TP2

- Master MVA ENS-Paris Saclay
- Balthazar Neveu
- balthazarneveu@gmail.com
- Web Version | Github

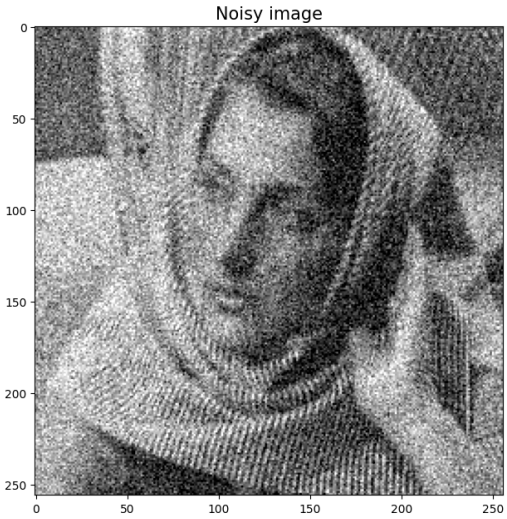
> Please note that the whole experiments are done on a single image...
> Conclusions would require more time

> Bonus part: not done in time

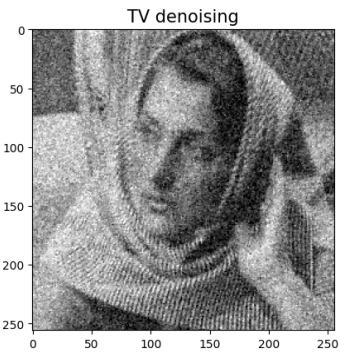
Default denoisers

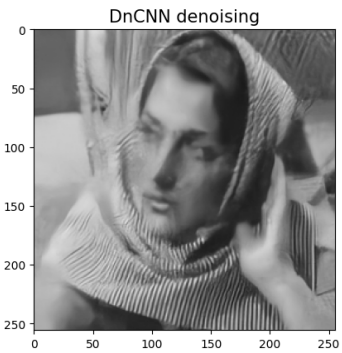
Evaluated on observation with additive white gaussian noise. $sigma_{255} = 40$

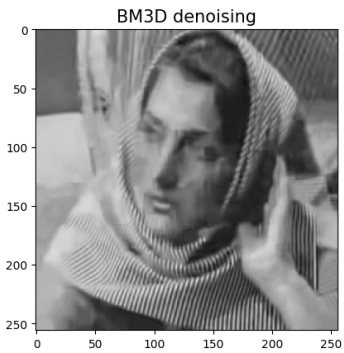
u = Original / $\tilde{u}$ = Noisy



Denoised with standard, non tuned denoisers



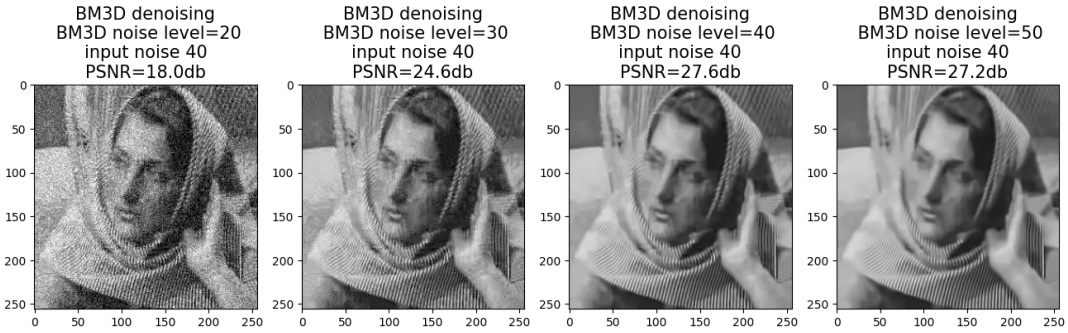




- TV-Denoising has not been tuned (and it can do better as we'll see later).

- DN-CNN trained with a noise level $sigma_{255} = 40$ is used here - so the denoising network is expected to work : there's no gap between its training conditions and evaluation conditions.
- BM3D is used with the right noise input level (`BM3DDenoiser(40)`), it is a non machine learning based algorithm.

BM3D with different tunings.



On the left, BM3D would expect a not too noisy image and so leave a lot of residual noise. On the right side, BM3D runs on an image where that was more noisy that reality so it ends up oversmoothing

