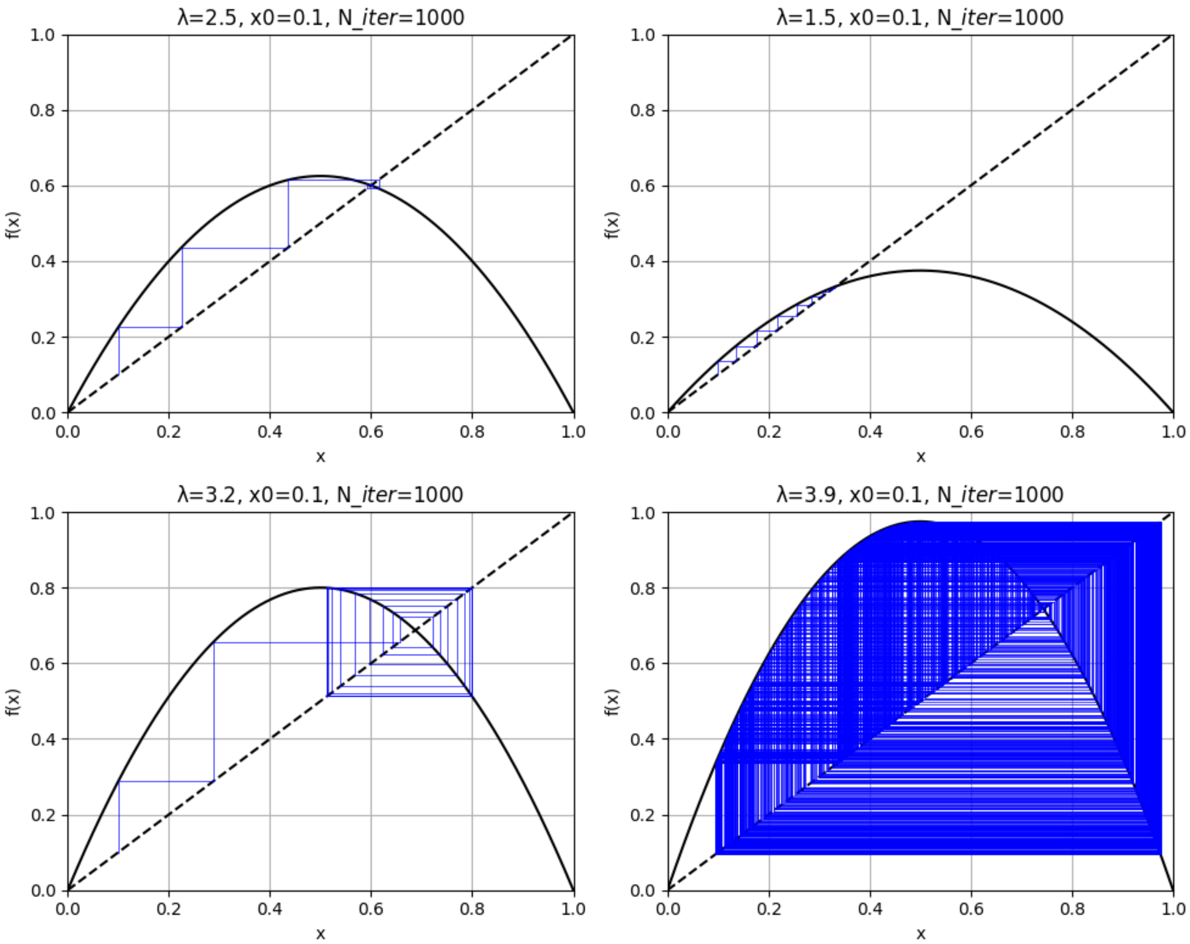
Lab 5 – Metody iteracyjne i rekurencyjne w przetwarzaniu sygnałów.

