International Master of Science in Electrical Engineering

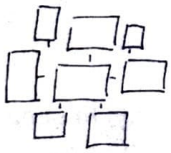
Embedded Signal Processing Systems

Real-Time Implementation of Digital Filters on Cortex-M Microcontrollers

Prof. Dr. C. Jakob

Hochschule Darmstadt
University of Applied Science Darmstadt

h_da



ESPS Laboratory #1 – Follow Up, Part 2

Version 1.4, 21 May 2025, created using WordTEX

Submission Deadline: -

For questions, comments or suggestions, please contact me by email:
christian.jakob@h-da.de

Darmstadt, May 2025

Digital Oscillator

Second Order IIR Filter

By placing a pair of complex-conjugate poles **exactly on** the unit circle, a discrete-time digital oscillator can be implemented that produces an undamped sinusoidal waveform in response to a single impulse excitation. This configuration corresponds to a second-order IIR system, where the pole angle directly defines the oscillation frequency, and the unit magnitude of the poles guarantees constant amplitude over time. Unlike bandpass filters or resonators that require a continuous input to maintain oscillation, this oscillator structure needs only an initial impulse to begin generating the sinusoidal signal. The transfer function of such an IIR oscillator is given by:

$$H(z) = \frac{\sin(\omega_0)\, z^{-1}}{1 - 2\cos(\omega_0)\, z^{-1} + z^{-2}}$$

A frequency-dependent gain factor in the numerator ensures that the output amplitude is correctly normalized. The resulting signal is a theoretically infinite-duration, steady-state sine wave whose frequency is precisely controlled by the angular position of the poles.

Hint: Use the provided MATLAB script to explore the behavior of the digital IIR oscillator in both the time and frequency domain. Observe how placing two complex-conjugate poles exactly on the unit circle results in a sustained sinusoidal output.

STM32G4 related tasks

Implement the digital oscillator in **floating-point arithmetic** within the **Timer Interrupt Service Routine (48 kHz ISR/TIM6)** on the STM32G4 MCU. The oscillator frequency should be set to **500 Hz.**

a) **Input Signal Source:**
   - Apply a single (scaled) impulse to start the oscillator.

b) **Filter Implementation:**
   - Implement the oscillator in C using floating-point arithmetic (float only) to simplify the implementation and avoid rounding artifacts. Fixed-point arithmetic will be addressed in future lab sessions.

c) **Output Conversion and DAC Handling:**
   - After the actual filter operation, convert the floating-point output back to the **integer domain** for DAC output (e.g., scale and cast to 12-bit unsigned integer). Output **only the the filter output** to **one of the two DAC channels.**

d) **Timing and Execution:**
   - Configure the **hardware timer TIM6** to trigger the ISR at a fixed sampling rate of **48 kHz.** Ensure that the ISR executes within the sampling period without overruns.

e) **Verification:**
   - Use an oscilloscope to observe the DAC output and verify that the correct frequency is being synthesized.