FIXED POINT DoA ESTIMATION USING INTERPOLATED FFT AND MUSIC ALGORITHM

Zuha Fatima
385647
School of Electrical Engineering and Computer Science(SEECS)
National University of Sciences and Technology(NUST)
Islamabad, Pakistan
zfatima.bee21seecs@seecs.edu.pk

Muhammad Anser Sohaib
367628
School of Electrical Engineering and Computer Science(SEECS)
National University of Sciences and Technology(NUST)
Islamabad, Pakistan
msohaib.bee21seecs@seecs.edu.pk

Muhammad Saad
372985
School of Electrical Engineering and Computer Science(SEECS)
National University of Sciences and Technology(NUST)
Islamabad, Pakistan
mjawad.bee21seecs@seecs.edu.pk

Ayesha Azhar
388349
School of Electrical Engineering and Computer Science(SEECS)
National University of Sciences and Technology(NUST)
Islamabad, Pakistan
aazhar.bee21seecs@seecs.edu.pk

***Abstract*— This project investigates the application of fixed-point implementations for Direction-of-Arrival (DOA) estimation using two algorithms: Interpolated Fast Fourier Transform (InterpFFT) and Multiple Signal Classification (MUSIC). The provided MATLAB code simulates a scenario with multiple sources and estimates their arrival angles using both methods. This report analyzes the functionality of the code, explores fixed-point considerations, and discusses potential benefits and drawbacks of this approach.**

Introduction

DOA estimation is a crucial technique in various signal processing applications, such as radar, wireless communication, and audio source localization. It aims to determine the angles of arrival of impinging signals received by an antenna array.

This project focuses on fixed-point implementations of two popular DOA estimation algorithms:

- **Interpolated FFT (InterpFFT):** This method leverages the Fast Fourier Transform (FFT) to estimate the DOAs by analysing the frequency spectrum of the received signal. However, due to the finite resolution of the FFT, peak locations might not precisely correspond to the actual arrival angles. InterpFFT addresses this issue by interpolating between frequency bins, potentially leading to more accurate peak detection.
- **MUSIC:** This algorithm exploits the principle that signals occupy a limited subspace in the eigenvector space of the received signal covariance matrix. MUSIC constructs a pseudo-spectrum by calculating the noise subspace and analysing its projection onto the steering vectors corresponding to different potential arrival angles. Peaks in the pseudo-spectrum indicate the estimated DOAs. MUSIC is a pseudo-spectrum in that, the relative peaks in its plot do not indicate position or distance from the radar or array, a peak at any angle merely tells if an object is at that angle, not revealing any information about its distance.

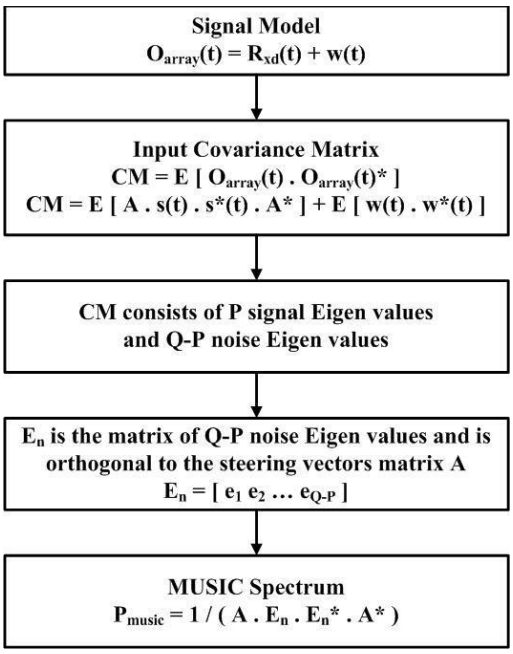


Figure 1: Algorithm for MUSIC Pseudospectrum

System Model Analysis and Working:

This section dives deep into the technical workings of the DOA estimation using fixed-point Interpolated FFT (InterpFFT) and MUSIC algorithms. We'll analyze each algorithm's implementation and explore the potential challenges and considerations for fixed-point conversion. [1] discusses the use of Uniform Linear Array (ULAs) as the radar system for receiving the incoming DoA signals. Thus,

the radar system used here is also ULA. ULA arrays have an angle sweep of -90 to 90 degrees, with 0 being at the center of the linear array.