Optimal TV Denoiser

Auto tuning TV denoiser

Question 1

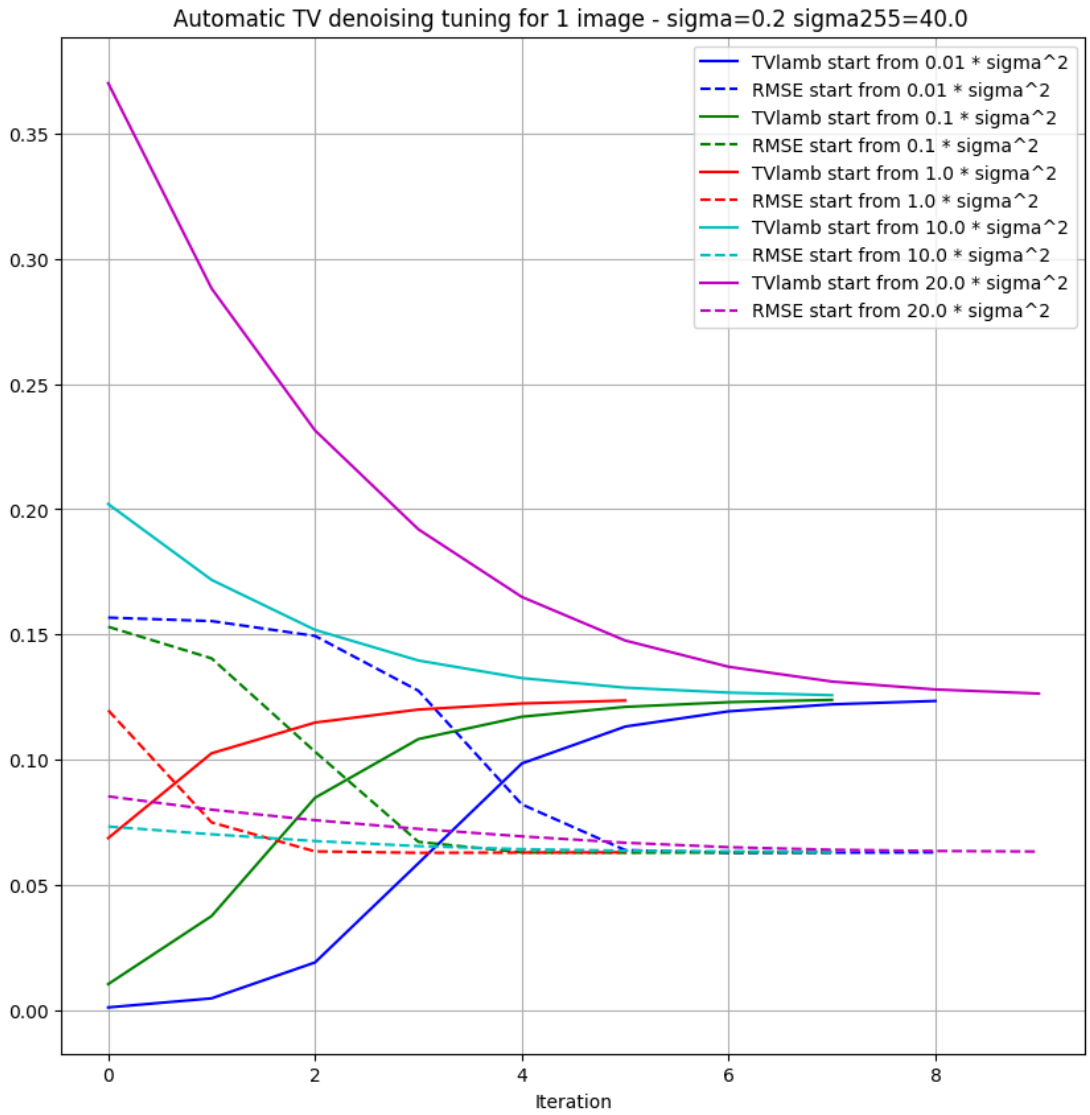
```
for i in range(1,100):
    uTV = TVDenoiser(lamb=TVlamb,niter=100).denoise(utilde)
    res = rmse(uTV,utilde)
    # Estimated residual between the noisy image and the denoised
image
    error = rmse(uTV,u)
    # True error between the prediction and the clean image
    non_gaussianity_error = sigma1*beta-res
    # if the error was gaussian with the right amount of noise,
this should be 0
    correction_factor = rho*non_gaussianity_error
    TVlamb *= np.exp(correction_factor)
    stop_cond= np.fabs(non_gaussianity_error) < 0.01 *
(sigma1*beta)
    if stop_cond:
        break
```

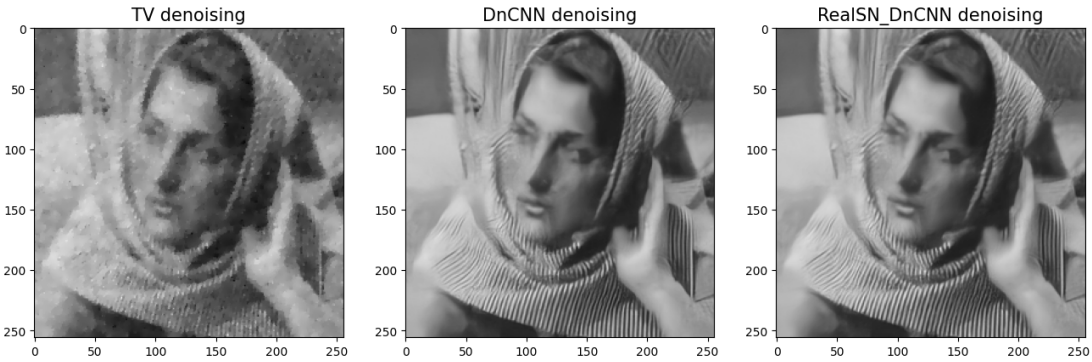
Question 2

Initial tuning : $\lambda_{\mathrm{n=0}}^{TV} = 0.024 = \sigma^2$

$\rightarrow \lambda_{n=6}^{TV} \approx 0.125 \approx 5.1\sigma^2$ to achieve a 1% tolerance (`residual/sigma = 0.955`)

Although simple, the proposed automatic tuning technique seems to be quite robust to initial values. Even if you start with a high value for λ like $10\sigma^2$ where the initial images are oversmoothed, the tuning parameter correction converges to the right value.





Denoiser	RMSE(normalized gray levels)	PSNR(dB)
Tuned TV Denoiser	0.0630	24.0 dB
DnCNN Denoiser	0.0450	26.9 dB
Real SN_DnCNN	0.0448	27.0 dB

`Real_SN_DnCNN` and `DnCNN` denoising have similar denoising quality, and way better than the correctly tuned TV denoiser

Question 3

`RealSN_DnCNN` has the same exact architecture as `DnCNN` but has been trained in a very specific fashion: The residual satisfies the Lipschitz constraint. This will not change anything to the denoising capabilities of the network under AWGN (this is what we observed in the previous table in question 2., additionnally the 2 denoised results look very much alike visually).