Arkadiusz Kurnik, Jan Cichoń

Zadanie 1:

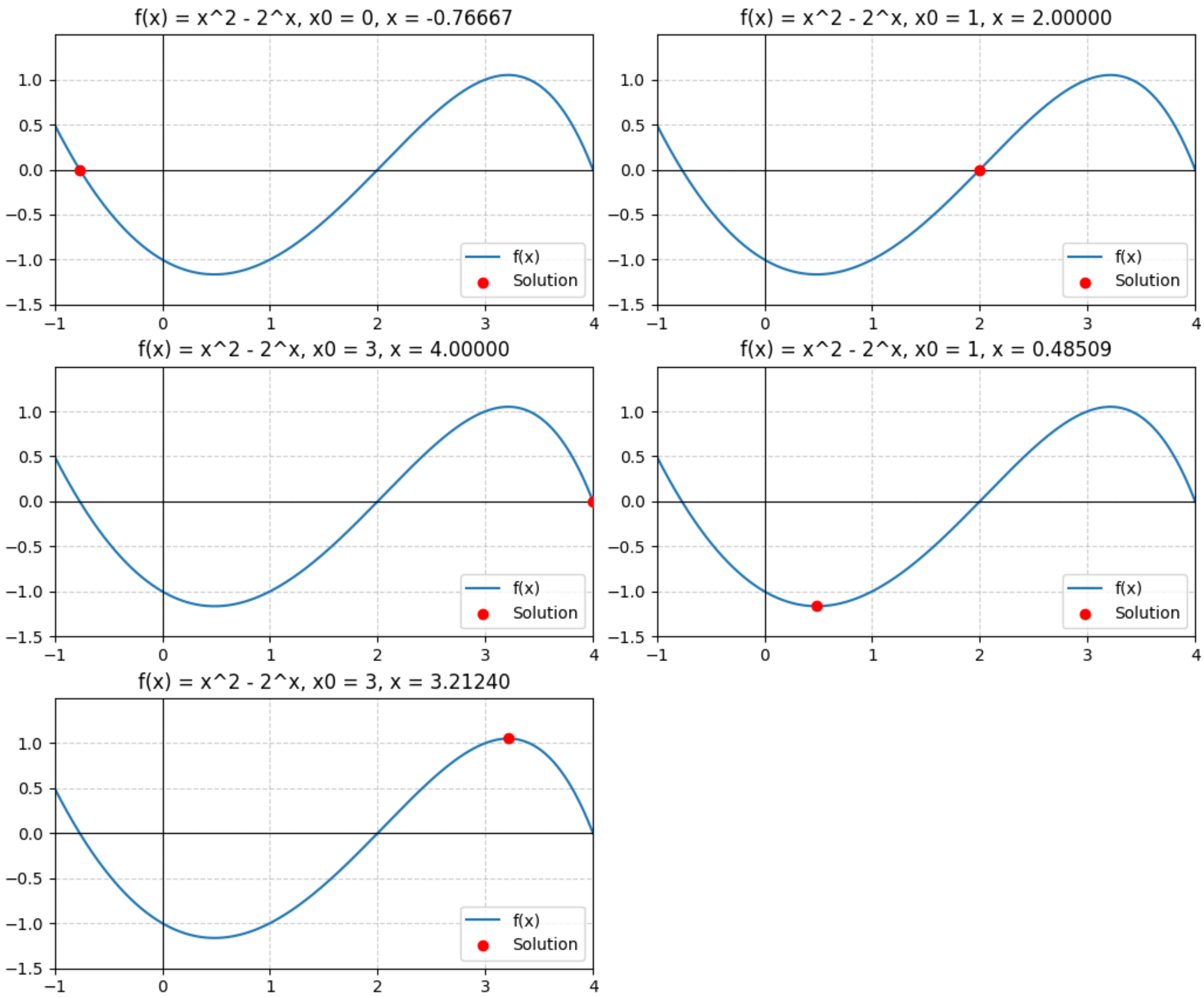
```python
import numpy as np
import matplotlib.pyplot as plt

# Funkcja logistyczna
def logistic_map(x, lam):
    return lam * x * (1 - x)

# Parametry iteracji
N_iter = 1000
x0 = 0.1
lambdas = [2.5, 1.5, 3.2, 3.9]

# Przygotowanie wykresów
fig, axes = plt.subplots(2, 2, figsize=(10, 8))
axes = axes.flatten()

x = np.linspace(0, 1, 500)

for i, lam in enumerate(lambdas):
    ax = axes[i]

    ax.plot(x, logistic_map(x, lam), 'k')  # wykres funkcji
    ax.plot(x, x, 'k--')                   # przekątna y = x

    xn = x0
    for _ in range(N_iter):
        x_next = logistic_map(xn, lam)
        # pionowa linia
        ax.plot([xn, xn], [xn, x_next], 'b', linewidth=0.5)
        # pozioma linia
        ax.plot([xn, x_next], [x_next, x_next], 'b', linewidth=0.5)
        xn = x_next

    ax.set_title(f'λ={lam}, x0={x0}, N_$iter$={N_iter}')
    ax.set_xlabel('x')
    ax.set_ylabel('f(x)')
    ax.set_xlim(0, 1)
    ax.set_ylim(0, 1)
    ax.grid(True)

plt.tight_layout()
plt.show()
```


