NASA Space Apps – Seismic Detection Across the Solar System

NASA International Space Apps Challenge

Team: OTU Innovators

Members: Yaseen Rehman, Aaron Emanuel

Ontario Tech University

Join us on a journey to uncover the secrets of the universe as we develop a groundbreaking Seismic Detection System to explore the hidden rhythms of the Moon and Mars!

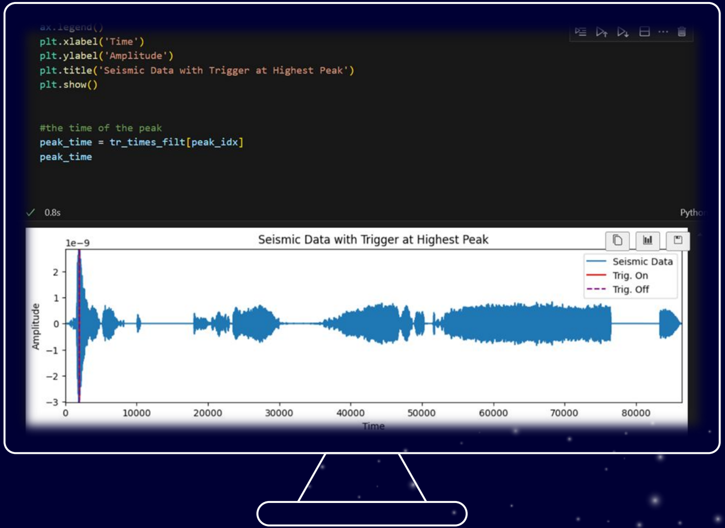
Our project focuses on creating a Seismic Detection System that analyzes seismic data from lunar and Martian environments, employing advanced data processing techniques to detect and visualize seismic events, thereby enhancing our understanding of extraterrestrial geological processes.

Python Code Breakdown

1. Import Libraries
pandas, numpy, obspy, datetime, matplotlib, os

THESE STEPS FOLLOW AFTER TRAINING TEST MODEL

2. Read Test Data .CSV
pandas to read file either Lunar or Mars



3. Extract Plot from Data
Using Relative Time and Velocity

4. Filter the Trace
Filter data using a bandpass filter between 0.01Hz to 0.5Hz

5. STA/LTA Detection Algorithm
Obtain and plot the characteristic function

6. Trigger Seismic Data at Peak
This will indicate when the seismic event has occurred

7. Save Data in .CSV

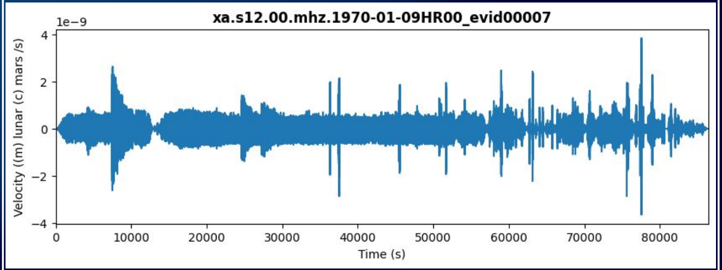
01-03 Read AND plot Test Data .CSV

We made it so you can read both Lunar and Mars and also automatically update directory path

