Programación Avanzada
Trabajo Práctico No. 4 - Apps en C++
Profesor: M.Sc. Mauro Baquero-Suárez
Programa: Ingeniería Mecatrónica

UNIVERSIDAD NACIONAL DE COLOMBIA

Aplicaciones prácticas en C++

10 %

1. Se tiene el ejercicio "8-puzzle" (o "rompecabezas de 8 piezas") que consiste en un tablero de 3x3 con 8 piezas numeradas del 1 al 8 y un espacio vacío. El objetivo es mover las piezas para alcanzar una configuración objetivo. Implemente una aplicación en C++ para encontrar la solución óptima al problema del 8-puzzle. La aplicación debe permitir al usuario ingresar la configuración inicial y mostrar la secuencia de movimientos necesarios para resolver el rompecabezas.

Fig. 1: Ejemplo de configuración inicial.

5	7	2
4	1	
3	8	6

Fig. 2: Solución.

1	2	3
8		4
7	6	5

50 %

2. Se desea implementar una aplicación de "'Hardware in the Loop (HIL)" en C/C++ para simular el comportamiento de una planta en tiempo real. En la aplicación se debe permitir la interacción con el modelo de la planta SISO de (1), y el derivador de orden superior de (2), ambos implementados en C/C++. El objetivo es evaluar el seguimiento de la referencia f_0 del derivador y las derivadas de 1er, 2do y 3er orden que entrega este algoritmo en diferentes condiciones de operación. La aplicación debe crear un archivo .txt donde se almacenen las señales de entrada y salida de la planta, así como las señales generadas por el derivador, para su posterior análisis. La implementación debe considerar los siguientes aspectos:

$$G(s) = \frac{7s^2 - 28s + 21}{s^3 + 9.8s^2 + 30.65s + 30.1}. \tag{1}$$

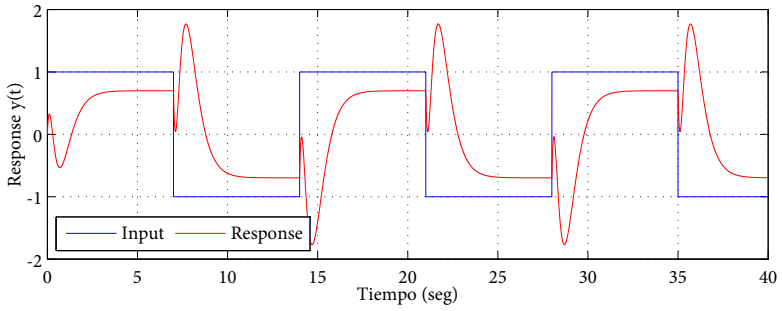

Fig. 3: Respuesta de la planta.

$$
\begin{aligned}
z_0(t_k+1) &= z_0(t_k) + \tau_s\varphi_0(z_0(t_k) - f(t_k)) + \tau_s z_1(t_k) + \frac{\tau_s^2}{2!}z_2(t_k) + \frac{\tau_s^3}{3!}z_3(t_k),\\
z_1(t_k+1) &= z_1(t_k) + \tau_s\varphi_1(z_0(t_k) - f(t_k)) + \tau_s z_2(t_k) + \frac{\tau_s^2}{2!}z_3(t_k),\\
z_2(t_k+1) &= z_2(t_k) + \tau_s\varphi_2(z_0(t_k) - f(t_k)) + \tau_s z_3(t_k),\\
z_3(t_k+1) &= z_3(t_k) + \tau_s\varphi_3(z_0(t_k) - f(t_k)),
\end{aligned}
\tag{2}
$$

donde $\varphi_i(z_0(t_k) - f(t_k)) = -\lambda_{n-i}L^{\frac{i+1}{n+1}}|z_0(t_k) - f(t_k)|^{\frac{n-i}{n+1}}\text{sign}(z_0(t_k) - f(t_k))$, con $i = 0, 1, 2, 3$, y $\lambda_0 = 1.3$, $\lambda_1 = 1.85$, $\lambda_2 = 2.79$, $\lambda_3 = 6.48$. La señal de referencia $f(t_k)$ puede ser una señal escalón, rampa o senoidal, $L = 1.8$ es la constante de Lipschitz y $\tau_s = 0.004$ s es el tiempo de muestreo.

La respuesta de la planta a una señal escalón unitario se muestra en la Fig. 3. Tenga en cuenta los siguientes puntos al desarrollar la aplicación:

1. Implemente el modelo de la planta en C/C++ utilizando un método de discretización adecuado para garantizar la estabilidad y precisión en tiempo real. Se sugiere utilizar alguna tarjeta de desarrollo compatible con C/C++ para la adquisición y generación de señales en tiempo real (por ejemplo, Arduino, Raspberry Pi, etc.). Esta tarjeta debe recibir la señal de entrada y enviar la señal de salida de la planta con una tasa de muestreo de 250 Hz.
2. Implemente el derivador de orden superior en otra tarjeta de desarrollo, sincronizada con la primera, asegurándose de que pueda manejar diferentes tipos de señales de referencia. Esta tarjeta debe recibir la señal de referencia y enviar las derivadas calculadas con la misma tasa de muestreo de 250 Hz.
3. Busque la manera para almacenar las señales de entrada y salida en tiempo real. Adicionalmente, cree una interfaz que permita visualizar las señales en tiempo real. El usuario puede seleccionar diferentes señales de referencia (escalón, rampa, senoidal) y ajustar parámetros del derivador.
4. Asegúrese de que la aplicación pueda ejecutarse en tiempo real.

Realice pruebas de la aplicación con diferentes señales de referencia (escalón, rampa, senoidal) y documente los resultados obtenidos.

40 % 3. Desarrolle una aplicación simple de Redes Neuronales Artificiales (RNA) en C++, para resolver un problema de clasificación de números naturales del 0 al 9. La aplicación recibe como entrada un conjunto de números previamente almacenados en el archivo `digitos.txt`. La aplicación debe reconocer cada número y clasificarlo correctamente en una de las siguientes categorías:

- Clase 0: Números pares (0, 2, 4, 6, 8).
- Clase 1: Números impares (1, 3, 5, 7, 9).
- Clase 2: Números primos (2, 3, 5, 7).
- Clase 3: Números compuestos (4, 6, 8, 9).

Se recomienda utilizar una arquitectura de RNA con una capa de entrada, una o más capas ocultas y una capa de salida.

Entrega

El trabajo práctico debe ser entregado en formato digital a través de un repositorio en la plataforma Github. Asegúrese de incluir un archivo README con instrucciones claras sobre cómo compilar y ejecutar su código. También, el archivo README debe explicar claramente la metodología que utiliza cada programa para resolver el problema dado. La fecha límite para la entrega es el 16 de Diciembre del 2025 a las 23:59 horas. No se aceptarán entregas tardías sin una justificación válida.