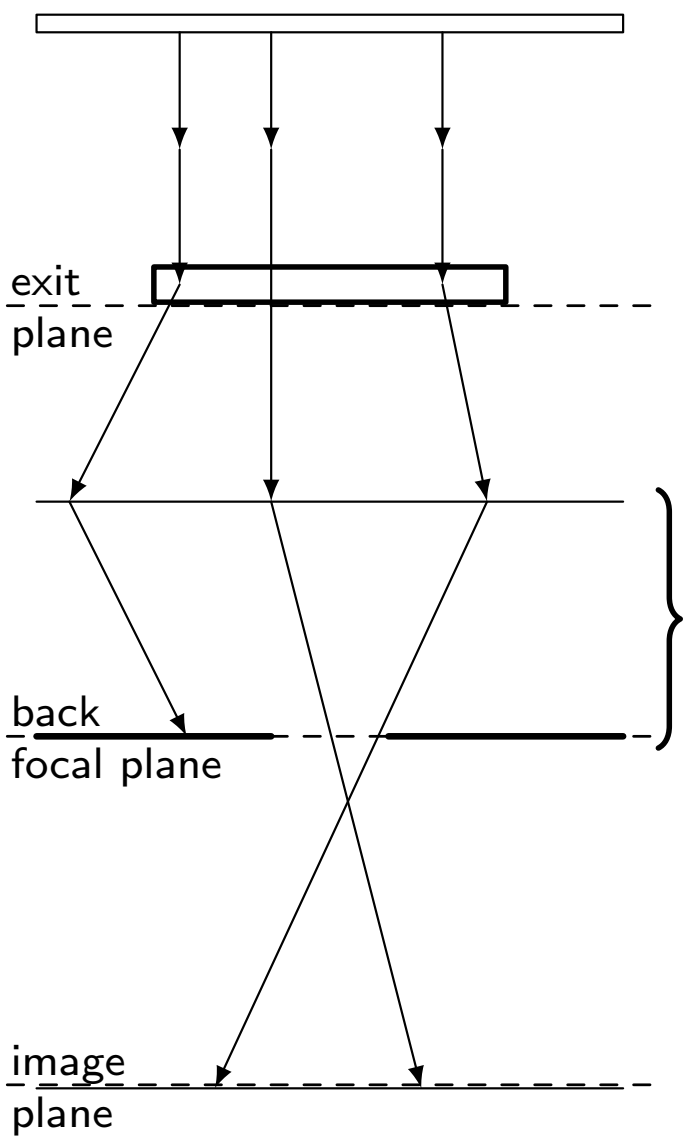


(1) Electron gun: emits and accelerates a highly coherent beam of electrons.

(2) Specimen: transmission and scattering of the electrons.

(3) Objective lens and aperture: focusing of the electron beam and removal of electrons that have been scattered to large angles.

Background: transmission electron microscopy



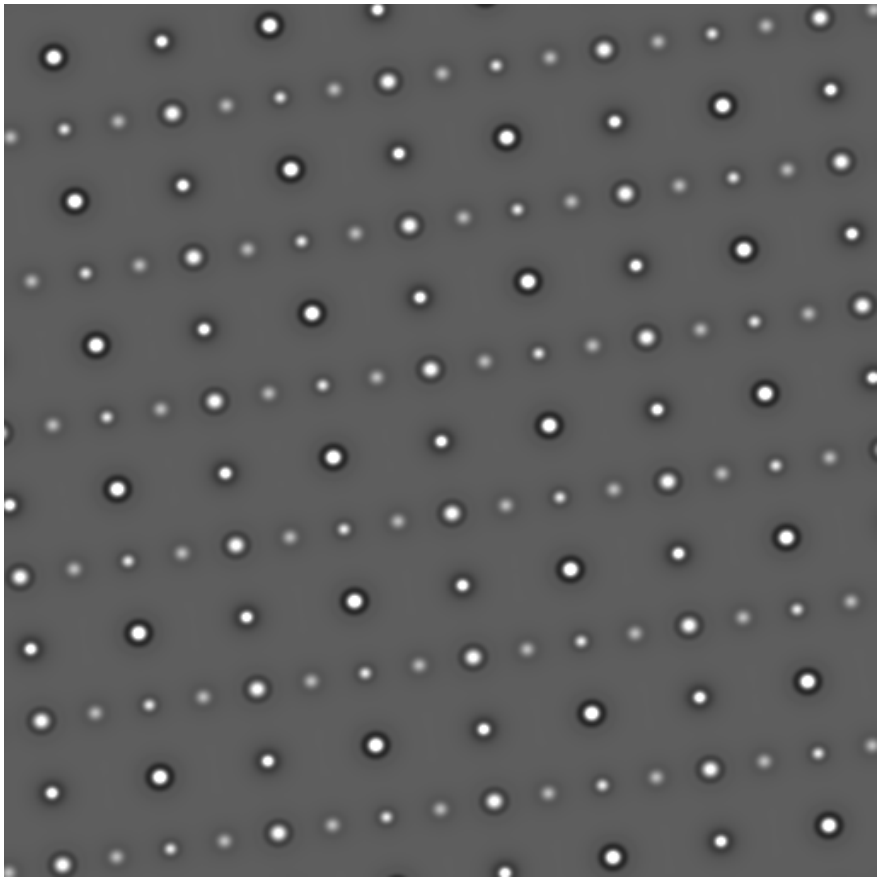
(1) Electron gun: emits and accelerates a highly coherent beam of electrons.

(2) Specimen: transmission and scattering of the electrons.

(3) Objective lens and aperture: focusing of the electron beam and removal of electrons that have been scattered to large angles.

(4) Camera: records the squared amplitude of the electron wave.

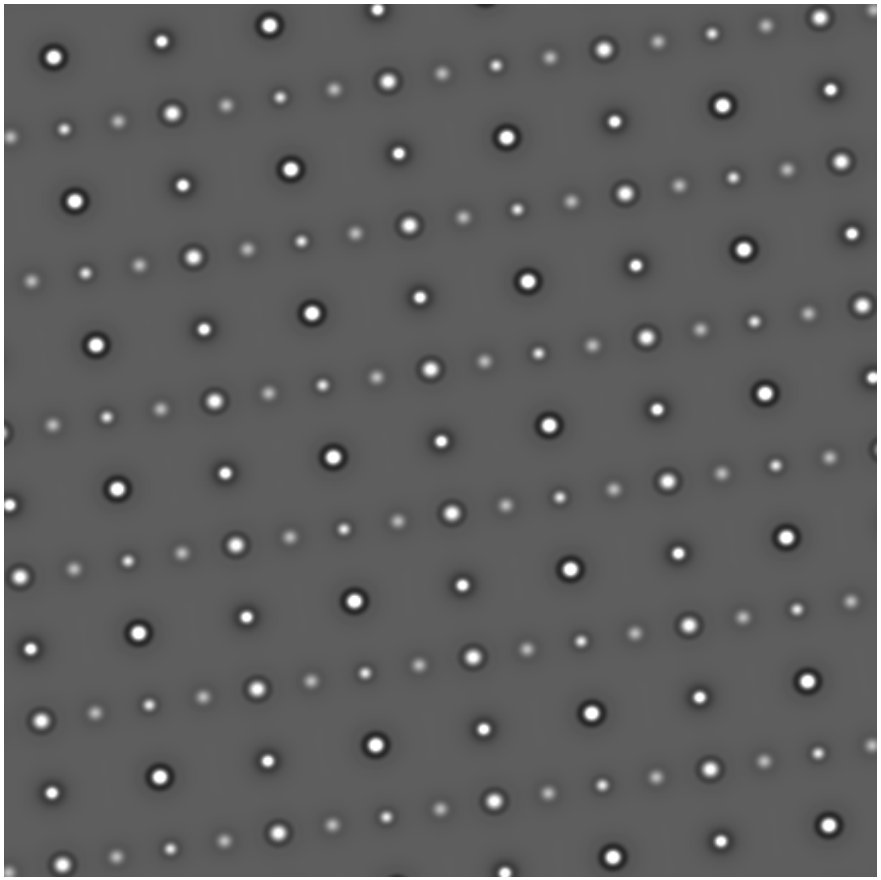
Background: transmission electron microscopy



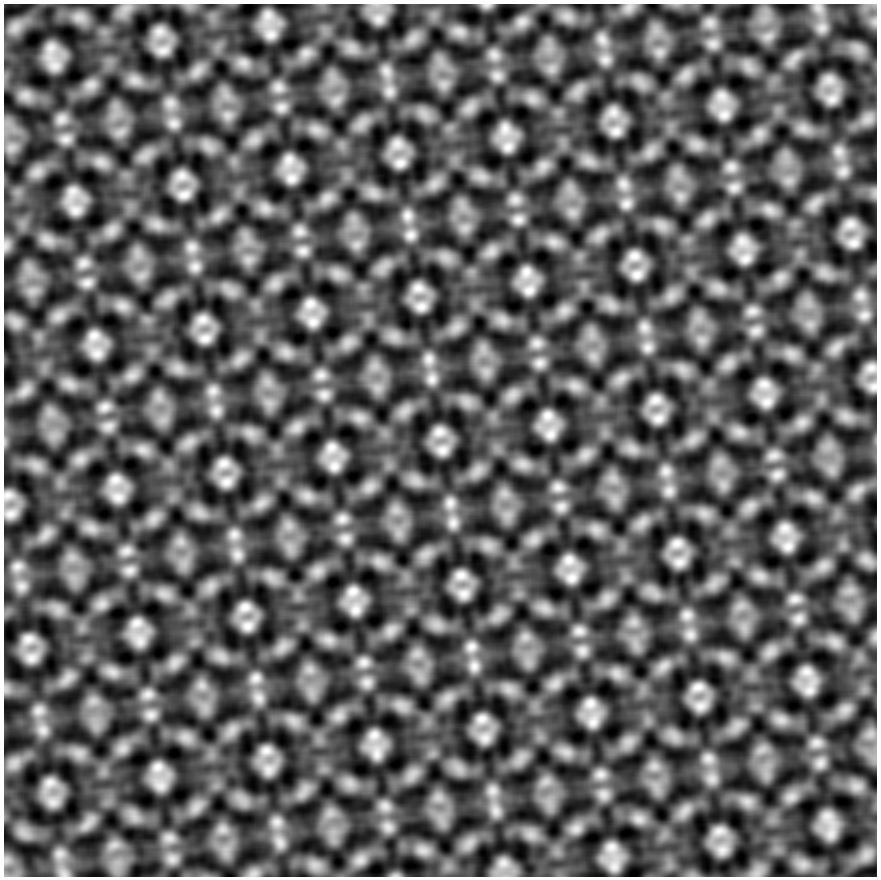
Ideal imaging conditions ($SrTiO_3$)

Images were simulated using DrProbe [Barthel Ultramic. '18]

Background: transmission electron microscopy



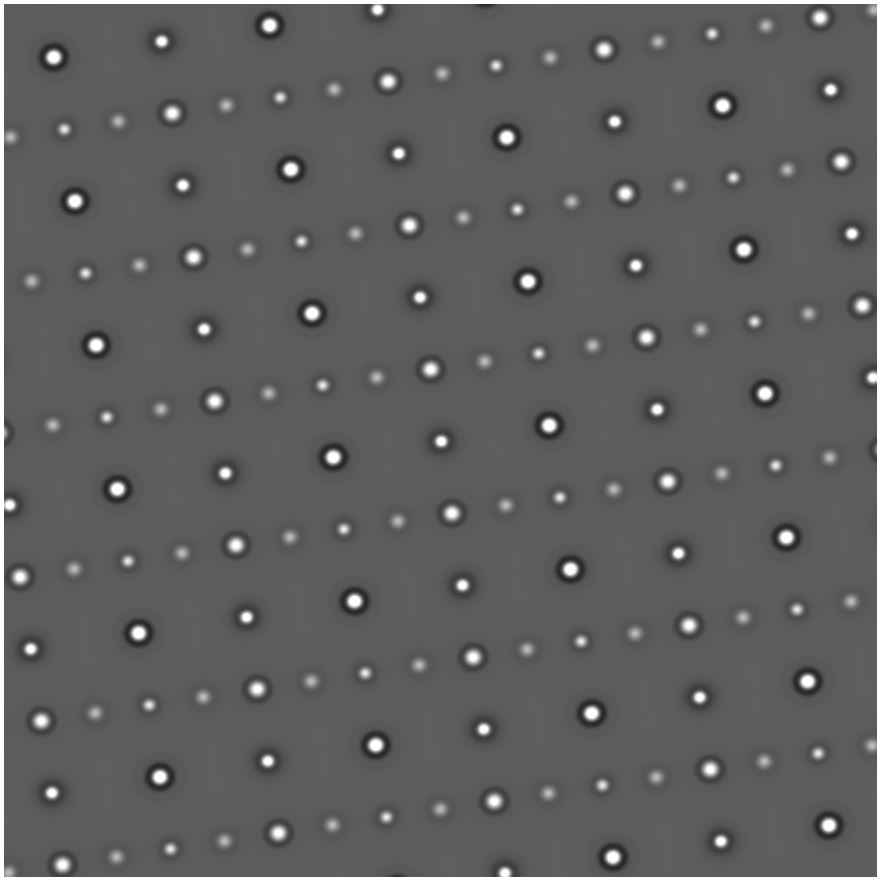
Ideal imaging conditions ($SrTiO_3$)



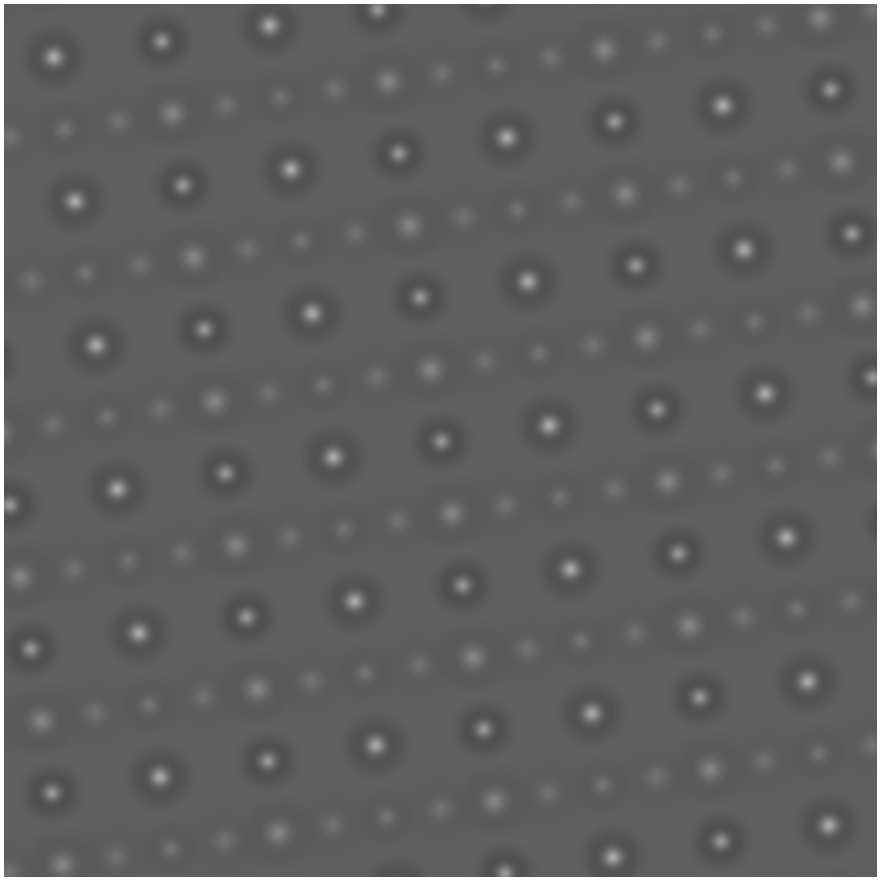
Spherical aberration

Images were simulated using DrProbe [Barthel Ultramic. '18]

Background: transmission electron microscopy



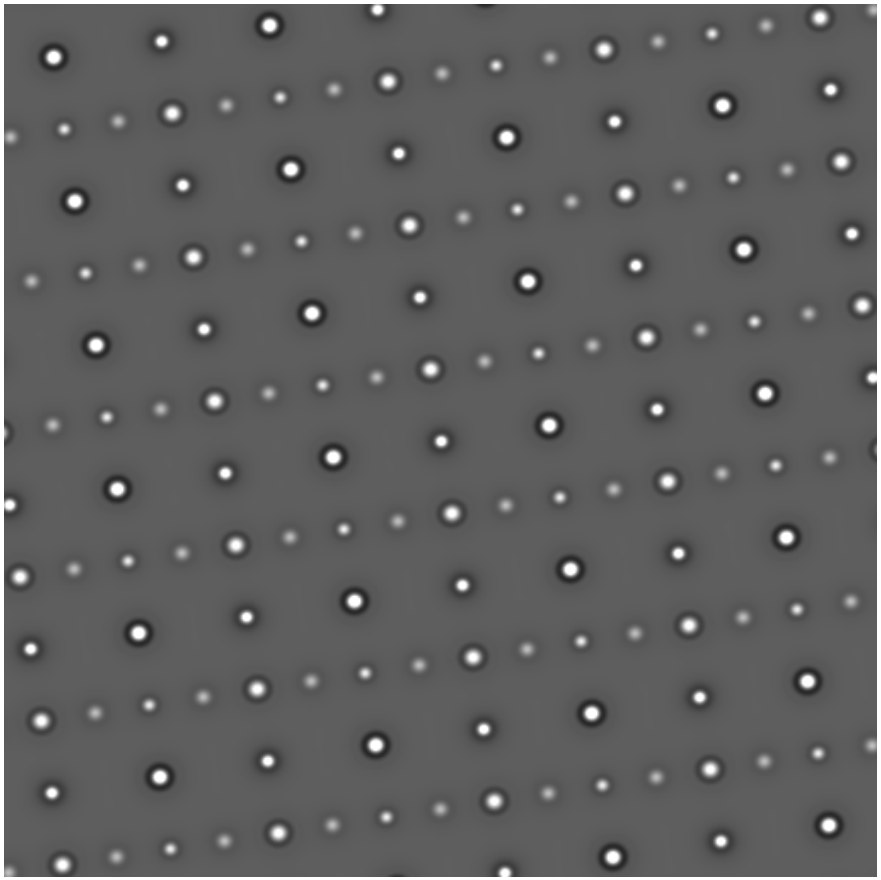
Ideal imaging conditions ($SrTiO_3$)



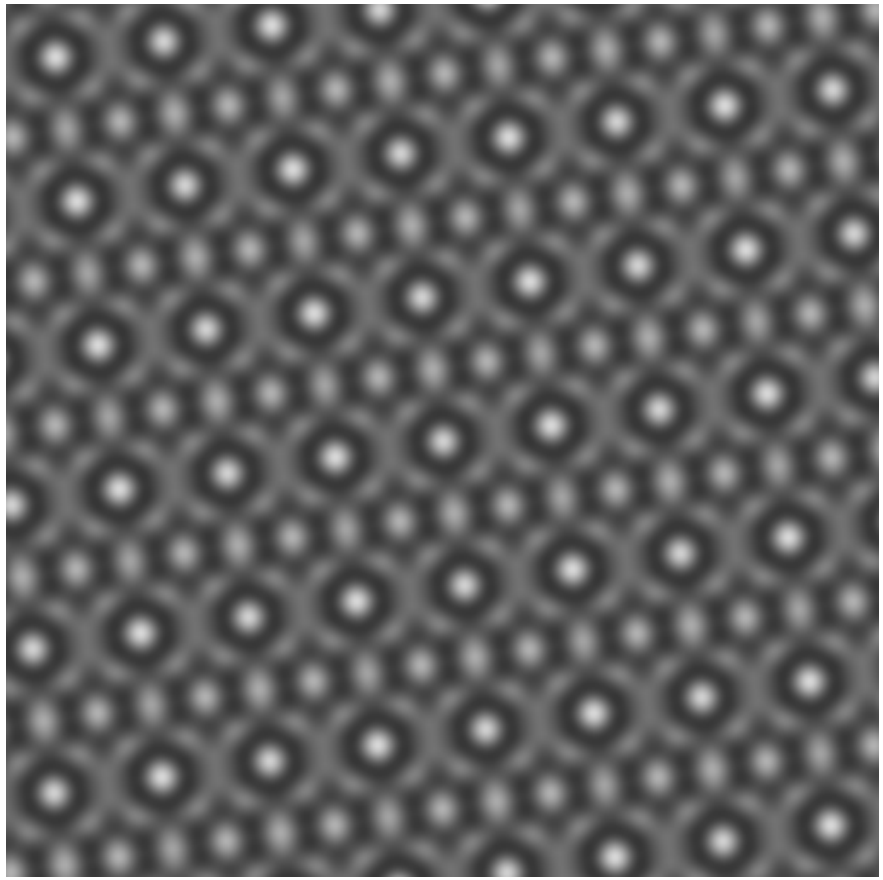
Partial coherence
(temporal and spatial)

Images were simulated using DrProbe [Barthel Ultramic. '18]

Background: transmission electron microscopy



Ideal imaging conditions ($SrTiO_3$)



Spherical aberration and partial coherence

Images were simulated using DrProbe [Barthel Ultramic. '18]

Background: transmission electron microscopy

TEM images are

- affected by aberrations of the objective lens
- blurred by partial coherence (temporal and spatial)
- real-valued, i. e. correspond to the amplitude of the image plane electron wave

Background: transmission electron microscopy

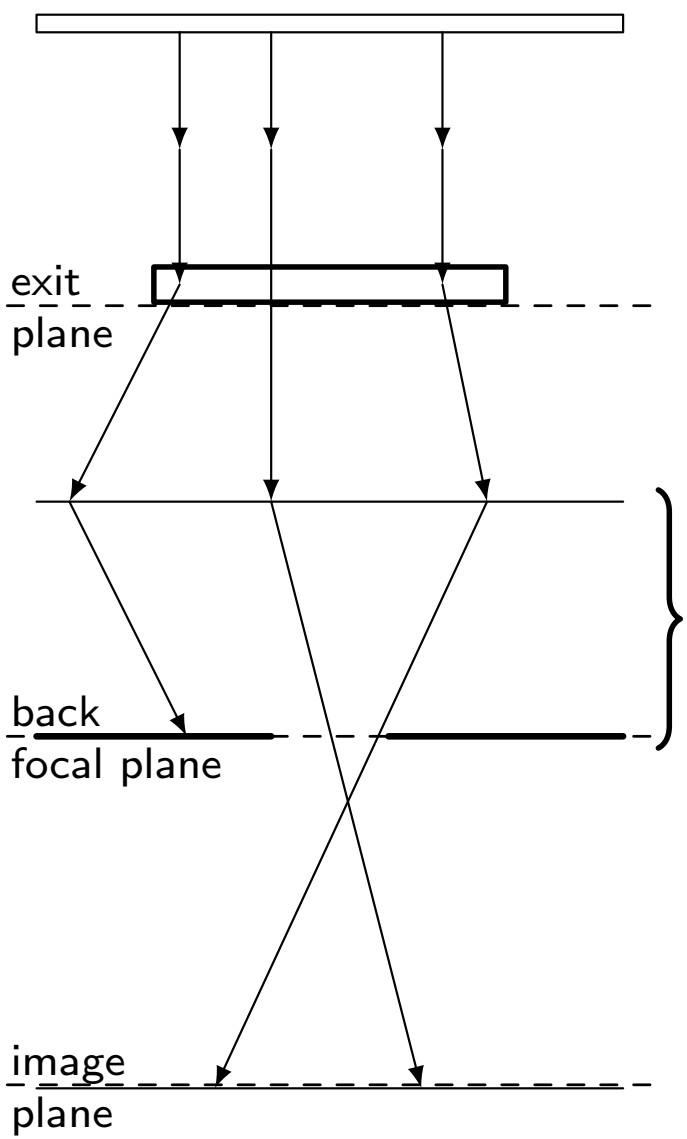
TEM images are

- affected by aberrations of the objective lens
- blurred by partial coherence (temporal and spatial)
- real-valued, i. e. correspond to the amplitude of the image plane electron wave

Definition

The electron wave at the exit plane of the specimen is called **exit wave**.

Background: transmission electron microscopy



(1) Electron gun: emits and accelerates a highly coherent beam of electrons.

(2) Specimen: transmission and scattering of the electrons.

(3) Objective lens and aperture: focusing of the electron beam and removal of electrons that have been scattered to large angles.

(4) Camera: records the squared amplitude of the electron wave.

Background: transmission electron microscopy

TEM images are

- affected by aberrations of the objective lens
- blurred by partial coherence (temporal and spatial)
- real-valued, i. e. correspond to the amplitude of the image plane electron wave

Definition

The electron wave at the exit plane of the specimen is called **exit wave**.

The exit wave is

- free from aberrations of the objective lens
- affected less strongly by partial temporal coherence
- complex-valued, i. e. consists of an amplitude and a phase

Background: transmission electron microscopy

TEM images are

- affected by aberrations of the objective lens
- blurred by partial coherence (temporal and spatial)
- real-valued, i. e. correspond to the amplitude of the image plane electron wave

Definition

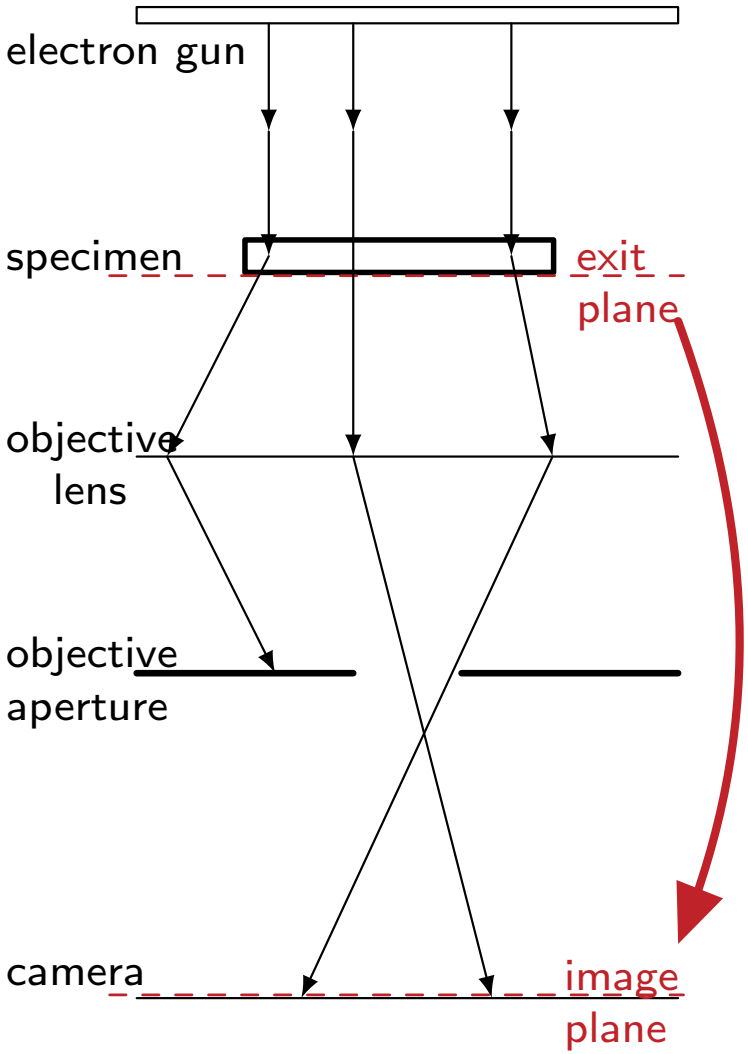
The electron wave at the exit plane of the specimen is called **exit wave**.

The exit wave is

- free from aberrations of the objective lens
- affected less strongly by partial temporal coherence
- complex-valued, i. e. consists of an amplitude and a phase

Question: How can we determine the exit wave?

The forward model: exit wave → TEM image



TEM image formation [Kirkland '10]:

$$\big(\mathcal{F}g\big)(x) = \int_{\mathbb{R}^2} \overline{\Psi(y)}\Psi(x+y)T(x+y,y)\,\mathrm{d}y$$

- g : real space TEM image $g \in L^2(\mathbb{R}^2, \mathbb{R}_{\geq 0})$
- Ψ : Fourier space exit wave $\Psi \in L^2(\mathbb{R}^2, \mathbb{C})$
- T : the microscope's transmission cross-coefficient (TCC) $T : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{C}$

$$(\mathcal{F}f)(\omega) = \int_{\mathbb{R}^d} f(x)e^{-2\pi i x\cdot\omega}\,\mathrm{d}x \text{ for } f \in L^1(\mathbb{R}^d, \mathbb{C})$$

The forward model: exit wave $\rightarrow$ TEM image

$$\big(\mathcal{F}g\big)(x) = \int_{\mathbb{R}^2} \overline{\Psi(y)}\Psi(x+y)T(x+y,y)\,\mathrm{d}y \qquad \forall\, x \in \mathbb{R}^2$$

The forward model: exit wave $\rightarrow$ TEM image

$$\big(\mathcal{F}g\big)(x) = \int_{\mathbb{R}^2} \overline{\Psi(y)}\Psi(x+y)T(x+y,y)\,\mathrm{d}y = \big(\Psi \star_T \Psi\big)(x) \quad \forall\, x \in \mathbb{R}^2$$

Definition

Let $f, g : \mathbb{R}^d \to \mathbb{C}$ and $w : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ be measurable functions. The weighted cross-correlation $f \star_w g$ is defined as

$$\big(f \star_w g\big)(x) := \int_{\mathbb{R}^d} \overline{f(y)}g(x+y)w(x+y,y)\,\mathrm{d}y$$

for all $x \in \mathbb{R}^d$ such that the integral is well-defined in the sense of Lebesgue-integrability.