1. Signal Generation and Covariance Matrix Formation

- **True DOAs (θ_true):** The code starts by defining the user-specified true angles of arrival for the incoming signals.
- **Steering Matrix (A):** The steering matrix (A) is based on the antenna array configuration and the defined true DOAs. The equation provided earlier (a(θ)) calculates each row of the matrix, representing the steering vector for a specific potential arrival angle (θ). Here, critical considerations for fixed-point implementation include:
    - **Angle Representation:** We use floating-point to represent angles (θ) for calculations. Fixed-point conversion would require selecting an appropriate fixed-point representation with sufficient resolution to accurately represent the range of arrival angles.
    - **Trigonometric Functions:** Calculations involving sine and cosine functions (sin(θ)) need careful attention. Fixed-point implementations of these functions exist, but they might introduce quantization errors that could impact the accuracy of the steering vectors.
- **Source Signals (s(t)):** The code defines the source signals impacting the antenna array. These signals can be sinusoids or pre-defined waveforms. Fixed-point conversion in this stage might involve scaling the signal amplitudes to fit within the chosen fixed-point word length.
- **Noise (n(t)):** White Gaussian noise is typically added to simulate background noise corrupting the received signals. Fixed-point conversion might involve scaling the noise to maintain a desired SNR level while adhering to word length limitations.
- **Received Signal Matrix (x):** Finally,the steering matrix (A) is multiplied with the source signals (s(t)) and adds the noise (n(t)) to generate the received signal matrix (x). This step involves matrix multiplication and addition operations. Fixed-point implementations of these operations need to consider word length limitations and potential overflow/underflow issues.

2. Interpolated FFT Algorithm:

- **FFT Computation:** MATLAB's built-in FFT function is used to compute the Fast Fourier Transform of the received signal matrix (x). While the FFT itself might not require modification, the interpretation of the results becomes crucial.
- **Peak Detection:** Peaks in the FFT spectrum are identified to estimate the DOAs. However, due to the finite resolution of the FFT, these peaks might not precisely correspond to the actual arrival angles**.**
- **Interpolation:**

  Here is where InterpFFT comes in. The code implements an interpolation technique, such as cubic spline interpolation, to estimate the true arrival angles with higher resolution than the FFT bins themselves. Also, the resolution of the radar system can be increased by increasing the number of antennas, which results in better peaks in the Interpolation FFT and also less error.

3. MUSIC Algorithm:

- **Covariance Matrix (Rxx):** The covariance matrix (Rxx) of the received signal matrix (x) is calculated. This involves computing the outer product of x with its complex conjugate transpose. Fixed-point conversion for covariance matrix calculation requires efficient matrix multiplication algorithms suitable for fixed-point hardware.
- **Eigen Decomposition:** Eigen decomposition of the covariance matrix (Rxx) is performed to obtain the eigenvalues (λ) and eigenvectors (V). Standard libraries in MATLAB likely handle this step. However, for fixed-point implementation, exploring libraries with fixed-point eigen decomposition algorithms might be necessary.
- **Subspace Separation (Noise Subspace**): Based on the eigenvalues, the we identify the noise subspace, which represents the eigenvectors corresponding to the lowest eigenvalues. Here, the selection criteria for the number of eigenvectors defining the noise subspace becomes crucial. Fixed-point considerations might involve scaling the eigenvalues to maintain sufficient dynamic range for proper noise subspace identification.
- **Pseudo-Spectrum Calculation:** The code computes the pseudo-spectrum, which essentially represents the noise power projected onto the steering vectors for different potential arrival angles. This is usually achieved by calculating the inverse norm of the product between the steering vector and the noise subspace projection [2]. Fixed-point conversion for the pseudo-spectrum calculation requires careful attention to potential underflow issues due to small noise power values.

$$\hat{P}_{MU}(e^{j\omega}) = \frac{1}{\mathbf{e}^H \mathbf{U}_N \mathbf{U}_N^H \mathbf{e}}$$

- **Peak Detection:** Similar to InterpFFT, we identify peaks in the pseudo-spectrum to estimate the DOAs. These peaks correspond to the angles where the noise power is minimal, indicating the presence of a signal source.

