ACOUSTOELECTRIC README

Herein lies code to support the figures and findings in the Acoustoelectric article. The data that goes with the code in this repository is located in Figshare and referenced in this readme. Before running the code, you will need to download the data files and place them in the folders that run the code that pertains to them.

Screenshots of the expected outputs are provided, as are Python, Matplotlib, Scipy and Numpy versions used.

To see all the files uploaded for this project see DOI: 10.6084/m9.figshare.c.6365098

Installation Note

```
Python 3.7.9
```

To run the code supplied with the figures, Python and associated scientific computing libraries will need to be installed according to:

```
pip install -r requirements.txt
```

Homogeneous Acoustoelectric Simulation

Fourier(k-space) Poisson solution for the Acoustoelectric effect. Please see the article supplementary section for the full derivation, with the simulation rationale described in methods. To solve the homogeneous solution of the equation:

$$\nabla \cdot \vec{E}_{ae} = -k\nabla P \cdot \vec{E}_0$$

we consider the Laplacian of the potential (since $\vec{E}_{ae} = -\nabla\phi_{ae}$) and consider the right-hand side as a source term for the Poisson diffusion equation:

$$\Delta \cdot (\phi_{ae}) = \nabla(\beta P) \cdot (\vec{E}_0)$$

1. Generate the acoustic field in matlab using focused.m. This will save out a two-dimensional transducer data file such as transducer_0.5MHz.mat. This implementation was written by Prof. Robin Cleveland.
2. Run python pressure_3Diser.py with the file that was generated above. This will put the 2D result into a three-dimensional grid, utilizing the axial symmetry.
3. Run python kspace_solver.py to create all the ae components. A resultant NPZ file will be created which contains final acoustoelectric fields as well as intermediary results.
4. To view the simulation, use the data viewing gui provided in Fig1and2-simulation_viewer_tool.
5. If you wish to plot the phasors, run the plot_phi_phasors.py file over the simulation output file.

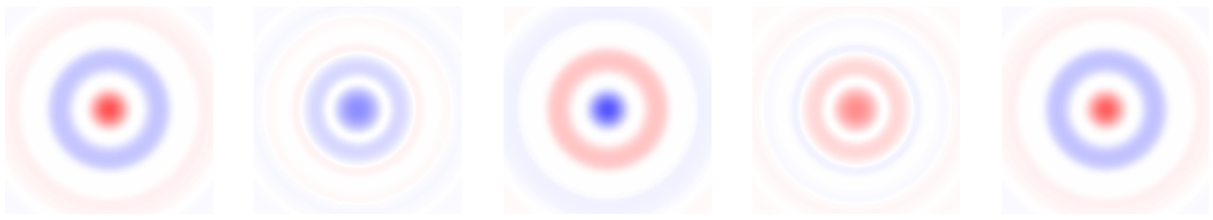


Figure 1 and 2: Simulation Results

Data to run with this code is included here: DOI: 10.6084/m9.figshare.21802165

```
python plot_viewer.py
```

Will plot the static 2D field, giving you a scroll bar to search through the 3rd dimension in either XY or XZ views. The viewer shows every variable in the equation simultaneously, to assist in understanding how the acoustoelectrically generated field changes. i.e. the reader may wish to see how the X,Y,Z acoustoelectric field components trend with pressure, or how the potential ϕ_{ae} changes with the gradient of pressure term.