The only difference is part of the training: compared to a "classical" MSE minimization with gradient descent the weights are modified during training so that the residual satisfies the Lipschitz constraint (Relu is 1-Lipschitz, Convolutions coefficients at each layer need to be normalized by their largest eigen value). RealSN stands for real Spectral Normalization and uses a "tricky" implementation (not naïvely performing SVD at each step, instead relying on an iterative power method = avoids computing SVD at every training step for all netwrok layers).

Question 4

$\alpha = \frac{\sigma^2}{\gamma}$

In [ ]:
```python
def prox_datafit_gaussian_denoising(x: np.ndarray, y: np.ndarray, alpha:
    """
    Proximal Operator for Gaussian denoising:

    f(x) = || x - y ||^2 / (2 s^2)

    prox_{alpha f} (x) = (x + y*alpha/s^2)/(1+alpha/s^2)

    Parameters:
        :x - the argument to the proximal operator.
        :y - the noisy observation (flattened).
        :opts - the kwargs for hyperparameters.
            :alpha - the value of alpha.
            :s - the standard deviation of the gaussian noise in y.
    """
    a = alpha/(s**2)
    v = (x+y*a)/(1+a)
    return v
```

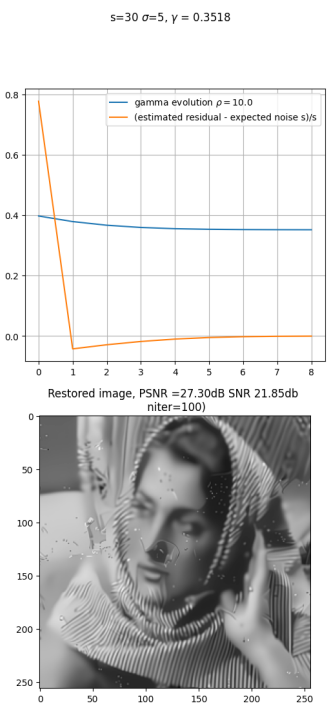
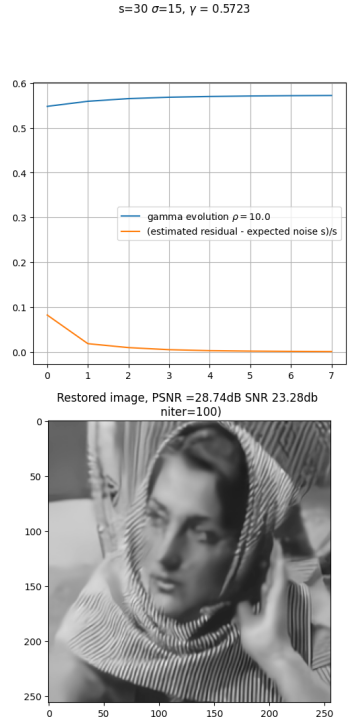
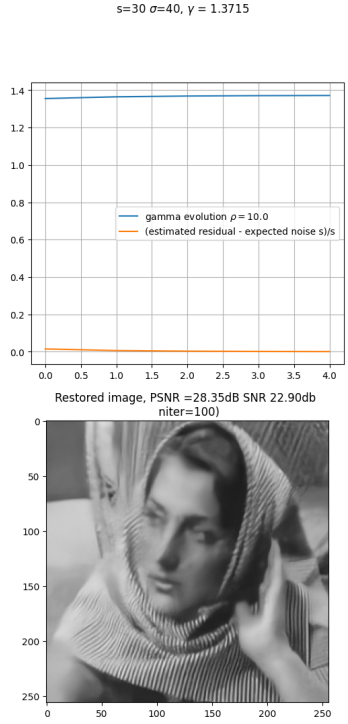
In the case where $s = \sigma$, the proximal term becomes $\text{prox}_{\alpha.F(x)} = \frac{\gamma x + y}{\gamma + 1}$ wich is a weighted sum of x and y (we're blending the denoised result from the Denoiser). When $\gamma = 1$, this is just the average $\frac{x+y}{2}$.

The γ parameter definitely acts as a tuning parameter for the denoiser as can be seen in the next figure. Good news is that compared to the "classic trick" (telling the network to process the image with a tweaked noise value), this tuning parameter actually seems to make sense.

$\gamma = 0.8$	$\gamma = 1.0$	$\gamma = 1.2$
		

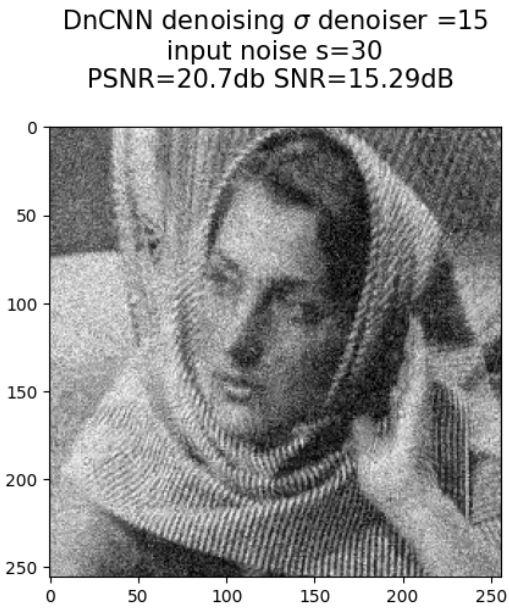
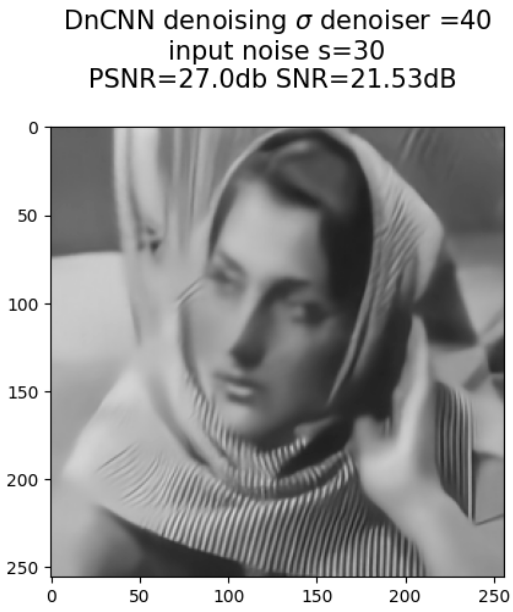
Auto tuning of PnP Gaussian denoising with $s \neq \sigma$

