SVD Analysis and TSVD Solutions

The purpose of this challenge is to illustrate the SVD analysis, which includes an inspection of the "basis matrices" and an inspection of the "Picard plot." We shall also compute TSVD solutions and study their behavior.

In a real deblurring problem we would, of course, never form the big coefficient matrix $\mathbf{A}$ explicitly; but we do it here for illustrative purposes. To keep computing times reasonably small, we must use small image dimensions; we suggest to use $m = n = 31$ leading to $N = 1024$.

1. Use your MATLAB function `OOFblur` to compute a PSF array $\mathbf{P}$ of size $n \times n$ for out-of-focus blur.

2. Create the $N \times N$ coefficient matrix $\mathbf{A}$ for zero boundary conditions, as defined in equation (4.5). You can use the MATLAB code shown on the next page.

3. Compute the SVD using MATLAB's `svd` function, and plot the singular values.

4. For selected values of the index i (some small, some medium and some large), construct the matrices/arrays $\mathbf{V}^{[i]}$ (use MATLAB's `reshape` function) and show them as images. You should see that the spatial frequencies in $\mathbf{V}^{[i]}$ tend to increase with i.

5. Now choose an $n \times n$ image $\mathbf{X}$ (e.g., a part of a test image) and compute $\mathbf{x} = \text{vec}(\mathbf{X})$ (use `x = X(:)` or `reshape`). Then compute the vector-version of the blurred image as $\mathbf{b} = \mathbf{A}\,\mathbf{x}$, and add Gaussian noise with a low noise level.

6. Plot the singular values σ_i together with the quantities $|\mathbf{u}_i^T\mathbf{b}|$ and $|\mathbf{u}_i^T\mathbf{b}|/\sigma_i$ in semilog-scale, and check if they behave according to the general description in the book (e.g., the coefficients $|\mathbf{u}_i^T\mathbf{b}|$ should first decay and then level off at the noise level).

7. Finally compute some TSVD solutions and show them as images. You should (hopefully) see that the optimal truncation parameter k corresponds roughly to the index i where the coefficients $|\mathbf{u}_i^T\mathbf{b}|$ start to level off.

8. Repeat 5–7 with a larger noise level, and check that the optimal k is now smaller than before.

MATLAB code for constructing the coefficient matrix $\mathbf{A}$ from the PSF array $\mathbf{P}$, according to equation (4.5) which assumes zero boundary conditions. The code is available in the file `P2Acode`, and it works only for odd values of `n`.

```
% Note: n must be odd.
B = zeros(n,n,n,2);
c = zeros(n,1);
r = zeros(n,1);
q = ceil(n/2);
for k=1:q
    c(1:q) = P(q:n,q+k-1);
    r(1:q) = P(q:-1:1,q+k-1);
    B(:,:,k,1) = toeplitz(c,r);
    c(1:q) = P(q:n,q+1-k);
    r(1:q) = P(q:-1:1,q+1-k);
    B(:,:,k,2) = toeplitz(c,r);
end

A = zeros(n^2,n^2);
for i=1:n
    A( (i-1)*n+1:i*n , (i-1)*n+1:i*n ) = B(:,:,1,1);
    for j=1:i-1
        A( (i-1)*n+1:i*n , (j-1)*n+1:j*n ) = B(:,:,i-j+1,1);
        A( (j-1)*n+1:j*n , (i-1)*n+1:i*n ) = B(:,:,i-j+1,2);
    end
end
```

Boundary Conditions and the SVD "Basis Images"

The "basis images" $\mathbf{V}^{[i]}$ computed above should all approach zero at the boundaries, because we assumed zero boundary conditions. In this challenge we shall study the "basis images" for the same problem but with periodic boundary conditions.

In the formulation of the book, we must now modify the matrix $\mathbf{A}$ to incorporate the nonzero boundary conditions, such that it has the form shown in equation (4.8).

Modify the MATLAB code above such that this is achieved. Then compute the SVD, inspect the "basis images" and check that they are periodic in the vertical and horizontal directions.