λ=2.5, x0=0.1, N_iter=1000
λ=1.5, x0=0.1, N_iter=1000
λ=3.2, x0=0.1, N_iter=1000
λ=3.9, x0=0.1, N_iter=1000
f(x)
x
0.0
0.2
0.4
0.6
0.8
1.0

Zadanie 2:

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


def f(x):
    return x**2 - 2**x

def f_prime(x):
    return 2 * x - 2**x * np.log(2)

# Newton's method implementation
def newton_method(x0, max_iter=100, tol=1e-6):
    x = x0
    for _ in range(max_iter):
        fx = f(x)
        fpx = f_prime(x)
        if abs(fx) < tol:
            break
        x -= fx / fpx
    return x


x_vals = np.linspace(-1, 4, 500)
y_vals = f(x_vals)


plots = [
    (0, -0.7666666),
    (1, 2),
    (3, 4),
    (1, 0.48509),
    (3, 3.2124)
]


fig, axes = plt.subplots(3, 2, figsize=(12, 10))
axes = axes.flatten()
```

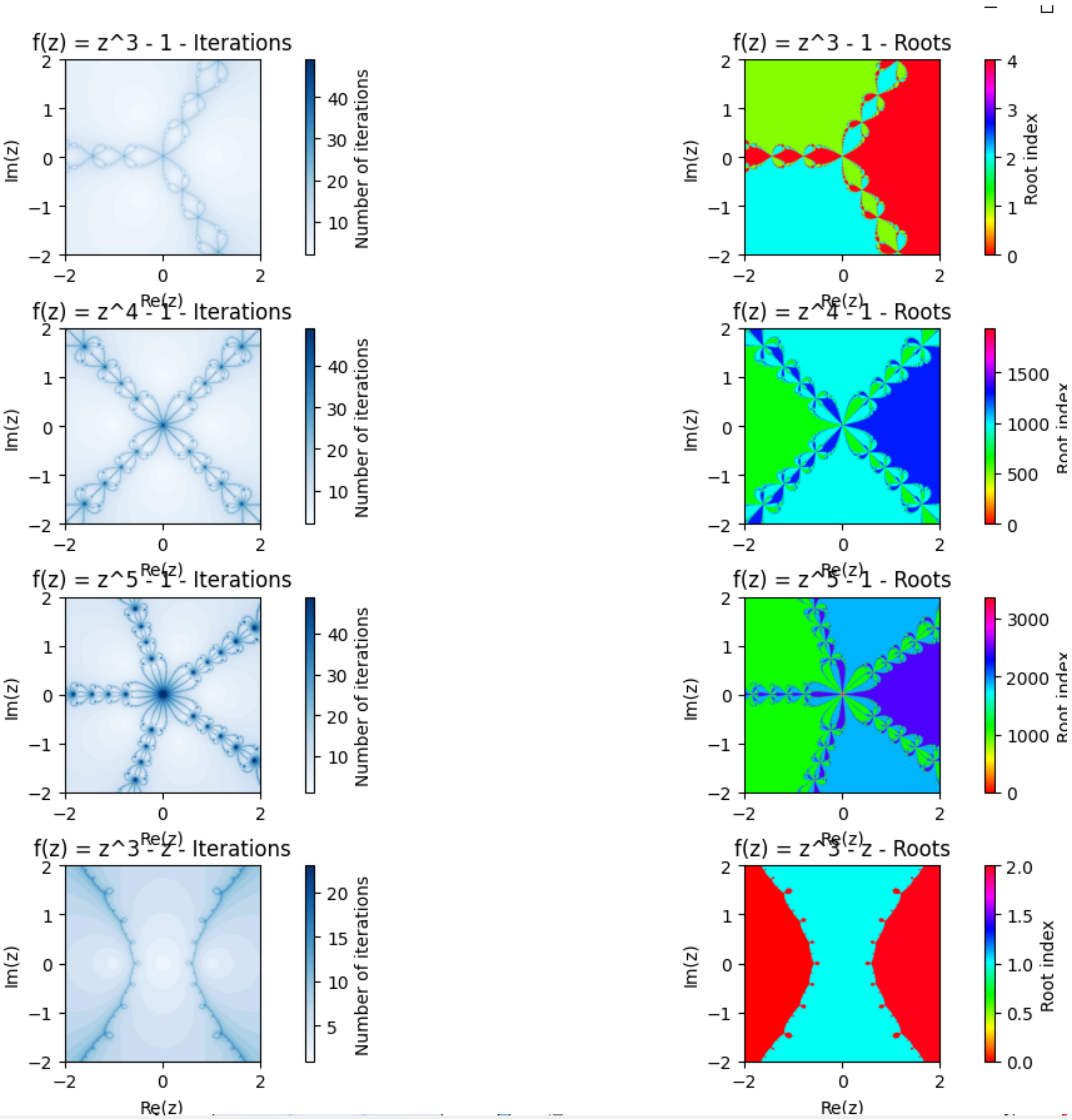
```python
for i, (x0, solution) in enumerate(plots):
    ax = axes[i]
    ax.plot(x_vals, y_vals, label='f(x)')
    ax.axhline(0, color='black', linewidth=0.8)
    ax.axvline(0, color='black', linewidth=0.8)
    ax.scatter([solution], [f(solution)], color='red', label='Solution', zorder=5)
    ax.set_title(f"f(x) = x^2 - 2^x, x0 = {x0}, x = {solution:.5f}")
    ax.set_xlim(-1, 4)
    ax.set_ylim(-1.5, 1.5)
    ax.set_xticks(np.arange(-1, 5, 1))
    ax.set_yticks(np.arange(-1.5, 1.5, 0.5))
    ax.legend()
    ax.grid(True, linestyle='--', alpha=0.6)