The forward model: exit wave $\rightarrow$ TEM image

$$\big(\mathcal{F}g\big)(x) = \int_{\mathbb{R}^2} \overline{\Psi(y)}\Psi(x+y)T(x+y,y)\,\mathrm{d}y = \big(\Psi \star_T \Psi\big)(x) \quad \forall\, x \in \mathbb{R}^2$$

Definition

Let $f, g : \mathbb{R}^d \to \mathbb{C}$ and $w : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ be measurable functions. The weighted cross-correlation $f \star_w g$ is defined as

$$\big(f \star_w g\big)(x) := \int_{\mathbb{R}^d} \overline{f(y)}g(x+y)w(x+y,y)\,\mathrm{d}y$$

for all $x \in \mathbb{R}^d$ such that the integral is well-defined in the sense of Lebesgue-integrability.

What are useful properties of the weight function w?

The forward model: weight function spaces

Let $U \subseteq \mathbb{R}^d$

- Measurable and bounded weight functions:

$$W(U) := \{w : U \times U \to \mathbb{C} \mid w \text{ measurable and bounded}\}.$$

The forward model: weight function spaces

Let $U \subseteq \mathbb{R}^d$

- Measurable and bounded weight functions:

$$W(U) := \{w : U \times U \to \mathbb{C} \mid w \text{ measurable and bounded}\}.$$

- A weight $w \in W(U)$ is called $\star$-separable, if there exists an $N \in \mathbb{N}$ and measurable and bounded $v_j : U \to \mathbb{C}$ such that

$$w(x,y) = \sum_{j=1}^{N} v_j(x)\overline{v_j(y)} \qquad \forall\, x, y \in U.$$

Reduction to the ordinary cross-correlation:

$$\big(f \star_w g\big)(x) = \int \overline{f(y)} g(x+y) w(x+y, y)\, \mathrm{d}y = \sum_{j=1}^{N} \big((f v_j) \star (g v_j)\big)(x).$$

The forward model: weight function spaces

Let $U \subseteq \mathbb{R}^d$

- Measurable and bounded weight functions:

$$W(U) := \{w : U \times U \to \mathbb{C} \mid w \text{ measurable and bounded}\}.$$

- A weight $w \in W(U)$ is called $\star$-separable, if there exists an $N \in \mathbb{N}$ and measurable and bounded $v_j : U \to \mathbb{C}$ such that

$$w(x, y) = \sum_{j=1}^{N} v_j(x)\overline{v_j(y)} \qquad \forall\, x, y \in U.$$

- Functions that can be uniformly approximated by $\star$-separable weights:

$$W^+(U) := \Big\{ w \in W(U) \;\Big|\; \forall N \in \mathbb{N}\ \exists\, w_N \in W(U)\ \star\text{-separable:} \lim_{N\to\infty} \|w - w_N\|_\infty = 0 \Big\}.$$

The forward model: weight function properties

I. $w \in W^+(U)$ for $U \subseteq \mathbb{R}^d$ bounded

- $\overline{w(x,y)} = w(y,x)$ for all $x, y \in U$.
- $\mathcal{F}^{-1}(f \star_w f)$ is real-valued and nonnegative for all $f \in L^2(\mathbb{R}^d, \mathbb{C})$.
- The functional

$$F : L^2(U, \mathbb{C}) \times \mathbb{R}^d \to \mathbb{R}, \quad (f, t) \mapsto \|f \star_w f - \mu_t c\|_{L^2}^2$$

is weakly lower semi-continuous for all $c \in L^2(U, \mathbb{C})$, where $\mu_t : \mathbb{R}^d \to \mathbb{C}, \;\; x \mapsto \exp(2\pi i x \cdot t)$.

The forward model: weight function properties

II. $w \in W(U)$ continuous for $U \subseteq \mathbb{R}^d$ open with $\mathrm{vol}(\partial U) = 0$

- $f \star_w g$ is continuous for $f \in L^p(U, \mathbb{C})$, $g \in L^q(U, \mathbb{C})$ with $\frac{1}{p} + \frac{1}{q} = 1$.

The forward model: weight function properties

II. $w \in W(U)$ continuous for $U \subseteq \mathbb{R}^d$ open with $\mathrm{vol}(\partial U) = 0$

- $f \star_w g$ is continuous for $f \in L^p(U, \mathbb{C})$, $g \in L^q(U, \mathbb{C})$ with $\frac{1}{p} + \frac{1}{q} = 1$.

III. $w \in W(U)$ for $U \subseteq \mathbb{R}^d$ bounded such that $\exists\, c > 0 : w(x, x) \geq c$ for all $x \in U$

- Assume that w additionally satisfies II. Then

$$\|f \star_w f\|_{L^2} = 0 \qquad \Longleftrightarrow \qquad \|f\|_{L^2} = 0$$

for all $f \in L^2(U, \mathbb{C})$.

The forward model: weight function properties

Desired properties of the TCC T: [Doberstein, B. IP '19][Doberstein '20]

Let $U \subseteq \mathbb{R}^2$ be an open and bounded set with $\mathrm{vol}(\partial U) = 0$.

I. $T \in W^+(U)$

II. T is continuous on $U \times U$

III. $\exists\, c > 0 : T(x, x) \geq c$ for all $x \in U$

The forward model: weight function properties

Desired properties of the TCC T: [Doberstein, B. IP '19][Doberstein '20]

Let $U \subseteq \mathbb{R}^2$ be an open and bounded set with $\mathrm{vol}(\partial U) = 0$.

I. $T \in W^+(U)$

II. T is continuous on $U \times U$

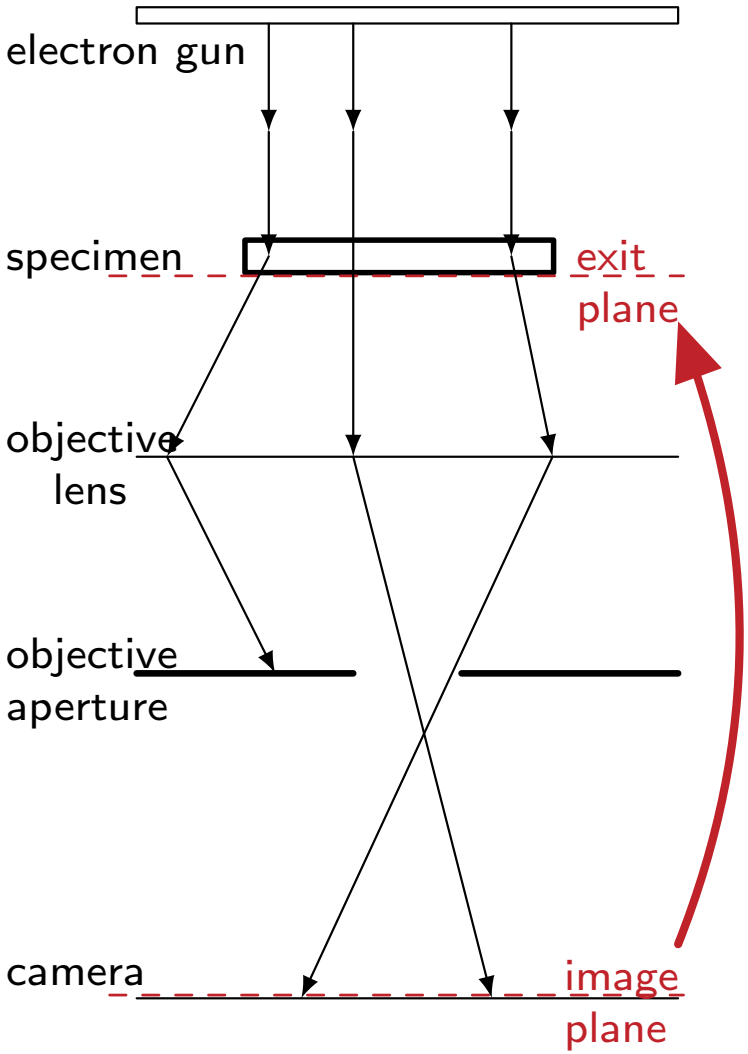
III. $\exists\, c > 0 : T(x, x) \geq c$ for all $x \in U$

In practice, the microscope's true TCC is unknown and approximations are used instead. The TCCs

- by Ishizuka [Ishizuka Ultramic. '80],
- from the MAL algorithm [Coene, Thust, Op de Beeck, Van Dyck Ultramic. '96] and
- from the quasi-coherent and coherent imaging approximations

all satisfy the properties I – III above.

The inverse problem: image series → exit wave



Exit wave reconstruction

Given: A series of $N \in \mathbb{N}$ images $g_1, \dots, g_N \in L^2(\mathbb{R}^2, \mathbb{R}_{\geq 0})$ and associated TCCs $T_1, \dots, T_N$.

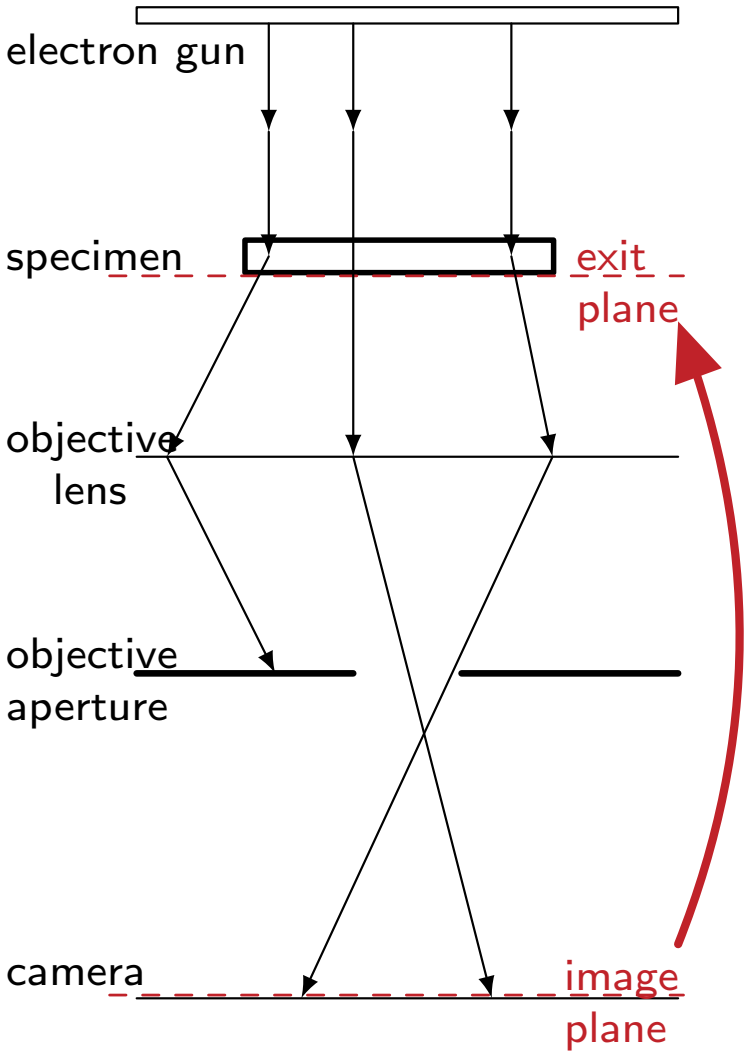
Task: determine a function (the exit wave) $\Psi \in L^2(\mathbb{R}^2, \mathbb{C})$ and image shifts $t_1, \dots, t_N \in \mathbb{R}^2$ such that

$$g_j \circ \phi_{t_j} \approx \mathcal{F}^{-1}(\Psi \star_{T_j} \Psi)$$

for all $j = 1, \dots, N$, where

$$\phi_y : \mathbb{R}^2 \to \mathbb{R}^2, \quad x \mapsto x + y \quad \forall\, y \in \mathbb{R}^2.$$

The inverse problem: image series → exit wave



Exit wave reconstruction

Given: A series of $N \in \mathbb{N}$ images $g_1, \dots, g_N \in L^2(\mathbb{R}^2, \mathbb{R}_{\geq 0})$ and associated TCCs $T_1, \dots, T_N$.

Task: determine a function (the exit wave) $\Psi \in L^2(\mathbb{R}^2, \mathbb{C})$ and image shifts $t_1, \dots, t_N \in \mathbb{R}^2$ such that

$$g_j \circ \phi_{t_j} \approx \mathcal{F}^{-1}(\Psi \star_{T_j} \Psi)$$

for all $j = 1, \dots, N$, where

$$\phi_y : \mathbb{R}^2 \to \mathbb{R}^2, \quad x \mapsto x + y \quad \forall\, y \in \mathbb{R}^2.$$

Here, $g_1, \dots, g_N$ will be a **focus** series.

The inverse problem: image series $\rightarrow$ exit wave

The variational approach:

- MIMAP: [Kirkland Ultramic. '84]

$$J^{\mathrm{MIMAP}}[\Psi] = \sum_{j=1}^{N} \left\| \Psi \star_{T_j^{\mathrm{MIMAP}}} \Psi - \mathcal{F}(g_j) \right\|_{L^2}^2 + c \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

- MAL: [Coene, Thust, Op de Beeck, Van Dyck Ultramic. '96]

$$J^{\mathrm{MAL}}[\Psi] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j^{\mathrm{MAL}}} \Psi - \mathcal{F}(g_j \circ \phi_{t_j}) \right\|_{L^2}^2$$

The inverse problem: image series $\rightarrow$ exit wave

The variational approach:

- MIMAP: [Kirkland Ultramic. '84]

$$J^{\text{MIMAP}}[\Psi] = \sum_{j=1}^{N} \left\| \Psi \star_{T_j^{\text{MIMAP}}} \Psi - \mathcal{F}(g_j) \right\|_{L^2}^2 + c \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

- MAL: [Coene, Thust, Op de Beeck, Van Dyck Ultramic. '96]

$$J^{\text{MAL}}[\Psi] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j^{\text{MAL}}} \Psi - \mathcal{F}(g_j \circ \phi_{t_j}) \right\|_{L^2}^2$$

- Joint reconstruction and registration: [Doberstein, B. Inverse Problems '19]

$$E_\sigma[\Psi, t] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j} \Psi - \mu_{t_j} \mathcal{F}(g_j) \right\|_{L^2}^2 + \sigma \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

Properties of the objective functional

Theorem If $\sigma > 0$ and the TCCs satisfy I, then there is a minimizer of

$$E_\sigma : L^2(A, \mathbb{C}) \times \overline{B_r(0)}^N \to \mathbb{R} \qquad \left(A = B_{r_a}(0) \text{ and } r, r_a > 0\right)$$

$$E_\sigma[\Psi, t] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j} \Psi - \mu_{t_j} \mathcal{F}(g_j) \right\|_{L^2}^2 + \sigma \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

Properties of the objective functional

Theorem If $\sigma > 0$ and the TCCs satisfy I, then there is a minimizer of

$$E_\sigma : L^2(A, \mathbb{C}) \times \overline{B_r(0)}^N \to \mathbb{R} \qquad \big(A = B_{r_a}(0) \text{ and } r, r_a > 0\big)$$

$$E_\sigma[\Psi, t] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j} \Psi - \mu_{t_j} \mathcal{F}(g_j) \right\|_{L^2}^2 + \sigma \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

Theorem Let $N \in \mathbb{N}$, $U \subseteq \mathbb{R}^2$ measurable with $\operatorname{int}(U) \neq \emptyset$ and $T_1, \ldots, T_N \in W(U)$ $\star$-separable TCCs with

$$T_j(v, w) = \sum_{k=1}^{K} h_{j,k}(v) \overline{h_{j,k}(w)} \qquad \forall\, v, w \in U$$

for $K \in \mathbb{N}$ and measurable, bounded $h_{j,k} : \mathbb{R}^2 \to \mathbb{C}$. If $\mathcal{F}^{-1}(h_{j,k}) \in L^1(\mathbb{R}^2, \mathbb{C})$ for all $1 \leq j \leq N$ and all $1 \leq k \leq K$, then

$$L^2(U, \mathbb{C}) \to \mathbb{R}, \qquad \Psi \mapsto \sum_{j=1}^{N} \left\| \Psi \star_{T_j} \Psi - G_j \right\|_{L^2}^2$$

is not coercive for any $G_1, \ldots, G_N \in L^2(\mathbb{R}^2, \mathbb{C})$.

The inverse problem: convexity

$$E_\sigma[\Psi, t] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j} \Psi - \mu_{t_j} \mathcal{F}(g_j) \right\|_{L^2}^2 + \sigma \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

Lemma

Let $t \in \overline{B_r(0)}^N$. For all $\Psi, \Phi \in L^2(A, \mathbb{C})$ with $\Phi \neq 0$ there are coefficients $(C^j_{\Psi,\Phi,t,\sigma})_{j=0,\dots,4} \in \mathbb{R}^5$ such that

$$E_\sigma[\Psi + \tau\Phi, t] = \sum_{j=0}^{4} C^j_{\Psi,\Phi,t,\sigma} \tau^j$$

for all $\tau \in \mathbb{R}$. If the TCCs satisfy II and III, then $C^4_{\Psi,\Phi,t,\sigma} > 0$.

The inverse problem: convexity

$$E_\sigma[\Psi, t] = \frac{1}{N} \sum_{j=1}^{N} \left\| \Psi \star_{T_j} \Psi - \mu_{t_j} \mathcal{F}(g_j) \right\|_{L^2}^2 + \sigma \left\| \Psi - \Psi_M \right\|_{L^2}^2$$

Lemma

Let $t \in \overline{B_r(0)}^N$. For all $\Psi, \Phi \in L^2(A, \mathbb{C})$ with $\Phi \neq 0$ there are coefficients $(C^j_{\Psi,\Phi,t,\sigma})_{j=0,\dots,4} \in \mathbb{R}^5$ such that

$$E_\sigma[\Psi + \tau\Phi, t] = \sum_{j=0}^{4} C^j_{\Psi,\Phi,t,\sigma} \tau^j$$

for all $\tau \in \mathbb{R}$. If the TCCs satisfy II and III, then $C^4_{\Psi,\Phi,t,\sigma} > 0$.

Theorem

Let $t \in \overline{B_r(0)}^N$ and assume that the TCCs satisfy II and III. If $g_j \neq 0$ for at least one $j \in \{1, \dots, N\}$, then $\Psi \mapsto E_0[\Psi, t]$ is not convex.

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

- phase transfer function $p_Z \in L^\infty(\mathbb{R}^2, \mathbb{C})$

$$p_Z(v) := \exp(-2\pi i \chi_Z(v))$$

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

- phase transfer function $p_Z \in L^\infty(\mathbb{R}^2, \mathbb{C})$

$$p_Z(v) := \exp(-2\pi i \chi_Z(v))$$

- isotropic aberration function $\chi_Z \in C^\infty(\mathbb{R}^2, \mathbb{R})$

$$\chi_Z(v) := \tfrac{1}{2} Z\lambda \|v\|_2^2 + \tfrac{1}{4} C_s \lambda^3 \|v\|_2^4$$

with focus Z, electron wavelength λ and spherical aberration coeff. C_s

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

- phase transfer function $p_Z \in L^\infty(\mathbb{R}^2, \mathbb{C})$

$$p_Z(v) := \exp(-2\pi i \chi_Z(v))$$

- isotropic aberration function $\chi_Z \in C^\infty(\mathbb{R}^2, \mathbb{R})$

$$\chi_Z(v) := \tfrac{1}{2} Z\lambda \|v\|_2^2 + \tfrac{1}{4} C_s \lambda^3 \|v\|_2^4$$

  with focus Z, electron wavelength λ and spherical aberration coeff. C_s

- aperture function $a \in L^\infty(\mathbb{R}^2, \mathbb{R})$

$$a(v) := \begin{cases} 1 & \lambda\|v\|_2 < \alpha_{\max}, \\ 0 & \text{else}, \end{cases}$$

  with maximum aperture semiangle $\alpha_{\max} > 0$

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

- spatial damping envelope $E_{s,Z} \in L^\infty(\mathbb{R}^2, \mathbb{R})$

$$E_{s,Z}(v) := \exp\left(-\left(\frac{\pi\alpha}{\lambda}\right)^2 \|\nabla\chi_Z(v)\|_2^2\right)$$

  for a sufficiently small half angle of beam convergence $\alpha > 0$.

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

- spatial damping envelope $E_{s,Z} \in L^\infty(\mathbb{R}^2, \mathbb{R})$

$$E_{s,Z}(v) := \exp\left(-\left(\frac{\pi\alpha}{\lambda}\right)^2 \|\nabla\chi_Z(v)\|_2^2\right)$$

  for a sufficiently small half angle of beam convergence $\alpha > 0$.

- temporal damping envelope $E_t \in L^\infty(\mathbb{R}^2 \times \mathbb{R}^2, \mathbb{R})$

$$E_t(v,w) := \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\Delta} \exp\left(-\frac{x^2}{2\Delta^2}\right) \exp\left(-\pi i\lambda x(\|v\|_2^2 - \|w\|_2^2)\right) \mathrm{d}x$$

  for a fixed focus spread $\Delta > 0$

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

- spatial damping envelope $E_{s,Z} \in L^\infty(\mathbb{R}^2, \mathbb{R})$

$$E_{s,Z}(v) := \exp\left(-\left(\frac{\pi\alpha}{\lambda}\right)^2 \|\nabla\chi_Z(v)\|_2^2\right)$$

  for a sufficiently small half angle of beam convergence $\alpha > 0$.

- temporal damping envelope $E_t \in L^\infty(\mathbb{R}^2 \times \mathbb{R}^2, \mathbb{R})$

$$E_t(v,w) := \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi}\Delta} \exp\left(-\frac{x^2}{2\Delta^2}\right) \exp\left(-\pi i\lambda x(\|v\|_2^2 - \|w\|_2^2)\right) \mathrm{d}x$$

  for a fixed focus spread $\Delta > 0$

- Note that $E_t(v,w) = \int_{\mathbb{R}} \widetilde{E}_t(v,x)\overline{\widetilde{E}_t(w,x)}\,\mathrm{d}x$ with

$$\widetilde{E}_t(v,x) := \sqrt{\frac{1}{\sqrt{2\pi}\Delta} \exp\left(-\frac{x^2}{2\Delta^2}\right)} \exp\left(-\pi i\lambda x\|v\|_2^2\right).$$

The TCC from the MAL algorithm

Assumes half angle of beam convergence α small ($\alpha < 2 \times 10^{-5}$ mrad)

$$T_Z(v,w) = p_Z(v)\overline{p_Z(w)}a(v)\overline{a(w)}E_{s,Z}(v)\overline{E_{s,Z}(w)}E_t(v,w).$$

Finally, with $t_Z(v,x) := p(v)a(v)E_{s,Z}(v)\widetilde{E}_t(v,x)$, we get

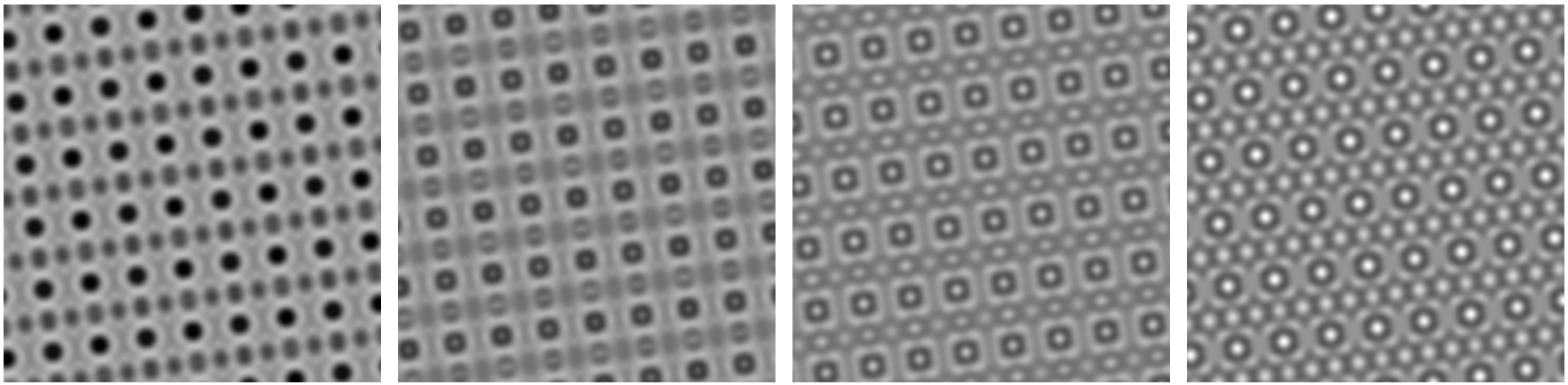
$$T_Z(v,w) = \int_{\mathbb{R}} t_Z(v,x)\overline{t_Z(w,x)}\,\mathrm{d}x\,.$$

Thus, $\Psi \star_{T_Z} \Psi$ can be recast by exploiting axial symmetry as follows

$$(\Psi \star_{T_Z} \Psi)(\omega) = \mathcal{F}\left(\int_{\mathbb{R}} |\mathcal{F}^{-1}(\Psi(\omega)t_Z(\omega,x))|^2\,\mathrm{d}x\right).$$

Numerical minimization of E_σ: artificial example

Input data

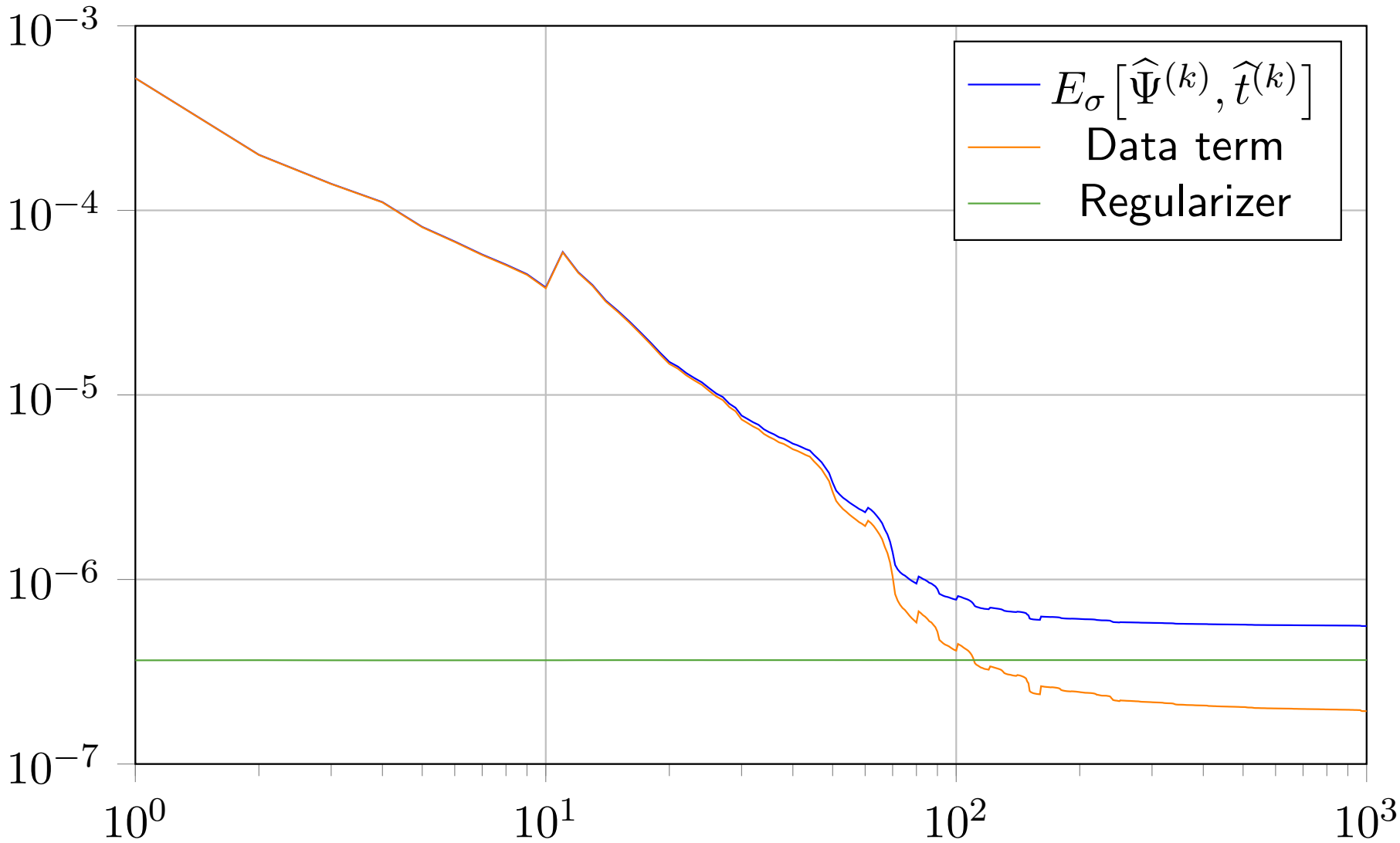


- $N = 10$ frames with varying focus, simulated with DrProbe
- $X = Y = 512$
- $T = T^{\mathsf{MAL}}$