Lunar uses time_rel(sec) and velocity(c/s) whereas Mars uses rel_time(sec) and velocity (c/s). This automatically figures which one to call

```python
# Define the file path (THIS IS WHERE YOU CHOOSE WHICH DATA SET YOU WOULD LIKE TO USE, AND THE REST WILL AUTOMATICALLY UPDATE)
cat_file = r'C: ......'

# Extract the directory path
directory_path = os.path.dirname(cat_file)

# Extract the file name from the path and remove the .csv extension
file_name_with_extension = os.path.basename(cat_file)
file_name, _ = os.path.splitext(file_name_with_extension)  # Split the name and extension

# Display the extracted directory path and filename without the .csv extension
print("Directory Path:", directory_path)
print("File Name (without extension):", file_name)

# Read the CSV file
cat = pd.read_csv(cat_file)
print(cat)
data_cat = pd.read_csv(cat_file)
```

```python
if 'time_rel(sec)' in data_cat.columns and 'velocity(m/s)' in data_cat.columns:
    csv_times = np.array(data_cat['time_rel(sec)'].tolist())
    csv_data = np.array(data_cat['velocity(m/s)'].tolist())
elif 'rel_time(sec)' in data_cat.columns and 'velocity(c/s)' in data_cat.columns:
    csv_times = np.array(data_cat['rel_time(sec)'].tolist())
    csv_data = np.array(data_cat['velocity(c/s)'].tolist())


# Output the results
print("CSV Times:", csv_times)
print("CSV Data:", csv_data)


# Plot the trace!
fig,ax = plt.subplots(1,1,figsize=(10,3))
ax.plot(csv_times,csv_data) #


# Make the plot pretty
ax.set_xlim([min(csv_times),max(csv_times)])
ax.set_ylabel('Velocity ((m) lunar (c) mars /s)')
ax.set_xlabel('Time (s)')
ax.set_title(f'{file_name}', fontweight='bold')
```

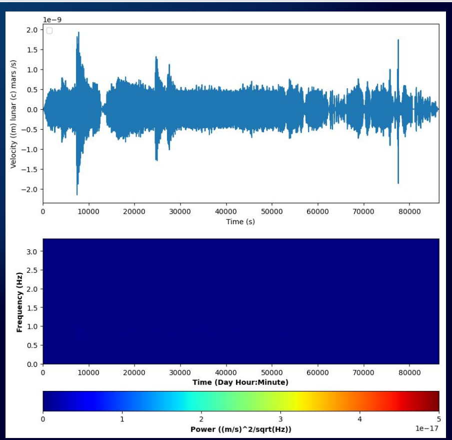


Example Shown Uses Lunar data set:
\S12_GradeB\xa.s12.00.mhz.1970-01-09HR00_evid00007.csv

04 Filter the Trace

```python
# Assuming 'st' is the stream read from a file, define it first
st = read(os.path.join(directory_path, file_name + '.mseed'))
# Set the minimum frequency
minfreq = 0.5
maxfreq = 1.0
# Going to create a separate trace for the filter data
st_filt = st.copy()
st_filt.filter('bandpass', freqmin=minfreq, freqmax=maxfreq)
tr_filt = st_filt.traces[0].copy()
tr_times_filt = tr_filt.times()
tr_data_filt = tr_filt.data

from matplotlib import cm
from scipy.signal import spectrogram
# Plot the time series and spectrogram
fig = plt.figure(figsize=(10, 10))
ax = plt.subplot(2, 1, 1)
# Plot trace
ax.plot(tr_times_filt,tr_data_filt)
# Mark detection
#ax.axvline(x = arrival_time_rel, color='red',label='Detection')
# Compute the spectrogram
f, t, sxx = spectrogram(tr_data_filt, fs=1/(tr_times_filt[1] - tr_times_filt[0]))
ax.legend(loc='upper left')
# Make the plot pretty
ax.set_xlim([min(tr_times_filt),max(tr_times_filt)])
ax.set_ylabel('Velocity ((m) lunar (c) mars /s)')
ax.set_xlabel('Time (s)')
ax2 = plt.subplot(2, 1, 2)
vals = ax2.pcolormesh(t, f, sxx, cmap=cm.jet, vmax=5e-17)
ax2.set_xlim([min(tr_times_filt),max(tr_times_filt)])
ax2.set_xlabel(f'Time (Day Hour:Minute)', fontweight='bold')
ax2.set_ylabel('Frequency (Hz)', fontweight='bold')
#ax2.axvline(x=arrival_time_rel, c='red')
cbar = plt.colorbar(vals, orientation='horizontal')
cbar.set_label('Power ((m/s)^2/sqrt(Hz))', fontweight='bold')
```

