Lecture 2 & 3: Introduction to Systems
Practice Exercises

Konstantinos Chatzilygeroudis
costashatz@upatras.gr

3 March 2026

1 Coding Exercises

Question 1.1

Let

$$x(t) = u(t) - u(t-2), \qquad h(t) = u(t) - u(t-1).$$

Approximate the continuous-time convolution

$$y(t) = (x * h)(t)$$

by sampling the signals on the interval $t \in [-1, 4]$ using 4000 equally spaced points.
Use `np.convolve` and the approximation

$$y(t) \approx \Delta t \cdot \texttt{convolve}(x, h),$$

where Δt is the sampling step. Continuous convolution is $y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\,d\tau$, which can be approximated by a Riemann sum when the signals are sampled with step Δt. Thus $y(t_n) \approx \sum_k x[k]h[n-k]\Delta t = \Delta t\,\mathrm{conv}(x, h)$, giving $y(t) \approx \Delta t\,\mathrm{np.convolve}(x, h)$.
Plot on the same figure:

$$x(t), \quad h(t), \quad y(t).$$

Then answer:

1. What is the support of $x(t)$ and $h(t)$?
2. Over what interval is the convolution nonzero?
3. What is the qualitative shape of $y(t)$?

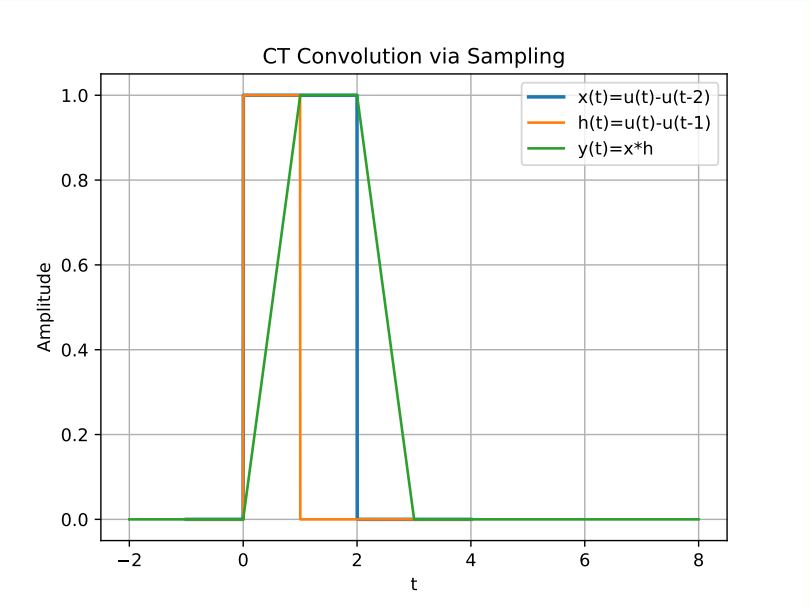
Solution (Code)

```python
import numpy as np
import matplotlib.pyplot as plt

t = np.linspace(-1, 4, 4000)
dt = t[1] - t[0]

u = lambda z: (z >= 0).astype(float)

x = u(t) - u(t - 2)
h = u(t) - u(t - 1)

y = dt * np.convolve(x, h, mode='full')
t_y = np.linspace(t[0] + t[0], t[-1] + t[-1], len(y))

plt.figure()
plt.plot(t, x, label='x(t)=u(t)-u(t-2)')
plt.plot(t, h, label='h(t)=u(t)-u(t-1)')
plt.plot(t_y, y, label='y(t)=x*h')
```

```
plt.xlabel('t')
plt.ylabel('Amplitude')
plt.title('CT Convolution via Sampling')
plt.grid(True)
plt.legend()
plt.show()
```

Solution

Plot:



Answers:

1. $x(t)$ is nonzero on $[0, 2]$ and $h(t)$ is nonzero on $[0, 1]$.
2. The convolution is nonzero on $[0, 3]$, since the support adds:

$$[0, 2] + [0, 1] = [0, 3].$$

3. $y(t)$ is trapezoidal: it increases linearly while overlap grows, stays constant while the shorter pulse is fully inside the longer one, and then decreases linearly as overlap shrinks.

Question 1.2

Let

$$x[n] = \{1, 2, 1\}, \qquad h[n] = \{1, -1\}.$$

Implement the discrete-time convolution manually, and also compute it by calling `np.convolve`. Plot both outputs on the same figure using stem plots and verify that they match exactly.
Then answer:

1. Why is comparing the manual result with `np.convolve` useful?
2. What does the sequence `h[n]={1,-1}` do to a slowly varying signal?
3. Is the output length what you expected from the length formula?

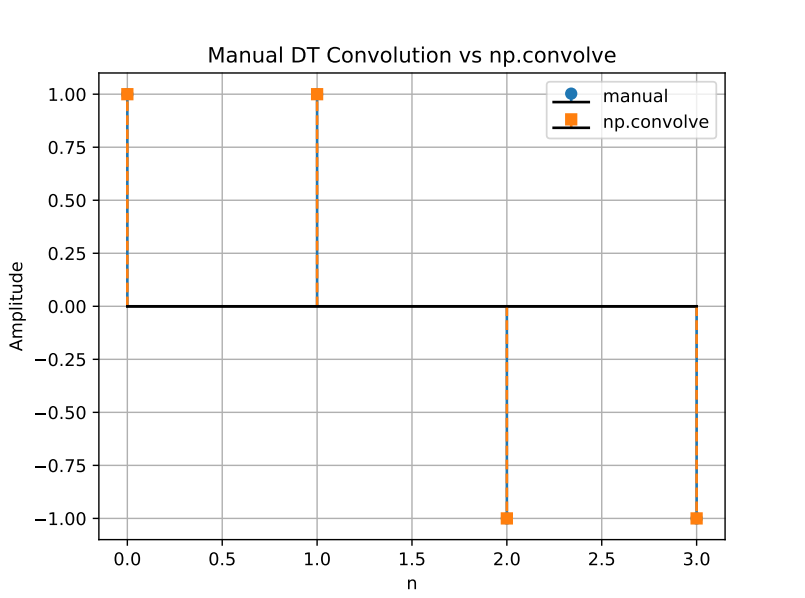
Solution (Code)

```
import numpy as np
import matplotlib.pyplot as plt

```

```python
x = np.array([1, 2, 1], dtype=float)
h = np.array([1, -1], dtype=float)

Ly = len(x) + len(h) - 1
y_manual = np.zeros(Ly)

for n in range(Ly):
    s = 0.0
    for k in range(len(x)):
        if 0 <= n - k < len(h):
            s += x[k] * h[n - k]
    y_manual[n] = s

y_np = np.convolve(x, h)

n = np.arange(Ly)

plt.figure()
plt.stem(n, y_manual, linefmt='C0-', markerfmt='C0o', basefmt='k-', label='manual')
plt.stem(n, y_np, linefmt='C1--', markerfmt='C1s', basefmt='k-', label='np.convolve')
plt.xlabel('n')
plt.ylabel('Amplitude')
plt.title('Manual DT Convolution vs np.convolve')
plt.grid(True)
plt.legend()
plt.show()
```

Solution

Plot:



Answers:

1. It checks that the manual implementation is correct and helps students trust both the formula and the code.

2. The filter $\{1, -1\}$ computes a first difference, so it emphasizes changes and reduces constant or slowly varying parts.

3. Yes. Since
$$L_x = 3, \qquad L_h = 2,$$
the output length should be
$$L_y = 3 + 2 - 1 = 4,$$

which matches the computed result.