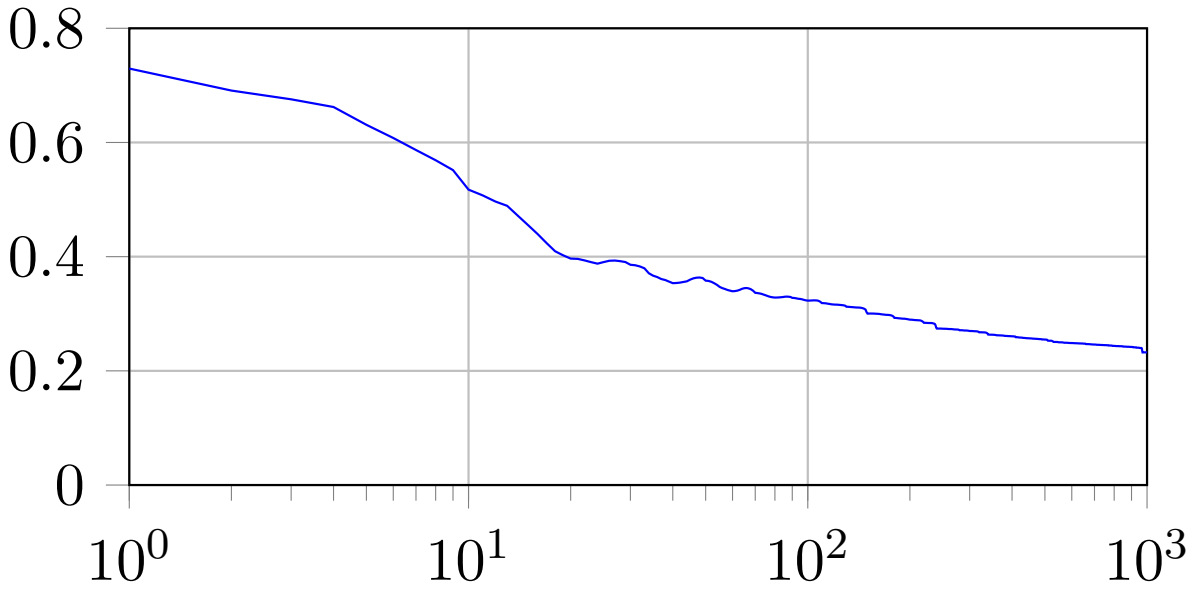
Exit wave reconstruction

- Joint reconstruction and image registration by minimizing E_σ
- Discretization as piecewise constant functions on a regular grid
- Minimization using nonlinear Fletcher-Reeves conjugate gradient descent
- Initial guess: constant exit wave and a rough translation estimate

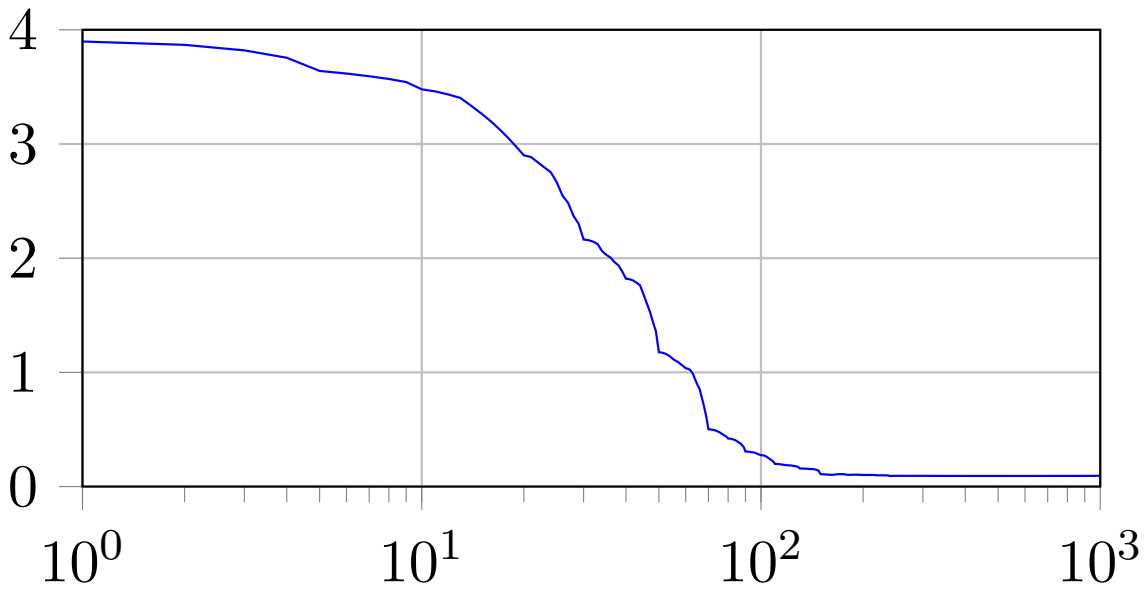
Numerical minimization of E_σ: artificial example (cont.)



Numerical minimization of E_σ: artificial example (cont.)



Distance of $\mathrm{IDFT}\left(\widehat{\Psi}^{(k)}\right)$ to the correct exit wave in the supremum norm

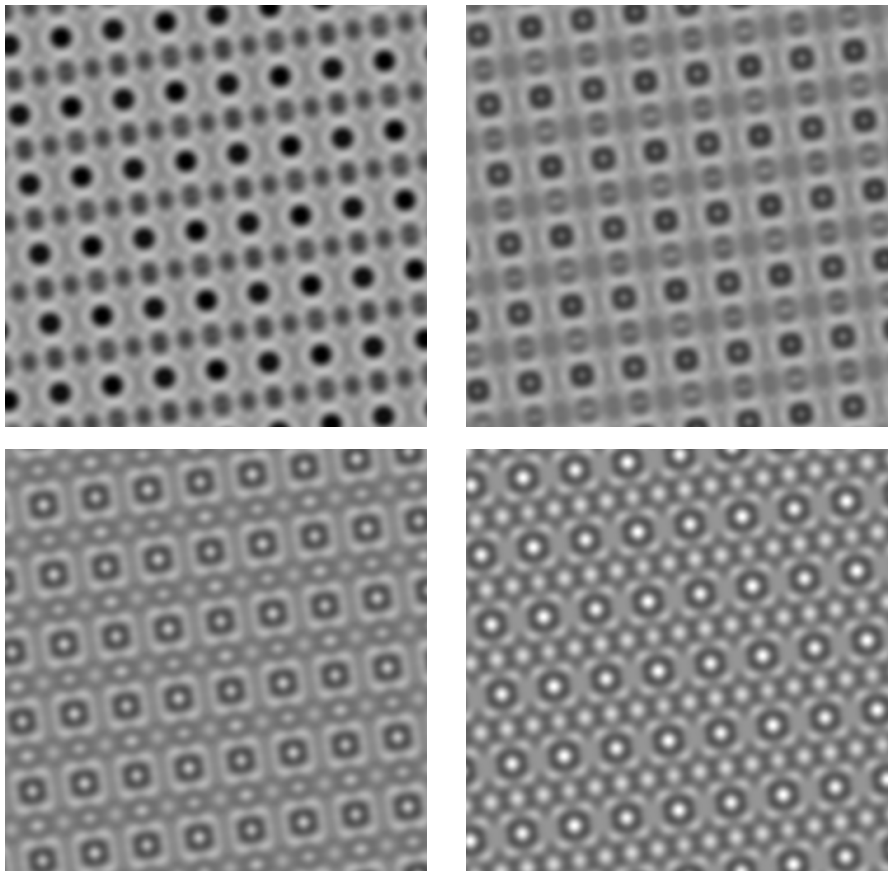


Distance of $\widehat{t}^{(k)}$ to the correct translations in the supremum norm (in pixel)

Numerical minimization of E_σ: artificial example (cont.)

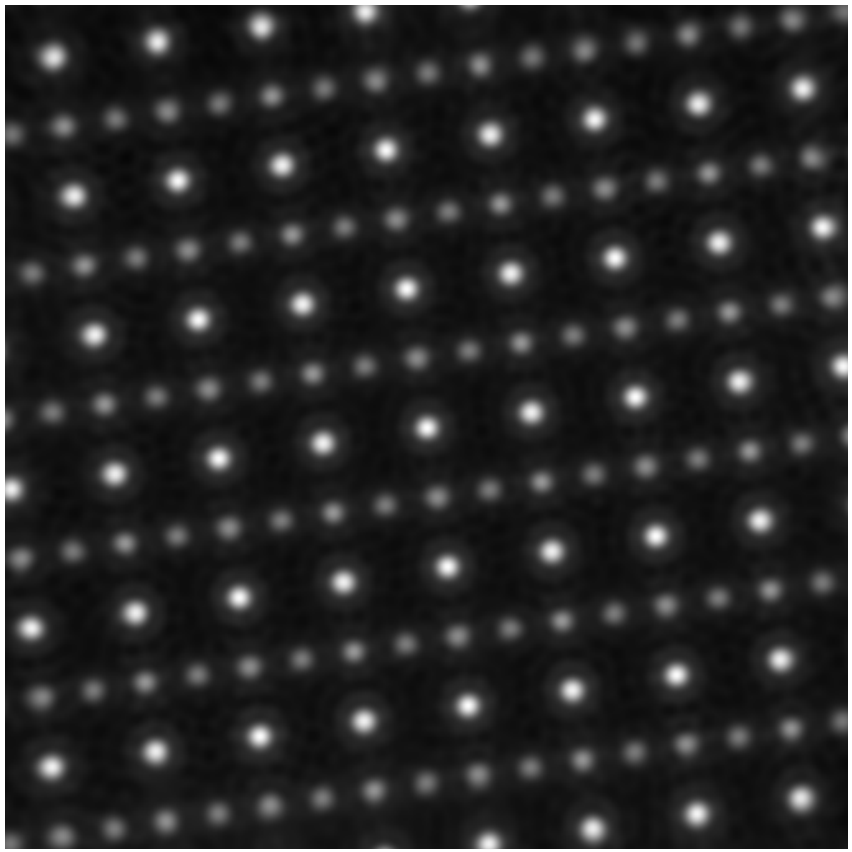
Input data (10 frames)

g_4 g_5



g_6 g_7

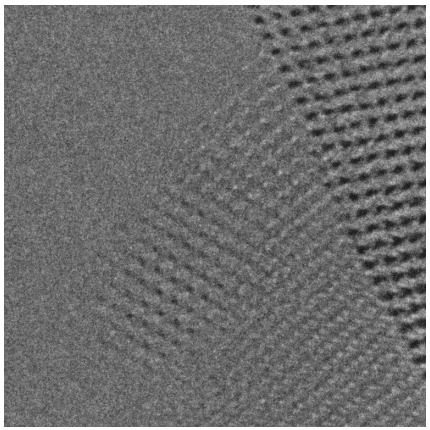
Reconstruction



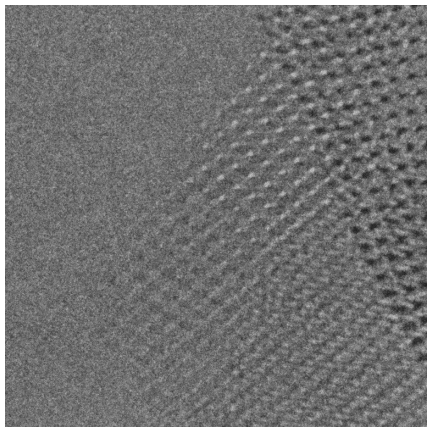
Phase(Ψ)

Preliminary results on real data

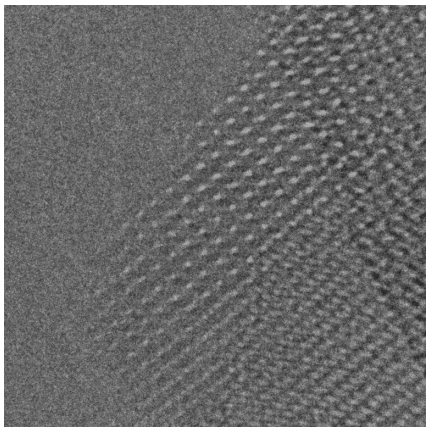
Input data (21 frames, courtesy of Angus Kirkland, Oxford)



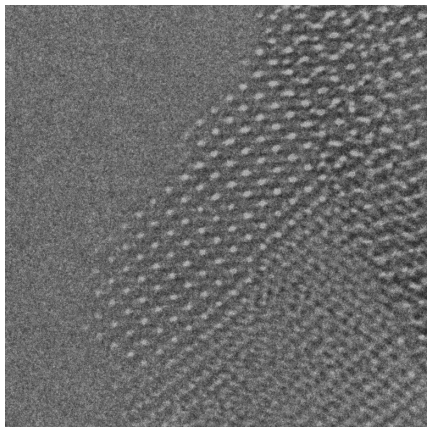
g_{10}



g_{11}

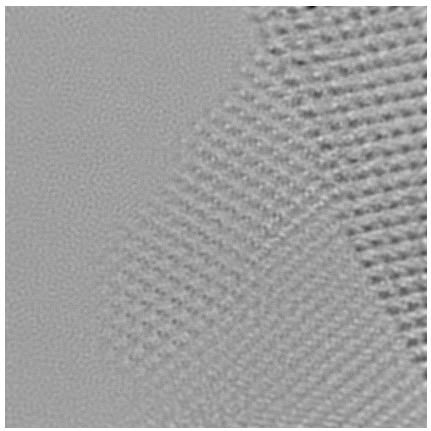


g_{12}

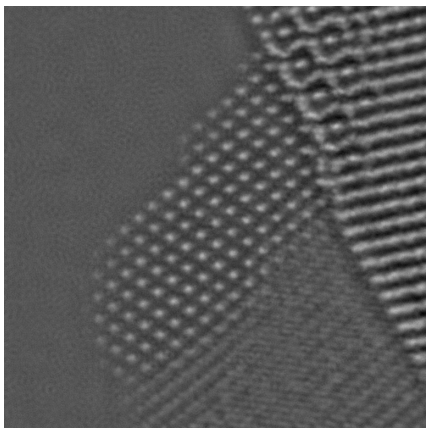


g_{13}

Reconstruction



Amplitude(Ψ)



Phase(Ψ)

Total Deep Variation for noisy exit wave reconstruction in transmission electron microscopy

joint work with:

C. Doberstein (University of SC)
T. Pinetz, A. Effland (TU Graz, now at University of Bonn)
E. Kobler (TU Graz)

Exit wave reconstruction using Total Deep Variation

The regularizer $\|\Psi - \Psi_M\|_{L^2}^2$ was just motivated by the need for coercivity.

Exit wave reconstruction using Total Deep Variation

The regularizer $\|\Psi - \Psi_M\|_{L^2}^2$ was just motivated by the need for coercivity.

Idea Learn the regularizer.

Exit wave reconstruction using Total Deep Variation

The regularizer $\|\Psi - \Psi_M\|_{L^2}^2$ was just motivated by the need for coercivity.

Idea Learn the regularizer.

Total deep variation was successfully applied to linear inverse problems.

[Kobler, Effland, Kunisch, Pock TPAMI'21]

$$\mathcal{R}(x, \theta) = \sum_{i=1}^{d} r(x, \theta)_i \text{ with } r(x, \theta) = w\mathcal{N}(Kx)$$

Exit wave reconstruction using Total Deep Variation

The regularizer $\|\Psi - \Psi_M\|_{L^2}^2$ was just motivated by the need for coercivity.

Idea Learn the regularizer.

Total deep variation was successfully applied to linear inverse problems.

[Kobler, Effland, Kunisch, Pock TPAMI'21]

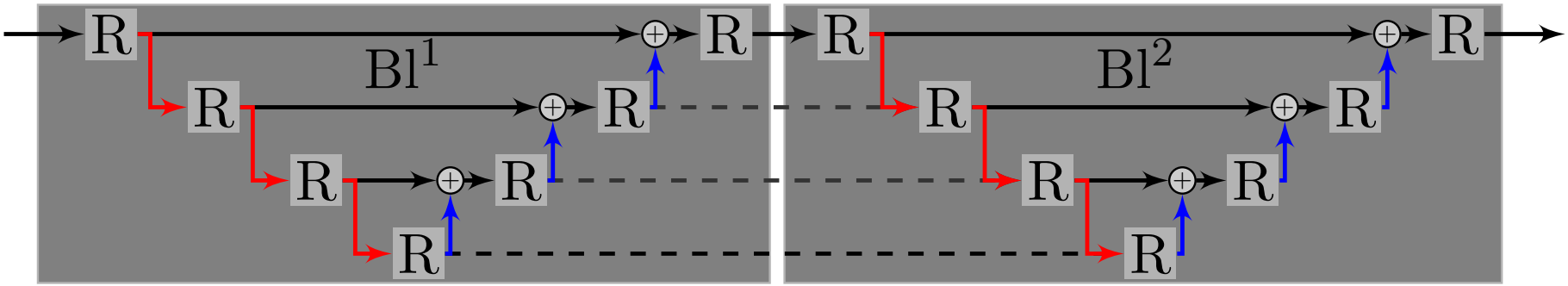
$$\mathcal{R}(x,\theta) = \sum_{i=1}^{d} r(x,\theta)_i \text{ with } r(x,\theta) = w\mathcal{N}(Kx)$$

Here, $\mathcal{F}^{-1}(\Psi) \cong x \in \mathbb{R}^{2d}$ using the identification $\mathbb{C}^d \cong \mathbb{R}^{2d}$,

- $K \in \mathbb{R}^{md \times 2d}$ matrix representation of learned 3×3 convolution kernels with m feature channels and zero-mean constraint
- $\mathcal{N} : \mathbb{R}^{md} \to \mathbb{R}^{md}$ sufficiently smooth multiscale convolutional neural network with compact support
- $w \in \mathbb{R}^{d \times md}$ matrix representation of learned 1×1 convolution kernels

θ denotes the learnable parameters of K, $\mathcal{N}$ and w.

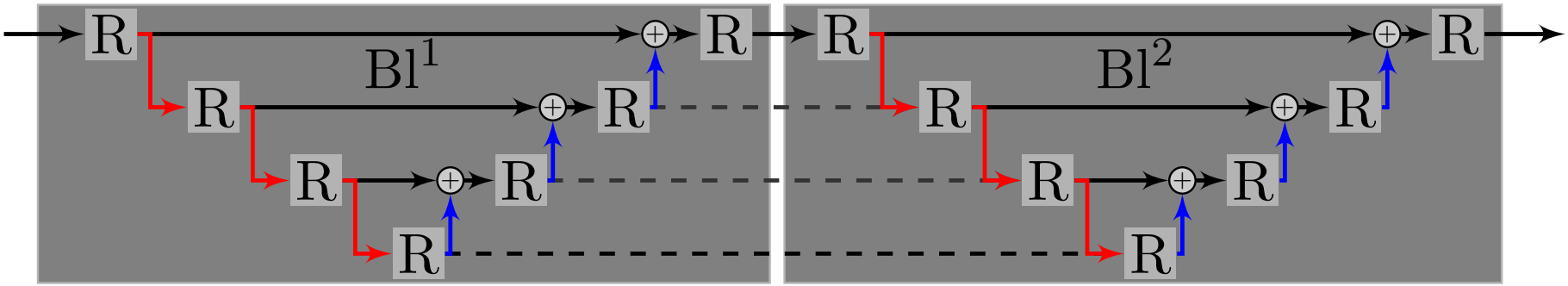
Exit wave reconstruction using Total Deep Variation (cont.)



Network architecture of $\mathcal{N}$.

- TDV is composed of 2 blocks Bl^i, $i = 1, 2$, where each block is designed as a U-Net with 7 residual blocks $\mathrm{R}_1^i, \ldots, \mathrm{R}_7^i$.

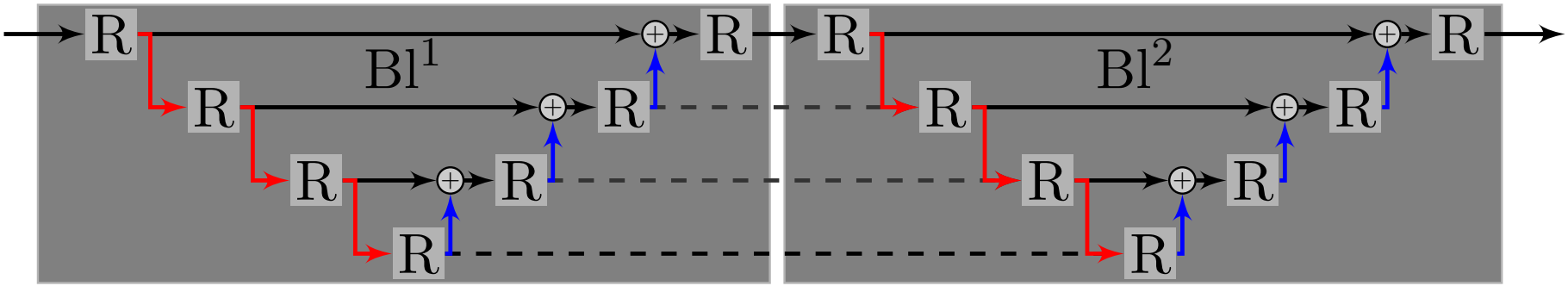
Exit wave reconstruction using Total Deep Variation (cont.)



Network architecture of $\mathcal{N}$.

- TDV is composed of 2 blocks Bl^i, $i = 1, 2$, where each block is designed as a U-Net with 7 residual blocks $\mathrm{R}_1^i, \ldots, \mathrm{R}_7^i$.
- Residual blocks on the same scale are linked with skip connections.

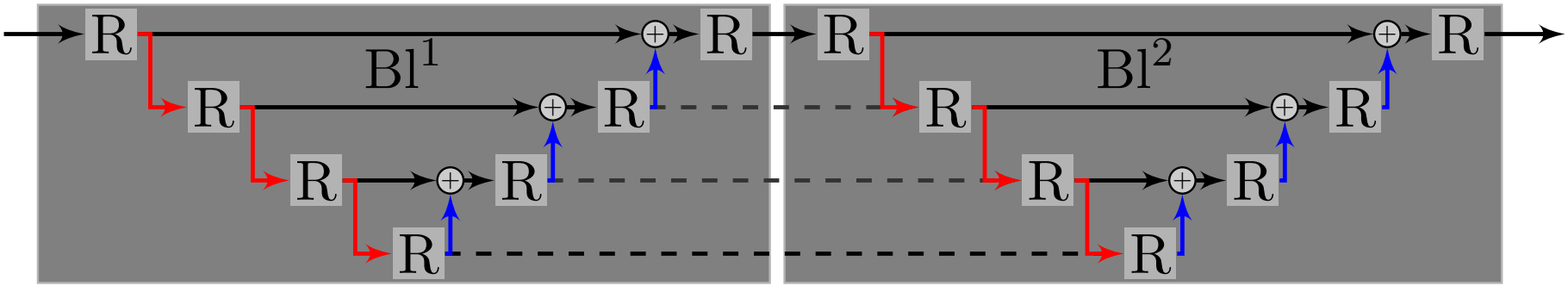
Exit wave reconstruction using Total Deep Variation (cont.)



Network architecture of $\mathcal{N}$.

- TDV is composed of 2 blocks Bl^i, $i = 1, 2$, where each block is designed as a U-Net with 7 residual blocks $\mathrm{R}_1^i, \ldots, \mathrm{R}_7^i$.
- Residual blocks on the same scale are linked with skip connections.
- Residual connections link residual blocks to consecutive blocks.

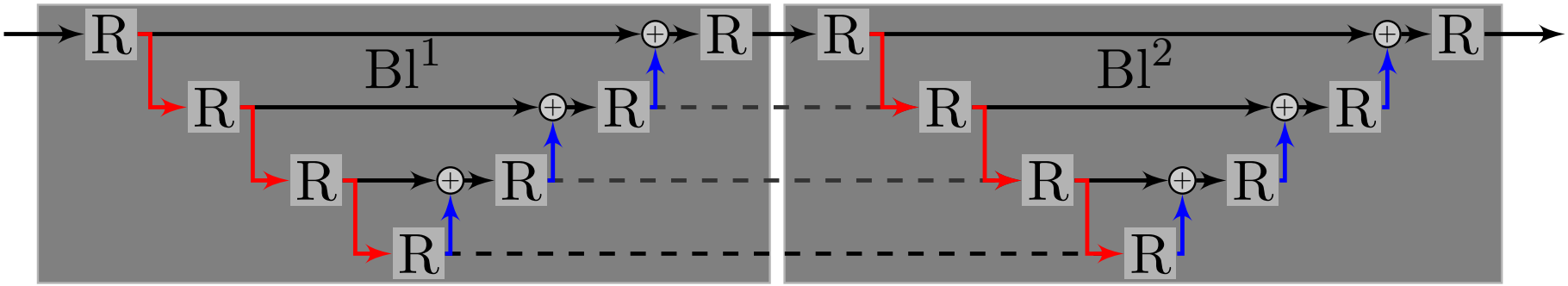
Exit wave reconstruction using Total Deep Variation (cont.)



Network architecture of $\mathcal{N}$.

- TDV is composed of 2 blocks Bl^i, $i = 1, 2$, where each block is designed as a U-Net with 7 residual blocks $\mathrm{R}_1^i, \ldots, \mathrm{R}_7^i$.
- Residual blocks on the same scale are linked with skip connections.
- Residual connections link residual blocks to consecutive blocks.
- Residual blocks are modeled as $\mathrm{R}_j^i(x) = x + K_{j,2}^i \Phi(K_{j,1}^i x)$ for 3×3 convolution operators with $m = 32$ feature channels and no bias.

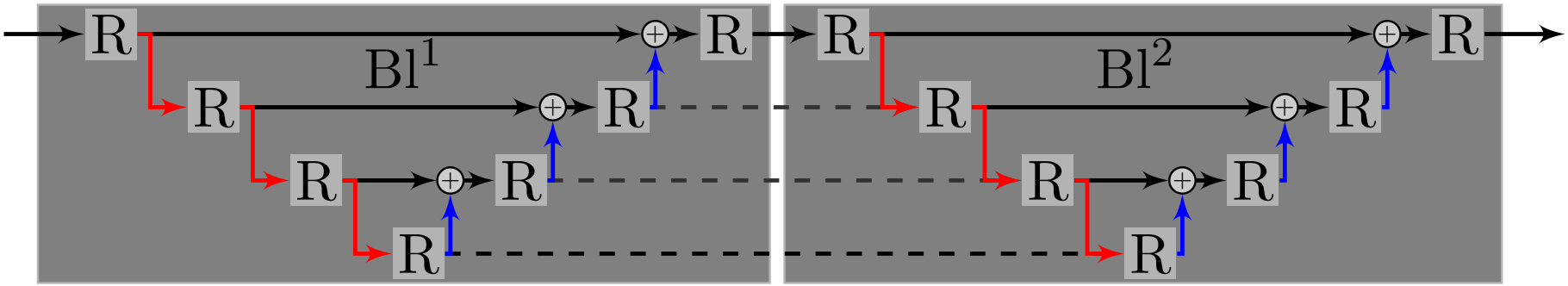
Exit wave reconstruction using Total Deep Variation (cont.)



Network architecture of $\mathcal{N}$.

- TDV is composed of 2 blocks Bl^i, $i = 1, 2$, where each block is designed as a U-Net with 7 residual blocks $\mathrm{R}_1^i, \ldots, \mathrm{R}_7^i$.
- Residual blocks on the same scale are linked with skip connections.
- Residual connections link residual blocks to consecutive blocks.
- Residual blocks are modeled as $\mathrm{R}_j^i(x) = x + K_{j,2}^i \Phi(K_{j,1}^i x)$ for 3×3 convolution operators with $m = 32$ feature channels and no bias.
- Activation function: log-Student-t-distribution $\phi(s) = \frac{1}{2} \log(1 + s^2)$.

Exit wave reconstruction using Total Deep Variation (cont.)



Network architecture of $\mathcal{N}$.

- TDV is composed of 2 blocks Bl^i, $i = 1, 2$, where each block is designed as a U-Net with 7 residual blocks $\mathrm{R}_1^i, \ldots, \mathrm{R}_7^i$.
- Residual blocks on the same scale are linked with skip connections.
- Residual connections link residual blocks to consecutive blocks.
- Residual blocks are modeled as $\mathrm{R}_j^i(x) = x + K_{j,2}^i \Phi(K_{j,1}^i x)$ for 3×3 convolution operators with $m = 32$ feature channels and no bias.
- Activation function: log-Student-t-distribution $\phi(s) = \frac{1}{2} \log(1 + s^2)$.
- To avoid aliasing, we use 3×3 convolutions and transposed convolutions with stride 2 and a blur kernel to realize down- and upsampling.

Exit wave reconstruction using Total Deep Variation (cont.)

- In total, we use $|\theta| \approx 4 \cdot 10^5$ learnable parameters.