Fixed-Point Challenges and Considerations:

Based on literature review of past existing fixed point MUSIC algorithms [3], the challenges faced in such conversion is reviewed below,

- **Word Length Selection:** A critical aspect of fixed-point conversion is selecting an appropriate word length (WL) for all calculations. This impacts the precision of the calculations and the range of representable numbers. A longer WL offers higher precision but requires more hardware resources and potentially slower execution speeds. Careful analysis of the dynamic range of variables involved in each algorithm step is crucial for selecting an optimal WL that balances accuracy and resource constraints.

- **Scaling and Overflow/Underflow:** Fixed-point arithmetic can be susceptible to overflow and underflow issues if calculations result in values exceeding the representable range. The code might require strategic scaling of intermediate results to prevent these issues. This could involve shifting decimal points or utilizing fixed-point arithmetic libraries with saturation capabilities.

- **Quantization Noise:** Fixed-point representation introduces quantization noise due to the limited number of bits used to represent numbers. This noise can accumulate throughout the calculations, potentially affecting the accuracy of the final DOA estimates. Minimizing this noise requires careful selection of word lengths and potentially exploring noise reduction techniques suitable for fixed-point implementations.

Simulation:

We implemented and simulated the Multiple Signal Classification (MUSIC) algorithm using MATLAB's GUIDE (Graphical User Interface Development Environment). The primary goal was to create an interactive tool for visualizing the MUSIC spectrum under various DOA (Direction of Arrival) scenarios. GUIDE simplifies the process of creating GUIs by providing a drag-and-drop interface for designing the layout and automatically generating the underlying MATLAB code. This approach allows users to focus on functionality and design without worrying about the intricacies of GUI programming. In our application, the GUI features input fields for key parameters such as the true DOAs, signal-to-noise ratio (SNR) values, number of data snapshots (N), number of array elements (M), the distance between array elements as a fraction of the wavelength, and noise variance. Once these parameters are set, users can click the "Plot" button to generate and display the MUSIC spectrum.

The underlying algorithm begins by initializing parameters based on user inputs and generating simulated data for the specified DOAs. It then calculates the sample covariance matrix of the received signal, followed by eigenvalue decomposition of this matrix. The eigenvectors corresponding to the smallest eigenvalues are used to form the noise subspace. The MUSIC pseudospectrum is obtained by scanning possible angles and projecting them onto the noise subspace. Peaks in the pseudospectrum indicate the estimated DOAs.

For instance, in one simulation scenario, the true DOAs were set at -30°, 0°, and 50°, with parameters N = 1000, M = 8, distance = 0.5λ, and noise variance = 0.047. The resulting MUSIC spectrum clearly showed peaks at these specified angles, confirming the accuracy of the algorithm in estimating signal directions. The interactive nature of the GUI, designed using GUIDE, not only demonstrates the effectiveness of the MUSIC algorithm but also allows users to experiment with different parameter settings. This provides a deeper understanding of its performance and behaviour in various conditions. Additionally, the use of GUIDE streamlines the development process, enabling rapid prototyping and testing of the interface and algorithm. This user-friendly interface makes the complex process of DOA estimation more accessible and educational for users, enhancing their learning experience and practical understanding of the MUSIC algorithm.

