22.46 Procesamiento Adaptativo de Señales Aleatorias
Laboratorio de descenso por gradiente

Hoy analizaremos el algoritmo de descenso por gradiente.

1. Implementar el filtrado óptimo Wiener con descenso por gradiente en la plantilla `steepest_descent.py`. Prestar atención al formato de salida.
2. Probar la función con los argumentos:

$$\mathbf{R} = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \mathbf{p} = \begin{bmatrix} 6 \\ 4 \end{bmatrix}, \mu = 0.1, N = 1000, N = 1000, \mathbf{w}(0) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \sigma_d^2 = 20$$

   Representar los coeficientes w_1 y w_2 en función del tiempo. Representar como referencia la solución del filtro óptimo Wiener correspondiente.
3. Representar la curva de error $J(\mathbf{w})$.
4. Determinar los autovalores de $\mathbf{R}$, y a partir de ellos determinar $\mu_{\max}$. Para un valor de μ cercano a $\mu_{\max}$ representar la evolución de los coeficientes $\mathbf{w}(t)$.
5. Considerar el escenario de $\mathbf{R}$ y $\mathbf{p}$ variables en el tiempo. Explicar los beneficios del descenso por gradiente en comparación con el método de Wiener.
6. El archivo `steepest_descent_test.py` filtra una señal de música $d(t)$ con un filtro Wiener obtenido por descenso por gradiente con diferentes valores de μ.

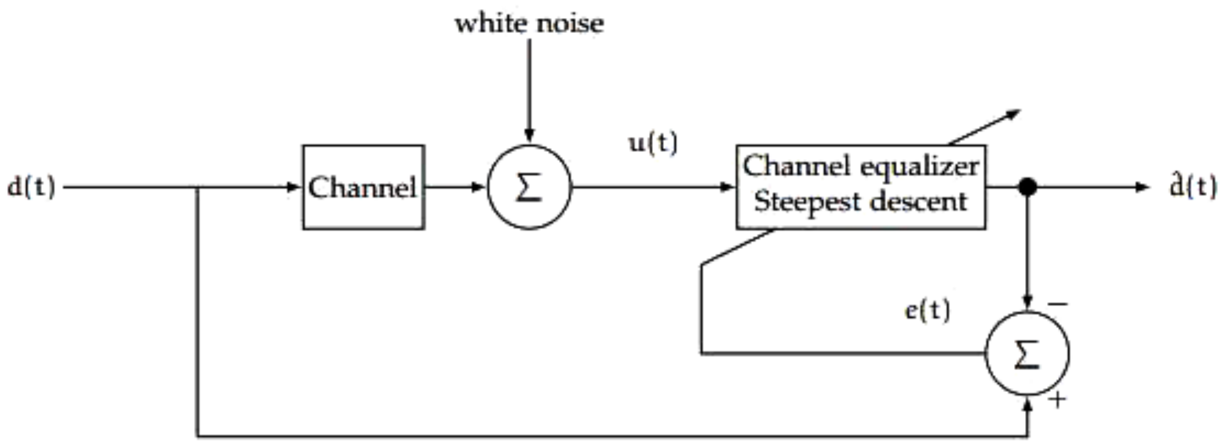

Figure 1: Channel equalization.

Analizar los gráficos generados. ¿Qué ocurre en el caso $\mu = 10^{-6}$?