Exit wave reconstruction using Total Deep Variation (cont.)

- In total, we use $|\theta| \approx 4 \cdot 10^5$ learnable parameters.
- To approximate the minimizer Ψ for given θ, we use the Landweber iteration (can be interpreted as a time-discretized gradient flow).

Exit wave reconstruction using Total Deep Variation (cont.)

- In total, we use $|\theta| \approx 4 \cdot 10^5$ learnable parameters.
- To approximate the minimizer Ψ for given θ, we use the Landweber iteration (can be interpreted as a time-discretized gradient flow).
- The parameters θ are trained with a sampled optimal control problem [Weinan E, Han, Li Res. Math. Sci. '19].

Exit wave reconstruction using Total Deep Variation (cont.)

- In total, we use $|\theta| \approx 4 \cdot 10^5$ learnable parameters.
- To approximate the minimizer Ψ for given θ, we use the Landweber iteration (can be interpreted as a time-discretized gradient flow).
- The parameters θ are trained with a sampled optimal control problem [Weinan E, Han, Li Res. Math. Sci. '19].
- We trained TDV for exit wave reconstruction using 100 exit waves from [Ede, Peters, Sloan, Beanland arXiv'20] and different noise type / levels.

Exit wave reconstruction using Total Deep Variation (cont.)

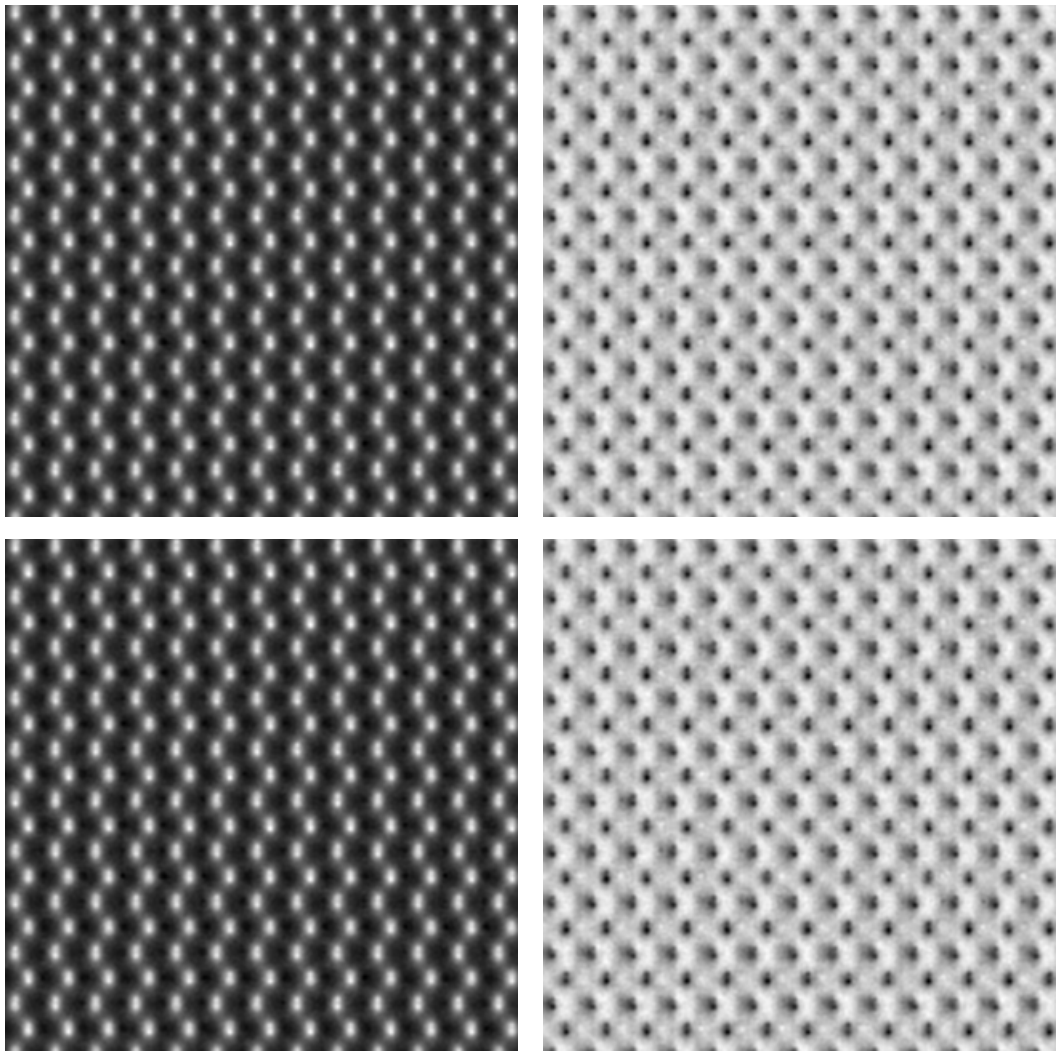
- In total, we use $|\theta| \approx 4 \cdot 10^5$ learnable parameters.
- To approximate the minimizer Ψ for given θ, we use the Landweber iteration (can be interpreted as a time-discretized gradient flow).
- The parameters θ are trained with a sampled optimal control problem [Weinan E, Han, Li Res. Math. Sci. '19].
- We trained TDV for exit wave reconstruction using 100 exit waves from [Ede, Peters, Sloan, Beanland arXiv'20] and different noise type / levels.

	noise-free	additive Gaussian noise	shot noise
initialization	1.27	1.27	1.27
MAL	0.01	0.83	0.66
DoBe19	0.01	0.82	0.65
TDV	0.01	0.42	0.46

Mean squared error for all considered noise types on validation data with 963 separate exit waves [Pinetz, Kobler, Doberstein, B., Effland SSVM'21].

Numerical results - noise free

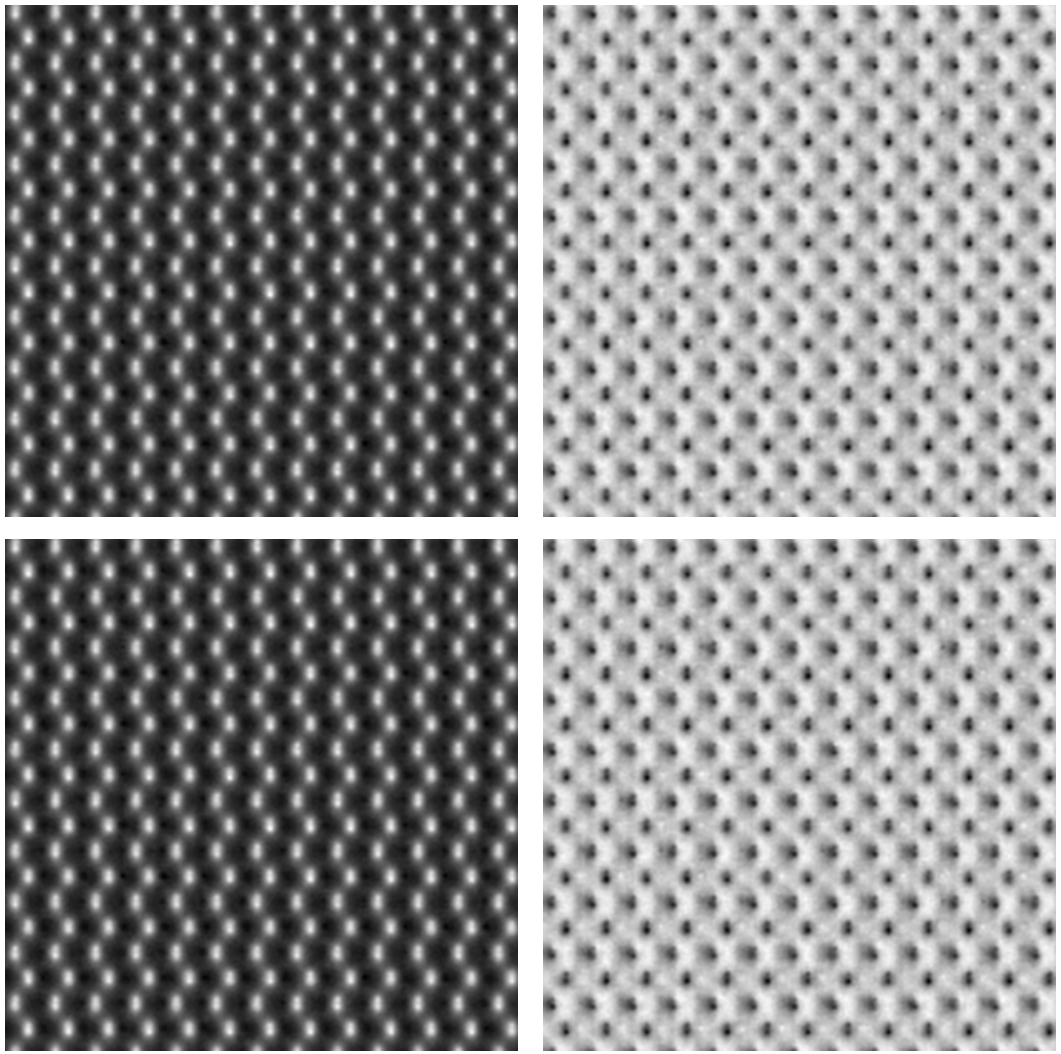
ground truth $\overline{\Psi}$



Ψ^{DoBe19} (MSE: $2 \cdot 10^{-5}$)

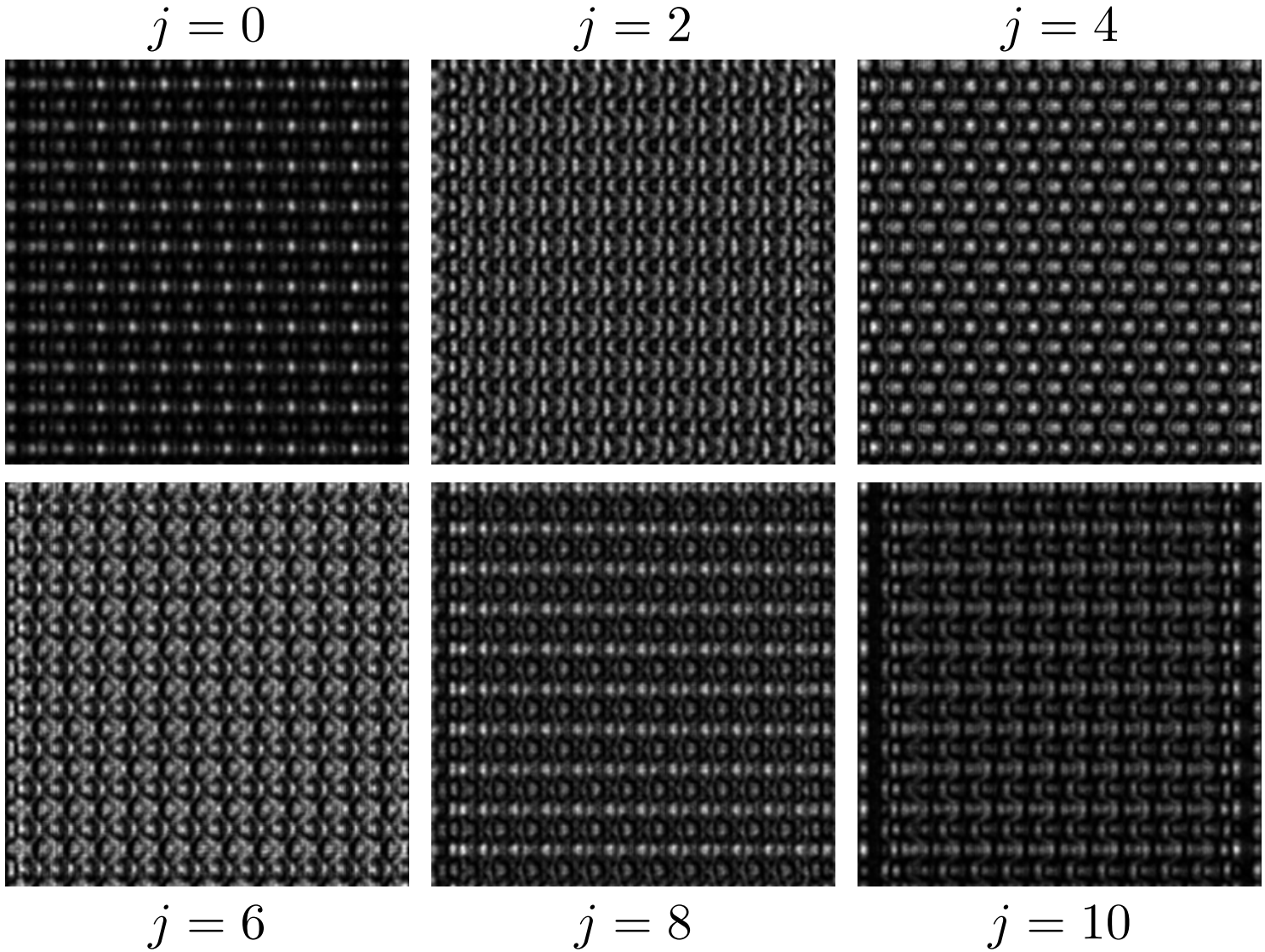
Numerical results - noise free

ground truth $\overline{\Psi}$



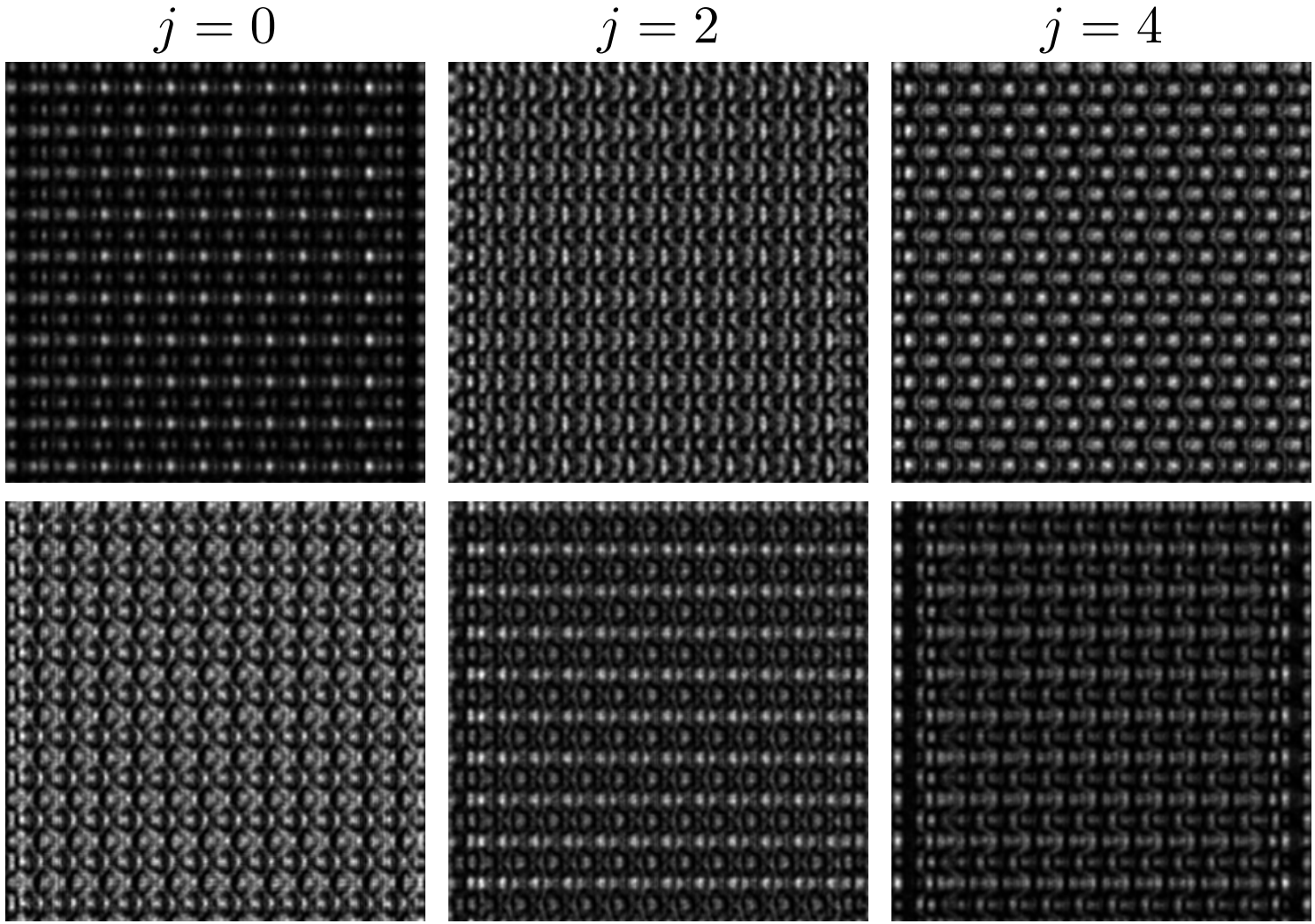
Ψ^{TDV} (MSE: $2 \cdot 10^{-5}$)

Numerical results - noise free (cont.)



input focal series $\mathbf{X}^{\exp}_{Z_j}$ (11 frames)

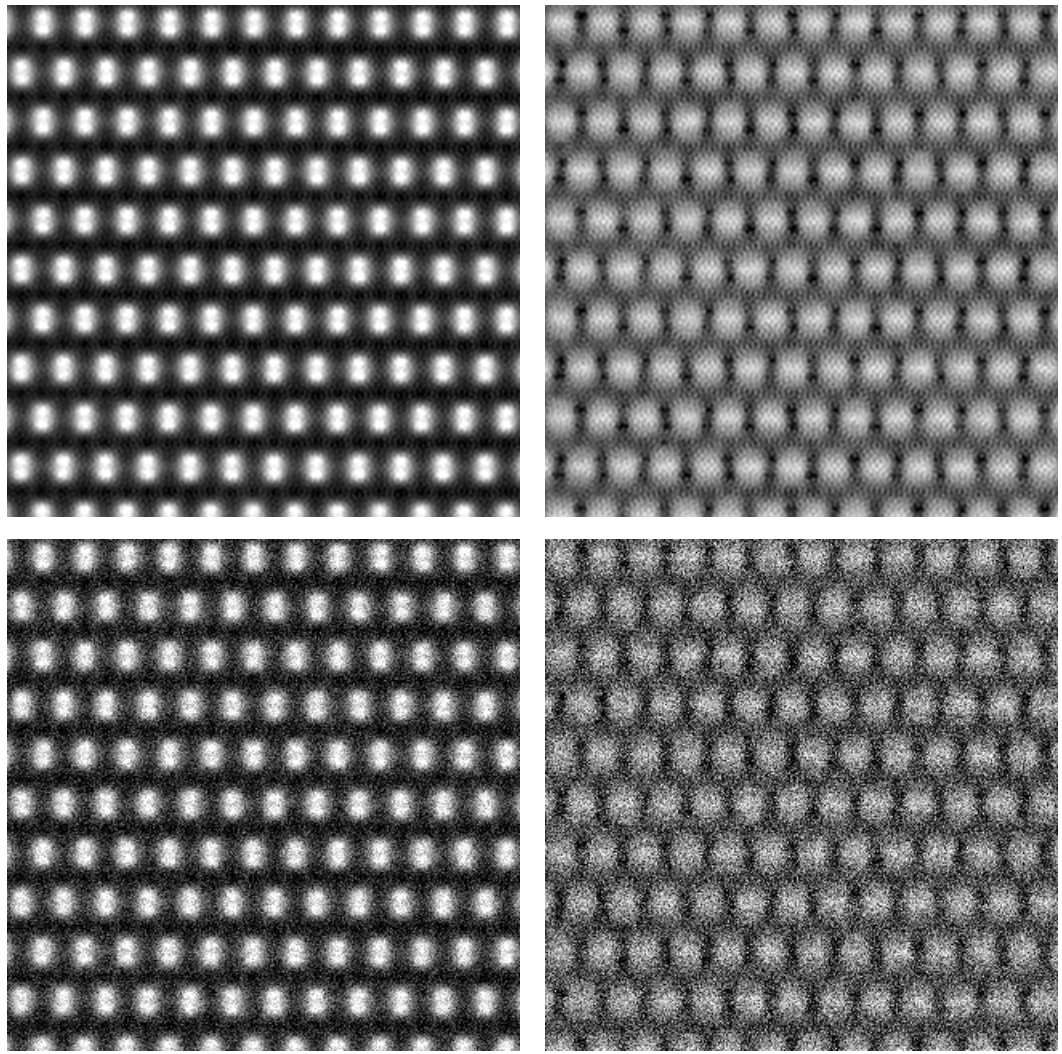
Numerical results - noise free (cont.)



reconstructed focal series $\mathbf{F}^{-1}(\mathbf{\Psi}^{\mathrm{TDV}} \star_{T_{Z_j}} \mathbf{\Psi}^{\mathrm{TDV}})$

Numerical results - additive white Gaussian noise

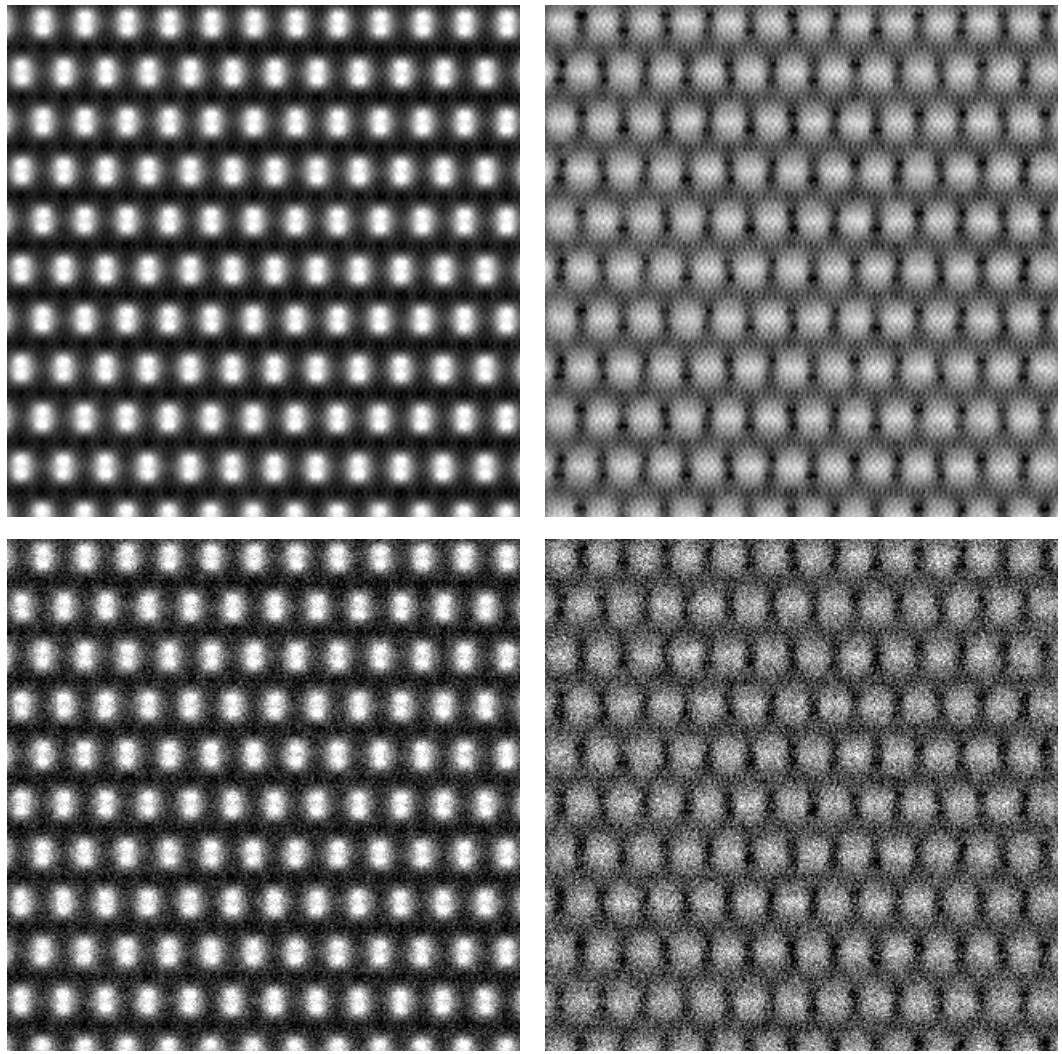
ground truth $\overline{\Psi}$



Ψ^{DoBe19} (MSE: 0.52)

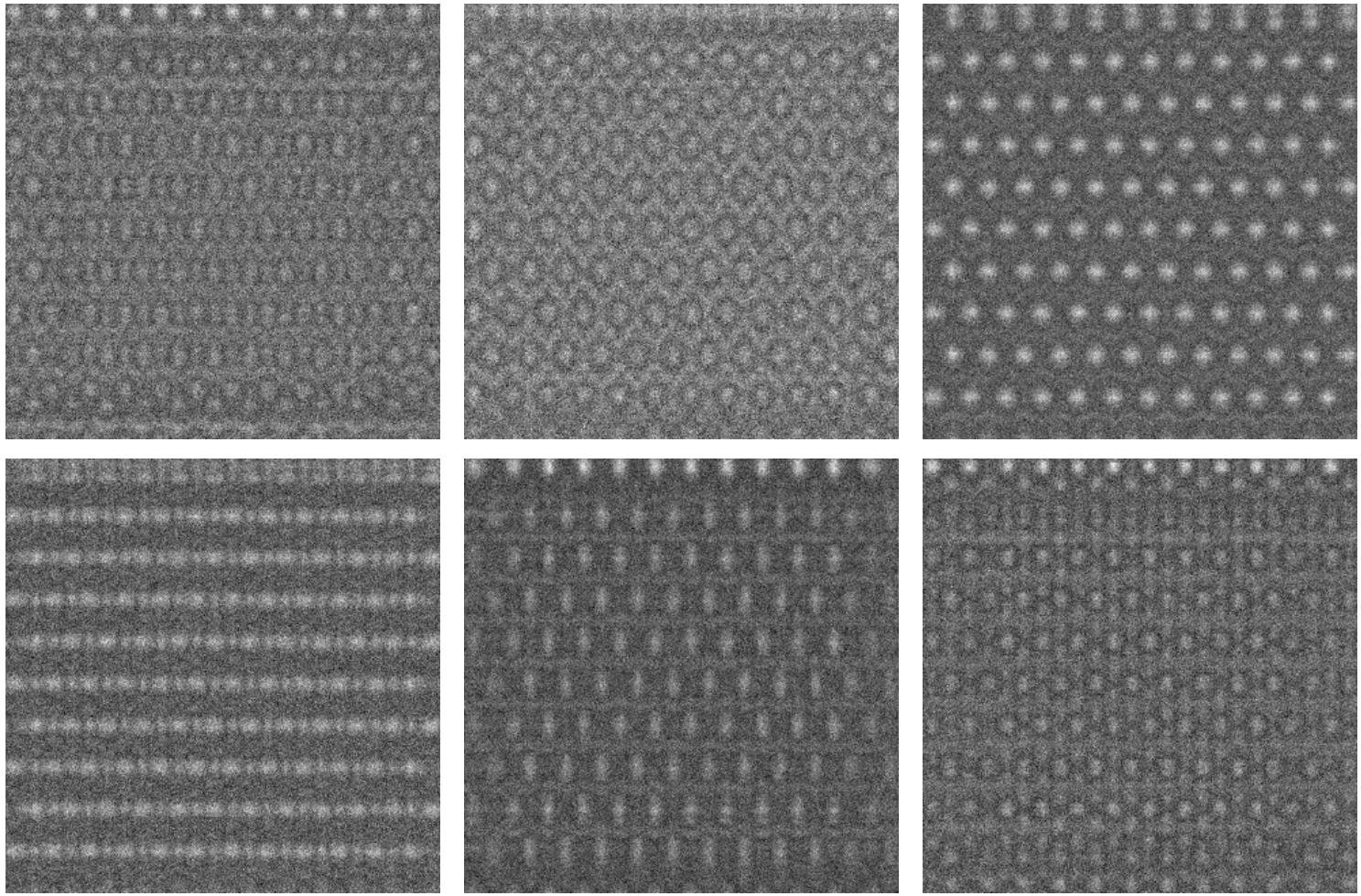
Numerical results - additive white Gaussian noise

ground truth $\overline{\Psi}$



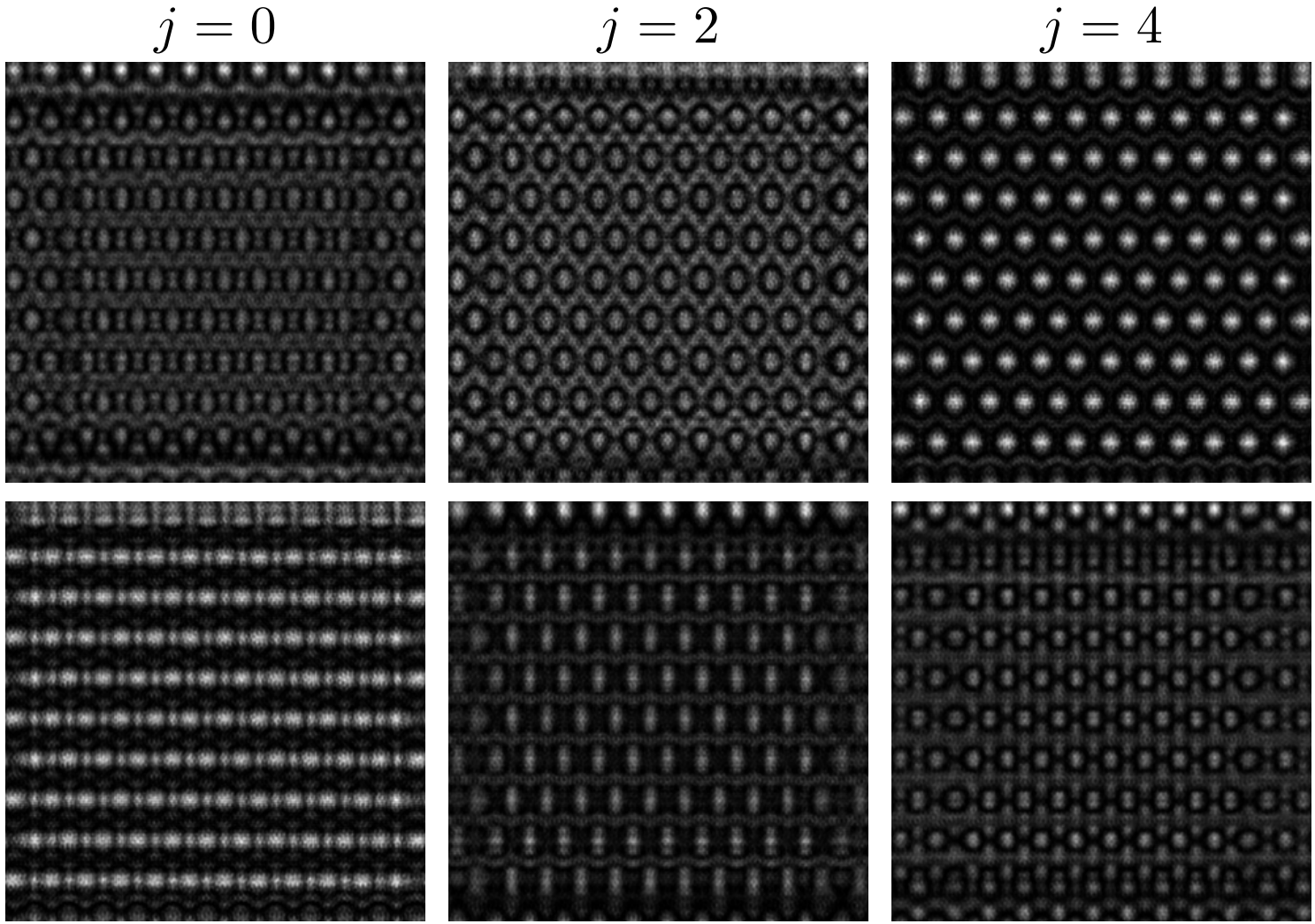
Ψ^{TDV} (MSE: 0.24)