Results using RealSN DnCNN

$\sigma = 5$	$\sigma = 15$	$\sigma = 40$
		

Observations:

- Regarding the image on the left side of the table containing a bunch of weird artifacts, this translates with the impossibility of convergence we'll shown in question 5.
- The results when using PnP DRS for Gaussian Denoising with DnCNN at $\sigma = 15$ and $\sigma = 40$ are much better than if we'd simply performed inference with the DnCNN $\sigma = 15$ or $\sigma = 40$

no PnP $\sigma = 15$	**no PnP** $\sigma = 40$
	

Question 5

> Disclaimer: *I solved question 5 and wrote the answers without notticing the "Check convergence conditions" section so I basically had to find everything by myself which roughly took me half a day*.

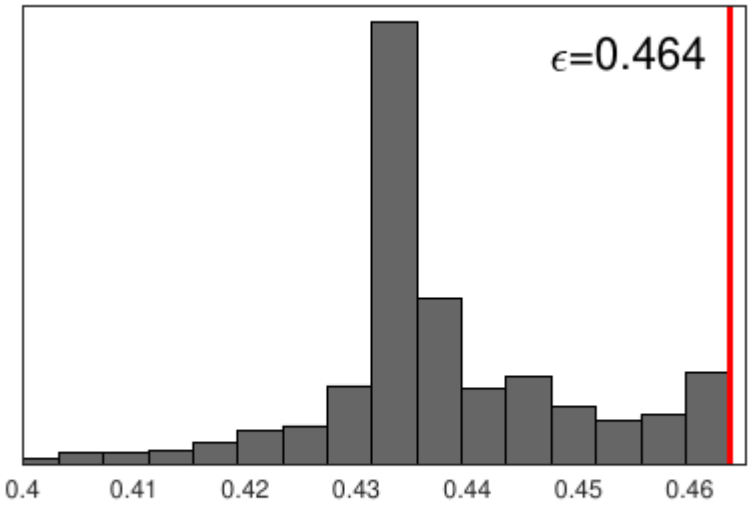
$\sigma = 5$	$\sigma = 15$	$\sigma = 40$
$\gamma = 0.351$	$\gamma = 0.5723$	$\gamma = 1.3715$
Condition $\gamma < 0.062$	Condition $\gamma < 0.55$	Condition $\gamma < 3.9$
SNR=21.85dB	SNR=23.28dB	SNR=22.90dB
PSNR=27.30dB	PSNR=28.74dB	PSNR=28.35dB

5.1 Convergence conditions

- Since $F(x) = \frac{\|x-y\|^2}{2s^2}$ is a quadaratic function, it is $\mu = \frac{1}{s^2}$ strictly -convex
- γ shall therefore satisfy the following conditions

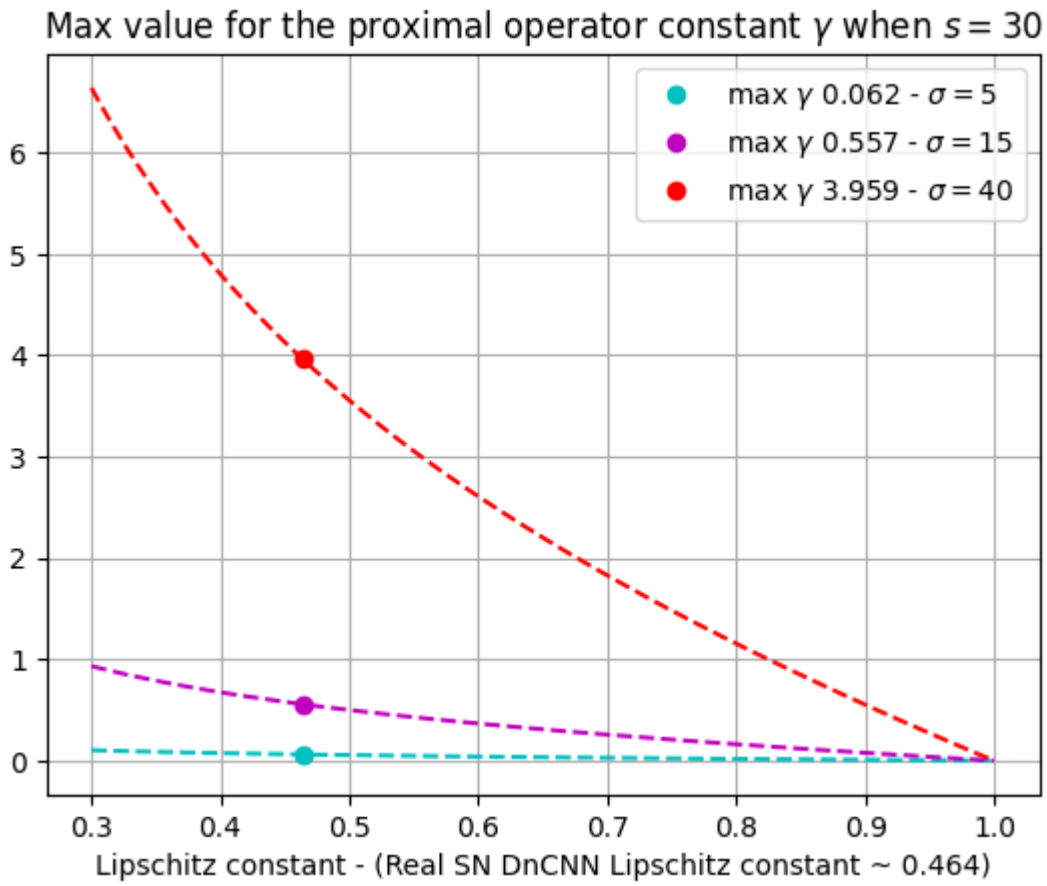
$$\gamma \leq \frac{\sigma^2}{s^2}.(\frac{1+L-2L^2}{L})$$

