Dark Ship Detection - Introduction Document

Violoni Gianluca

March 13, 2025

1 Introduction

This document provides a brief explanation of the formulas used and the structure of the code.

2 Used formulas

2.1 Distance [m]

The Euclidean distance between a hydrophone and a ship is calculated as:

$$\Delta x = x_1 - x_0 \tag{1}$$

$$\Delta y = y_1 - y_0 \tag{2}$$

$$distance = \sqrt{\Delta x^2 + \Delta y^2} \tag{3}$$

2.2 Pressure Attenuation [dB]

The pressure attenuation was computed as a function of distance using the following equation:

$$attenuation = 20 * \log_{10}(distance) \tag{4}$$

2.3 Received Pressure Level [dB re 1 µPa]

The received acoustic pressure level at the hydrophone is computed by subtracting the attenuation from the ship's source level:

$$P_{received\_dB} = P_{source\_dB} - attenuation \tag{5}$$

2.4 Linear Pressure [µPa]

To convert the received pressure level from decibels to linear units (required for summing contributions from multiple ships):

$$P_{linear} = 10^{P_{received\_dB}/20} \tag{6}$$

2.5 Decibel Pressure [dB re 1 µPa]

To convert the linear noise to decibels, the following equation is used:

$$P_{[dB]} = 10 * \log_{10}(P_{linear}) \tag{7}$$

2.6 Distance from Pressure [m]

To calculate the distance from a hydrophone to a ship based on the received pressure, the following equation is used:

$$distance = 10^{\frac{P_{\text{source\_dB}} - P_{\text{received\_dB}}}{20}} - 1 * 10^{-9} \tag{8}$$

3 Project structure

The project is organized into the following directories and files:

3.1 Root Directory

The root directory contains the following files and folders:

- `config/` – Configuration files to run the simulation with predefined settings
- `requirements.txt` – Dependencies of the project
- `src/` – Source code of the project

3.2 Source Directory: `src/`

The `src/` directory contains the following files:

- `main.py` – Main entry point for running the simulation
- `core.py` – Main detection functions (currently contains 1 function)

- `utils.py` – Utility functions for the project
- `acoustic_calculator.py` – Contains all acoustic-related physical formula functions
- `simulation.py` – Contains Hydrophone and Ship classes and all the function to run and plot the simulation (will be refactored into multiple files)

4 Simulation

Class `Hydrophone`

Description: Represents an underwater acoustic sensor.
Attributes:

- `id`: Unique identifier for the hydrophone.
- `x`, `y`: Coordinates of the hydrophone.
- `observed_pressure`: Measured acoustic pressure (dB re 1μPa).
- `expected_pressure`: Predicted acoustic pressure (from AIS data).

Class `Ship`

Description: Represents a vessel with acoustic properties.
Attributes:

- `id`: Unique identifier for the ship.
- `x`, `y`: Coordinates of the ship.
- `speed`: Speed of the ship (in knots).
- `is_dark`: Indicates if the ship is not transmitting AIS.
- `base_pressure`: Acoustic pressure at 1m distance (modified based on speed).

Class `SimulationManager`

Description: Manages the simulation environment, including configuration loading and handling ships and hydrophones.

Main Functions:

- `initialize_environment`: Sets up the simulation environment by creating hydrophones and ships (both manual and random) reading from the config file and calculating acoustic pressures.
- `estimate_ds_positions`: Estimates the position of "Dark Ships" using the functions in the core module.
- `plot_environment`: Plots ships and hydrophones on a map.
- `plot_simulation`: Displays the simulation using matplotlib.

5 Acoustic Calculator

Description: The class provides methods to convert between dB and linear pressure scales, calculate attenuation, and compute pressures and distances in an acoustic environment.

Methods

`db_to_linear(pressure_db)`

Description: impl of 6

`linear_to_db(pressure_linear)`

Description: impl of 7

`calculate_attenuation(distance)`

Description: impl of 4.