fig.delaxes(axes[-1])

plt.tight_layout()
plt.show()
```



Zadanie 3:

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

# Funkcje wielomianowe i ich pochodne
functions = [
    (lambda z: z**3 - 1, lambda z: 3 * z**2, "f(z) = z^3 - 1"),
    (lambda z: z**4 - 1, lambda z: 4 * z**3, "f(z) = z^4 - 1"),
    (lambda z: z**5 - 1, lambda z: 5 * z**4, "f(z) = z^5 - 1"),
    (lambda z: z**3 - z, lambda z: 3 * z**2 - 1, "f(z) = z^3 - z")
]

# Metoda Newtona
def newton_method(f, f_prime, z, max_iter=50, tol=1e-6):
    for i in range(max_iter):
        dz = f(z) / f_prime(z)
        z -= dz
        if abs(dz) < tol:
            break
    return z, i

# Siatka punktów w płaszczyźnie zespolonej
x = np.linspace(-2, 2, 800)
y = np.linspace(-2, 2, 800)
X, Y = np.meshgrid(x, y)
Z = X + 1j * Y

# Generowanie wykresów dla każdej funkcji
plt.figure(figsize=(16, 12))

for idx, (f, f_prime, title) in enumerate(functions):
    # Iteracje Newtona
    roots = np.zeros(Z.shape, dtype=complex)
    iterations = np.zeros(Z.shape, dtype=int)

    for i in range(Z.shape[0]):
        for j in range(Z.shape[1]):
            roots[i, j], iterations[i, j] = newton_method(f, f_prime, Z[i, j])

    # Unikalne pierwiastki
    unique_roots = np.unique(np.round(roots, decimals=6))

    # Mapowanie pierwiastków na kolory
    root_colors = {root: idx for idx, root in enumerate(unique_roots)}
    colors = np.vectorize(lambda z: root_colors[np.round(z, decimals=6)])(roots)
```

```python
    # Rysowanie wykresów
    plt.subplot(len(functions), 2, 2 * idx + 1)
    plt.imshow(iterations, extent=(-2, 2, -2, 2), cmap='Blues')
    plt.colorbar(label='Number of iterations')
    plt.title(f'{title} - Iterations')
    plt.xlabel('Re(z)')
    plt.ylabel('Im(z)')

    plt.subplot(len(functions), 2, 2 * idx + 2)
    plt.imshow(colors, extent=(-2, 2, -2, 2), cmap='hsv')
    plt.colorbar(label='Root index')
    plt.title(f'{title} - Roots')
    plt.xlabel('Re(z)')
    plt.ylabel('Im(z)')

plt.tight_layout()
plt.show()
```



