Doppler Localization Algorithm

By Bas Calders -- Version 1.5

The Doppler Localization algorithm is designed to estimate the position of a UE based on the observed Doppler shifts of signals emitted from satellites. This version uses Iridium satellites as a testing set with global coverage.

Dependencies

- MATLAB (R2020a or later)
- Satellite Communications Toolbox

Features

- Uses Iridium satellite constellation as a test case for global coverage
- Visualizes estimated positions during iterations
- Includes a satellite scenario for visualization

Usage

1. Clone or download the repository.
2. Open MATLAB and navigate to the directory containing the algorithm.
3. Run the main script (`workingScript.m`).
4. Observe the output, including the initial estimated position, position estimations after each iteration, and the final estimated position.
5. Visualize the satellite scenario if desired.

Notes

- The simulation is set to run for 10 minutes, but can be adjusted by modifying the `simTime` variable.
- The start time of the simulation is set to April 16th, 2023 at 16:05:33, but can be adjusted by modifying the `startTime` variable.

- The initial position estimation is set to Rome for added difficulty.
- Starlink satellites are also available in the repository but note that using them may result in longer simulation times.

Main Loop

1. Initialize variables and set up the satellite scenario.
2. Set up a loop to iterate through each time step in the simulation.
    - For each time step:
        1. Initialize variables for the focused satellite and the current satellite.
        2. Loop through all satellites.
            - If the satellite is in view:
                1. Determine the satellite's position and velocity at the current time.
                2. Calculate the relative velocity between the satellite and ground station for the current and previous time steps.
                3. Calculate the acceleration of the satellite.
                4. Calculate the observed Doppler shift based on the relative velocity.
                5. Calculate range vector, unit vector (e), rho, rhoDot, and rhoDotDot.
                6. Calculate eDot and append everything to the matrix H.
                7. Calculate the predicted Doppler shift, deltaDoppler, and deltaD, then append deltaD to the deltaDMatrix.
                8. Increment the focused satellite counter.
        3. Calculate deltaX by inverting matrix H and multiplying it by deltaDMatrix.
        4. Update the estimated state with deltaX.
        5. Store the estimated state for the current time step.
        6. Convert the estimated state's ECEF coordinates to LLA coordinates and store them.
        7. Display the estimated position for the current time step.
3. Once the loop is completed, display the final results and plot the estimated positions.

Functions

- `speed2Dop` : Converts the emitted frequency and relative velocity into the observed Doppler shift.
- `dop2Speed` : Converts the emitted frequency and observed Doppler shift into the relative velocity.
- `calcRelVel` : Calculates the relative velocity between the satellite and ground station.

Conclusion

Overall, this code implements a simple algorithm for estimating the position of a receiver using Doppler shift measurements from one or more satellites. However, the accuracy and stability of the solution may be limited by the assumptions and simplifications used in the model, as well as the quality of the measurements themselves.