`calculate_linear_pressure(hydro, ship)`

Description: Calculates the linear pressure received by a hydrophone from a ship. **Procedure**:

- Calculate the distance between the hydrophone and the ship, using 3.

- Calculate the attenuation based on the distance, using the previous method.
- Subtract the attenuation from the ship's base pressure.
- Convert the resulting pressure from dB to linear scale (µPa).

`calculate_pressures(hydrophones, ships, config, include_base_noise_level = False)`

Description: Calculates expected and observed pressures for all hydrophones.
Procedure:

- For each hydrophone, calculate the total observed and expected pressures from all ships.
- Convert the pressures to dB re 1 µPa.
- Optionally, add random noise to the observed pressure.

`calculate_distance_from_pressure(hydro_pressure, ship_base_pressure)`

Description: impl of 8.

`compute_pressure_delta(hydro)`

Description: Computes the difference between the observed and expected acoustic pressure at a hydrophone.

6 Core

The current implementation focuses on a multilateralization (MLAT) algorithm.

`mlat(hydrophones)`

Description: Estimates the position and source level of a dark ship minimizing the error between observed and estimated pressure values to determine the most likely position and acoustic signature of the dark ship.

Procedure: The localization process involves the following steps:

1. A loss function computes the total squared error between observed and estimated pressure readings across all hydrophones.

2. For each hydrophone, the algorithm:
    - Calculates the distance between the hypothesized dark ship position and the hydrophone
    - Computes the expected acoustic pressure from the dark ship at that distance.
    - Converts both the expected dark ship pressure and the hydrophone's expected pressure (from known ships) from decibels to linear scale
    - Combines these pressures in linear scale to obtain the total estimated pressure
    - Converts the combined pressure back to decibel scale
    - Calculates the squared error between this estimated total pressure and the actually observed pressure
3. The algorithm returns the optimized position coordinates and acoustic source level.

Limitations:

- The algorithm requires at least three hydrophones for reliable triangulation.
- It assumes the presence of only one dark ship; in case of more than one dark ship the results will be inaccurate.

```python
def mlat(hydrophones):
    """
    Estimate the position of the ship using triangulation based on hydrophone pressure differences.

    :param hydrophones: List of hydrophone objects with observed and expected acoustic pressure properties.
    :return: Estimated (x, y) position of the ship and estimated base pressure.
    """

    def loss_function(params):
        """Calculate error between estimated and observed pressure deltas."""
        ship_x, ship_y, ship_pressure = params
```

```
            ship_pos = Position(ship_x, ship_y)

            total_error = 0

            for hydro in hydrophones:
                # Calculate distance from ship to hydrophone
                distance = Position.distance(ship_pos, hydro)

                # Compute expected pressure delta using the inverse model
                darkship_observed_pressure = ship_pressure - AcousticCalculator.calculate_attenuation(distance)

                # Computing the observed value as the sum of the expected value and the darkship observed pressure
                darkship_observed_linear = AcousticCalculator.db_to_linear(darkship_observed_pressure)
                hydro_expected_linear = AcousticCalculator.db_to_linear(hydro.expected_pressure)

                new_observation = AcousticCalculator.linear_to_db(darkship_observed_linear + hydro_expected_linear)

                # Compute squared error
                total_error += (new_observation - hydro.observed_pressure) ** 2

            return total_error

        # Initial guess (centered in the middle of hydrophones)
        x0 = np.mean([h.x for h in hydrophones])
        y0 = np.mean([h.y for h in hydrophones])

        DEFAULT_PRESSURE = 150

```

```
        # Optimize (x, y) position and base acoustic
           pressure of the ship
        result = minimize(loss_function, [x0, y0,
           DEFAULT_PRESSURE], method='Nelder-Mead')

        # Return estimated ship position and its
           estimated base pressure

        return result.x[:2], result.x[2]
```