Numerical results - additive white Gaussian noise (cont.)



input focal series $\mathbf{X}^{\exp}_{Z_j}$ (11 frames)

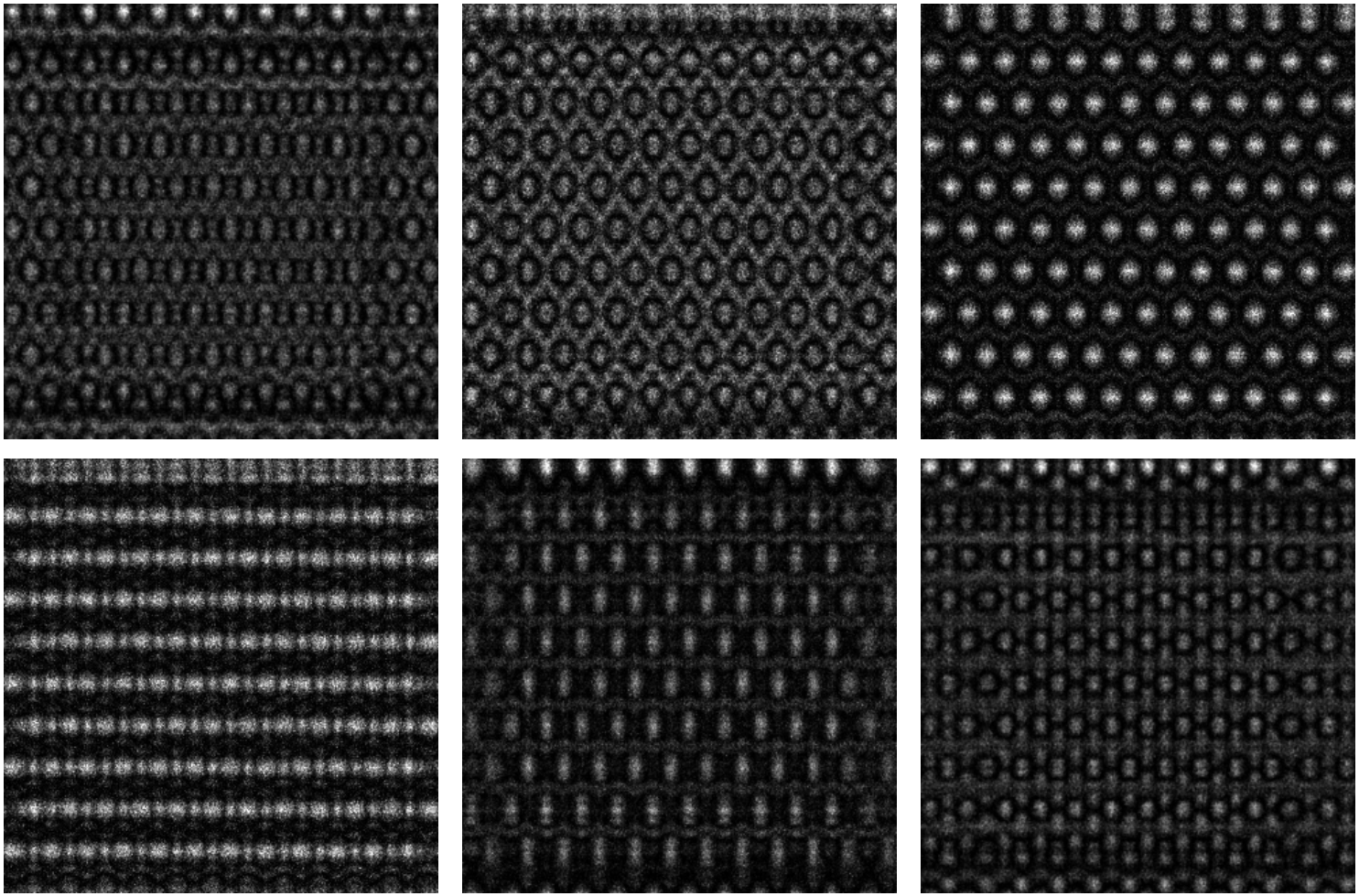
Numerical results - additive white Gaussian noise (cont.)



ground truth focal series $\mathbf{F}^{-1}(\overline{\mathbf{\Psi}} \star_{T_{Z_j}} \overline{\mathbf{\Psi}})$

Numerical results - additive white Gaussian noise (cont.)

$j = 0$, MSE: 0.28 $j = 2$, MSE: 0.49 $j = 4$, MSE: 0.69

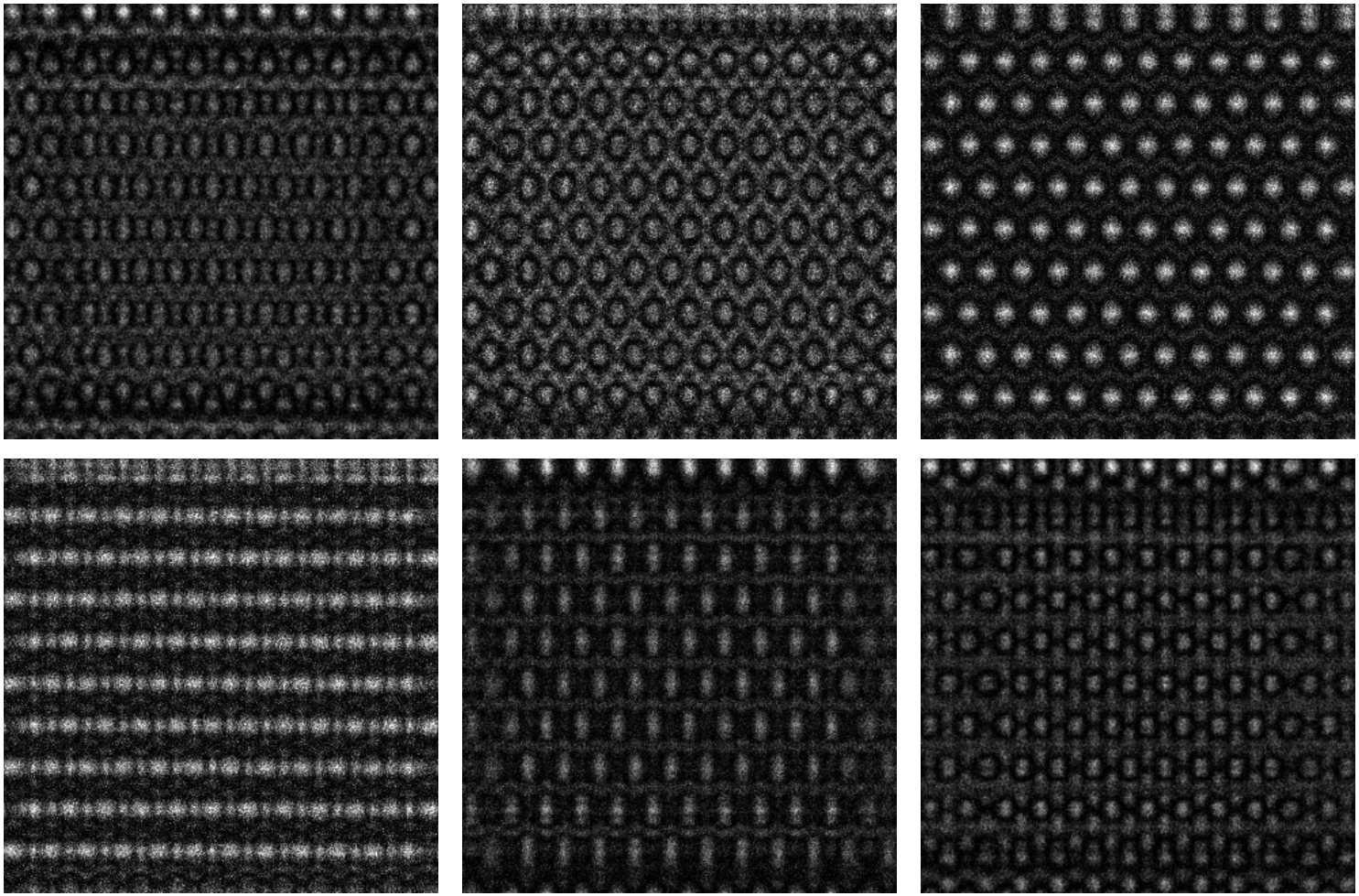


$j = 6$, MSE: 0.71 $j = 8$, MSE: 0.47 $j = 10$, MSE: 0.28

reconstructed focal series $\mathbf{F}^{-1}(\mathbf{\Psi}^{\mathrm{TDV}} \star_{T_{Z_j}} \mathbf{\Psi}^{\mathrm{TDV}})$

Numerical results - additive white Gaussian noise (cont.)

$j = 0$, MSE: 0.71 $j = 2$, MSE: 0.98 $j = 4$, MSE: 1.30

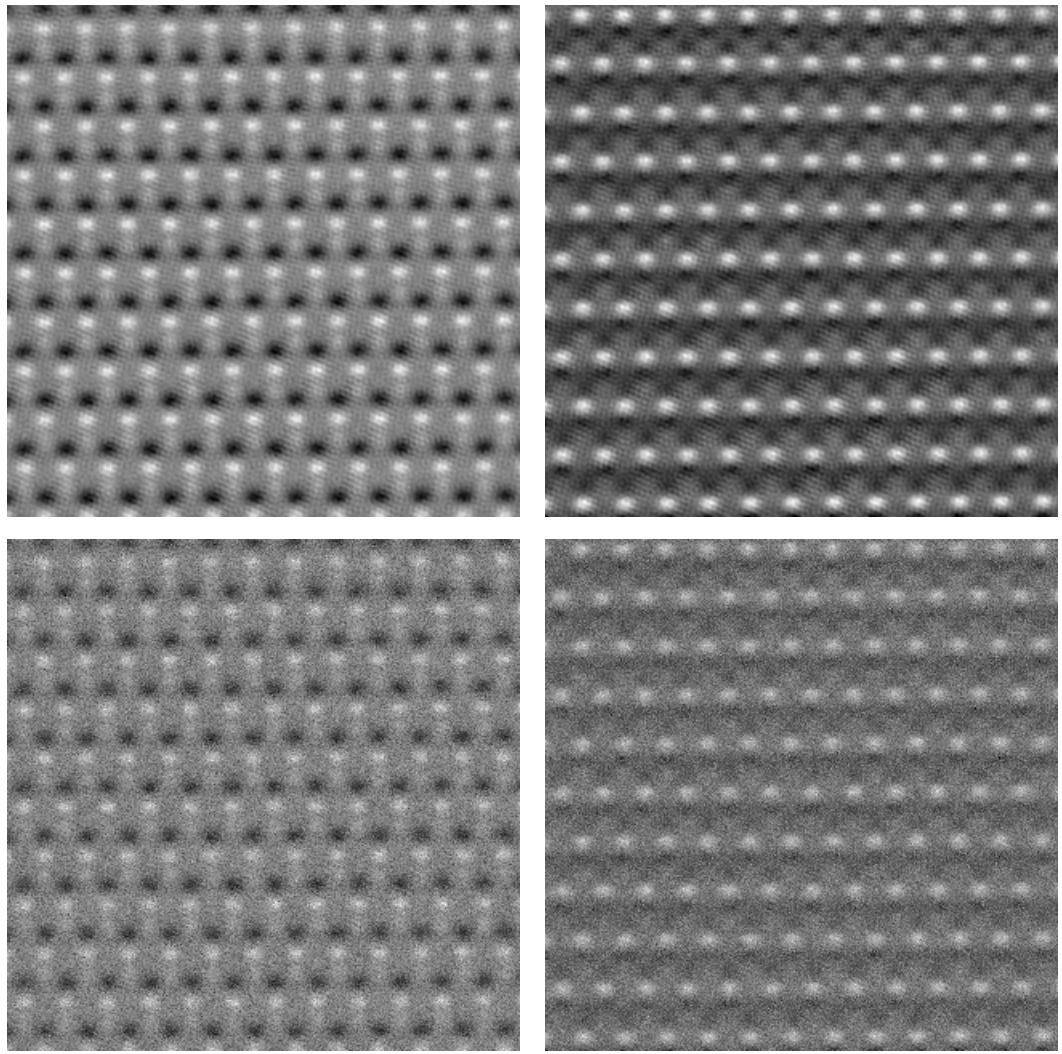


$j = 6$, MSE: 1.23 $j = 8$, MSE: 0.98 $j = 10$, MSE: 0.67

reconstructed focal series $\mathbf{F}^{-1}(\mathbf{\Psi}^{\mathrm{DoBe19}} \star_{T_{Z_j}} \mathbf{\Psi}^{\mathrm{DoBe19}})$

Numerical results - shot noise

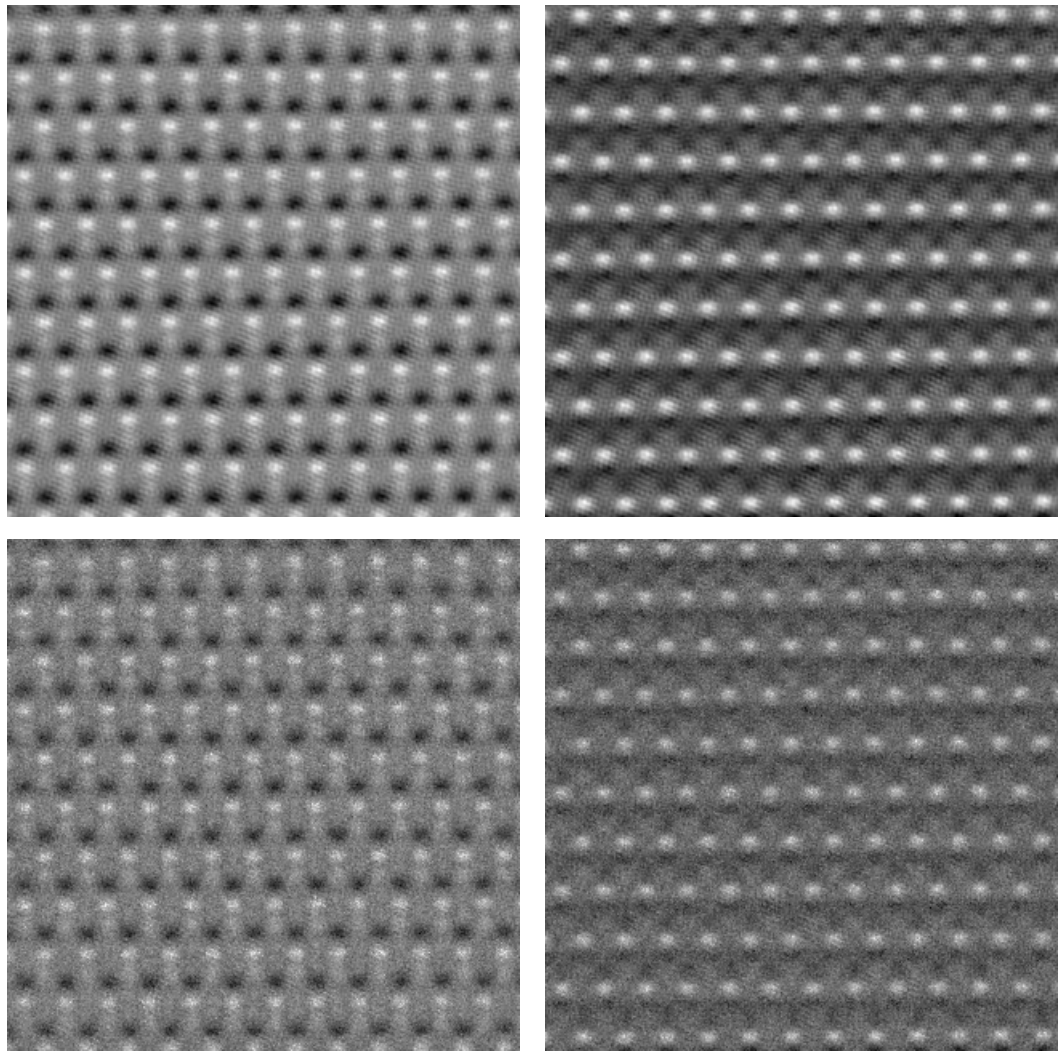
ground truth $\overline{\Psi}$



Ψ^{DoBe19} (MSE: 0.55)

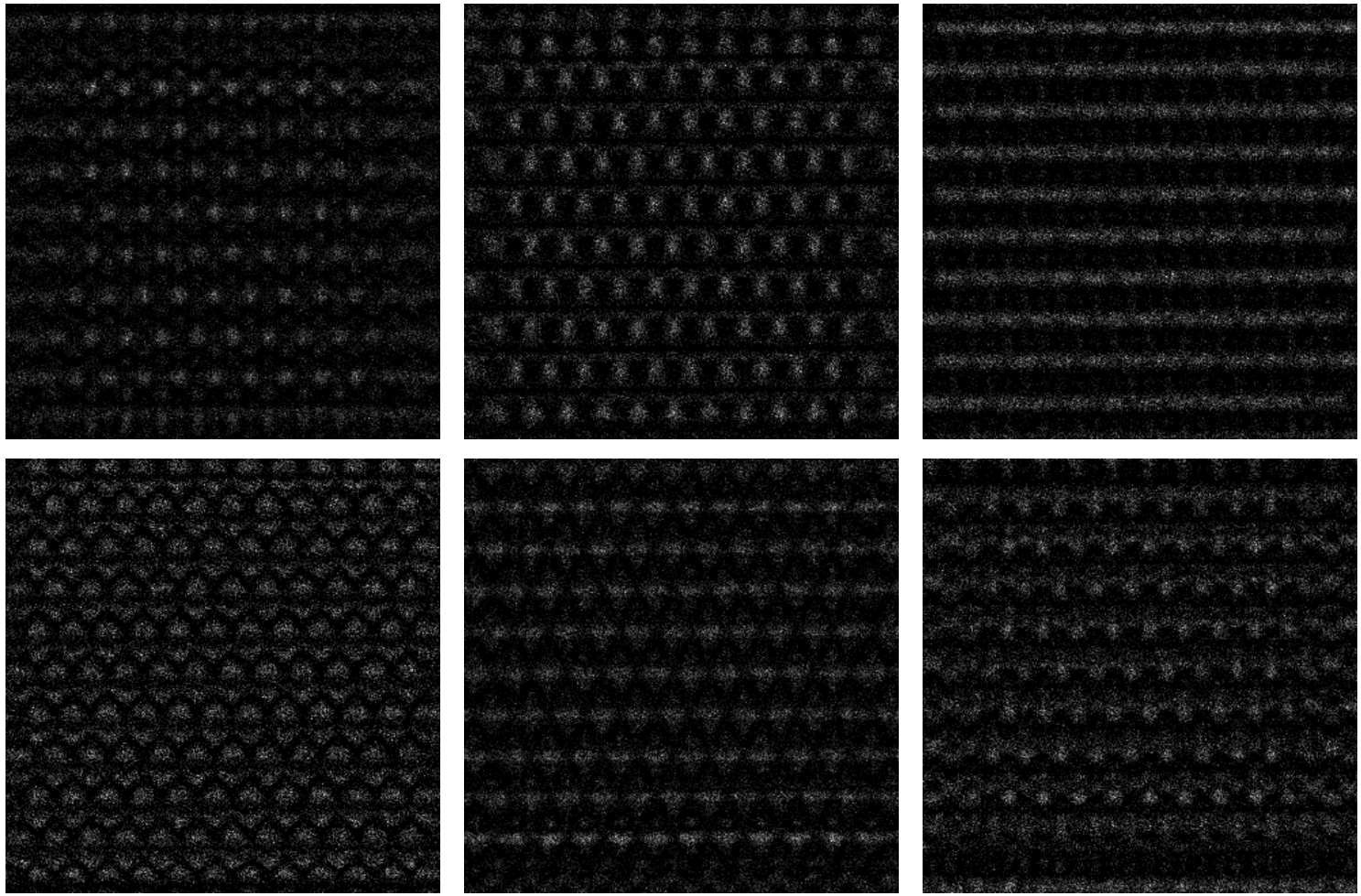
Numerical results - shot noise

ground truth $\overline{\Psi}$



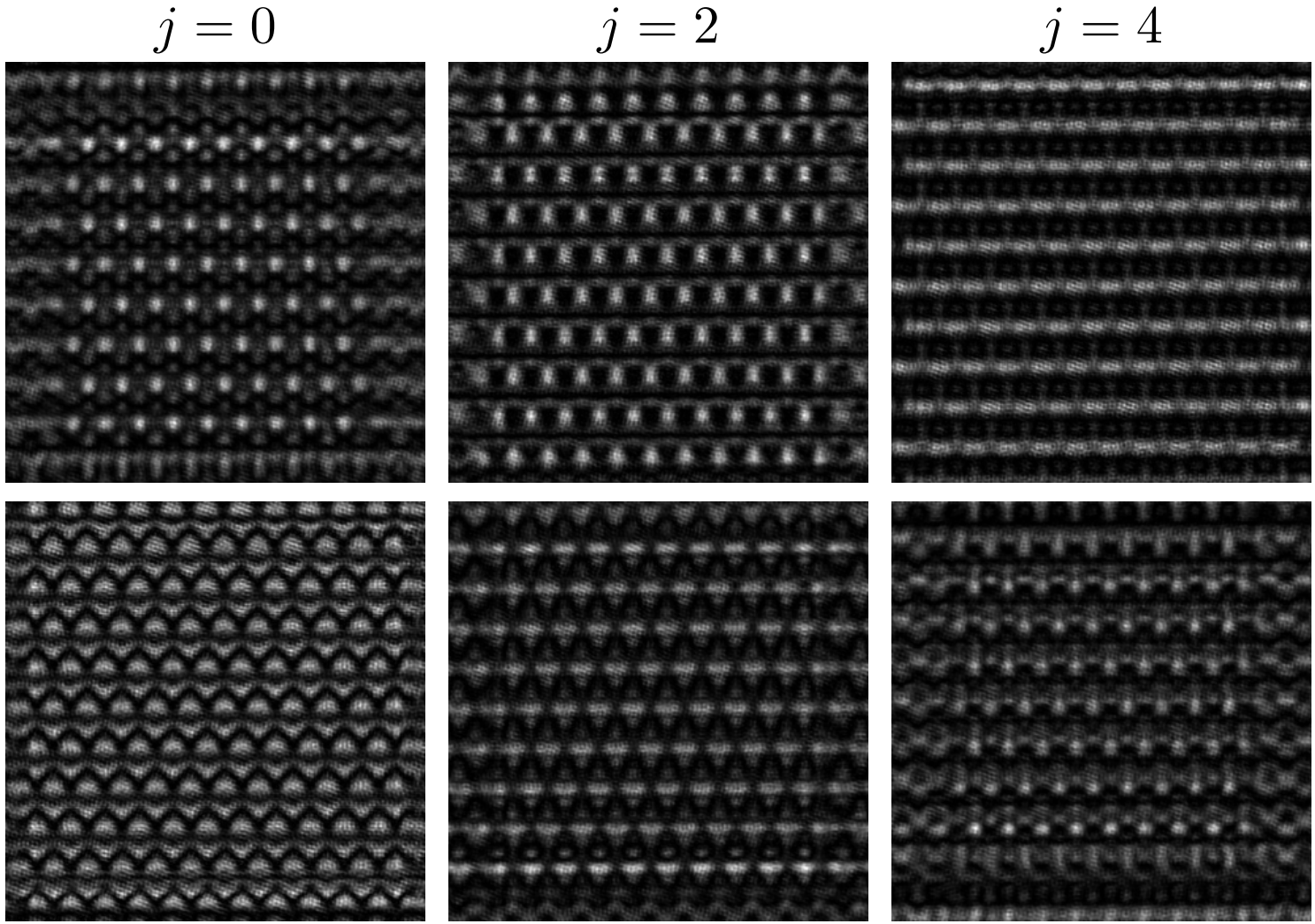
Ψ^{TDV} (MSE: 0.29)

Numerical results - shot noise (cont.)



input focal series $\mathbf{X}^{\mathrm{exp}}_{Z_j}$ (11 frames)

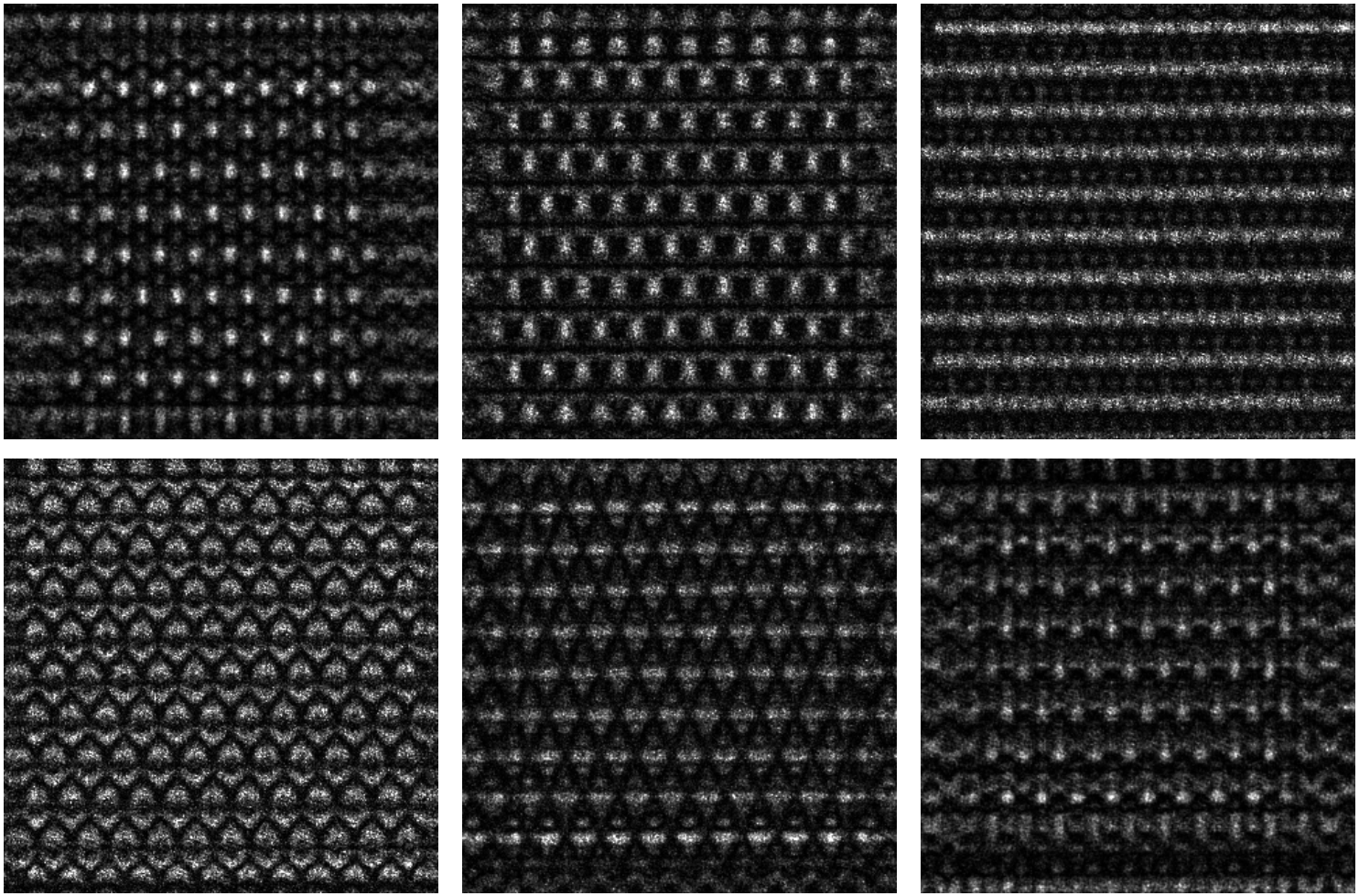
Numerical results - shot noise (cont.)



ground truth focal series $\mathbf{F}^{-1}(\overline{\mathbf{\Psi}} \star_{T_{Z_j}} \overline{\mathbf{\Psi}})$

Numerical results - shot noise (cont.)

