E9 222 Signal Processing in Practice

Linear and Circular Convolution

Instructor: Chandra Sekhar Seelamantula

Due: 14 January 2026

Learning Objectives

- Implement 1D and 2D linear and circular convolution
- Implement the inverse filter and Wiener filter for deconvolution

Problem Statement

This assignment covers the fundamental operations of convolution for 1D and 2D. You will implement convolution from scratch to understand boundary conditions, then apply linear and time-invariant techniques to restore signals degraded by reverberation (1D) and optical blur (2D).

1 Part 1: Discrete Convolution Implementation

The discrete convolution of a signal $x[n]$ of length N and a filter $h[n]$ of length M is defined as:

$$y[n] = (x * h)[n] = \sum_{k \in \mathbb{Z}} x[k]h[n-k]$$

Tasks:

1. **Manual Implementation:** Write a function `manual_convolve(x, h, mode)` using nested loops. Do not use `numpy.convolve` or `scipy.signal.convolve`.
2. **Mode Support:** Implement the following boundary modes:
   - '`full`': Output size $N + M - 1$. Standard linear convolution.
   - '`same`': Output size N. Central crop of the full convolution.
   - '`valid`': Output size $N - M + 1$. Returns only parts where signals fully overlap.

2 Part 2: 1D Deconvolution (Audio De-reverberation)

In audio processing, the multipath propagation of sound is termed reverberation. The goal is to recover the clean speech signal $x[n]$ from a recorded reverberated signal $y[n]$ modelled as:

$$y[n] = x[n] * h[n] + \eta[n]$$

where h is the Room Impulse Response (RIR) and η is background noise.

Tasks:

1. **Restoration:** Implement the 1D Wiener deconvolution, given in the frequency domain:

$$\hat{X}(\omega) = \left[\frac{H^*(\omega)}{|H(\omega)|^2 + K} \right] Y(\omega)$$

2. **Comparison:** Compare the auditory and visual results (waveforms) of:
   - The Naive Inverse Filter (setting $K \approx 0$).
   - The Wiener Filter (with tuned K).

3 Part 3: 2D Image Degradation Model

The degradation process for images is modeled as:

$$g(x, y) = (h * f)(x, y) + \eta(x, y)$$

where $f(x, y)$ is the original image, $h(x, y)$ is the point-spread function (PSF), and $\eta(x, y)$ is additive Gaussian noise.

Tasks:

1. **Image Loading:** Load a grayscale image. Convert pixel intensities to floating-point values in the range $[0, 1]$.
2. **Kernel Generation:** Create a $k \times k$ motion blur kernel h. This can be approximated as a normalized diagonal line.
3. **Frequency-domain Convolution:** Compute the degraded image using the FFT:

$$g = \mathcal{F}^{-1}\{\mathcal{F}\{h\} \cdot \mathcal{F}\{f\}\}$$

   Note: Ensure h is padded to the dimensions of f prior to the FFT to prevent aliasing.

4. **Noise Addition:** Add Gaussian noise ($\mu = 0, \sigma = 0.01$) to the convolved output. Display the final degraded image $g(x, y)$.

4 Part 4: Inverse Filtering (2D)

In the absence of noise, convolution in the spatial domain corresponds to multiplication in the frequency domain: $G(u, v) = H(u, v)F(u, v)$. The inverse filter estimate is given by:

$$\hat{F}(u, v) = \frac{G(u, v)}{H(u, v)}$$

Tasks:

1. Implement the inverse filter on
   (a) the blurry image without noise
   (b) the noisy, degraded image
2. **Comparison:** Report the peak signal-to-noise ratio (PSNR) and structural similarity index (SSIM).

5 Part 5: Wiener Filtering (2D)

The Wiener filter minimizes the mean square error (MSE) between the estimated image and the original image. The filter response in the frequency domain is:

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + K} \right] G(u,v)$$

Tasks:

1. Implement the Wiener filter.
2. Evaluate the performance for different regularization constants K (e.g., $0.01, 0.001, 0.0001$).
3. **Comparison:** Visualize the Inverse Filter result alongside the Wiener Filter result. Report the PSNR and SSIM.

6 Part 6: Parameter Estimation

The objective is to perform blind deconvolution, i.e., deconvolution without the knowledge of the blur kernel. Consider a motion blurred version of `page.png`. From this point on, the true blur kernel is unknown.

1. Assume a Gaussian PSF model.
2. Manually tune the size, standard deviation σ of the Gaussian kernel and the Wiener constant K to maximize legibility.
3. Report the parameters yielding the optimal visual reconstruction.