Student: Ben stone

Course: ECE 3210

Subject: Lab 03, Numerical Convolution

Date: September 29, 2023

---

1 Introduction

During this lab we implemented a convolution function in C and Python, compared the results from the functions against an analytically solved convolution, and compared the time it takes for the C function to run against the time for the Python function to run. Convolution plays a fundamental role in understanding the behavior of signals and systems, this lab is intended to help gain a deeper understanding of the underlying mathematics behind the convolution integral.

2 Theory

In this lab, we analytically evaluate the convolution for the following equations

$$f(t) = e^{-t/2}\left(u(t-1) - u(t-10)\right) \tag{1}$$

$$h(t) = e^{-t}\left(u(t-2) - u(t-5)\right) \tag{2}$$

$$y(t) = f(t) * h(t) \tag{3}$$

We calculate the convolution by separating the convolution into 5 different time intervals.

$$y(t) = \begin{cases} 0 & \text{if } t < 3 \\ e^{-t}\int_{1}^{t-2} e^{\tau/2}\, d\tau & \text{if } 3 < t < 6 \\ e^{-t}\int_{t-5}^{t-2} e^{\tau/2}\, d\tau & \text{if } 6 < t < 12 \\ e^{-t}\int_{t-5}^{10} e^{\tau/2}\, d\tau & \text{if } 12 < t < 15 \\ 0 & \text{if } t > 15 \end{cases} \tag{4}$$

After performing the integrations above we yield the result listed below.

$$y(t) = \begin{cases} 0 & \text{if } t < 3 \\ 2(e^{\frac{t}{2}} - e^{\frac{3}{2}})e^{-t-1} & \text{if } 3 < t < 6 \\ (2e^{-1} - 2e^{\frac{-5}{2}})e^{\frac{-t}{2}} & \text{if } 6 < t < 12 \\ -2e^{-2t}(e^{t+5} - 1) & \text{if } 12 < t < 15 \\ 0 & \text{if } t > 15 \end{cases} \tag{5}$$

3 Results

3.1 C vs. Analytical

The comparison between the analytical solution, as seen in Eq. 5, and our C-based solution yields identical results, as shown in Fig. 1.

3.2 Implementation Time

The following results refer to Fig. 2:
When comparing the execution times of the C and Python functions, we find that the Python function takes longer to run, as expected. Additionally, it is noteworthy that as the sampling rate decreases, the implementation times increase.
Another interesting observation emerges when we change the sampling rate. As the sampling rate gets smaller the Python and C implementation times get closer together, and conversely, as the sampling rate gets larger, the implementation times get further apart.

4 Discussion and Conclusions

4.1 C vs. Analytical

Fig. 1 depicts the convolution analitical results, showcasing its alignment with C implementations. The graph clearly highlights the initial ascent as the two functions converge, followed by a decline as the shifted h(t) function begins to move away from the f(t) function.

4.2 Implementation Time

Python is expected to run slower than c because of the way each program runs, Python is an interpreted language, which is much slower than C, a compiled language. As the sampling rate decreases, the number of samples increases, so the smaller the sampling rate is the more samples we will have, thus our arrays will be larger and take more time to run through, resulting in an increased run time.

The convergence of Python and C implementation times at smaller sampling rates is likely due to the differing coding approaches. The Python implementation utilizes slicer notation and NumPy arrays, which in the case of large arrays can run nearly as efficiently as a double for loop which is used in the C implementation. This efficiency with large NumPy arrays significantly contributes to why both implementations take nearly the same amount of time to run when the sampling rate is very small.

4.3 Conclusion

In conclusion, convolution is a very helpful tool in understanding how signals behave and interact with each other. Convolutions can be determined in a variety of ways including, graphical analysis, integration, and the use of computing tools such as C or Python.

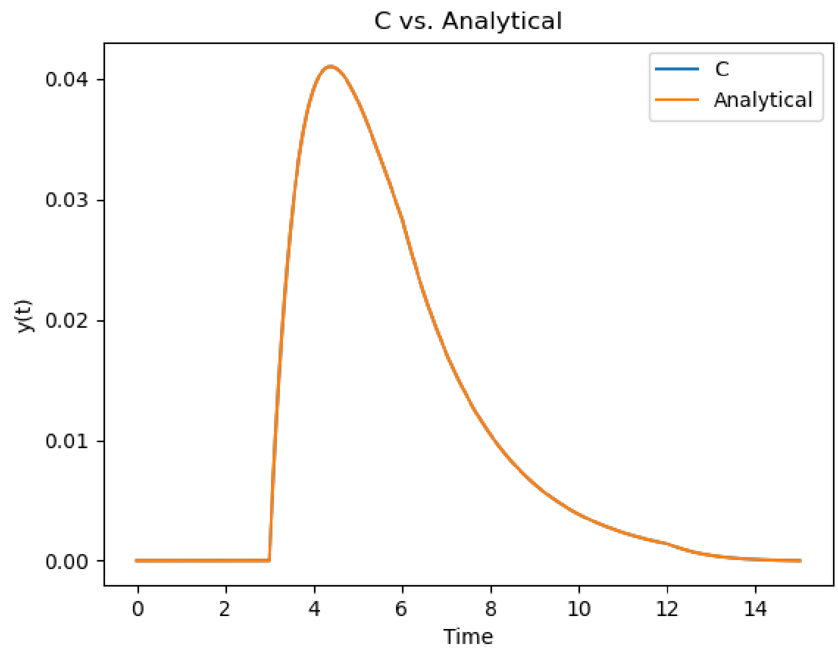

Figure 1: C vs. Analytical convolution of Eq. 1 and Eq. 2

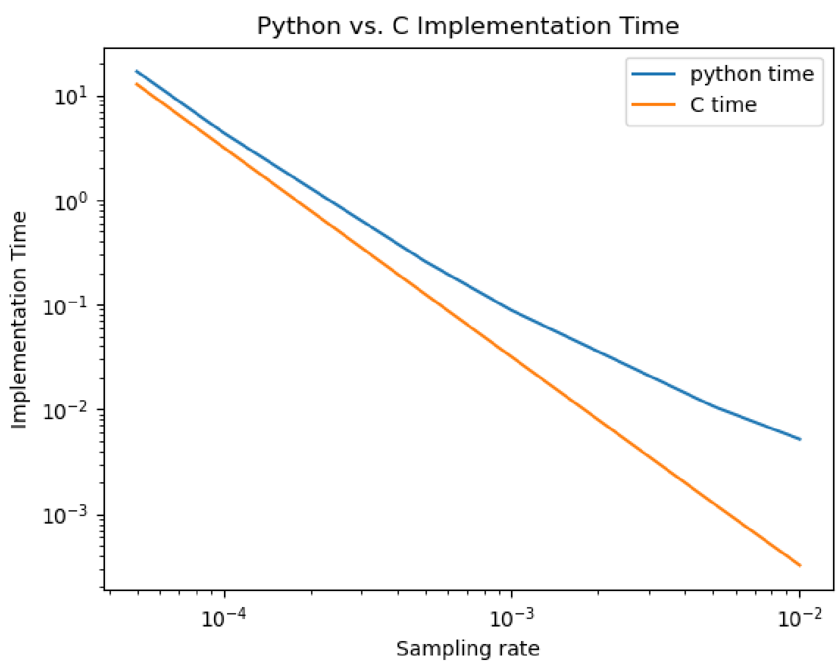

Figure 2: Implementation time comparison between C and Python function run time.