$j = 0$, MSE: 0.31 $j = 2$, MSE: 0.53 $j = 4$, MSE: 0.85

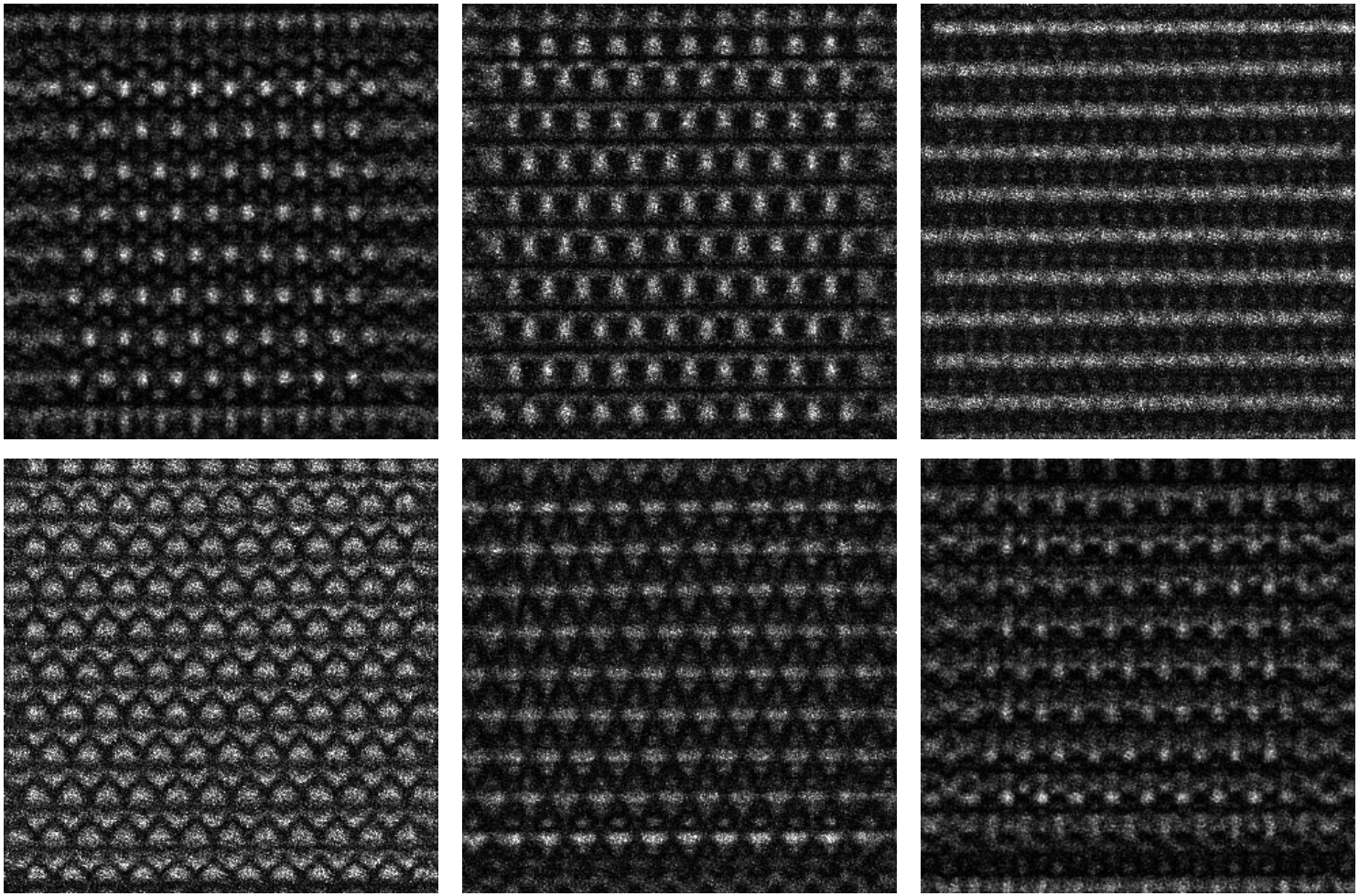


$j = 6$, MSE: 0.72 $j = 8$, MSE: 0.54 $j = 10$, MSE: 0.30

reconstructed focal series $\mathbf{F}^{-1}(\mathbf{\Psi}^{\mathrm{TDV}} \star_{T_{Z_j}} \mathbf{\Psi}^{\mathrm{TDV}})$

Numerical results - shot noise (cont.)

$j = 0$, MSE: 0.55 $j = 2$, MSE: 0.87 $j = 4$, MSE: 1.09



$j = 6$, MSE: 0.94 $j = 8$, MSE: 0.86 $j = 10$, MSE: 0.58

reconstructed focal series $\mathbf{F}^{-1}(\mathbf{\Psi}^{\mathrm{DoBe19}} \star_{T_{Z_j}} \mathbf{\Psi}^{\mathrm{DoBe19}})$

Deep unfolding

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.
- However, standard deep network architectures, like convolutional neural networks (CNNs), neglect most of the available domain knowledge and compensate this with large amounts of training data.

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.
- However, standard deep network architectures, like convolutional neural networks (CNNs), neglect most of the available domain knowledge and compensate this with large amounts of training data.
- Moreover, these deep networks are mostly black boxes and their behavior is not really understood.

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.
- However, standard deep network architectures, like convolutional neural networks (CNNs), neglect most of the available domain knowledge and compensate this with large amounts of training data.
- Moreover, these deep networks are mostly black boxes and their behavior is not really understood.
- On the other hand, there are many hand crafted model-based approaches whose behavior is well understood, but that are nowadays empirically outperformed by deep learning based approaches.

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.
- However, standard deep network architectures, like convolutional neural networks (CNNs), neglect most of the available domain knowledge and compensate this with large amounts of training data.
- Moreover, these deep networks are mostly black boxes and their behavior is not really understood.
- On the other hand, there are many hand crafted model-based approaches whose behavior is well understood, but that are nowadays empirically outperformed by deep learning based approaches.
- How can both worlds, i.e., deep learning and model-based approaches, be connected?

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.
- However, standard deep network architectures, like convolutional neural networks (CNNs), neglect most of the available domain knowledge and compensate this with large amounts of training data.
- Moreover, these deep networks are mostly black boxes and their behavior is not really understood.
- On the other hand, there are many hand crafted model-based approaches whose behavior is well understood, but that are nowadays empirically outperformed by deep learning based approaches.
- How can both worlds, i.e., deep learning and model-based approaches, be connected?
- For model-based approaches that are solved using *iterative* algorithms, there is a general strategy to connect both worlds:

Unfolding

Deep unfolding / algorithm unrolling

- The immense success of deep learning is undeniable.
- However, standard deep network architectures, like convolutional neural networks (CNNs), neglect most of the available domain knowledge and compensate this with large amounts of training data.
- Moreover, these deep networks are mostly black boxes and their behavior is not really understood.
- On the other hand, there are many hand crafted model-based approaches whose behavior is well understood, but that are nowadays empirically outperformed by deep learning based approaches.
- How can both worlds, i.e., deep learning and model-based approaches, be connected?
- For model-based approaches that are solved using *iterative* algorithms, there is a general strategy to connect both worlds:

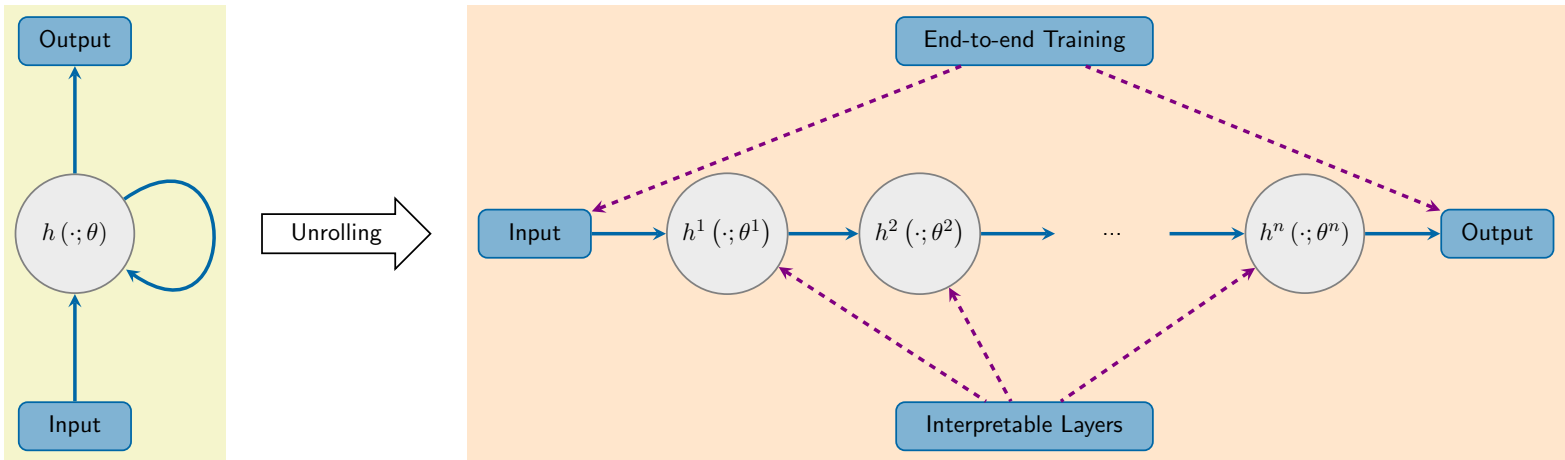
  Unfolding

- Combinations of data-driven learning with model-based approaches are also called *hybrid* methods.

Deep unfolding / algorithm unrolling (cont.)

Idea Given an *iterative* algorithm, [Monga, Li, Eldar '21]

- fix the number of iterations L and
- interpret each iteration of the algorithm as layer of a neural network.

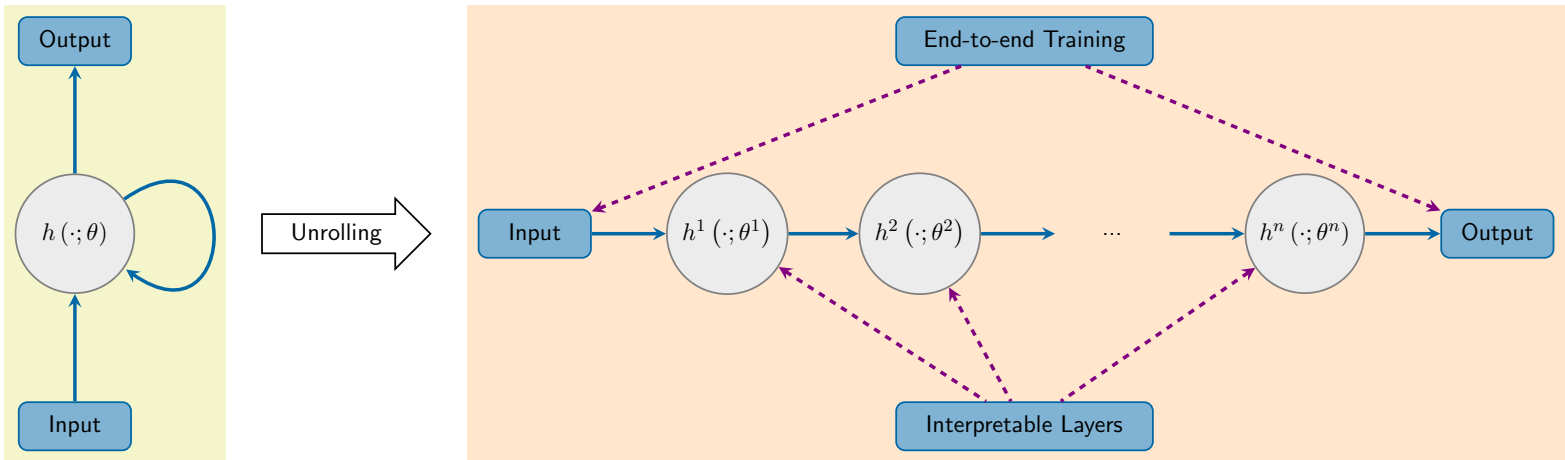


High level overview of deep unfolding

Deep unfolding / algorithm unrolling (cont.)

Idea Given an *iterative* algorithm, [Monga, Li, Eldar '21]

- fix the number of iterations L and
- interpret each iteration of the algorithm as layer of a neural network.



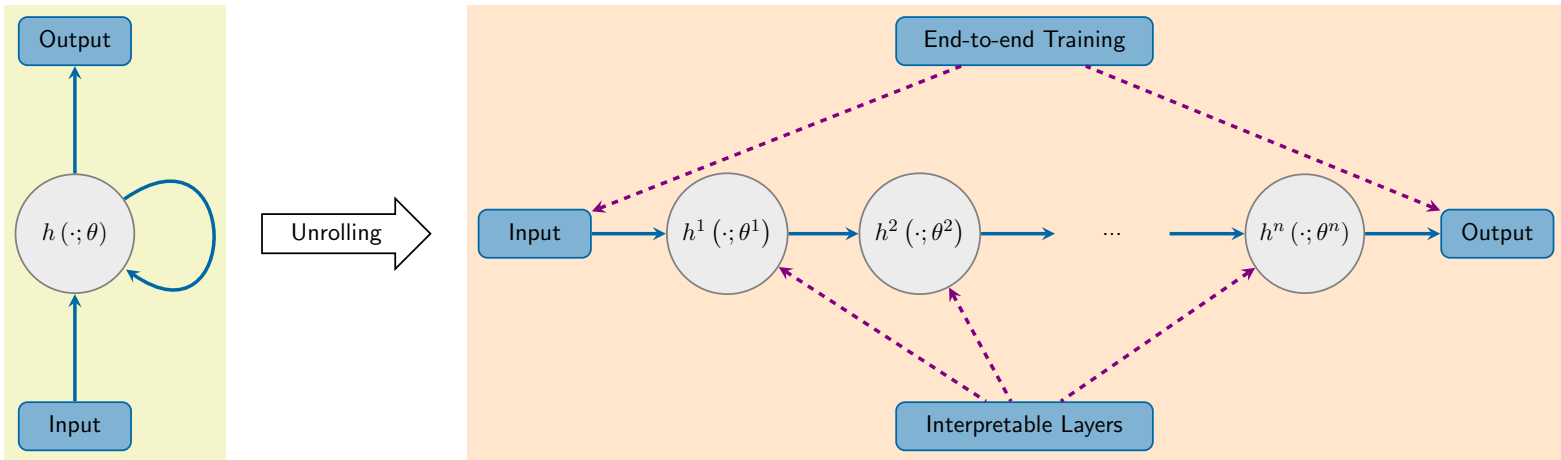
High level overview of deep unfolding

- In this sense, the algorithm is unfolded in a network with L layers.

Deep unfolding / algorithm unrolling (cont.)

Idea Given an *iterative* algorithm, [Monga, Li, Eldar '21]

- fix the number of iterations L and
- interpret each iteration of the algorithm as layer of a neural network.



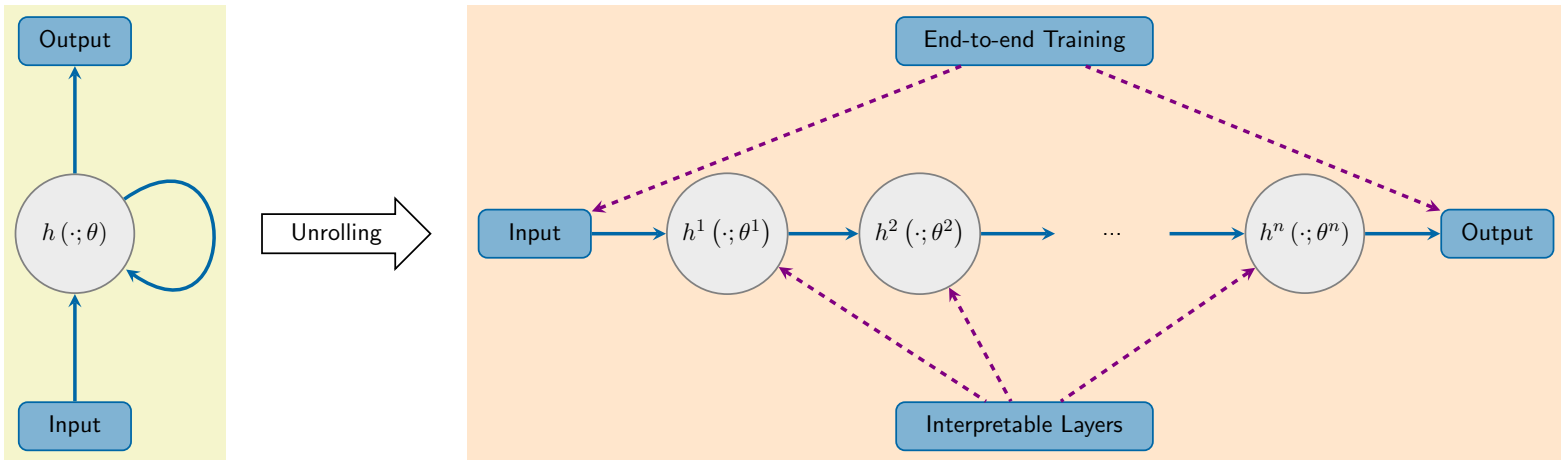
High level overview of deep unfolding

- In this sense, the algorithm is unfolded in a network with L layers.
- This process is called *deep unfolding* or also *algorithm unrolling*.

Deep unfolding / algorithm unrolling (cont.)

Idea Given an *iterative* algorithm, [Monga, Li, Eldar '21]

- fix the number of iterations L and
- interpret each iteration of the algorithm as layer of a neural network.



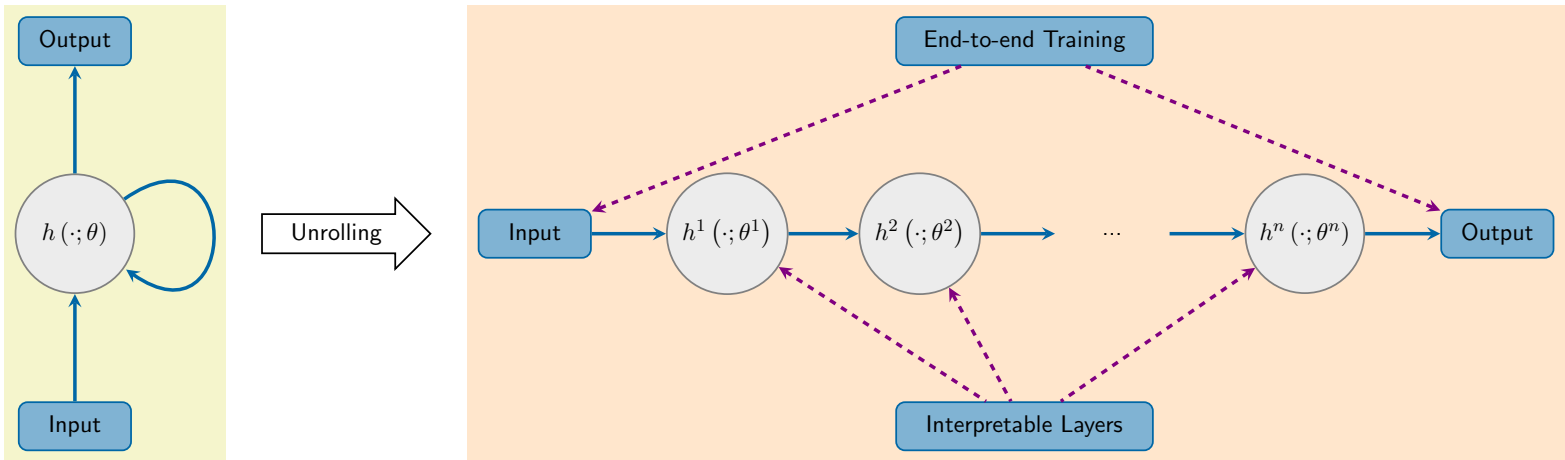
High level overview of deep unfolding

- In this sense, the algorithm is unfolded in a network with L layers.
- This process is called *deep unfolding* or also *algorithm unrolling*.
- The unfolding is just an equivalent reinterpretation of the algorithm, but allows one to look at the problem from a different perspective.

Deep unfolding / algorithm unrolling (cont.)

Idea Given an *iterative* algorithm, [Monga, Li, Eldar '21]

- fix the number of iterations L and
- interpret each iteration of the algorithm as layer of a neural network.



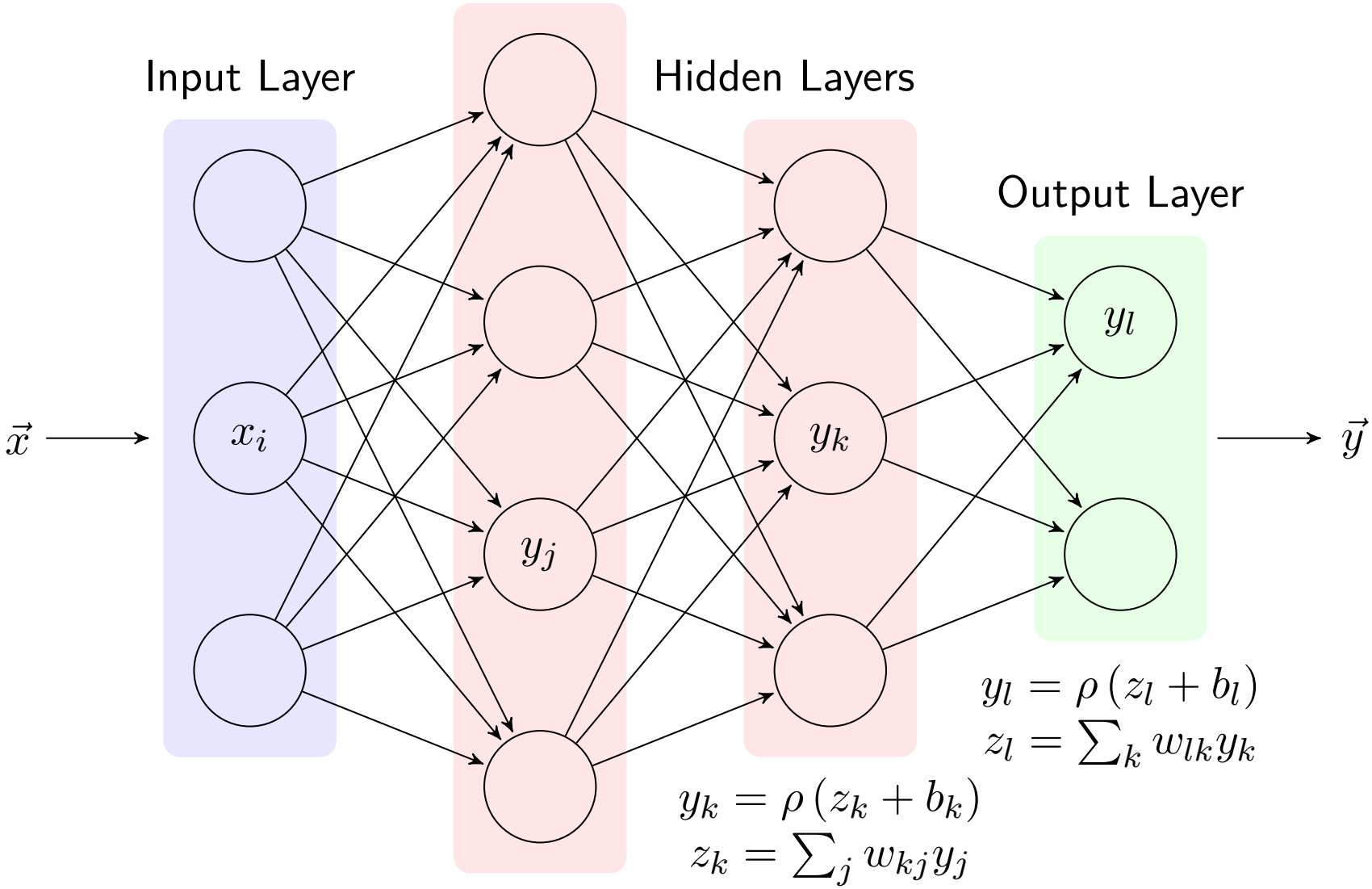
High level overview of deep unfolding

- In this sense, the algorithm is unfolded in a network with L layers.
- This process is called *deep unfolding* or also *algorithm unrolling*.
- The unfolding is just an equivalent reinterpretation of the algorithm, but allows one to look at the problem from a different perspective.
- One can make some of network parameters inherited from the algorithm trainable, which introduces data-driven learning.

Deep unfolding / algorithm unrolling (cont.)

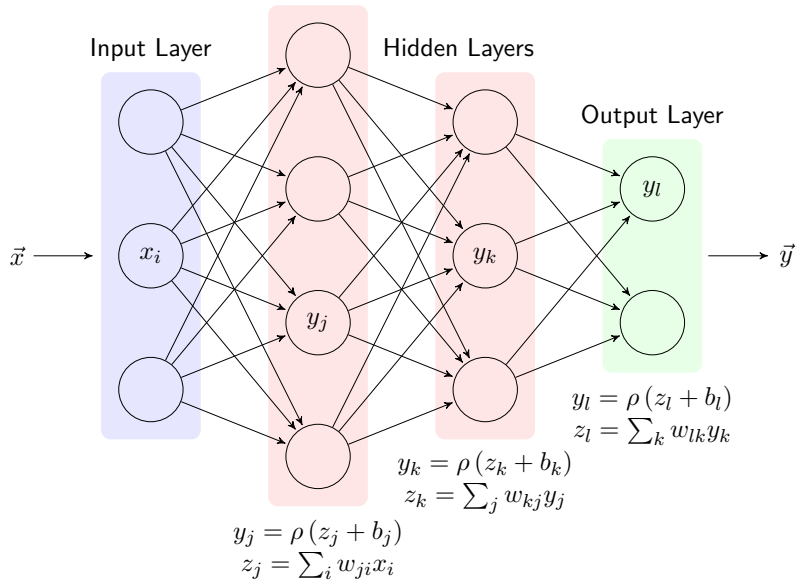
How to interpret an iteration as layer (or layers) of neural network?

Deep unfolding / algorithm unrolling (cont.)



Deep unfolding / algorithm unrolling (cont.)

How to interpret an iteration as layer (or layers) of neural network?

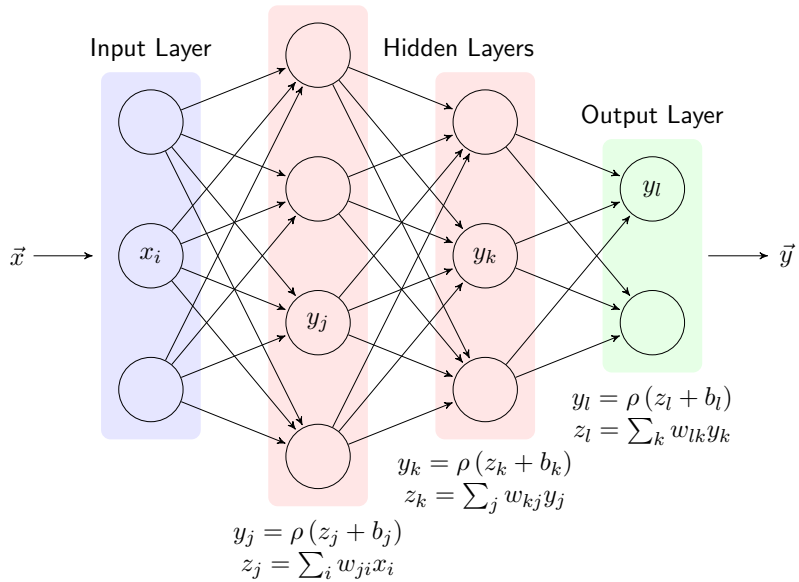


Identify the building blocks of a neural network layer in the iteration:

- *linearities*, i.e. $x \mapsto Wx + b$, where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and
- *element-wise nonlinearities*, i.e. $(x_1, \ldots, x_n) \mapsto (\rho(x_1), \ldots, \rho(x_n))$.