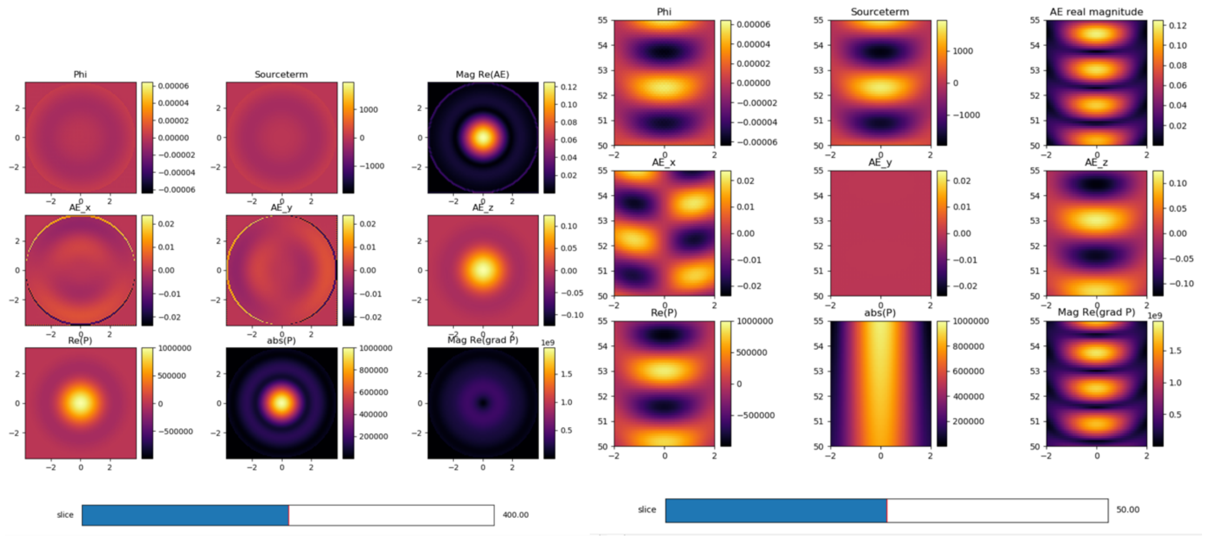


Figure 3: Acoustoelectric frequency mixing in phantom

Data to run with this code is included here: DOI: 10.6084/m9.figshare.21802171

```
python plot_fmixing_df492khz.py
```

Note: You should ensure the filename matches the file you downloaded.

An example python script is provided which reads in the supplied data recordings, and makes the plots which are contained in Figure 3 of the article.

Below are some of the outputs of the scripts, showing the frequency mixing nature of the generated acoustoelectric field, the first where the applied electric field is 8khz, the second where the applied electric field is at 499.99khz.

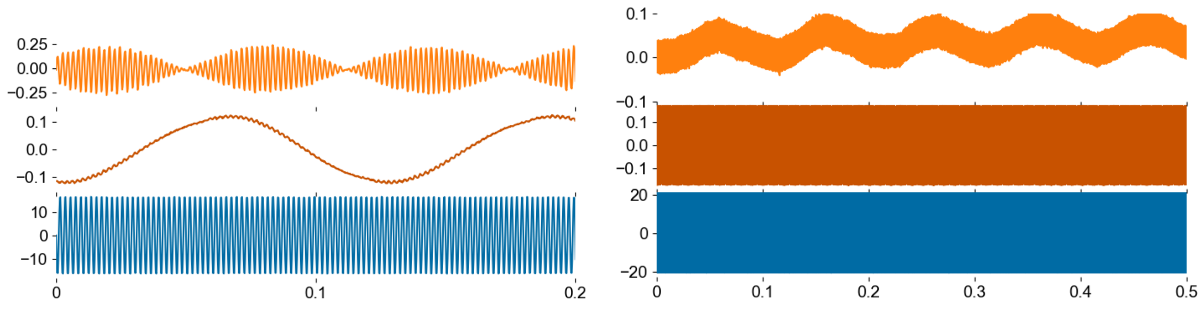


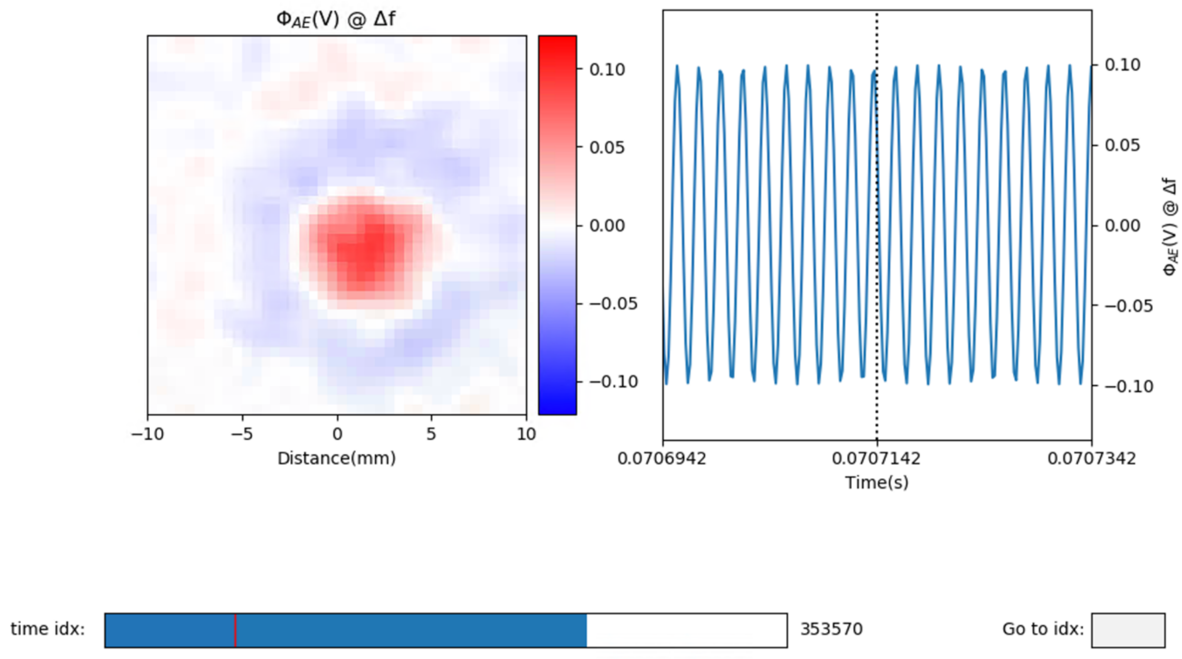
Figure 4: Acoustoelectric 2D maps in phantom

Data to run with this code is included here: DOI: 10.6084/m9.figshare.21802183

```
python time_scroller_xy_2d.py
```

The python viewing tools contained here enable the reader to scroll through the time series potential data of the generated acoustoelectric field, and observe how it changes with time, as well as how it changes based on the angle between the applied electric field and pressure field. There are a few different tools provided to explore the large files that are provided.

