NTNU
Norwegian University of Science and Technology
Institute of ICT and Natural Sciences

AIS2201 - Signal Processing

Individual Project

Improved Frequency Detection

Introduction

In **Exercise 6** we implemented a simple frequency detection algorithm on a STM32 microcontroller, with the aim of detecting the dominant frequency $\hat{f}$ of an analog voltage signal in real time. The algorithm in question employed a N-point DFT to transform the input signal $x[n]$ to the frequency domain, and identified the frequency of the strongest sinusoidal component. Mathematically, this algorithm can be described as in equation 1, where $X[m]$ is the DFT of the N most recent samples of the input signal $x[n]$ and f_s is the sampling frequency.

$$\hat{f} = \underset{m \in \{0,1,\ldots,\frac{N}{2}\}}{\arg\max} \left(|X[m]|\right) \times \frac{f_s}{N} \tag{1}$$

While the algorithm described in equation 1 works quite well to detect the frequency of simple sinusoidal input, there are some limitations:

1 The estimated frequency is confined to a discrete range of values, resulting in a relatively imprecise measure of frequency.

2 The algorithm does not take noise into account, which may result in false detection if there is sufficiently strong measurement noise present.

3 The algorithm is not ideal for estimating the *fundamental frequency* f_0 of a signal as we explored in assignment 1. The reason being the frequency component with the highest magnitude will merely be one among many harmonics.

The main technical goal of the individual project is to make improvements to the algorithm in equation 1 with the aim of addressing these limitations. Before being implemented on a microcontroller, the algorithm should be thoroughly tested with test data to evaluate it's potential for improved performance.

Project Overview

For this project you will devise (at least) three modifications you can make to the frequency detection system in equation 1 which you expect will improve it's performance in one (or more) of the three areas mentioned in the introduction. Your modifications could be as simple as using zero padding to increase the spectral resolution, as complex as implementing an advanced algorithm for pitch detection you found in a research paper, or anything in between. It is not a requirement that you address all three points mentioned in the introduction, you may for instance use three modifications all aimed at improving frequency resolution.

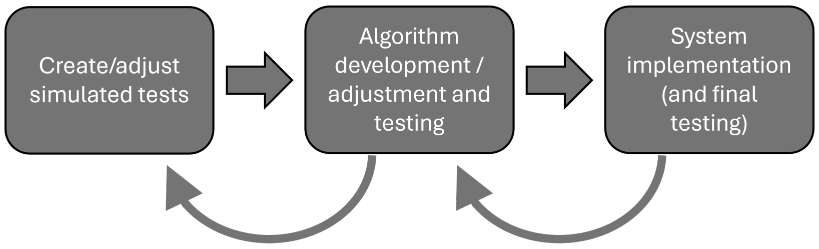
Design Parameters

Here follows a list of parameters/limitations for the final system should aim to satisfy:

- The final system should produce a fundamental frequency estimate $\hat{f}_0$ of a signal within the frequency range $25\text{Hz} \leq f \leq 4200\text{Hz}$. It is not a requirement to be able to handle signals which fall outside this category as a separate "case".
- The estimate "refresh interval" must be at most 0.2 seconds, although it is permissable for samples taken more than 0.2 seconds ago to influence the estimate.
- The algorithm must be able to adapt to changes in fundamental frequency within 0.5 seconds, with samples more than 0.5 seconds old having negligible influence (e.g. $\leq 10\%$).
- You are free to choose any sampling frequency you may wish, as long as it is achievable on the STM32 hardware.

Development Process

When working on a project such as this it is extremely useful to be able to verify that your system produces the expected results at every stage of the process through simulations. If you jump straight to implementation, and begin troubleshooting once the finished system does not produce the excpected results, you risk wasting a lot of time trying to work out where the problems originate. Therefore, the project work will be separated into three main stages: developing test framework, algorithm development, and system implementation.