Deep unfolding / algorithm unrolling (cont.)

How to interpret an iteration as layer (or layers) of neural network?



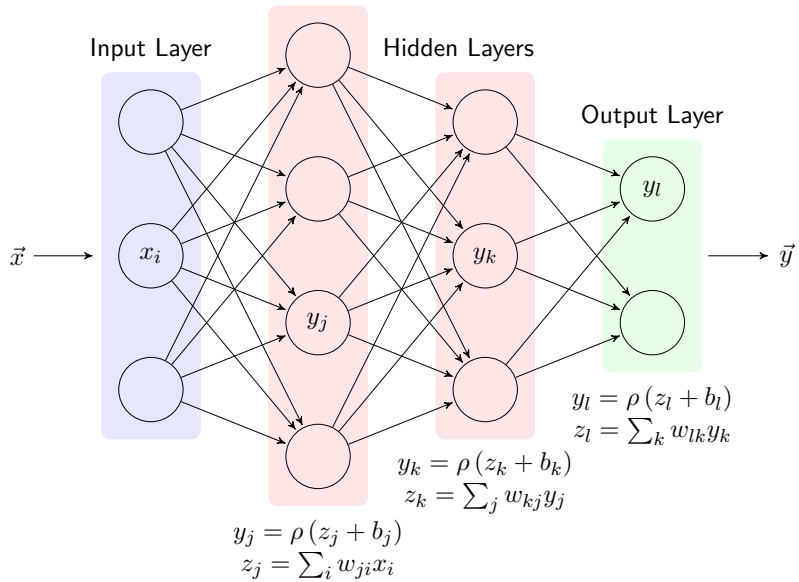
Identify the building blocks of a neural network layer in the iteration:

- *linearities*, i.e. $x \mapsto Wx + b$, where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and
- *element-wise nonlinearities*, i.e. $(x_1, \ldots, x_n) \mapsto (\rho(x_1), \ldots, \rho(x_n))$.

Thus, many, but not all, iterative algorithm are suitable for unfolding.

Deep unfolding / algorithm unrolling (cont.)

How to interpret an iteration as layer (or layers) of neural network?



Identify the building blocks of a neural network layer in the iteration:

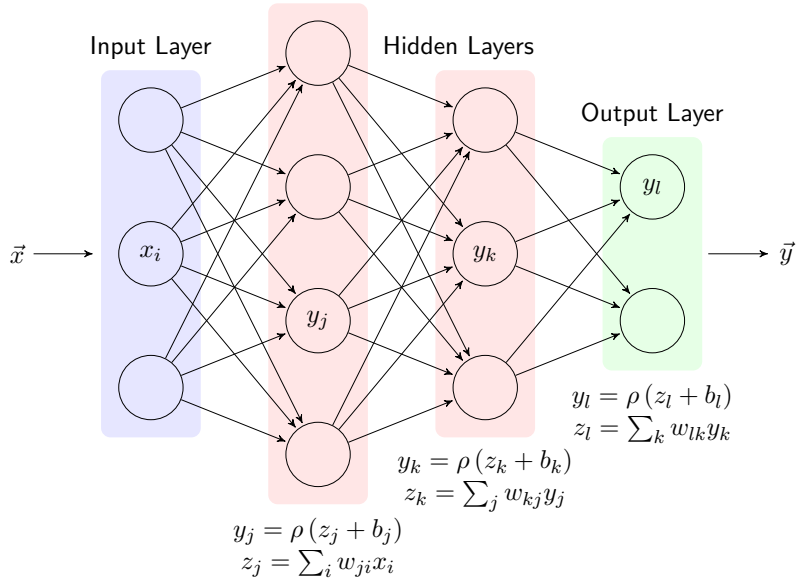
- *linearities*, i.e. $x \mapsto Wx + b$, where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and
- *element-wise nonlinearities*, i.e. $(x_1, \ldots, x_n) \mapsto (\rho(x_1), \ldots, \rho(x_n))$.

Thus, many, but not all, iterative algorithm are suitable for unfolding.

- The iterative soft thresholding algorithm (ISTA) is such an algorithm.

Deep unfolding / algorithm unrolling (cont.)

How to interpret an iteration as layer (or layers) of neural network?



Identify the building blocks of a neural network layer in the iteration:

- *linearities*, i.e. $x \mapsto Wx + b$, where $W \in \mathbb{R}^{m \times n}$, $x \in \mathbb{R}^n$, $b \in \mathbb{R}^m$ and
- *element-wise nonlinearities*, i.e. $(x_1, \ldots, x_n) \mapsto (\rho(x_1), \ldots, \rho(x_n))$.

Thus, many, but not all, iterative algorithm are suitable for unfolding.

- The iterative soft thresholding algorithm (ISTA) is such an algorithm.
- ISTA was the first unrolled algorithm $\rightarrow$ "Learned ISTA" (LISTA) [Gregor, LeCun '10].

Unfolding ISTA - LISTA [Gregor, LeCun '10]

Task Given $y \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times n}$ ($m < n$, overcomplete dictionary), find a *sparse* $x \in \mathbb{R}^n$ so that $y = Wx$ or $y \approx Wx$.

Unfolding ISTA - LISTA [Gregor, LeCun '10]

Task Given $y \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times n}$ ($m < n$, overcomplete dictionary), find a *sparse* $x \in \mathbb{R}^n$ so that $y = Wx$ or $y \approx Wx$.

$\rightarrow$ LASSO (least absolute shrinkage and selection operator) problem:

$$\min_x \tfrac{1}{2}\|y - Wx\|_2^2 + \lambda \|x\|_1.$$

Here, $\lambda > 0$ is a regularization parameter.

Unfolding ISTA - LISTA [Gregor, LeCun '10]

Task Given $y \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times n}$ ($m < n$, overcomplete dictionary), find a *sparse* $x \in \mathbb{R}^n$ so that $y = Wx$ or $y \approx Wx$.

$\rightarrow$ LASSO (least absolute shrinkage and selection operator) problem:

$$\min_x \tfrac{1}{2}\|y - Wx\|_2^2 + \lambda \|x\|_1.$$

Here, $\lambda > 0$ is a regularization parameter.

ISTA solves the LASSO problem and is given by the iteration:

$$x^{l+1} = \mathcal{S}_\lambda \left(x^l + \frac{1}{\mu} W^T (y - W x^l) \right).$$

Unfolding ISTA - LISTA [Gregor, LeCun '10]

Task Given $y \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times n}$ ($m < n$, overcomplete dictionary), find a *sparse* $x \in \mathbb{R}^n$ so that $y = Wx$ or $y \approx Wx$.

$\rightarrow$ LASSO (least absolute shrinkage and selection operator) problem:

$$\min_x \tfrac{1}{2}\|y - Wx\|_2^2 + \lambda \|x\|_1.$$

Here, $\lambda > 0$ is a regularization parameter.

ISTA solves the LASSO problem and is given by the iteration:

$$x^{l+1} = \mathcal{S}_\lambda \left(x^l + \frac{1}{\mu} W^T (y - W x^l) \right).$$

Here, $\mathcal{S}_\lambda$ is the element-wise soft thresholding operator

$$\mathcal{S}_\lambda(z) = \operatorname{sign}(z) \cdot \max\{|z| - \lambda, 0\}$$

and μ is the largest EV of $W^T W$.

Unfolding ISTA - LISTA [Gregor, LeCun '10]

Task Given $y \in \mathbb{R}^m$ and $W \in \mathbb{R}^{m \times n}$ ($m < n$, overcomplete dictionary), find a *sparse* $x \in \mathbb{R}^n$ so that $y = Wx$ or $y \approx Wx$.

$\rightarrow$ LASSO (least absolute shrinkage and selection operator) problem:

$$\min_x \tfrac{1}{2}\|y - Wx\|_2^2 + \lambda \|x\|_1.$$

Here, $\lambda > 0$ is a regularization parameter.

ISTA solves the LASSO problem and is given by the iteration:

$$x^{l+1} = \mathcal{S}_\lambda \left(x^l + \frac{1}{\mu} W^T (y - W x^l) \right).$$

Here, $\mathcal{S}_\lambda$ is the element-wise soft thresholding operator

$$\mathcal{S}_\lambda(z) = \operatorname{sign}(z) \cdot \max\{|z| - \lambda, 0\}$$

and μ is the largest EV of $W^T W$. Rearranging the ISTA iteration gives

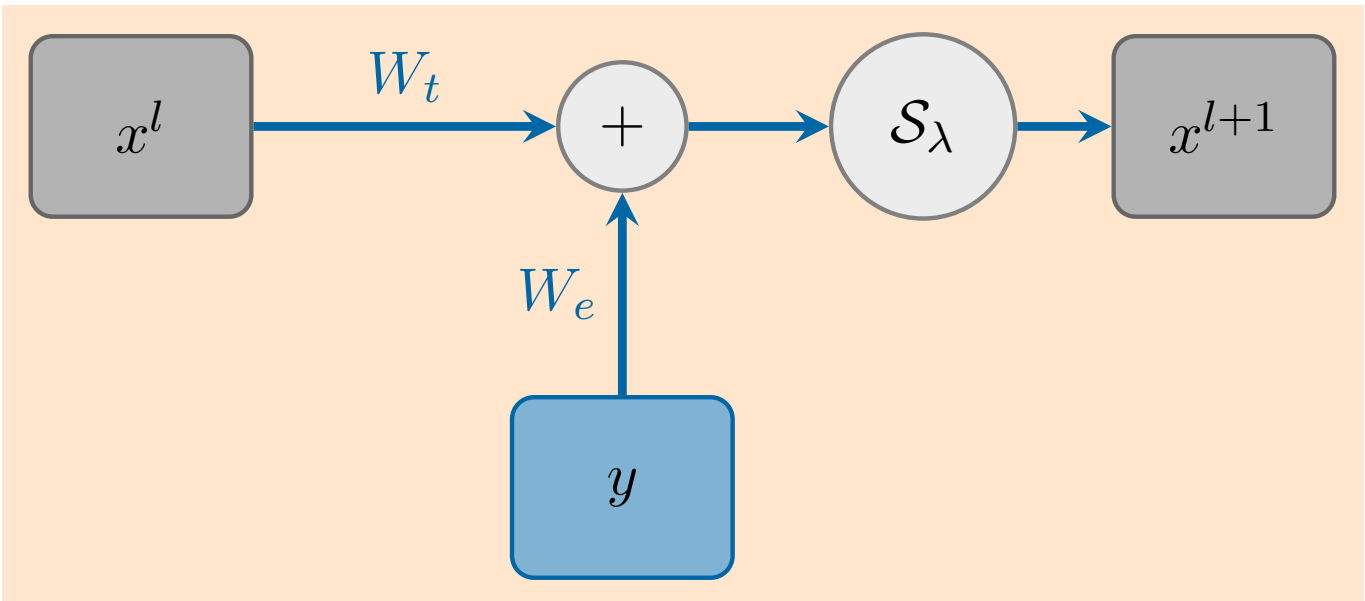
$$x^{l+1} = \mathcal{S}_\lambda \left(\left(\mathcal{I} - \frac{1}{\mu} W^T W \right) x^l + \frac{1}{\mu} W^T y \right).$$

LISTA (cont.)

Defining $W_t := \mathcal{I} - \frac{1}{\mu} W^T W$ and $W_e := \frac{1}{\mu} W^T$, the ISTA iteration is

$$x^{l+1} = \mathcal{S}_\lambda \left(W_t x^l + W_e y \right).$$

We can represent this as one layer of a network:



Stacking L layers, we get the unrolled ISTA.

LISTA (cont.)

Now that we have the unrolled network, we need trainable parameters.

LISTA (cont.)

Now that we have the unrolled network, we need trainable parameters.

Gregor and LeCun proposed to learn

- the matrices W_t, W_e and
- the threshold λ, replaced by a vector of thresholds $\theta \in \mathbb{R}^n$.

LISTA (cont.)

Now that we have the unrolled network, we need trainable parameters.

Gregor and LeCun proposed to learn

- the matrices W_t, W_e and
- the threshold λ, replaced by a vector of thresholds $\theta \in \mathbb{R}^n$.

Training

Let $h(y; W_t, W_e, \theta)$ denote the network output for input y and parameters W_t, W_e, θ. Given training vectors $y^1, \ldots, y^N$ and corresponding ground truth sparse vectors $\hat{x}^1, \ldots, \hat{x}^N$, minimize the squared loss, i.e.

$$\min_{W_t, W_e, \theta} \sum_{n=1}^{N} \|h(y^n; W_t, W_e, \theta) - \hat{x}^n\|_2^2.$$

LISTA (cont.)

Now that we have the unrolled network, we need trainable parameters.

Gregor and LeCun proposed to learn

- the matrices W_t, W_e and
- the threshold λ, replaced by a vector of thresholds $\theta \in \mathbb{R}^n$.

Training

Let $h(y; W_t, W_e, \theta)$ denote the network output for input y and parameters W_t, W_e, θ. Given training vectors $y^1, \ldots, y^N$ and corresponding ground truth sparse vectors $\hat{x}^1, \ldots, \hat{x}^N$, minimize the squared loss, i.e.

$$\min_{W_t, W_e, \theta} \sum_{n=1}^{N} \|h(y^n; W_t, W_e, \theta) - \hat{x}^n\|_2^2.$$

In the low iteration regime, LISTA reaches prediction errors comparable to roughly $20L$ "Fast ISTA" (FISTA) iterations.

LISTA (cont.)

Now that we have the unrolled network, we need trainable parameters.

Gregor and LeCun proposed to learn

- the matrices W_t, W_e and
- the threshold λ, replaced by a vector of thresholds $\theta \in \mathbb{R}^n$.

Training

Let $h(y; W_t, W_e, \theta)$ denote the network output for input y and parameters W_t, W_e, θ. Given training vectors $y^1, \ldots, y^N$ and corresponding ground truth sparse vectors $\hat{x}^1, \ldots, \hat{x}^N$, minimize the squared loss, i.e.

$$\min_{W_t, W_e, \theta} \sum_{n=1}^{N} \|h(y^n; W_t, W_e, \theta) - \hat{x}^n\|_2^2.$$

In the low iteration regime, LISTA reaches prediction errors comparable to roughly $20L$ "Fast ISTA" (FISTA) iterations.

In other words, the number of iterations is reduced by a factor of 20.

Examples of rigorous statements on unfolded networks

- Convergence rate (of the output for $L \to \infty$) of LISTA unknown.

Examples of rigorous statements on unfolded networks

- Convergence rate (of the output for $L \to \infty$) of LISTA unknown.
- With layer dependent weights and the coupling condition

$$W_t^\ell = \mathcal{I} - W_e^\ell W$$

unrolled ISTA (LISTA-CP) can attain linear convergence.

[Chen, Liu, Wang, Yin '18]

Recall: ISTA/FISTA have sublinear convergence in general cases.

Examples of rigorous statements on unfolded networks

- Convergence rate (of the output for $L \to \infty$) of LISTA unknown.
- With layer dependent weights and the coupling condition

$$W_t^\ell = \mathcal{I} - W_e^\ell W$$

unrolled ISTA (LISTA-CP) can attain linear convergence.

[Chen, Liu, Wang, Yin '18]

Recall: ISTA/FISTA have sublinear convergence in general cases.

- Analytic LISTA (ALISTA): The weight matrix in LISTA can be computed as solution to a data-free optimization problem, while retaining linear convergence. Here, only the step size and threshold parameters are learned. [Liu, Chen, Wang, Yin '19]

Examples of rigorous statements on unfolded networks

- Convergence rate (of the output for $L \to \infty$) of LISTA unknown.
- With layer dependent weights and the coupling condition

$$W_t^\ell = \mathcal{I} - W_e^\ell W$$

unrolled ISTA (LISTA-CP) can attain linear convergence.

[Chen, Liu, Wang, Yin '18]

Recall: ISTA/FISTA have sublinear convergence in general cases.

- Analytic LISTA (ALISTA): The weight matrix in LISTA can be computed as solution to a data-free optimization problem, while retaining linear convergence. Here, only the step size and threshold parameters are learned. [Liu, Chen, Wang, Yin '19]
- For unfolded ISTA with a sparsity basis as network parameter, generalization bounds can be derived.

[Behboodi, Rauhut, Schnoor '20]

Potential benefits of unfolding

Interpretability:

- Unfolded networks are structured much like the (interpretable) algorithms that inspired them.

Potential benefits of unfolding

Interpretability:

- Unfolded networks are structured much like the (interpretable) algorithms that inspired them.

Computational efficiency:

- Unfolded networks typically need fewer layers than the number of iterations required in the algorithm.
  (One layer is usually comparable in computational cost to one iteration of the algorithm.)

Potential benefits of unfolding

Interpretability:

- Unfolded networks are structured much like the (interpretable) algorithms that inspired them.

Computational efficiency:

- Unfolded networks typically need fewer layers than the number of iterations required in the algorithm.
  (One layer is usually comparable in computational cost to one iteration of the algorithm.)

Training efficiency:

- Unfolded networks typically have far fewer parameters than conventional deep networks, since they benefit from the domain knowledge of the algorithms.
- This allows them to be trained using less data.

Potential benefits of unfolding

Interpretability:

- Unfolded networks are structured much like the (interpretable) algorithms that inspired them.

Computational efficiency:

- Unfolded networks typically need fewer layers than the number of iterations required in the algorithm.
  (One layer is usually comparable in computational cost to one iteration of the algorithm.)

Training efficiency:

- Unfolded networks typically have far fewer parameters than conventional deep networks, since they benefit from the domain knowledge of the algorithms.
- This allows them to be trained using less data.

Analyzability:

- Unfolded networks are typically more accessible to a theoretical understanding and rigorous analysis than conventional deep networks.

Unfolding exit wave reconstruction

A simplified exit wave model - Phase retrieval

Exit wave reconstruction objective (assuming no image shifts)

$$\mathcal{E}[\Psi] = \frac{1}{K} \sum_{j=1}^{K} \left\| \Psi \star_{T_j} \Psi - \mathcal{F}(g_j) \right\|_{L^2}^2 + \mathcal{R}(\Psi),$$

where $\mathcal{R}$ is a suitable regularizer.

A simplified exit wave model - Phase retrieval

Exit wave reconstruction objective (assuming no image shifts)

$$\mathcal{E}[\Psi] = \frac{1}{K}\sum_{j=1}^{K} \left\| \Psi \star_{T_j} \Psi - \mathcal{F}(g_j) \right\|_{L^2}^2 + \mathcal{R}(\Psi),$$

where $\mathcal{R}$ is a suitable regularizer.

Assuming there are $w_j : \mathbb{R}^d \to \mathbb{C}$ with $T_j(x,y) = \overline{w_j(x)} w_j(y)$, we get

$$\begin{aligned} &\Psi \star_{T_j} \Psi = (\Psi w_j) \star (\Psi w_j) \\ &\Rightarrow \quad \mathcal{F}^{-1}(\Psi \star_{T_j} \Psi) = \overline{\mathcal{F}^{-1}(\Psi w_j)} \mathcal{F}^{-1}(\Psi w_j) = \left| \mathcal{F}^{-1}(\Psi w_j) \right|^2 \end{aligned}$$

A simplified exit wave model - Phase retrieval

Exit wave reconstruction objective (assuming no image shifts)

$$\mathcal{E}[\Psi] = \frac{1}{K}\sum_{j=1}^{K} \left\| \Psi \star_{T_j} \Psi - \mathcal{F}(g_j) \right\|_{L^2}^2 + \mathcal{R}(\Psi),$$

where $\mathcal{R}$ is a suitable regularizer.

Assuming there are $w_j : \mathbb{R}^d \to \mathbb{C}$ with $T_j(x,y) = \overline{w_j(x)} w_j(y)$, we get

$$\begin{aligned} &\Psi \star_{T_j} \Psi = (\Psi w_j) \star (\Psi w_j) \\ \Rightarrow\ &\mathcal{F}^{-1}(\Psi \star_{T_j} \Psi) = \overline{\mathcal{F}^{-1}(\Psi w_j)} \mathcal{F}^{-1}(\Psi w_j) = \left| \mathcal{F}^{-1}(\Psi w_j) \right|^2 \end{aligned}$$

Since $\mathcal{F}$ is a unitary transform, $\mathcal{E}$ can be rephrased to

$$\mathcal{E}[\Psi] = \frac{1}{K}\sum_{j=1}^{K} \left\| \left| \mathcal{F}^{-1}(\Psi w_j) \right|^2 - g_j \right\|_{L^2}^2 + \mathcal{R}(\Psi).$$

A simplified exit wave model - Phase retrieval

Exit wave reconstruction objective (assuming no image shifts)

$$\mathcal{E}[\Psi] = \frac{1}{K}\sum_{j=1}^{K} \left\| \Psi \star_{T_j} \Psi - \mathcal{F}(g_j) \right\|_{L^2}^2 + \mathcal{R}(\Psi),$$

where $\mathcal{R}$ is a suitable regularizer.

Assuming there are $w_j : \mathbb{R}^d \to \mathbb{C}$ with $T_j(x,y) = \overline{w_j(x)} w_j(y)$, we get

$$\begin{aligned} &\Psi \star_{T_j} \Psi = (\Psi w_j) \star (\Psi w_j) \\ \Rightarrow\ &\mathcal{F}^{-1}(\Psi \star_{T_j} \Psi) = \overline{\mathcal{F}^{-1}(\Psi w_j)} \mathcal{F}^{-1}(\Psi w_j) = \left| \mathcal{F}^{-1}(\Psi w_j) \right|^2 \end{aligned}$$

Since $\mathcal{F}$ is a unitary transform, $\mathcal{E}$ can be rephrased to

$$\mathcal{E}[\Psi] = \frac{1}{K}\sum_{j=1}^{K} \left\| \left| \mathcal{F}^{-1}(\Psi w_j) \right|^2 - g_j \right\|_{L^2}^2 + \mathcal{R}(\Psi).$$

$\Rightarrow$ With the simplified T_j, we have a *phase retrieval* problem.

A simplified exit wave model - Discretization

Discretization with N nodes leads to following objective function structure

$$E[\psi] = \underbrace{\frac{1}{2KN} \sum_{j=1}^{K} \|\varphi(A_j\psi) - G_j\|_2^2}_{=:D(\psi)} + R(\psi),$$

where

- $\psi \in \mathbb{C}^N$ discrete exit wave
- $A_j \in \mathbb{C}^{N\times N}$ inverse DFT + weighting with a discretization of w_j
- $G_j \in \mathbb{R}^N$ discretization of g_j
- R discretization of the regularizer $\mathcal{R}$
- $\varphi : \mathbb{C} \to \mathbb{R}, z \mapsto |z|^2$
- $\varphi(G)$ for $G \in \mathbb{C}^N$ means that φ is applied element-wise

A simplified exit wave model - Discretization

Discretization with N nodes leads to following objective function structure

$$E[\psi] = \underbrace{\frac{1}{2KN} \sum_{j=1}^{K} \|\varphi(A_j \psi) - G_j\|_2^2}_{=:D(\psi)} + R(\psi),$$

where

- $\psi \in \mathbb{C}^N$ discrete exit wave
- $A_j \in \mathbb{C}^{N \times N}$ inverse DFT + weighting with a discretization of w_j
- $G_j \in \mathbb{R}^N$ discretization of g_j
- R discretization of the regularizer $\mathcal{R}$
- $\varphi : \mathbb{C} \to \mathbb{R}, z \mapsto |z|^2$
- $\varphi(G)$ for $G \in \mathbb{C}^N$ means that φ is applied element-wise

How to compute a minimizer of E for a *sparsity promoting* R?

A simplified exit wave model - Discretization

Discretization with N nodes leads to following objective function structure

$$E[\psi] = \underbrace{\frac{1}{2KN} \sum_{j=1}^{K} \|\varphi(A_j \psi) - G_j\|_2^2}_{=:D(\psi)} + R(\psi),$$

where

- $\psi \in \mathbb{C}^N$ discrete exit wave
- $A_j \in \mathbb{C}^{N \times N}$ inverse DFT + weighting with a discretization of w_j
- $G_j \in \mathbb{R}^N$ discretization of g_j
- R discretization of the regularizer $\mathcal{R}$
- $\varphi : \mathbb{C} \to \mathbb{R}, z \mapsto |z|^2$
- $\varphi(G)$ for $G \in \mathbb{C}^N$ means that φ is applied element-wise

How to compute a minimizer of E for a *sparsity promoting* R?

Note: Sparsity promoting regularizers are often not differentiable!

A simplified exit wave model - Discretization

Discretization with N nodes leads to following objective function structure

$$E[\psi] = \underbrace{\frac{1}{2KN} \sum_{j=1}^{K} \|\varphi(A_j \psi) - G_j\|_2^2}_{=:D(\psi)} + R(\psi),$$

where

- $\psi \in \mathbb{C}^N$ discrete exit wave
- $A_j \in \mathbb{C}^{N \times N}$ inverse DFT + weighting with a discretization of w_j
- $G_j \in \mathbb{R}^N$ discretization of g_j
- R discretization of the regularizer $\mathcal{R}$
- $\varphi : \mathbb{C} \to \mathbb{R}, z \mapsto |z|^2$
- $\varphi(G)$ for $G \in \mathbb{C}^N$ means that φ is applied element-wise

How to compute a minimizer of E for a *sparsity promoting* R?

Note: Sparsity promoting regularizers are often not differentiable!

$\Rightarrow$ Classical gradient based methods cannot be used.

A simplified exit wave model - Discretization

Discretization with N nodes leads to following objective function structure

$$E[\psi] = \underbrace{\frac{1}{2KN} \sum_{j=1}^{K} \|\varphi(A_j \psi) - G_j\|_2^2}_{=:D(\psi)} + R(\psi),$$

where

- $\psi \in \mathbb{C}^N$ discrete exit wave
- $A_j \in \mathbb{C}^{N \times N}$ inverse DFT + weighting with a discretization of w_j
- $G_j \in \mathbb{R}^N$ discretization of g_j
- R discretization of the regularizer $\mathcal{R}$
- $\varphi : \mathbb{C} \to \mathbb{R}, z \mapsto |z|^2$
- $\varphi(G)$ for $G \in \mathbb{C}^N$ means that φ is applied element-wise

How to compute a minimizer of E for a *sparsity promoting* R?

Note: Sparsity promoting regularizers are often not differentiable!

$\Rightarrow$ Classical gradient based methods cannot be used.

Idea: Use convexity instead of differentiability.

A simplified exit wave model - Minimization

A typical approach to compute a minimizer of

$$E = D + R$$

with R convex, D convex, differentiability is

A simplified exit wave model - Minimization

A typical approach to compute a minimizer of

$$E = D + R$$

with R convex, D convex, differentiability is

$$\psi^{l+1} = \mathrm{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

which is called *proximal gradient algorithm*. Here,

- ∇ is the Wirtinger gradient (essentially interpreting $\mathbb{C}$ as $\mathbb{R}^2$)
- prox is the proximal operator

A simplified exit wave model - Minimization

A typical approach to compute a minimizer of

$$E = D + R$$

with R convex, D convex, differentiability is

$$\psi^{l+1} = \mathrm{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

which is called *proximal gradient algorithm*. Here,

- ∇ is the Wirtinger gradient (essentially interpreting $\mathbb{C}$ as $\mathbb{R}^2$)
- prox is the proximal operator

The proximal gradient algorithm is a forward-backward splitting.

A simplified exit wave model - Minimization

A typical approach to compute a minimizer of

$$E = D + R$$

with R convex, D convex, differentiability is

$$\psi^{l+1} = \operatorname{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

which is called *proximal gradient algorithm*. Here,

- ∇ is the Wirtinger gradient (essentially interpreting $\mathbb{C}$ as $\mathbb{R}^2$)
- prox is the proximal operator

The proximal gradient algorithm is a forward-backward splitting.

For D as before, one gets

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j\psi) - G_j \odot (A_j\psi)\right),$$

where

- $\hat{\varphi} : \mathbb{C} \to \mathbb{C}, z \mapsto 2\,|z|^2\, z$
- $\odot$ is the Hadamard product, i.e. element-wise multiplication

A simplified exit wave model - Minimization

A typical approach to compute a minimizer of

$$E = D + R$$

with R convex, D convex, differentiability is

$$\psi^{l+1} = \operatorname{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

which is called *proximal gradient algorithm*. Here,

- ∇ is the Wirtinger gradient (essentially interpreting $\mathbb{C}$ as $\mathbb{R}^2$)
- prox is the proximal operator

The proximal gradient algorithm is a forward-backward splitting.

