Practical Examples 2

Dr. G. Omprakash

Assistant Professor, ECE, KLEF

Table of Contents

Random Process — Stationarity and Non-Stationarity

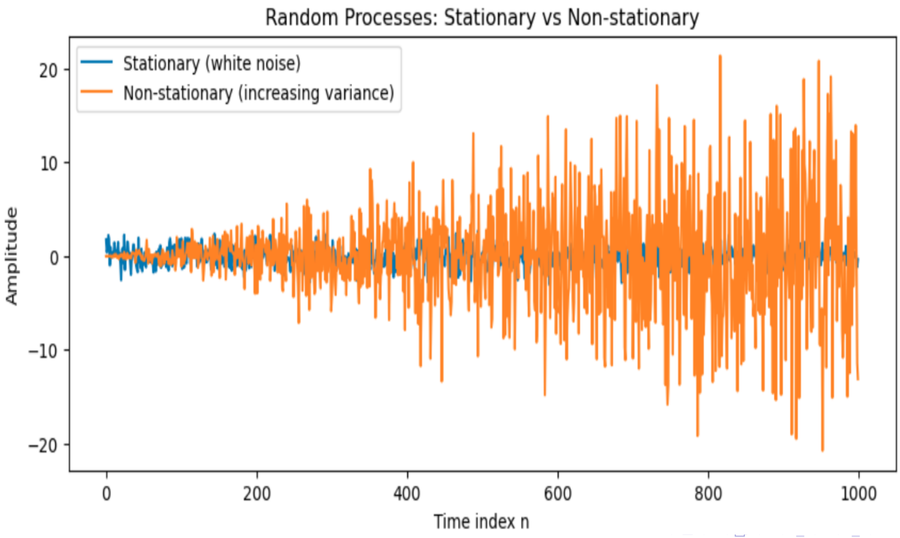
Random Process

- A process is **stationary** if its mean and variance do not change with time.
- **Observations:** $x[n]$ (blue) has constant mean ≈ 0 and variance ≈ 1 (stationary).
- $y[n]$ (orange) has increasing spread (variance increases over time) — non-stationary.

Random Process

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

np.random.seed(0)

# Generate two random processes
N = 1000
# (a) Stationary white noise (zero-mean, constant variance)
x = np.random.randn(N)

# (b) Non-stationary process: variance increases with time
t = np.arange(N)
y = (0.01 * t) * np.random.randn(N)

# Plot both processes
plt.figure(figsize=(10,4))
plt.plot(x, label='Stationary (white noise)')
plt.plot(y, label='Non-stationary (increasing variance)')
plt.title("Random Processes: Stationary vs Non-stationary")
plt.xlabel("Time index n"); plt.ylabel("Amplitude"); plt.legend(); plt.show()
```

Random Process



Measuring Dependence: Correlation and Covariance

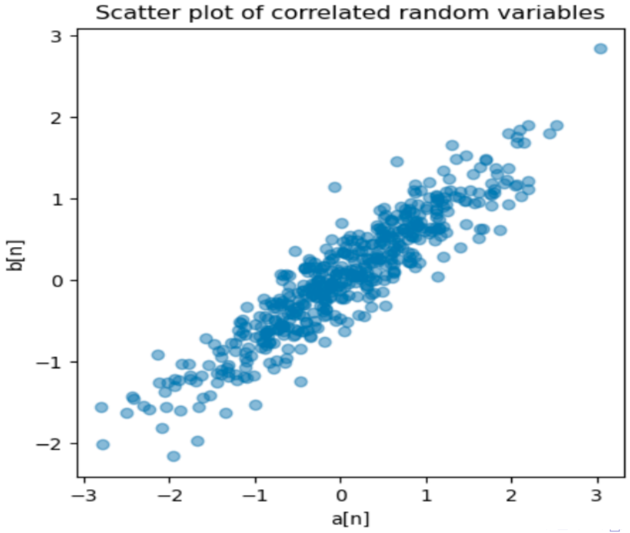
Measuring Dependence: Correlation and Covariance

- **Covariance** measures joint variability — how two random variables change together.
- **Correlation coefficient** = normalized covariance $\in [-1, 1]$; +1 means perfectly linear relationship.
- **Task**: Change coefficient 0.7 to 0.1 and 0.9 — observe scatter plot tightening or spreading.

Measuring Dependence

```
import numpy as np
import matplotlib.pyplot as plt
# Create two related random sequences
np.random.seed(1)
N = 500
a = np.random.randn(N)
b = 0.7 * a + 0.3 * np.random.randn(N)    # correlated with a