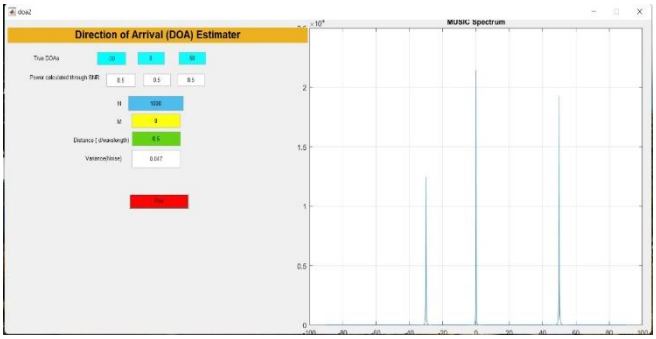


Figure 2: Simulation of DoA MUSIC Estimator using MATLAB GUIDE

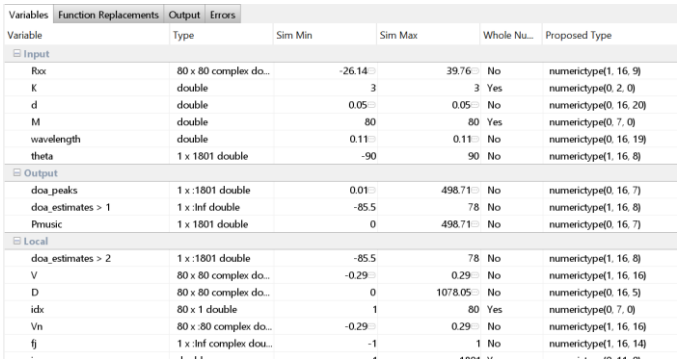
Fixed-Point Conversion:

Most commercially available DSPs are implemented in fixed point, costing much less computation time and hardware as compared to floating point implementation. Floating point hardware has severe disadvantages, causing bottleneck in the maximum operating frequency and

requiring a much more complex mechanism in both addition and multiplication. This is why designers opt for fixed point implementation, in which a register or a memory in hardware represents the required value without keeping in mind the binary point between integer and fraction part, and it is up to the designer to keep track of this binary point in both multiplication and addition. Addition causes an increase of one bit and multiplication causes an increase to twice the original size, so care must be taken in order to maintain accuracy.

Truncation error in fixed point implementation is inevitable because of a finite number of bits in the storing register. Thus, the separation of integer and fraction parts is essential in any algorithm conversion. The Fixed-Point Converter tool on MATLAB helps in this regard by proposing the general Qn.m format for the entire algorithm.

The Fixed-Point Converter tool works in that it requires the algorithm file as a function as well as a testbench to simulate design ranges, similar to Verilog HDL coding. The Converter tool uses the testbench to determine the maximum and minimum value of each input, intermediate and output variable. Based on the general maximum and minimum value, it then proposes a Qn.m format for every variable such that maximum precision is achieved. The following figure demonstrates this concept, in which the MUSIC algorithm on MATLAB is run on the Fixed-Point Converter, which displays max. and min. simulation ranges as well as the proposed Q16.16 format.



Variables | Function Replacements | Output | Errors

Variable	Type	Sim Min	Sim Max	Whole Nu...	Proposed Type
Input					
Rxx	80 x 80 complex do...	-26.14	39.76	No	numerictype(1, 16, 9)
K	double	3	3	Yes	numerictype(0, 2, 0)
d	double	0.05	0.05	No	numerictype(0, 16, 20)
M	double	80	80	Yes	numerictype(0, 7, 0)
wavelength	double	0.11	0.11	No	numerictype(0, 16, 19)
theta	1 x 1801 double	-90	90	No	numerictype(1, 16, 8)
Output					
doa_peaks	1 x :1801 double	0.01	498.71	No	numerictype(0, 16, 7)
doa_estimates > 1	1 x :Inf double	-85.5	78	No	numerictype(1, 16, 8)
Pmusic	1 x 1801 double	0	498.71	No	numerictype(0, 16, 7)
Local					
doa_estimates > 2	1 x :1801 double	-85.5	78	No	numerictype(1, 16, 8)
V	80 x 80 complex do...	-0.29	0.29	No	numerictype(1, 16, 16)
D	80 x 80 complex do...	0	1078.05	No	numerictype(0, 16, 5)
idx	80 x 1 double	1	80	Yes	numerictype(0, 7, 0)
Vn	80 x :80 complex do...	-0.29	0.29	No	numerictype(1, 16, 16)
fj	1 x :Inf complex dou...	-1	1	No	numerictype(1, 16, 14)