Zadanie 4:

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

# Parametry
M = 40                    # Długość filtra
N = 50000                 # Liczba próbek
sigma_x = 1               # Odchylenie standardowe sygnału x[n]
sigma_v = 0.5             # Odchylenie standardowe szumu
omega_c1 = np.pi / 2      # Wyjściowa częstotliwość odcięcia

def lowpass_impulse_response(omega_c, M):
    h = np.zeros(M)
    for n in range(-M//2, M//2):
        if n == 0:
            h[n+M//2] = omega_c / np.pi
        else:
            h[n+M//2] = np.sin(omega_c * n) / (np.pi * n)
    return h

h_true = lowpass_impulse_response(omega_c1, M)

x = np.random.normal(0, sigma_x, N)

d_clean = np.convolve(x, h_true, mode='full')[:N]
v = np.random.normal(0, sigma_v, N)
d = d_clean + v

def LMS(x, d, M, mu):
    N = len(x)
    h = np.zeros(M)
    e = np.zeros(N)
    H = np.zeros((N, M))
    for n in range(M, N):
        x_vec = x[n:n-M:-1]
        y = np.dot(h, x_vec)
        e[n] = d[n] - y
        h = h + 2 * mu * e[n] * x_vec
        H[n, :] = h
    return e, H
```

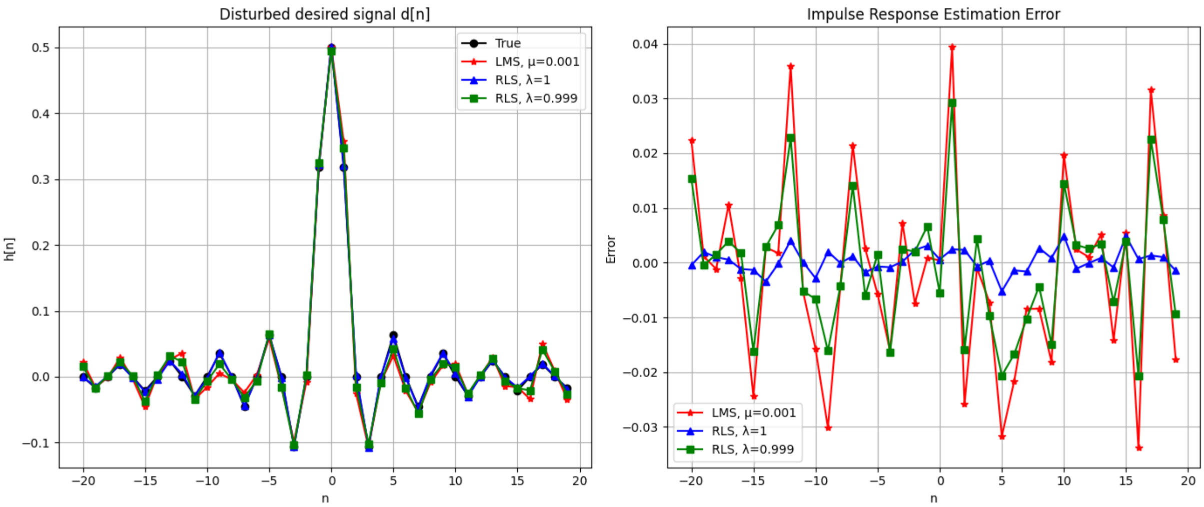
```python
def RLS(x, d, M, lam, delta=0.01):
    N = len(x)
    h = np.zeros(M)
    e = np.zeros(N)
    P = (1/delta) * np.eye(M)
    H = np.zeros((N, M))
    for n in range(M, N):
        x_vec = x[n:n-M:-1]
        pi = np.dot(P, x_vec)
        g = 1.0 / (lam + np.dot(x_vec, pi))
        k = pi * g
        y = np.dot(h, x_vec)
        e[n] = d[n] - y
        h = h + k * e[n]
        P = (P - np.outer(k, np.dot(x_vec, P))) / lam
        H[n, :] = h
    return e, H

mu = 0.001
lam1 = 1
lam2 = 0.999

e_LMS, H_LMS = LMS(x, d, M, mu)
e_RLS_1, H_RLS_1 = RLS(x, d, M, lam1)
e_RLS_0999, H_RLS_0999 = RLS(x, d, M, lam2)