- We know that the SN_DnCNN has been trained with the right Lipschitz constraint, we have $0 < L < 1$ we find that the constant is approximately .
  - From the Ryu paper, `Plug-and-Play Methods Provably Converge with Properly Trained Denoisers`, we can get the estimations of the Lipschitz constant $L \approx 0.464$ for DnCNN.



(e) RealSN-DnCNN

- We can therefore deduce the maximum value of Gamma for each noise level of the pretrained denoisers.



- It's clear that the optimum $\gamma = 0.35 > 0.062$ found when $\sigma = 5$ violates the constraint, which probably explain why we observe a bunch of artifacts.
- At $\sigma = 15$, the optimum $\gamma = 0.57 \approx 0.557$ is close to the limit conditions but works correctly.

5.2 Convergence conditions assessment

Convergence is theoretically guaranteed in all cases for `Real SN DnCNN` as long as the constraint shown above is respected on γ. In practice when γ is small, you almost remove the regularization term so not much will happen. But empirically, the iterative search to get the best regularization parameter γ at $\sigma = 5$ ends up with a non satisfied condition.

- PnP **BM3D is not guaranteed converged as the Lipschitz** constant is $L > 1$

5.3 Best results at $\sigma = 15$

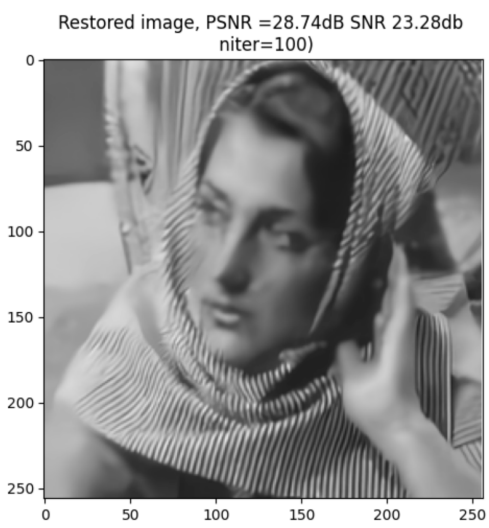
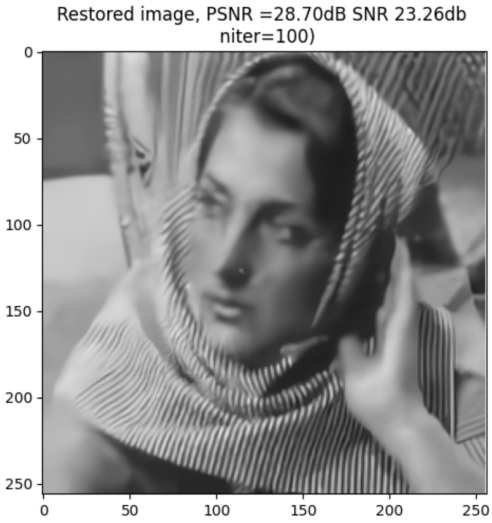
Best quantitative results are obtained for $\sigma = 15$ (a denoiser trained for noises lower than $s = 30$). It's also confirmed in terms of quality as textures are also better preserved. The explanation is not truly straightforward but first of all the proximal operator for regularization is just approximated by the denoiser. Let's get an intuition of what

- In the case of $\sigma = 15$, the first denoiser iteration will end up with an image which has a lot of residual noise (but less than the original image)... and the mechanism will go on, noise will be progressively removed.
- In the case of $\sigma = 40$, at the first iteration the denoiser will oversmooth textures (denoise too much). Then data term will add back a bit of the original noise but you'll then fall back into the same trap at next iteration (the signal is not noisy enough to what the denoiser is expecting - therefore the denoiser removes texture and content considered as noise rather than true signal).

If we recall what was said about Tweedie's formula, everytime we're using the MMSE denoiser and remove a bit of the residual noise, we're trying to project the noisy image into an improved version according to a smoothed version of the posterior distribution: Basically, by using the denoiser, we increase the probability of the denoised image of belonging to a smoothed version of the posterior image distribution. When σ is high, the distribution may actually be too "blurry" therefore potentially limiting the results. Having too small of a σ may result in preventing correct convergence as the posterior distribution may be peaky and multimodal.

Question 6

- When using DnCNN (without the Lipschitz constraint at training time), there are more visible local artifacts but the overall trend is similar to what we observed in question 5 for the `Real SN DnCNN`. See that bright spot on the nose for instance. This was discussed in question 3 already but here we see the limitations visually when the Lipschitz constraints where not taken into account at training time.
- To be honest, **this is not too critical** but we have to keep in mind that when we look at Ryu's estimation of the Lipschitz constant of the denoiser residual, the DnCNN L ($epsilon$ in the paper) was close to the one trained with the spectral normalization trick.