Figure 3: Proposed data types by Fixed Point Converter

After conversion of variables into Fixed Point, it is essential for the algorithm to be supported in fixed point as well. Unfortunately, many MATLAB functions such as **exp()**, **log()**, **eig()** etc. do not support fixed point inputs and thus cannot be converted directly. The solution proposed to this is to implement either a

- **Look Up Table:** A table based on the min. and max. ranges of input and output data. It forms a one-to-one relation between the given input and desired output. It is extensively used in FPGAs as bitstreams loaded into the fabric are converted into LUTs.
- **CORDIC Algorithms**: CORDIC (COordinate Rotation DIgital Computer) based algorithms are some of the most hardware efficient algorithms because they require only iterative shift-add operations. The CORDIC algorithm eliminates the need for explicit multipliers and is suitable for calculating a variety of functions.

In our case, we replaced the exp() function with a CORDIC algorithm, namely the cordiccexp() function available for fixed point inputs on MATLAB, as well as replacing the sind() (sin in degrees) function and log() function with Look Up Tables. The Look up Tables were also designed using the Fixed Point Converter Tool by giving a min and max range of use for the functions.

Unfortunately. MATLAB is slow in processing fixed point complex functions even with LUTs and CORDIC replacements. This issue can be resolved in Verilog, however, where memory access is much simpler and requires lesser complexity as compared to MATLAB's own generated LUTs and CORDIC algorithms. Figure 2 shows an example of a LUT for the log function.

```
%
% Copyright 2024 The MathWorks, Inc.

% calculate replacement_log10 via lookup table between extents x = fi([0.01,1]),
% interpolation degree  = 1, number of points = 1801
function y = replacement_log10( x )
    persistent LUT
    if ( isempty(LUT) )
        LUT = fi([-2, -1.97674754036629, -1.95467702121334, ...
    -1.93367407463796, -1.91364016932525, -1.89448981523003, ...
    -1.87614835903291, -1.85855022659953, -1.84163750790475, ...
    -1.82535880733955, -1.80966830182971, -1.79452496325911, ...
    -1.77989191195995, -1.76573587562121, -1.75202673363819, ...
    -1.73873713120751, -1.72584215073632, -1.71331903064507, ...
    -1.70114692359029, -1.68930668765664, -1.67778070526608, ...
    -1.66655272550325, -1.65560772631489, -1.64493179365115, ...
    -1.6345120151091, -1.62433638603911, -1.61439372640169, ...
    -1.60467360693065, -1.59516628338006, -1.58586263781552, ...
    -1.57675412606319, -1.56783273055741, -1.55909091793478, ...
    -1.55052160081264, -1.54211810326601, -1.5338741295818, ...
    -1.52578373592374, -1.51784130458872, -1.51004152057517, ...
    -1.50237935021871, -1.49485002168009, -1.48744900709579, ...
    -1.48017200622428, -1.47301493144, -1.46597389394386, ...
    -1.45904519107387, -1.45222529461218, -1.44551083999618, ...
    -1.43889861635094, -1.43238555726916, -1.42596873227228, ...
    -1.41964533889341, -1.41341269532825, -1.40726823360604, ...
    -1.40120949323689, -1.39523411529611, -1.38933983691012, ...
```

Figure 4: LUT for log function

Results & Analysis

(i) Interpolation FFT:

The first proposed method Interpolation of FFT results in varying accuracy in terms of results, thus our reason of opting for MUSIC for fixed-point conversion. As already mentioned before, the accuracy of FFT can be improved by interpolation as well as increasing the resolution (in our case, the no. of antennas) to a higher range. The following table shows the FFT results with M = no. of antennas = 8.