Create/adjust simulated tests
Algorithm development / adjustment and testing
System implementation (and final testing)

Figur 1: Workflow for individual project

Part 1: Test framework

Before jumping into developing new modified DSP algorithms, it is important to have a clear idea of how to quantify the performance of our algorithm. The first milestone of this project is therefore to create a series of tests which use simulated signal input, and can measure the accuracy of the frequency estimate at the output. This is most easily accomplished using a scripting language with extensive support for numerical calculations such as Python (or MATLAB).

For this project we will use the following test input signals to evaluate the performance of the fundamental frequency estimation algorithm. We will begin by running the simple detection algorithm in equation 1 through the tests to establish a baseline.

- Pure sine wave with varying levels of noise
- Sine wave with abrupt change in frequency
- Musical instrument with varying levels of noise
- Vocal

To help you get started, exercise 7 will provide test audio and include a more detailed description of the tests involved. Exercise 7 has a approval deadline on **thursday the 24th of October**, but you are naturally not prohibited from making later adjustments to your test framework if the need arises.

Part 2: Algorithm Development

Use what you have learned throughout the semester in Signal Processing to choose (at least) three modifications you wish to make to the algorithm in equation 1 with the aim of improving performance. Use the test framework from part 1 to compare the effect of each modification separately with the baseline (i.e. equation 1. Finally, run the tests on the fully modified system and compare the final results with previous tests. Should you feel the need to add your own tests to evaluate a specific modification this is highly encouraged.
P.S. if it is not possible to isolate one modification from another this is OK, and they can still count as two separate modifications as long as you make a note of this in the report.

It should be noted that pitch detection (also often referred to as fundamental frequency estimation) is still an active field of research, and there are no "best" solutions which outperform other algorithms for all possible test cases. The purpose of this project section is not that you find the "correct" solution to a question, but that you can demonstrate an ability to identify what concepts from DSP theory can potentially be used to solve a specific practical problem, and use programming simulations to put your theories to the test.

When writing your project report, here are some questions which should be answered:

- Why have you chosen to make this modification, and what improvement do you expect to see in the simulated tests?
- How was the modification implemented in Python/Matlab (or other preferred language)?
- Did the modification have the desired effect?
- Did the tests uncover any unexpected side-effects, and why may they have occurred?

Part 3: Implementation

Now that you have finalized a new and improved algorithm for fundamental frequency estimation, it is time to implement it on a Microcontroller. Once done, you can use e.g. a function generator to provide various waveform input signals as a final functional test. Points will be awarded for considering the efficiency of your code (i.e. using fast convolution in favor of direct convolution etc..)

Please note:

- If you are unable to complete a working STM32 project this is OK, and does not translate to an automatic fail. The most important part of the project is to provide a well-documented overview of your algorithm development process. Providing incomplete "main.c" code alongside an explanation of *how* you intended to implement your improved algorithm will go a long way.
- You are not expected to be an expert in using STM's HAL library, ARM's CMSIS-DSP library etc, finding good documentation for these functions is not an easy task. Ask your teacher if you need help, and make use of any resources which may aid inn programming the STM (AI included). You are however still expected to be able to understand what your program does, and present an overview of the algorithm implementation in your report.
- If you have time to spare, consider using the built-in MEMS microphone on the STM32 to test audio input. A STM32 project template which uses the built-in microphone on the STM32 to gather audio will be made available.

Project Submission

Your completed project will be submitted as a project report alongside the rest of the course portfolio by the final exam deadline, **december 9th 2024 at 12:00**. Prioritize keeping the report brief rather than providing lengthy theory explanations of what e.g. a window function is. The main emphasis of your report should be the following:

- A description of the improvements you have chosen and why you chose them (e.g. what do you expect this improvement to accomplish and why).
- Presentation and discussion of simulated test results, tying the observed behavior of your improvements to DSP principles covered in the course.
- An overview of your final system implementation, explaining the flow of sampled information from ADC to fundamental frequency estimate. System diagrams are encouraged.

Provide Python-code from parts 1 & 2, as well as the code of your "main.c" file as separate attachments to the project report.