You can use the arrow keys to scroll through the data, or the slider bar, or enter a specific index in the file you are interested in. This provides a more intuitive way to understand how the acoustoelectric field, based on experimental phantom measurements, changes with time.



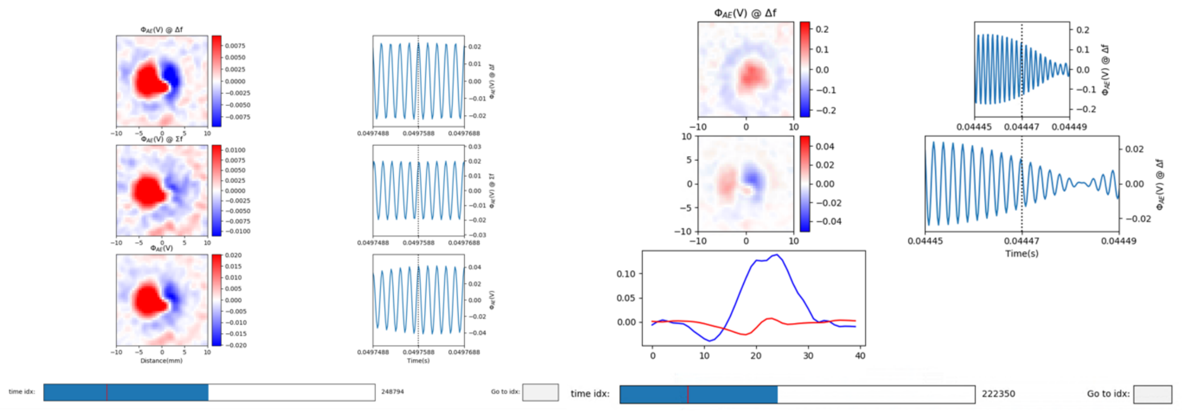


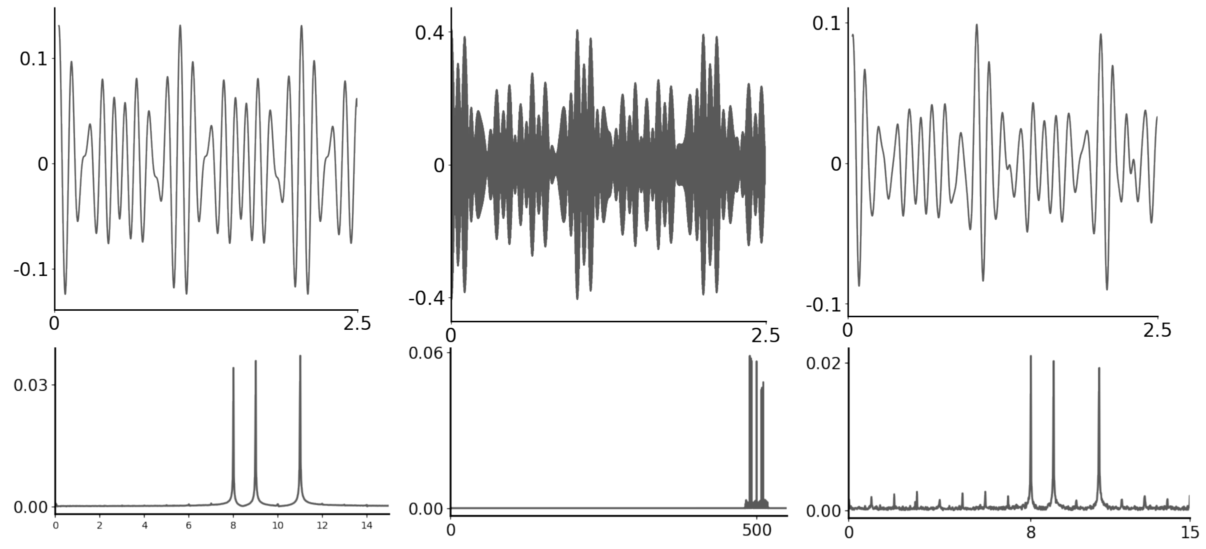
Figure 5: Acoustoelectric demodulation and signal recovery in phantom

Data to run with this code is included here: DOI: 10.6084/m9.figshare.21802174

```
python demodulation.py
```

Note: ensure the filename referenced in the code matches the one that you downloaded.

This code provides a data set where a complex ionic signal is created in the phantom (8khz + 9khz + 11khz), is upmodulated via the applied ultrasound, and then decoded using IQ demodulation. The code is supplied for this, as is the calculation of the cross-correlation metric.



License