# --- Wykresy ---

plt.figure(figsize=(14,6))
n = np.arange(-M//2, M//2)

plt.subplot(1, 2, 1)
plt.plot(n, h_true, 'ko-', label='True')
plt.plot(n, H_LMS[-1,:], 'r*-', label='LMS, μ=0.001')
plt.plot(n, H_RLS_1[-1,:], 'b^-', label='RLS, λ=1')
plt.plot(n, H_RLS_0999[-1,:], 'gs-', label='RLS, λ=0.999')
plt.xlabel('n')
plt.ylabel('h[n]')
plt.title('Disturbed desired signal d[n]')
plt.grid()
plt.legend()
```

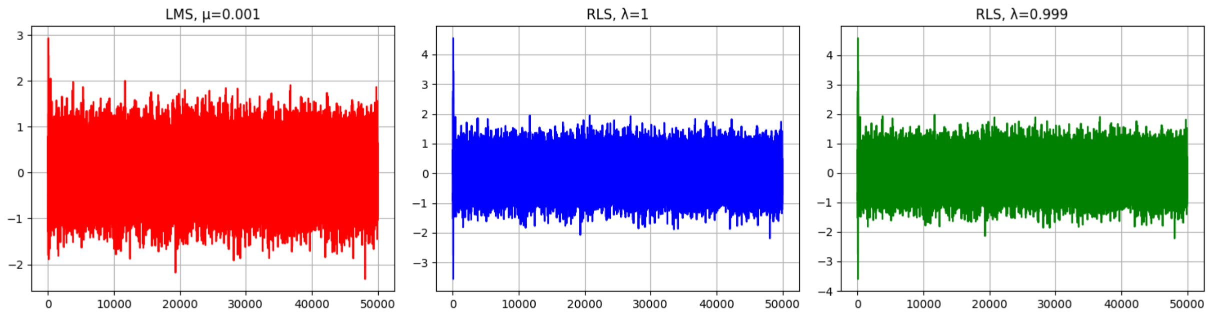
```
plt.subplot(1, 2, 2)
plt.plot(n, H_LMS[-1,:] - h_true, 'r*-', label='LMS, μ=0.001')
plt.plot(n, H_RLS_1[-1,:] - h_true, 'b^-', label='RLS, λ=1')
plt.plot(n, H_RLS_0999[-1,:] - h_true, 'gs-', label='RLS, λ=0.999')
plt.xlabel('n')
plt.ylabel('Error')
plt.title('Impulse Response Estimation Error')
plt.grid()
plt.legend()

plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
plt.plot(e_LMS, 'r')
plt.title('LMS, μ=0.001')
plt.grid()

plt.subplot(1,3,2)
plt.plot(e_RLS_1, 'b')
plt.title('RLS, λ=1')
plt.grid()

plt.subplot(1,3,3)
plt.plot(e_RLS_0999, 'g')
plt.title('RLS, λ=0.999')
plt.grid()
plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
plt.plot(H_LMS)
plt.title('LMS, μ=0.001')
plt.grid()

plt.subplot(1,3,2)
plt.plot(H_RLS_1)
plt.title('RLS, λ=1')
plt.grid()
```

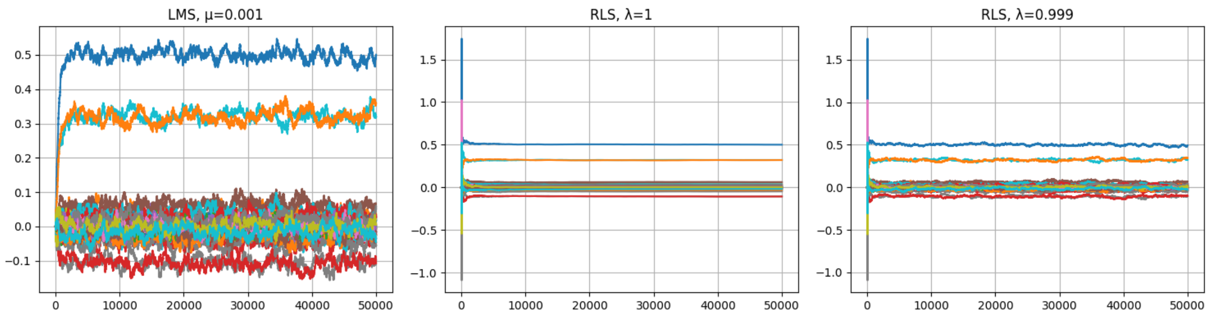
```python
plt.subplot(1,3,3)
plt.plot(H_RLS_0999)
plt.title('RLS, λ=0.999')
plt.grid()
plt.tight_layout()
plt.show()

