

Advanced Digital Signal Processing (ADSP) Lab - Python Lab Manual

Course Code: EEE G613

Instructor in Charge: Dr. Rajesh Kumar Tripathy

Teaching Assistant: Shaswati Dash

Lab Technician: Ramesh Pokanati

▼ Experiment No. - 7

Levinson-Durbin Recursion

For the given two $r_x(k)$ autocorrelation sequences

- a) $r_x = [1, 0.5, 0.5, 0.25]^T$
- b) $r_x = [2, 0.5(1+j), 0.5j]^T$

Use Levinson-Durbin Recursion to solve the autocorrelation normal equations for a second-order all-pole model. Write your own program to find the filter coefficients and error.

▼ Python Code-

```
#import libraries
import numpy as np

#define functions
def rtoa(r):
    r = np.squeeze(r)
    p = len(r) - 1
    a = np.array([1.0])
    epsilon = r[0]
    for j in range(2, p+2):
        gamma = -np.sum(r[1:j] * np.flipud(a)) / epsilon
        a = np.append(a, 0) + gamma * np.append(0, np.flipud(np.conj(a)))
        epsilon = epsilon * (1 - np.abs(gamma) ** 2)
    return a, epsilon

def rtog(r):
    a, epsilon = rtoa(r)
    gamma = atog(a)
    return gamma, epsilon

def atog(a):
    a = np.squeeze(a)
    p = len(a)
    a = a[1:p] / a[0]
    gamma = np.zeros(p - 1, dtype=complex)
    gamma[p - 2] = a[p - 2]
    for j in range(p - 2, 0, -1):
        a = (a[0:j] - gamma[j] * np.flipud(np.conj(a[0:j]))) / (1 - np.abs(gamma[j]) ** 2)
        gamma[j - 1] = a[j - 1]
    return gamma
```

```

# Part-(a)
r = np.array([1, 0.5, 0.5, 0.25])
a, epsilon1 = rtoa(r)
gamma, epsilon2 = rtog(r)
b = np.sqrt(epsilon1)
print("a = " , a)
print("b = " , b)
print("error = ")
print(epsilon1)

a = [ 1.      -0.375 -0.375  0.125]
b = 0.8100925873009825
error =
0.65625

```

```

# Part-(b)
r = np.array([2, 0.5 * (1 + 1j), 0.5j])
a, epsilon1 = rtoa(r)
gamma, epsilon2 = rtog(r)
b = np.sqrt(epsilon1)
print("a = " , a)
print("b = " , b)
print("error = ")
print(epsilon1)

a = [ 1.      +0.j      -0.21428571-0.21428571j  0.      -0.14285714j]
b = (1.3093073414159542+0j)
error =
(1.7142857142857142+0j)

```