PnP DRS , denoiser = Real SN-DnCNN $\sigma = 15$	**PnP DRS, denoiser DnCNN** $\sigma = 15$
	

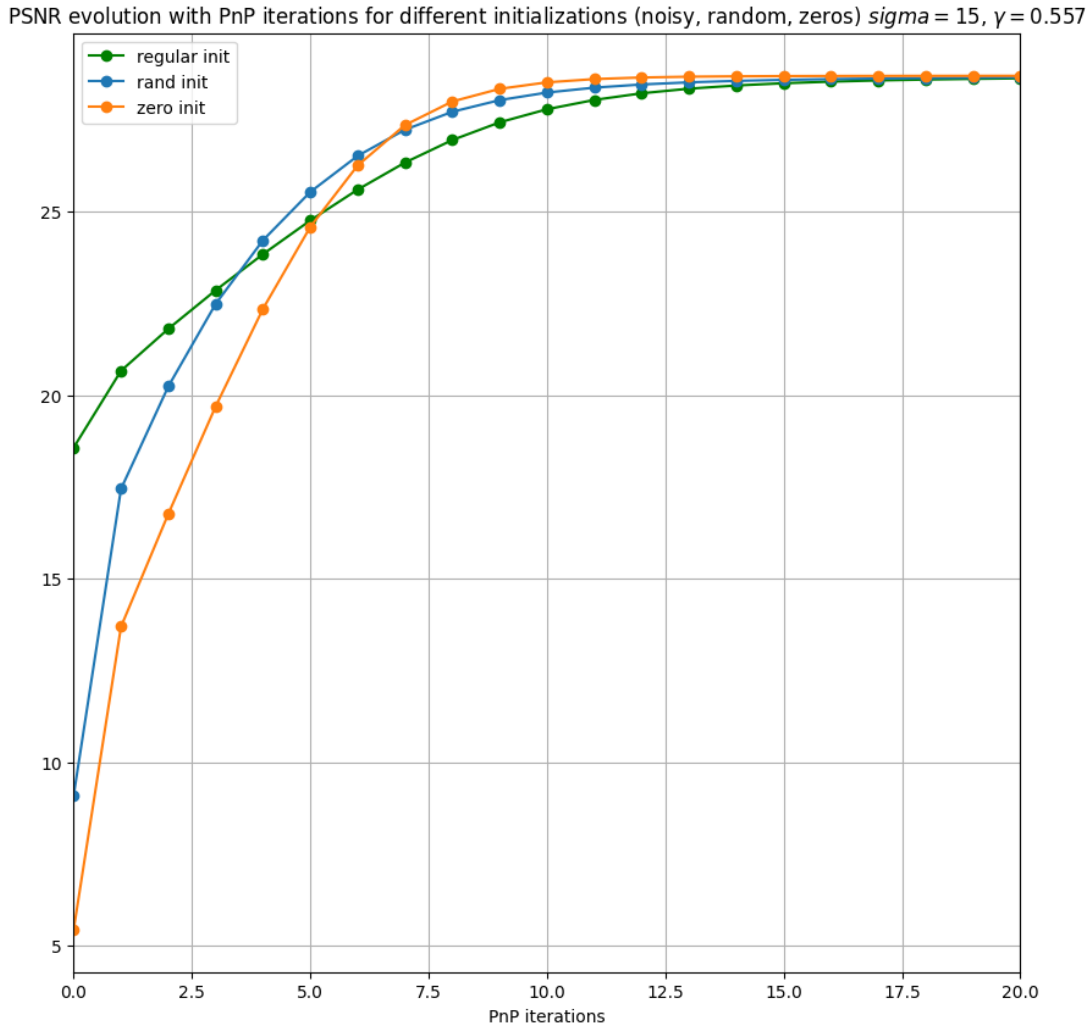
PnP BM3D	**PnP DnCNN**	**PnP Real SN DnCNN**
$\sigma = 40, N = 20, \gamma = 1.374$	$\sigma = 40, N = 20, \gamma = 1.379$	$\sigma = 40, N = 20, \gamma = 1.371$
PSNR=29.1dB	PSNR=28.32dB	PSNR=28.35dB

Best quantitative quality (PSNR) is achieved with PnP BM3D although the convergence condition was not even granted. Qualitatively, PnP BM3D leads to more artifacts (see the unexpected patterns on the cheek for instance). *We have to keep in mind that PSNR is not a perfect measurement of image quality*. I can find no clear and honnest explanation of the mismatch between theory stating there's no convergence for PnP BM3D and empirical results. It make come from the fact that BM3D roughly denoises in all conditions (whereas neural networks trained on a single noise value behave kind of badly as soon as they're out of their training distribution).

$s = 30$ **PnP DRS , denoiser = BM3D** $\sigma = 40$ **, DnCNN** $\sigma = 40$ **, Real SN-DnCNN** $\sigma = 40$

Question 7: (in)sensitivity to initialization

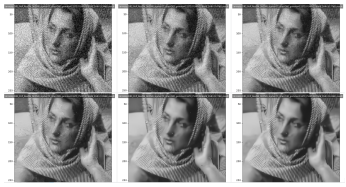
The PnP algorithm is **almost unsensitive to initialization conditions**. We get similar results no matter initializing with zeros, random uniform noise (between 0 and 1). This is a remarkable property and the convergence of DRS was proved no matter the initialization.



Sensitivity to initialization: Normal vs Zeros	**Crazy vs Random noise**
Top row : "normal" init with noisy image	Top row : "crazy" noise between -100 and 100,
	
Bottom row: init with zeros	Bottom row: uniform noise between 0 and 1
Left to right $\sigma = 5, 15, 40$	Left to right $\sigma = 5, 15, 40$

The $\sigma = 5$ case (with no guaranty of convergence) becomes more problematic with bad initialization.