For D as before, one gets

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi) \right),$$

where

- $\hat{\varphi} : \mathbb{C} \to \mathbb{C}, z \mapsto 2 |z|^2 z$
- $\odot$ is the Hadamard product, i.e. element-wise multiplication

Note: For $R = 0$ and spectral initialization of ϕ^0, this is *Wirtinger flow*.

Interpretation as neural network

$$\psi^{l+1} = \mathrm{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi)\right),$$

Interpretation as neural network

$$\psi^{l+1} = \mathrm{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi) \right),$$

- $\psi^l - \tau_l \nabla D(\psi^l)$ can be seen as layer in a (residual) neural network with the complex activation function $\hat{\varphi}$

Interpretation as neural network

$$\psi^{l+1} = \mathrm{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi) \right),$$

- $\psi^l - \tau_l \nabla D(\psi^l)$ can be seen as layer in a (residual) neural network with the complex activation function $\hat{\varphi}$
- step size τ_l is a parameter of the network

Interpretation as neural network

$$\psi^{l+1} = \operatorname{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi) \right),$$

- $\psi^l - \tau_l \nabla D(\psi^l)$ can be seen as layer in a (residual) neural network with the complex activation function $\hat{\varphi}$
- step size τ_l is a parameter of the network

For a suitable R, $\operatorname{prox}_{\tau_l R}$ can also be seen as network layer.

Interpretation as neural network

$$\psi^{l+1} = \operatorname{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi)\right),$$

- $\psi^l - \tau_l \nabla D(\psi^l)$ can be seen as layer in a (residual) neural network with the complex activation function $\hat{\varphi}$
- step size τ_l is a parameter of the network

For a suitable R, $\operatorname{prox}_{\tau_l R}$ can also be seen as network layer.

$R = \|\cdot\|_1 \Rightarrow \operatorname{prox}_{\tau_l R} =$ soft thresholding
(can be seen as application of another activation function)

Interpretation as neural network

$$\psi^{l+1} = \operatorname{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\hat{\varphi}(A_j \psi) - G_j \odot (A_j \psi)\right),$$

- $\psi^l - \tau_l \nabla D(\psi^l)$ can be seen as layer in a (residual) neural network with the complex activation function $\hat{\varphi}$
- step size τ_l is a parameter of the network

For a suitable R, $\operatorname{prox}_{\tau_l R}$ can also be seen as network layer.

$R = \|\cdot\|_1 \Rightarrow \operatorname{prox}_{\tau_l R} =$ soft thresholding
(can be seen as application of another activation function)

Note:

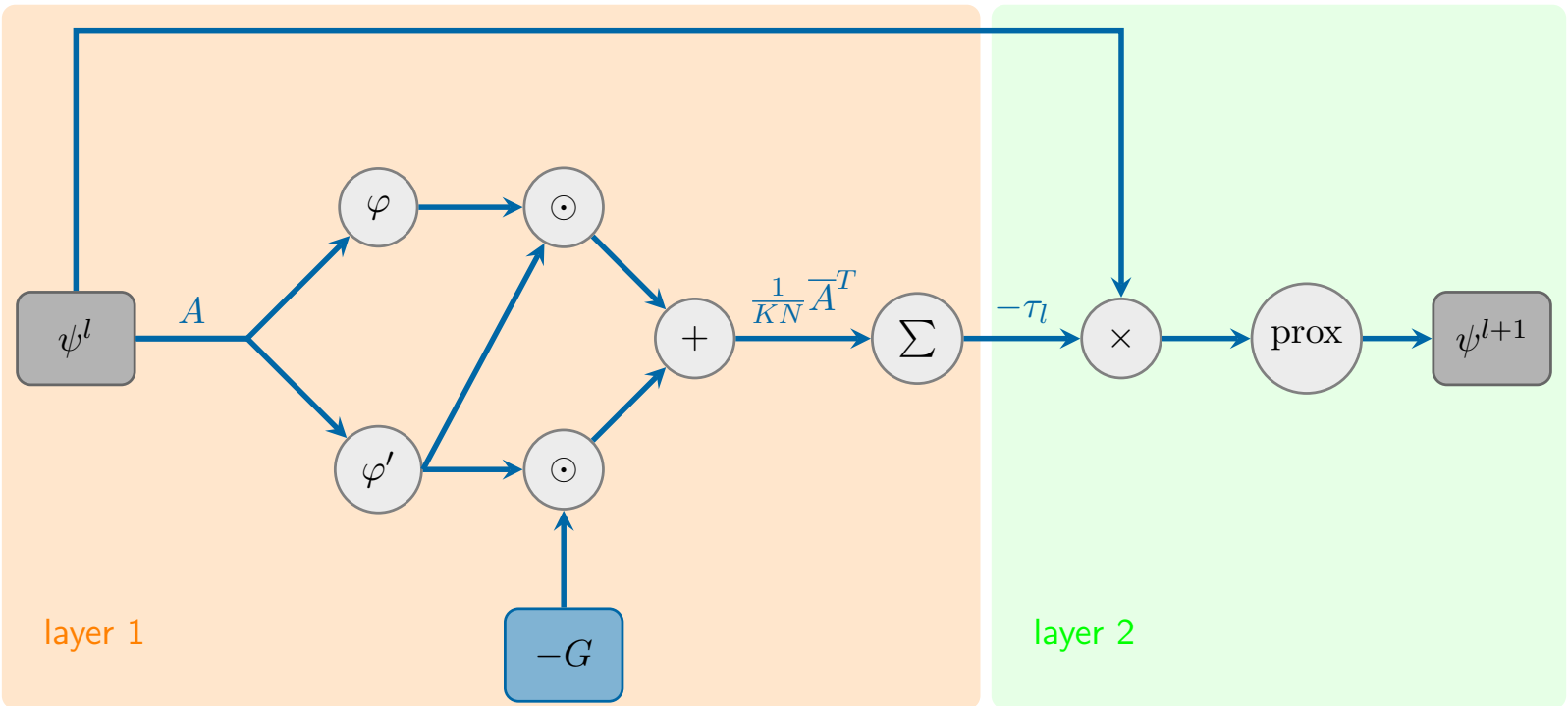
- For the application, assuming (convolutional) sparsity of ψ reasonable.
- The corresponding dictionary could be understood as further parameters of the network.

Interpretation as neural network (cont.)

The specific $\varphi : \mathbb{C} \to \mathbb{R}, z \mapsto |z|^2$ can be replaced by a general $\varphi : \mathbb{C} \to \mathbb{R}$. This changes the derivative to

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\varphi(A_j\psi) - G_j\right) \odot \varphi'(A_j\psi)$$

Here, two potential nonlinearities appear, φ and φ'.



Many approaches covered by the splitting

$$\psi^{l+1} = \mathrm{prox}_{\tau_l R}(\psi^l - \tau_l \nabla D(\psi^l)),$$

$$\nabla D(\psi) = \frac{1}{KN} \sum_{j=1}^{K} \overline{A_j^T} \left(\varphi(A_j\psi) - G_j\right) \odot \varphi'(A_j\psi)$$

Unregularized case ($R = 0$)

WF: $\varphi(z) = |z|^2$

RWF: $\varphi(z) = |z|$

TAF: $\varphi(z) = |z|$ + truncated ∇D (encoded in φ')

Regularized case ($R \neq 0$)

SPARTA: TAF + $R = I_{C_s}$, where C_s is the set of s-sparse vectors

IHTA $R = \lambda \|\cdot\|_0$, $\varphi(z) = z$, $K = 1$

ISTA $R = \lambda \|\cdot\|_1$, $\varphi(z) = z$, $K = 1$
+ $A_1 = A\Phi$ with a measurement matrix A and a dictionary matrix Φ $\longrightarrow$ [Behboodi, Rauhut, Schnoor '20]

Nonlinearites in the unfolded network

Forward part of the splitting $(\psi^l - \tau_l \nabla D(\psi^l)) \to \varphi\varphi'$ and φ'

	$\varphi(z)$	$\varphi'(z)$	$\varphi(z)\varphi'(z)$
WF	$\lvert z\rvert^2$	$2z$ (linear)	$2\lvert z\rvert^2 z$
RWF	$\lvert z\rvert$	$\frac{z}{\lvert z\rvert}$	z (linear)
	Huber	nonlinear	nonlinear

Backward part of the splitting $\to \operatorname{prox}_{\tau_l R}$

R	$\operatorname{prox}_{\tau_l R}(y)$
$R = 0$	identity (linear)
$R = \lambda \lVert\cdot\rVert_1$	soft thresholding
$R = \lambda \lVert\cdot\rVert_0$	hard thresholding
$R = I_{C_s}$	top-s operator

Potentially learnable parameters

- (scalar) step sizes τ_l
- shared positive semi-definite matrix S
  (change step sizes to $\tau_l S$)
- measurement weights w_j
- input weights c_j
  (changing the data term to $\sum_{j=1}^{K} \|c_j \odot (\varphi(A_j\psi) - G_j)\|_2^2$)
- operator conjugate $\overline{A_j^T}$
  (combined with scaling s.t. $\|(\varphi(A_j\psi) - G_j) \odot \varphi'(A_j\psi)\|_2 \leq 1$)
- regularizes weight λ
- dictionary matrix Φ (replacing A_j with $A_j\Phi$)
- focus values z_j
- ...

Minimization using the proximal operator

The proximal operator

Proposition Let X be a reflexive Banach space and $J \in \Gamma_0(X)$, i.e., J is a closed proper convex functional on X. Then, the mapping

$$\operatorname{prox}_J : X \to X, y \mapsto \operatorname*{argmin}_{u \in X} \left(J[u] + \frac{1}{2} \|u - y\|^2 \right)$$

is well-defined (i.e. there is a unique minimizer) and is called *proximal mapping / proximal operator*.

The proximal operator

Proposition Let X be a reflexive Banach space and $J \in \Gamma_0(X)$, i.e., J is a closed proper convex functional on X. Then, the mapping

$$\operatorname{prox}_J : X \to X, y \mapsto \operatorname*{argmin}_{u \in X} \left(J[u] + \frac{1}{2} \|u - y\|^2 \right)$$

is well-defined (i.e. there is a unique minimizer) and is called *proximal mapping / proximal operator*.

- Existence of a minimizer can be shown with the direct method.

The proximal operator

Proposition Let X be a reflexive Banach space and $J \in \Gamma_0(X)$, i.e., J is a closed proper convex functional on X. Then, the mapping

$$\operatorname{prox}_J : X \to X, y \mapsto \operatorname*{argmin}_{u \in X} \left(J[u] + \frac{1}{2} \|u - y\|^2 \right)$$

is well-defined (i.e. there is a unique minimizer) and is called *proximal mapping / proximal operator*.

- Existence of a minimizer can be shown with the direct method.
- $\Gamma_0(X)$ was constructed such that the direct method can be applied.

The proximal operator

Proposition Let X be a reflexive Banach space and $J \in \Gamma_0(X)$, i.e., J is a closed proper convex functional on X. Then, the mapping

$$\operatorname{prox}_J : X \to X, y \mapsto \operatorname*{argmin}_{u \in X} \left(J[u] + \frac{1}{2} \|u - y\|^2 \right)$$

is well-defined (i.e. there is a unique minimizer) and is called *proximal mapping / proximal operator*.

- Existence of a minimizer can be shown with the direct method.
- $\Gamma_0(X)$ was constructed such that the direct method can be applied.
- Uniqueness of the minimizer follows from the strict convexity of $\|\cdot\|^2$.

The proximal operator

Proposition Let X be a reflexive Banach space and $J \in \Gamma_0(X)$, i.e., J is a closed proper convex functional on X. Then, the mapping

$$\operatorname{prox}_J : X \to X, y \mapsto \operatorname*{argmin}_{u \in X} \left(J[u] + \frac{1}{2} \|u - y\|^2 \right)$$

is well-defined (i.e. there is a unique minimizer) and is called *proximal mapping / proximal operator*.

- Existence of a minimizer can be shown with the direct method.
- $\Gamma_0(X)$ was constructed such that the direct method can be applied.
- Uniqueness of the minimizer follows from the strict convexity of $\|\cdot\|^2$.

Note: $J \in \Gamma_0(X)$ doesn't ensure $\operatorname*{argmin}_{x \in X} J[x] \neq \emptyset$, e.g. $X = \mathbb{R}$, $J[x] = x$.

The proximal operator

Proposition Let X be a reflexive Banach space and $J \in \Gamma_0(X)$, i.e., J is a closed proper convex functional on X. Then, the mapping

$$\operatorname{prox}_J : X \to X, y \mapsto \operatorname*{argmin}_{u \in X} \left(J[u] + \frac{1}{2} \|u - y\|^2 \right)$$

is well-defined (i.e. there is a unique minimizer) and is called *proximal mapping / proximal operator*.

- Existence of a minimizer can be shown with the direct method.
- $\Gamma_0(X)$ was constructed such that the direct method can be applied.
- Uniqueness of the minimizer follows from the strict convexity of $\|\cdot\|^2$.

Note: $J \in \Gamma_0(X)$ doesn't ensure $\operatorname*{argmin}_{x \in X} J[x] \neq \emptyset$, e.g. $X = \mathbb{R}$, $J[x] = x$.

Corollary Let $J \in \Gamma_0(X)$ and $\tau > 0$. Then,

$$y^* \in \operatorname*{argmin}_{y \in X} J[y] \Leftrightarrow y^* = \operatorname{prox}_{\tau J}[y^*].$$

The proximal operator - Examples

Now, confine to $X = \mathbb{R}^n$ and $\|\cdot\| = \|\cdot\|_2$, i.e. *Discretize Then Optimize*.

- For $J \equiv c \in \mathbb{R}$, we have

$$\operatorname{prox}_{\tau J}(y) = \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(\tau c + \frac{1}{2} \|u - y\|_2^2 \right) = y.$$

The proximal operator - Examples

Now, confine to $X = \mathbb{R}^n$ and $\|\cdot\| = \|\cdot\|_2$, i.e. *Discretize Then Optimize*.

- For $J \equiv c \in \mathbb{R}$, we have

$$\operatorname{prox}_{\tau J}(y) = \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(\tau c + \frac{1}{2} \|u - y\|_2^2 \right) = y.$$

- Let $g \in \mathbb{R}^n$ and $J(y) = \frac{1}{2} \sum_{i=1}^{n} (y_i - g_i)^2 = \frac{1}{2} \|y - g\|_2^2$. Then,

$$u^* := \operatorname{prox}_{\tau J}(y) = \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(\frac{\tau}{2} \|u - g\|_2^2 + \frac{1}{2} \|u - y\|_2^2 \right)$$

$$\Rightarrow 0 = \tau(u^* - g) + (u^* - y) \Rightarrow \operatorname{prox}_{\tau J}(y) = \frac{y + \tau g}{1 + \tau}$$

The proximal operator - Examples

Now, confine to $X = \mathbb{R}^n$ and $\|\cdot\| = \|\cdot\|_2$, i.e. *Discretize Then Optimize*.

- For $J \equiv c \in \mathbb{R}$, we have

$$\operatorname{prox}_{\tau J}(y) = \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(\tau c + \frac{1}{2} \|u - y\|_2^2 \right) = y.$$

- Let $g \in \mathbb{R}^n$ and $J(y) = \frac{1}{2} \sum_{i=1}^{n} (y_i - g_i)^2 = \frac{1}{2} \|y - g\|_2^2$. Then,

$$u^* := \operatorname{prox}_{\tau J}(y) = \operatorname*{argmin}_{u \in \mathbb{R}^n} \left(\frac{\tau}{2} \|u - g\|_2^2 + \frac{1}{2} \|u - y\|_2^2 \right)$$

$$\Rightarrow 0 = \tau(u^* - g) + (u^* - y) \Rightarrow \operatorname{prox}_{\tau J}(y) = \frac{y + \tau g}{1 + \tau}$$

- For $J(y) = \sum_{i=1}^{n} J_i(y_i)$ with $J_i \in \Gamma_0(\mathbb{R})$, we have

$$\operatorname{prox}_{\tau J}(y) = (\operatorname{prox}_{\tau J_1}(y_1), \ldots, \operatorname{prox}_{\tau J_n}(y_n)).$$

The proximal operator - Examples (cont.)

- For $J(y) := \|y\|_1$, $\mathrm{prox}_{\tau J}(y)$ is the *soft threshold* operator, i.e.

$$(\mathrm{prox}_{\tau J}(y))_i = \begin{cases} y_i - \tau & y_i \geq \tau \\ 0 & |y_i| < \tau \\ y_i + \tau & y_i \leq -\tau \end{cases} = \mathcal{S}_\tau(y_i).$$

The proximal operator - Examples (cont.)

- For $J(y) := \|y\|_1$, $\operatorname{prox}_{\tau J}(y)$ is the *soft threshold* operator, i.e.

$$(\operatorname{prox}_{\tau J}(y))_i = \begin{cases} y_i - \tau & y_i \geq \tau \\ 0 & |y_i| < \tau \\ y_i + \tau & y_i \leq -\tau \end{cases} = \mathcal{S}_\tau(y_i).$$

- If $C \subset \mathbb{R}^n$ is a nonempty, closed, convex set, then

$$I_C : \mathbb{R}^n \to \mathbb{R}_\infty, y \mapsto \begin{cases} 0 & y \in C \\ \infty & y \notin C, \end{cases}$$

  is called *indicator function* of C in $\Gamma_0(\mathbb{R}^n)$

The proximal operator - Examples (cont.)

- For $J(y) := \|y\|_1$, $\operatorname{prox}_{\tau J}(y)$ is the *soft threshold* operator, i.e.

$$(\operatorname{prox}_{\tau J}(y))_i = \begin{cases} y_i - \tau & y_i \geq \tau \\ 0 & |y_i| < \tau \\ y_i + \tau & y_i \leq -\tau \end{cases} = \mathcal{S}_\tau(y_i).$$

- If $C \subset \mathbb{R}^n$ is a nonempty, closed, convex set, then

$$I_C : \mathbb{R}^n \to \mathbb{R}_\infty, y \mapsto \begin{cases} 0 & y \in C \\ \infty & y \notin C, \end{cases}$$

is called *indicator function* of C in $\Gamma_0(\mathbb{R}^n)$ and we have

$$\operatorname{prox}_{\tau I_C}(y) = \Pi_C(y),$$

where Π_C is the Euclidean projection to C, i.e.

$$\Pi_C(y) = \operatorname*{argmin}_{z \in C} \|z - y\|_2 .$$

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\mathrm{prox}_{\tau J}$ are equivalent:

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\mathrm{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \mathrm{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\mathrm{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \mathrm{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

If a minimizer of J exists, y^k converges to the set of minimizers and $J(y^k)$ to the optimal value.

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\operatorname{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \operatorname{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

If a minimizer of J exists, y^k converges to the set of minimizers and $J(y^k)$ to the optimal value.

If $J \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$, then $y^{k+1} = \operatorname{prox}_{\tau J}(y^k)$ is characterized by

$$0 = \tau \nabla J(y^{k+1}) + (y^{k+1} - y^k) \;\Rightarrow\; y^{k+1} = y^k - \tau \nabla J(y^{k+1})$$

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\mathrm{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \mathrm{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

If a minimizer of J exists, y^k converges to the set of minimizers and $J(y^k)$ to the optimal value.

If $J \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$, then $y^{k+1} = \mathrm{prox}_{\tau J}(y^k)$ is characterized by

$$0 = \tau \nabla J(y^{k+1}) + (y^{k+1} - y^k) \;\Rightarrow\; y^{k+1} = y^k - \tau \nabla J(y^{k+1})$$

This is the backward Euler discretization of the gradient descent of J.

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\mathrm{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \mathrm{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

If a minimizer of J exists, y^k converges to the set of minimizers and $J(y^k)$ to the optimal value.

If $J \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$, then $y^{k+1} = \mathrm{prox}_{\tau J}(y^k)$ is characterized by

$$0 = \tau \nabla J(y^{k+1}) + (y^{k+1} - y^k) \;\Rightarrow\; y^{k+1} = y^k - \tau \nabla J(y^{k+1})$$

This is the backward Euler discretization of the gradient descent of J.

Thus, for differentiable J, the proximal point algorithm is equivalent to the fully implicit gradient descent.

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\operatorname{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \operatorname{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

If a minimizer of J exists, y^k converges to the set of minimizers and $J(y^k)$ to the optimal value.

If $J \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$, then $y^{k+1} = \operatorname{prox}_{\tau J}(y^k)$ is characterized by

$$0 = \tau \nabla J(y^{k+1}) + (y^{k+1} - y^k) \;\Rightarrow\; y^{k+1} = y^k - \tau \nabla J(y^{k+1})$$

This is the backward Euler discretization of the gradient descent of J.

Thus, for differentiable J, the proximal point algorithm is equivalent to the fully implicit gradient descent.

Note: For most image processing problems, this algorithm is not very practical, since $\operatorname{prox}_{\tau J}$ can't be evaluated efficiently.

The proximal point algorithm

Finding a minimizer of $J \in \Gamma_0$ and a fixed point of $\operatorname{prox}_{\tau J}$ are equivalent:

Proximal point algorithm

$$y^{k+1} = \operatorname{prox}_{\tau J}(y^k)$$

for a step size $\tau > 0$ and an initial value value $y^0 \in \mathbb{R}^n$.

If a minimizer of J exists, y^k converges to the set of minimizers and $J(y^k)$ to the optimal value.

If $J \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$, then $y^{k+1} = \operatorname{prox}_{\tau J}(y^k)$ is characterized by

$$0 = \tau \nabla J(y^{k+1}) + (y^{k+1} - y^k) \;\Rightarrow\; y^{k+1} = y^k - \tau \nabla J(y^{k+1})$$

This is the backward Euler discretization of the gradient descent of J.

Thus, for differentiable J, the proximal point algorithm is equivalent to the fully implicit gradient descent.

Note: For most image processing problems, this algorithm is not very practical, since $\operatorname{prox}_{\tau J}$ can't be evaluated efficiently.

$\Rightarrow$ Operator Splitting.

The proximal gradient algorithm

For $J = G + H$, we consider the optimization problem

$$\min_{y \in \mathbb{R}^n} (G(y) + H(y)),$$

where $G \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$ and $H \in \Gamma_0(\mathbb{R}^n)$.

The proximal gradient algorithm

For $J = G + H$, we consider the optimization problem

$$\min_{y \in \mathbb{R}^n} (G(y) + H(y)),$$

where $G \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$ and $H \in \Gamma_0(\mathbb{R}^n)$.

Proximal gradient algorithm

$$y^{k+1} = \operatorname{prox}_{\tau_k H}\left(y^k - \tau_k \nabla G(y^k)\right)$$

for step sizes $\tau_k > 0$ and an initial value $y^0 \in \mathbb{R}^n$.

The proximal gradient algorithm

For $J = G + H$, we consider the optimization problem

$$\min_{y \in \mathbb{R}^n} (G(y) + H(y)),$$

where $G \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$ and $H \in \Gamma_0(\mathbb{R}^n)$.

Proximal gradient algorithm

$$y^{k+1} = \operatorname{prox}_{\tau_k H}\left(y^k - \tau_k \nabla G(y^k)\right)$$

for step sizes $\tau_k > 0$ and an initial value $y^0 \in \mathbb{R}^n$. Using

$$\mathcal{F}_\tau(y) = \frac{1}{\tau}\left(y - \operatorname{prox}_{\tau H}\left(y - \tau \nabla G(y)\right)\right),$$

we get

$$y^{k+1} = y^k - \tau_k \mathcal{F}_{\tau_k}(y^k).$$

The proximal gradient algorithm

For $J = G + H$, we consider the optimization problem

$$\min_{y \in \mathbb{R}^n} (G(y) + H(y)),$$

where $G \in C^1(\mathbb{R}^n) \cap \Gamma_0(\mathbb{R}^n)$ and $H \in \Gamma_0(\mathbb{R}^n)$.

Proximal gradient algorithm

$$y^{k+1} = \operatorname{prox}_{\tau_k H}\left(y^k - \tau_k \nabla G(y^k)\right)$$

for step sizes $\tau_k > 0$ and an initial value $y^0 \in \mathbb{R}^n$. Using

$$\mathcal{F}_\tau(y) = \frac{1}{\tau}\left(y - \operatorname{prox}_{\tau H}\left(y - \tau \nabla G(y)\right)\right),$$

we get

$$y^{k+1} = y^k - \tau_k \mathcal{F}_{\tau_k}(y^k).$$

This is also called *forward-backward splitting*, since it combines

- a forward Euler gradient descent step in G with
- a proximal point algorithm step in H
  (equivalent to a backward Euler gradient descent step in H).

The proximal gradient algorithm - Properties

Assumptions as before and ∇G Lipschitz continuous with constant $L > 0$.

The proximal gradient algorithm - Properties

Assumptions as before and ∇G Lipschitz continuous with constant $L > 0$.

Lemma For all $y, z \in \mathbb{R}^n$ and $\tau \in [0, \frac{1}{L}]$, it holds that

$$J(y - \tau \mathcal{F}_\tau(y)) \leq J(z) + \mathcal{F}_\tau(y) \cdot (y - z) - \frac{\tau}{2} \|\mathcal{F}_\tau(y)\|_2^2 .$$

The proximal gradient algorithm - Properties

Assumptions as before and ∇G Lipschitz continuous with constant $L > 0$.

Lemma For all $y, z \in \mathbb{R}^n$ and $\tau \in [0, \frac{1}{L}]$, it holds that

$$J(y - \tau \mathcal{F}_\tau(y)) \leq J(z) + \mathcal{F}_\tau(y) \cdot (y - z) - \frac{\tau}{2} \|\mathcal{F}_\tau(y)\|_2^2 .$$

Theorem Let $\tau_k \in [\tau_{\min}, \frac{1}{L}]$, where $\tau_{\min} \in (0, \frac{1}{L}]$, and let $\operatorname*{argmin}_{x \in X} J \neq \emptyset$. Then, the proximal gradient algorithm converges. More precisely,

$$0 \leq J(y^k) - J(y^*) \leq \frac{1}{2k\tau_{\min}} \left\|y^0 - y^*\right\|_2^2 = O\left(\tfrac{1}{k}\right)$$

for $y^* \in \operatorname*{argmin}_{x \in X} J$. Moreover, $(y^k)_k$ converges to the set of minimizers, i.e.

$$\lim_{k \to \infty} \operatorname{dist}(y^k, \operatorname{argmin} J) = 0.$$

The proximal gradient algorithm - Special cases

The above also proves convergence of other methods.

The proximal gradient algorithm - Special cases

The above also proves convergence of other methods.

- $G = 0$, $H = J \Rightarrow$ proximal point algorithm
  Since $\nabla 0$ is Lipschitz continuous with constant 0, it follows the convergence for arbitrary, bounded step sizes.

The proximal gradient algorithm - Special cases

The above also proves convergence of other methods.

- $G = 0$, $H = J \Rightarrow$ proximal point algorithm
  Since $\nabla 0$ is Lipschitz continuous with constant 0, it follows the convergence for arbitrary, bounded step sizes.
- $G = J$, $H = 0 \Rightarrow$ fully explicit gradient descent
  If ∇J is Lipschitz continuous, we get convergence for suitable τ_n.

The proximal gradient algorithm - Special cases

The above also proves convergence of other methods.

- $G = 0$, $H = J \Rightarrow$ proximal point algorithm
  Since $\nabla 0$ is Lipschitz continuous with constant 0, it follows the convergence for arbitrary, bounded step sizes.
- $G = J$, $H = 0 \Rightarrow$ fully explicit gradient descent
  If ∇J is Lipschitz continuous, we get convergence for suitable τ_n.
- $C \subset \mathbb{R}^n$ nonempty, convex and closed, $G = J$, $H = I_C$
  $\Rightarrow$ *projected gradient descent*, which minimizes $J(y)$ under the constraint $y \in C$. If ∇J is Lipschitz continuous, we get convergence for suitable τ_n.

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem
- Key for the mathematical analysis is to interpret the forward model as weighted cross-correlation

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem
- Key for the mathematical analysis is to interpret the forward model as weighted cross-correlation
- Total deep variation can the used as data driven regularizer

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem
- Key for the mathematical analysis is to interpret the forward model as weighted cross-correlation
- Total deep variation can the used as data driven regularizer
- Unfolding can be used to convert iterative algorithms to neural nets

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem
- Key for the mathematical analysis is to interpret the forward model as weighted cross-correlation
- Total deep variation can the used as data driven regularizer
- Unfolding can be used to convert iterative algorithms to neural nets
- Unfolding can add data-driven learning to model-based approaches

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem
- Key for the mathematical analysis is to interpret the forward model as weighted cross-correlation
- Total deep variation can the used as data driven regularizer
- Unfolding can be used to convert iterative algorithms to neural nets
- Unfolding can add data-driven learning to model-based approaches
- Proximal gradient algorithm suitable for sparsity promoting regularizers and unfolding

Summary

- Exit wave reconstruction (EWR) is a non-linear inverse problem
- Key for the mathematical analysis is to interpret the forward model as weighted cross-correlation
- Total deep variation can the used as data driven regularizer
- Unfolding can be used to convert iterative algorithms to neural nets
- Unfolding can add data-driven learning to model-based approaches
- Proximal gradient algorithm suitable for sparsity promoting regularizers and unfolding

Thank you! Questions?

Contact: berkels@aices.rwth-aachen.de