# Compute covariance and correlation
cov_ab = np.cov(a, b, ddof=0)[0,1]
corr_ab = np.corrcoef(a, b)[0,1]

print(f"Covariance = {cov_ab:.3f}, Correlation = {corr_ab:.3f}")

plt.figure(figsize=(5,5))
plt.scatter(a, b, alpha=0.5)
plt.title("Scatter plot of correlated random variables")
plt.xlabel("a[n]"); plt.ylabel("b[n]")
plt.show()
```

Measuring Dependence



Target Detection using correlation

Target Detection

Target Detection Using correlation

- When a radar or sonar transmits a random (or pseudo-random) signal $X(t)$, the echo from a target is a delayed and possibly attenuated version: $Y(t) = \alpha X(t - \tau)$
  - where $alpha$ is reflection strength; τ is time delay corresponding to target distance $(d = c\tau/2)$
- Compute the cross-correlation between $X(t)$ and $Y(t)$ to find the target delay peak
  - Equivalently the autocorrelation of $X(t)$ with delayed copies of itself

Target Detection

RVSP

Dr. G. Omprakash

Random Process — Stationarity and Non-Stationarity

Measuring Dependence: Correlation and Covariance

Target Detection using correlation

```
# Target detection using autocorrelation demo
import numpy as np
import matplotlib.pyplot as plt

np.random.seed(0)

# 1 Define random process X(t)
fs = 1000            # sampling frequency (Hz)
T = 1                # total time (1 second)
t = np.linspace(0, T, int(T*fs), endpoint=False)
X = np.random.randn(len(t))      # zero-mean Gaussian noise process

# 2 Define reflected echo Y(t) with time delay τ (target)
true_delay = 0.015             # 15 ms delay
attenuation = 0.7
delay_samples = int(true_delay * fs)
Y = np.zeros_like(X)
Y[delay_samples:] = attenuation * X[:-delay_samples]

# Add small noise
Y += 0.1 * np.random.randn(len(Y))

#Y2 = 0.5*X[:-400]  # second echo
#Y[400:] += Y2
```

Target Detection

```python
# 3️⃣ Compute autocorrelation between X and delayed version
corr = np.correlate(Y, X, mode='full')
lags = np.arange(-len(X)+1, len(X)) / fs

# Find estimated delay
est_delay = lags[np.argmax(corr)]

# 4️⃣ Plot signals and correlation
plt.figure(figsize=(12,3))
plt.plot(t, X, label="Transmitted X(t)")
plt.plot(t, Y, label="Received Y(t) (delayed)")
plt.xlim(0, 0.1)
plt.xlabel("Time (s)"); plt.title("Transmitted vs Received Signal")
plt.legend(); plt.show()

plt.figure(figsize=(8,3))
plt.plot(lags, corr)
plt.title("Cross-correlation between X(t) and Y(t)")
plt.xlabel("Lag (s)"); plt.ylabel("Correlation amplitude")
plt.axvline(est_delay, color='r', linestyle='--', label=f"Estimated delay = {est_delay*1000:.1f} ms")
plt.legend(); plt.show()

print(f"True delay = {true_delay*1000:.2f} ms, Estimated delay = {est_delay*1000:.2f} ms")
```

Target Detection

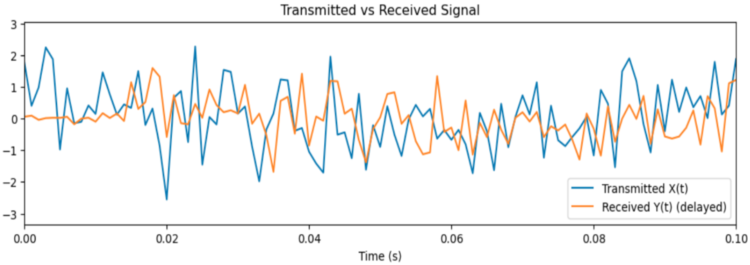


Figure: Transmitted and received signals

Target Detection

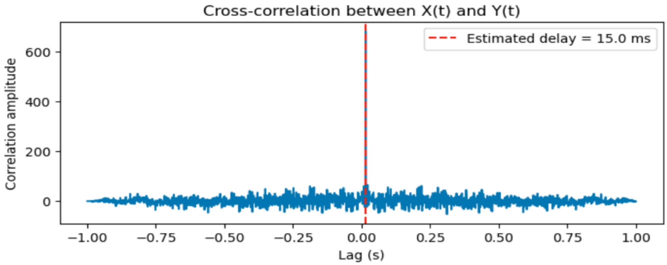


Figure: Correlation vs lag — sharp peak marks where the two align best.

[2]:
```python
c = 3e8  # speed of light (m/s)
distance = c * est_delay / 2
print(f"Estimated target distance ≈ {distance:.1f} meters")
```

```
Estimated target distance ≈ 2250000.0 meters
```

Acknowledge various sources for the images.

Thankyou