Visualizing convergence visually:
$n_{\text{iterations PnP}} = [0, 3, 6, 9, 20, 100], \gamma = 0.57, \sigma = 15.$

Random uniform noise initialization	Regular initialization from noisy image	Initialization from zeros
		

Question 8: PnP ADMM ⇔ PnP DRS

PnP DRS (Douglas–Rachford splitting) vs ADMM (alternating directions method of multipliers)

PnP-DRS

1. $\mathbf{v}^k = \text{prox}_{F/\rho}(\overline{\mathbf{u}}^{k-1})$.
2. $\mathbf{x}^k = D_\sigma(2\mathbf{v}^k - \overline{\mathbf{u}}^{k-1})$.
3. $\overline{\mathbf{u}}^k = \overline{\mathbf{u}}^{k-1} + (\mathbf{x}^k - \mathbf{v}^k)$. multiplier

PnP ADMM algorithm:
Find $\mathbf{x}^* = \arg\min_{\mathbf{x}} \alpha F(\mathbf{x}) + \sigma^2 R(\mathbf{x})$

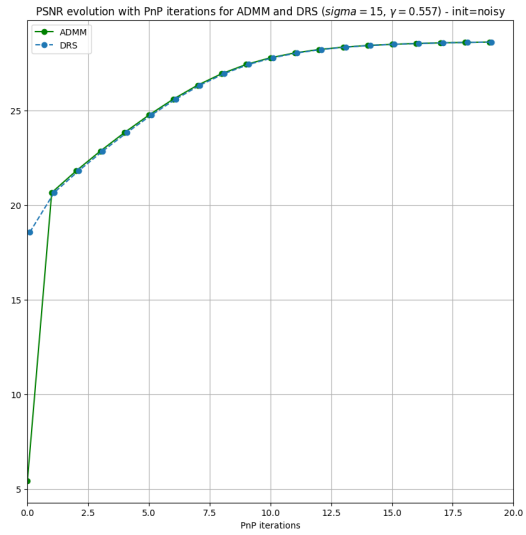
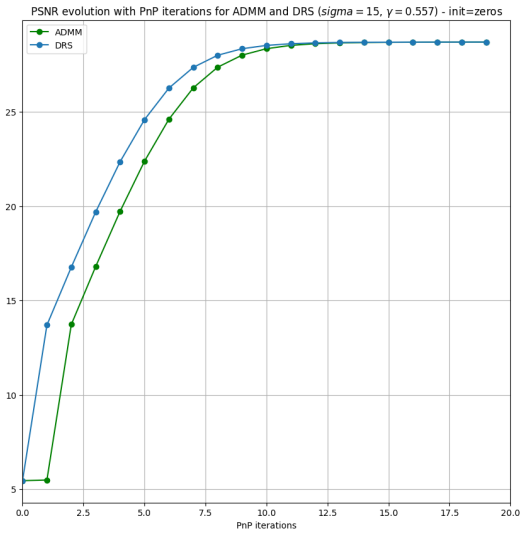
1. $\mathbf{x}^k = D_\sigma(\mathbf{v}^{k-1} - \overline{\mathbf{u}}^{k-1})$.
2. $\mathbf{v}^k = \text{prox}_{\alpha F}(\mathbf{x}^k + \overline{\mathbf{u}}^{k-1})$.
3. $\overline{\mathbf{u}}^k = \overline{\mathbf{u}}^{k-1} + (\mathbf{x}^k - \mathbf{v}^k)$.

Class slides

$$
\begin{aligned}
x^{k+1/2} &= \text{Prox}_{\alpha f}(z^k) \\
x^{k+1} &= H_\sigma(2x^{k+1/2} - z^k) \\
z^{k+1} &= z^k + x^{k+1} - x^{k+1/2}
\end{aligned}
\qquad
\begin{aligned}
x^{k+1} &= H_\sigma(y^k - u^k) \\
y^{k+1} &= \text{Prox}_{\alpha f}(x^{k+1} + u^k) \\
u^{k+1} &= u^k + x^{k+1} - y^{k+1}
\end{aligned}
$$

Ryu et al - Plug-and-Play Methods Provably Converge with Properly Trained Denoisers

---

Sanity check PnP ADMM vs DRS - noisy init	Sanity check PnP ADMM vs DRS - zero init
	

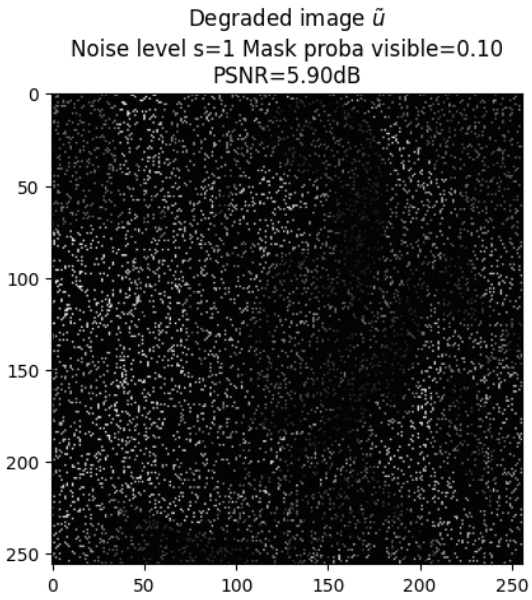
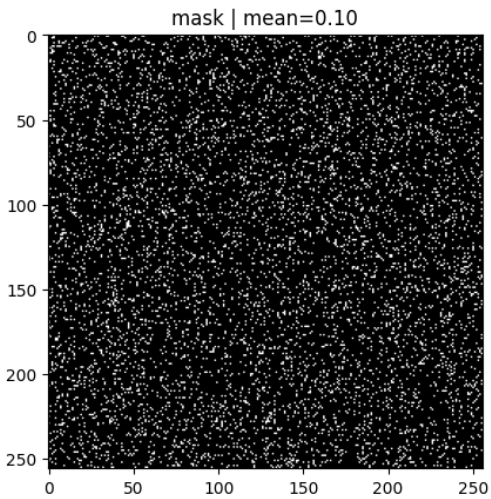
Convergence sanity check - slight difference at the first iteration (due to the order of operations). See the code for details.