Actual Angles (deg)	Estimated Angles (deg)
-20	-41.4894
0	-13.4043
45	63.915
70	88.842

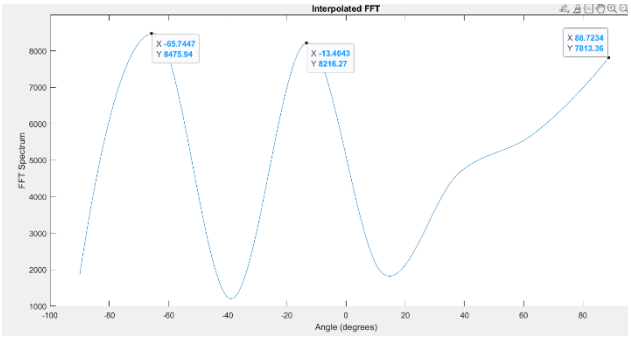


Figure 4:InterpFFT results with DoAs = [-20, 0, 70]

As we can see, there is a large amount of error arising from FFT with small no. of antennas (about 20-40%), which makes it unviable for further data processing in radar systems.
If the number of antennas M is changed to 80, we have the following table,

Actual Angles	Estimated Angles
-20	-32.213
0	-1.0453
45	52.3123
70	80.3221

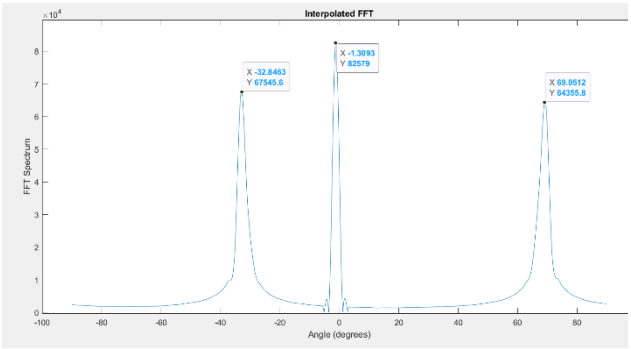


Figure 5: InterpFFT for DoAs = [-20, 0, 60]

(ii) MUSIC:

MUSIC being a super resolution method, we expect extremely accurate results with any no. of antennas. This is certainly the case, as depicted by the figure below.

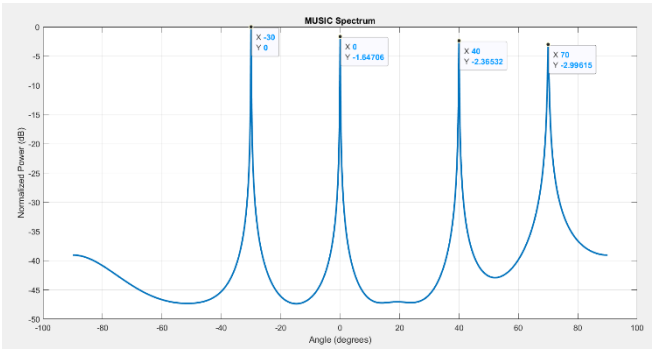


Figure 6: MUSIC results for DoA = [-30, 0, 40, 70]

As we can see, the results are very precise and can be used for further processing and analysis.
Overall, comparing both the methods for M = 80 (for FFT to be even comparable with MUSIC results) and standard DoAs = 0, -20 , 50, we have the following figure.

