COMP0114 Inverse Problems in Imaging. MiniProject - Matlab Hints

1. Calculate the Radon transform of an image and test the back-projection method

The file `SLphan.mat` contains the image. You just need to load this with `load SLphan`.

If $\boldsymbol{f}$ is a 2D image, then the following code finds its Radon transform in 1-degree intervals from $0 - 179$.

```
angles = [0:179];
g = radon(f,angles);
```

The following call implements (Unfiltered) Backprojection

```
bg = iradon(g,angles,'linear','none');
```

Since the size of `bg` is not equal to your input $\boldsymbol{f}_{\mathsf{true}}$, we need to use the extra parameter 'OUTPUT_SIZE' to control the size of the back-projected image. (You need to set 'FREQUENCY_SCALING'=1) Thus

```
n = size(f,1);
bg = iradon(g,angles,'linear','none',1,n);
```

will give an image the same size as $\boldsymbol{f}_{\mathsf{true}}$.

The simple call

```
ig = iradon(g,angles);
```

computes the filtered-backprojection.

2. Calculate an explicit matrix form of the Radon transform and investigate its SVD

Let's assume we have an image $\boldsymbol{f}$ of dimensions 64×64 and we want to compute the Radon transform for the angle vector with 45 angles:

```
angles = [0:4:179];
```

The measurements g (sinogram) are of dimension 95×45. Where 45 is the "number of angles" and 95 is the "number of projection samples" or in other words the resolution of your measurement detector. In vectorized form $\boldsymbol{f}$ is then of dimension $64^2 = 4096$ and $\boldsymbol{g}$ will be of size $45 \cdot 95 = 4275$. Your matrix representation of the radon transform then needs to map the vectorized image to the vectorized measurement, that is $\mathsf{A} : \mathbb{R}^{4096} \to \mathbb{R}^{4275}$, in other words your matrix is of size 4275×4096 (rows $\times$ columns).

You can use '`svds`' to only compute the singular values.

3. Implement a matrix-free regularised least-squares solver for the Radon Transform

To determine the regularisation parameter you could use for instance the discrepancy principle from coursework 2. We note that in this case the parameter is typically larger than for the deconvolution.

```
y = iradon(radon(f,angles),angles,'linear','none',1,n);
```

is the forward-backward operator with input $\boldsymbol{f}$ and output $\boldsymbol{y}$. Take care as usual with reshaping images to 1D vectors and back, for the purpose of calling a builtin Krylov solver.

4. Write a Haar wavelet denoiser Wavelets are a so-called multi-scale decomposition of your image. For an image of size 128×128 you will get 7 scales, each scale will be have three "images" of gradually decreasing resolution. These images are the wavelet coefficients and represent the horizontal, vertical, and diagonal components.

You can compute the coefficients for your image $\boldsymbol{f}$ by calling

```
[a,h,v,d] = haart2(f);
```

You can reconsruct the image by calling

```
f_rec = ihaart2(a,h,v,d)
```

For denoising write a threshold function, for instance as the following:

```
range = [1:maxRange];
tVal  = 1;
[hT,vT,dT] = thresholdFunction(h,v,d,range,tVal,otherParameter);
```

In the thresholding function you can make use of MATLAB's `wthresh`. Then a denoised image is reconstructed by

```
f_denoise = ihaart2(a,hT,vT,dT)
```

In your thresholding function the `range` denotes the scales you want to threshold. Setting

```
maxRange = 7;
range = [1:maxRange];
```

would mean that you loop through all scales and threshold on each scale. It might be a good idea to limit the range to the higher finer coefficients that mostly consist of the noise components. For instance `range` $= [1:4]$; would only threshold the first 4 scales.

How to choose a thresholding parameter `tVal`: The parameter should be chosen such that it thresholds the noise components of your coefficients. This can be done either by visual examination of the coefficients or a more stable option would be to determine a value to threshold the lowest few (x) percent of parameters. To do that, collect all coefficients for all scales and directions into one large vector, sort them by absolute value and then determine the value for which x percent of the coefficients are smaller. This is your threshold value.

The `otherParameter` is not essential and was only added in case you want further personalise your thresholding function.

5. Iterative soft-thresholding for X-ray tomography To clarify the choice of step length λ. During the iterations you can keep the step length constant. To find a suitable choice of λ

start with a small value, like 10^{-4} and see if the iterations are stable, i.e. they are not heavily oscillating. To check this you can just plot the reconstruction in each iteration and monitor the progress. If you found a stable choice for λ increase it as long as you still have a stable algorithm, this will speed up convergence, since larger λ will lead to faster convergence.