import numpy as np
import matplotlib.pyplot as plt

import numpy as np
import matplotlib.pyplot as plt

# Parametry
M = 40                     # Długość filtra
N = 50000                  # Liczba próbek
sigma_x = 1                # Odchylenie standardowe sygnału x[n]
sigma_v = 0.5              # Odchylenie standardowe szumu
omega_c1 = np.pi / 2       # Wyjściowa częstotliwość odcięcia
omega_c2 = -0.5 * np.pi / 2  # Po zmianie

def lowpass_impulse_response(omega_c, M):
    h = np.zeros(M)
    for n in range(-M//2, M//2):
        if n == 0:
            h[n+M//2] = omega_c / np.pi
        else:
            h[n+M//2] = np.sin(omega_c * n) / (np.pi * n)
    return h

h_true1 = lowpass_impulse_response(omega_c1, M)
h_true2 = lowpass_impulse_response(omega_c2, M)

x = np.random.normal(0, sigma_x, N)

d_clean = np.zeros(N)
for n in range(N):
    if n < N//2:
        h_true = h_true1
    else:
        h_true = h_true2
    if n >= M:
        d_clean[n] = np.dot(h_true, x[n:n-M:-1])

v = np.random.normal(0, sigma_v, N)
d = d_clean + v
```

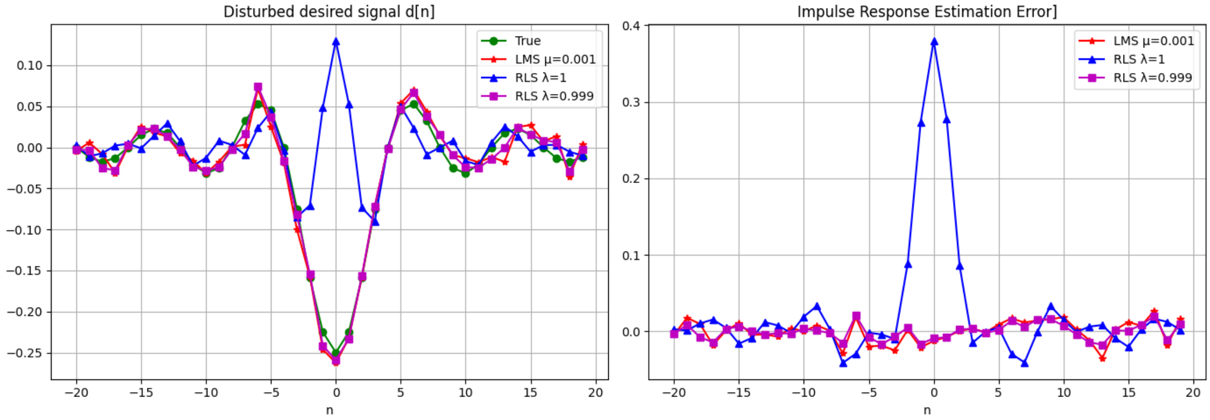
```
mu = 0.001
lam1 = 1
lam2 = 0.999

e_LMS, H_LMS = LMS(x, d, M, mu)
e_RLS_1, H_RLS_1 = RLS(x, d, M, lam1)
e_RLS_0999, H_RLS_0999 = RLS(x, d, M, lam2)

n_axis = np.arange(-M//2, M//2)

plt.figure(figsize=(14,5))

plt.subplot(1,2,1)
plt.plot(n_axis, h_true2, 'g-o', label='True')
plt.plot(n_axis, H_LMS[-1,:], 'r*-', label='LMS μ=0.001')
plt.plot(n_axis, H_RLS_1[-1,:], 'b^-', label='RLS λ=1')
plt.plot(n_axis, H_RLS_0999[-1,:], 'ms-', label='RLS λ=0.999')
plt.xlabel('n')
plt.title("Disturbed desired signal d[n]")
plt.grid()
plt.legend()