> Proof of the equivalency between DRS and ADMM : **not found by myself.** - Tried for almost 2 hours and could not find the mechanic. Ryu et al Section 9.1 has the proof, this is smart.

Inpainting / Missing pixels

Question 9

Degradation

Degradation	Mask A
	

Potential function

Notations: Random variables defined at a given pixel location i

- Y_i noisy degraded observation with realization y_i.
- X_i unknown ideal gray level (groundtruth) with realization x_i - sampled from the prior image distribution.
- M_i mask with realization m_i: the mask is known, $\mathrm{p}(N_i = n_i) = 1$. This means we have access to an **oracle on the inpainting mask**.
- N_i additive white gaussian noise with realization n_i . $N_i \sim \mathcal{N}(0, s^2)$

$$Y_i = M_i X_i + N_i$$

For a given observed pixel y_i , $y_i = m_i x_i + n_i$ where $m_i \sim \mathrm{Ber}(p)$ is known and is additive white gaussian noise.

$\mathrm{p}_{Y_i|X_i}(Y_i = y_i|X_i = x_i) = \mathrm{p}_{Y_i|X_i}(Y_i = y_i|X_i = x_i, M_i = m_i) = \mathrm{p}_{N_i}(N_i = y_i - m_i$ since the noise distribution of N_i is Gaussian.

We assume that the noise is i.i.d so this formula stands for the whole image of size (H, W) considered as a vector $\backslash\mathrm{R}^{H.W}$ (where the noise covariance matrix is a diagonal full of σ^2 and the degradation mask matrix can also be expressed as a diagonal matrix A filled with 0 and 1).

Potential for a single pixel: $F(x_i, y_i) = \frac{1}{2s^2}(m_i . x_i - y_i)^2$ which can be re-written for the whole matrix using vector notations.

$F(x, y) = \frac{1}{2s^2}\|A . x - y\|^2$, note that the degradation operator A is known.

Proximal operator

Let's use a single pixel to avoid heavy pointless matrix computations here as all elements are independant.

$\mathrm{prox}_{\alpha F}(x) = \mathrm{argmin}_v \frac{1}{2}\|v - x\|^2 + \alpha F(v) = \mathrm{argmin}_v \frac{1}{2}(v - x)^2 + \alpha\frac{1}{2s^2}(m . v - y)$.

- If $m = 0$, $\mathrm{prox}_{\alpha F}(x) = x$ (*keep previously impainted value*)
- If $m = 1$, $\mathrm{prox}_{\alpha F}(x)$ can be found by zero-ing the derivate of function F. $\frac{df}{dx} = (v - x) + \frac{\alpha}{s^2}(v - y) = 0$, $v = \frac{s^2 x + \alpha y}{s^2 + \alpha}$

$$\mathrm{prox}_{\alpha F}(x) = \frac{s^2 x + \alpha y}{s^2 + \alpha}$$

(*add back a bit of the noisy version where pixels are not masked*)

Finally, the closed form can be written with a single liner (thus vectorized, no need for if conditions).

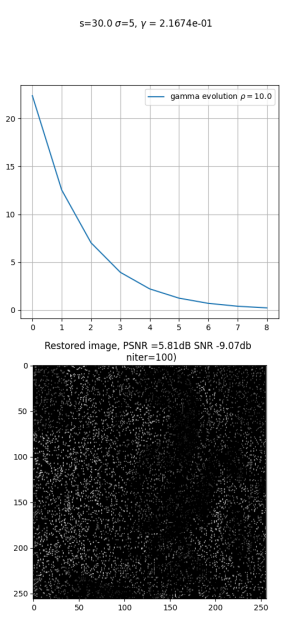
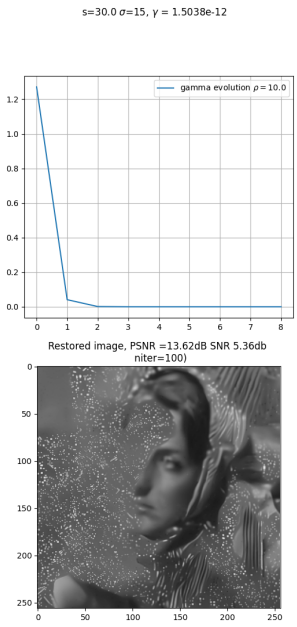
$$\mathrm{prox}_{\alpha F}(x) = m * \frac{s^2 x + \alpha y}{s^2 + \alpha} + (1 - m) * x$$

Results analyzis (inpainting)

Using the DRS algorithm

$\sigma = 5$		$\sigma = 15$	$\sigma = 40$
$\gamma = 0.2167$	$\gamma = 1.5e - 12$		$\gamma = 8.10^{-19}$
PSNR=5.81dB		PSNR=13.62dB	PSNR 24.09dB
SNR=-9.07dB		PSNR = 5.36dB	SNR = 18.65dB

Results are not good unless using the RealSN DnCNN with $\sigma = 40$. Problem looks difficult though to the human eye, results are still impressive.

$\sigma = 5$	$\sigma = 15$	$\sigma = 40$
		

Appendix

- Ryu et al - Plug-and-Play Methods Provably Converge with Properly Trained Denoisers