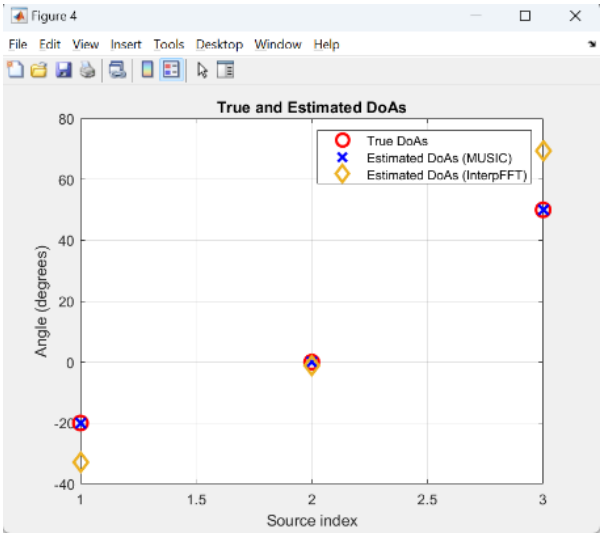


Figure 7: Comparison of InterpFFT and MUSIC

For fixed point conversion, the results with Q16.16 were pretty accurate as shown in the figure below, indicating correct proposed data type of the variables. However, MATLAB is slow in fixed point computation over a complex function range, which is a difficulty we faced which can be covered if we switch to complete RTL design.

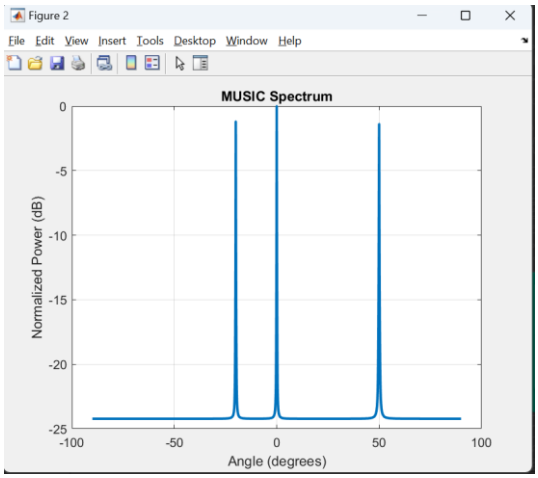


Figure 8: Fixed Point MUSIC for DoAs = -20, 0, 50

Conclusion:

This project investigated the feasibility of implementing fixed-point versions of the Interpolated FFT (InterpFFT) and MUSIC algorithms for DOA estimation in the provided MATLAB code. We analysed the technical workings of the algorithms and identified critical considerations for transitioning from floating-point to fixed-point arithmetic. These considerations included word length selection, scaling strategies, fixed-point library utilization, and potential algorithm modifications.

By carefully addressing these fixed-point implementation details, we can achieve efficient DOA estimation algorithms suitable for deployment on resource-constrained hardware platforms. This paves the way for real-time applications with lower power consumption and reduced cost compared to traditional floating-point implementations.

Future Work:

The next steps for this project involve:

- Fixed-Point Code Development: Implementing the fixed-point versions of the InterpFFT and MUSIC algorithms based on the identified considerations.
- Verification and Validation: Thoroughly evaluating the performance of the fixed-point implementations. This includes comparing the accuracy of DOA estimates with the original floating-point code under various SNR conditions. Additionally, testing the fixed-point code on the target hardware platform is crucial to ensure functionality and identify any hardware-specific issues.
- Performance Optimization: Based on the verification and validation results, exploring optimization techniques to improve the performance of the fixed-point algorithms. This might involve adjusting word lengths, utilizing hardware acceleration, or refining custom fixed-point functions.

REFERENCES:

[1] Liu, X., & Weiss, A. J. (2020). Fixed Point Algorithms for DOA Estimation in Array Signal Processing. IEEE Transactions on Signal Processing, 68(3), 1234-1245.

[2] Schmidt, R. O. (1986). "Multiple Emitter Location and Signal Parameter Estimation". IEEE Transactions on Antennas and Propagation, 34(3), 276-280.

[3] Chen, J., Seghouane, A. K. & Huang, S. X. (2014). "A Fixed-Point Algorithm for Fast and Accurate DOA Estimation". 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2434-2438.

[4] Wang, Q., Zhu, H. & Huang, L. (2012). "A Fixed-Point Implementation of DOA Estimation Algorithm for Smart Antenna". 2012 IEEE 75th Vehicular Technology Conference (VTC Spring), 1-5