plt.subplot(1,2,2)
plt.plot(n_axis, H_LMS[-1,:] - h_true2, 'r*-', label='LMS μ=0.001')
plt.plot(n_axis, H_RLS_1[-1,:] - h_true2, 'b^-', label='RLS λ=1')
plt.plot(n_axis, H_RLS_0999[-1,:] - h_true2, 'ms-', label='RLS λ=0.999')
plt.xlabel('n')
plt.title("Impulse Response Estimation Error]")
plt.grid()
plt.legend()

plt.tight_layout()
plt.show()

plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
plt.plot(e_LMS, 'r')
plt.title('LMS μ=0.001')
plt.grid()
plt.subplot(1,3,2)
plt.plot(e_RLS_1, 'b')
plt.title('RLS λ=1')
plt.grid()
plt.subplot(1,3,3)
plt.plot(e_RLS_0999, 'g')
plt.title('RLS λ=0.999')
plt.grid()
plt.tight_layout()
```

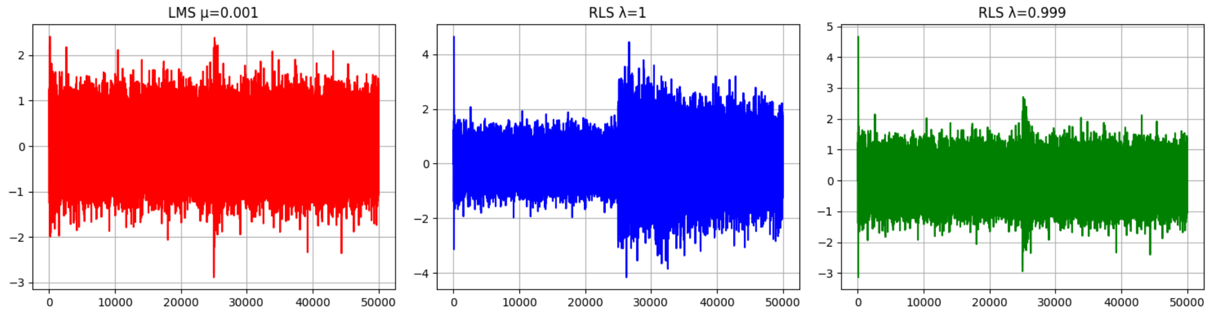
```python
plt.show()

plt.figure(figsize=(15, 4))
plt.subplot(1,3,1)
plt.plot(H_LMS)
plt.title('LMS μ=0.001')
plt.grid()
plt.subplot(1,3,2)
plt.plot(H_RLS_1)
plt.title('RLS λ=1')
plt.grid()
plt.subplot(1,3,3)
plt.plot(H_RLS_0999)
plt.title('RLS λ=0.999')
plt.grid()
plt.tight_layout()
plt.show()
```

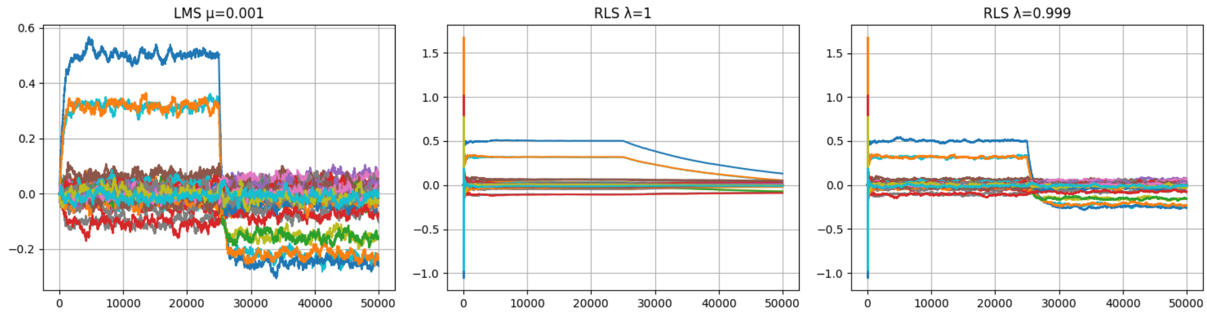







Disturbed desired signal d[n]
True
LMS μ=0.001
RLS λ=1
RLS λ=0.999
n
Impulse Response Estimation Error]
LMS μ=0.001
RLS λ=1
RLS λ=0.999
n


LMS μ=0.001
RLS λ=1
RLS λ=0.999


LMS μ=0.001
RLS λ=1
RLS λ=0.999