Filter using a bandpass filter between 0.01 Hz and 0.5 Hz

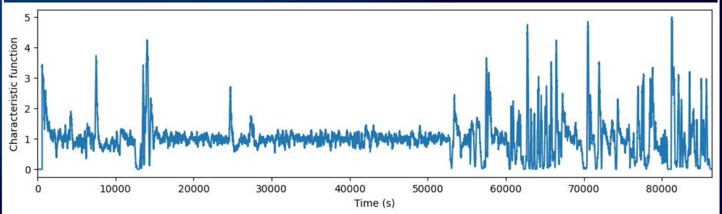


Example Shown Uses Lunar data set:
\S12_GradeB\xa.s12.00.mhz.1970-01-09HR00_evid00007.csv

05 STA/LTA Detection Algorithm

A STA/LTA algorithm moves two time windows of two lengths (one short, one long) across the seismic data. The algorithm calculates the average amplitude in both windows, and calculates the ratio between them. If the data contains an earthquake, then the short-term window containing the earthquake will be much larger than the long-term window - resulting in a detection

```
from obspy.signal.invsim import cosine_taper
from obspy.signal.filter import highpass
from obspy.signal.trigger import classic_sta_lta, plot_trigger, trigger_onset

# Sampling frequency of our trace
tr = tr_filt
df = tr.stats.sampling_rate

# How long should the short-term and long-term window be, in seconds?
sta_len = 120
lta_len = 600

# Run Obspy's STA/LTA to obtain a characteristic function
# This function basically calculates the ratio of amplitude between the short-term
# and long-term windows, moving consecutively in time across the data
cft = classic_sta_lta(tr_data_filt, int(sta_len * df), int(lta_len * df))

# Plot characteristic function
fig,ax = plt.subplots(1,1,figsize=(12,3))
ax.plot(tr_times_filt, cft)
ax.set_xlim([min(tr_times_filt), max(tr_times_filt)])
ax.set_xlabel('Time (s)')
ax.set_ylabel('Characteristic function')
```



Example Shown Uses Lunar data set:
\S12_GradeB\xa.s12.00.mhz.1970-01-09HR00_evid00007.csv

06-07 Trigger Seismic Data at Peak and Save in .CSV File

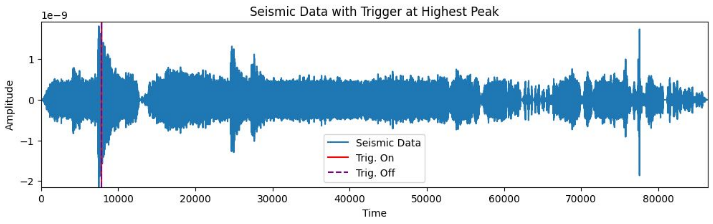
FIND THE INDEX OF THE MAXIMUM VALUE IN THE DATA AND EXTRACT THE TIME OF PEAK

```python
# Find the index of the maximum value in the seismic data
peak_idx = np.argmax(tr_data_filt)
# Plot the seismogram
fig, ax = plt.subplots(1, 1, figsize=(12, 3))
# Plot the seismogram
ax.plot(tr_times_filt, tr_data_filt, label='Seismic Data')
ax.axvline(x=tr_times_filt[peak_idx], color='red', label='Trig. On')  # Trigger at peak
ax.axvline(x=tr_times_filt[peak_idx], color='purple', linestyle='dashed', label='Trig. Off')  # Off trigger same as on
# Set plot limits and legends
ax.set_xlim([min(tr_times_filt), max(tr_times_filt)])
ax.set_ylim([min(tr_data_filt), max(tr_data_filt)])  # Set limits based on data
ax.legend()
plt.xlabel('Time')
plt.ylabel('Amplitude')
plt.title('Seismic Data with Trigger at Highest Peak')
plt.show()
# The time of the peak
peak_time = tr_times_filt[peak_idx]
```

```python
class MockRow:
    filename = file_name
class MockTrace:
    stats = type('stats', (), {'starttime': datetime(1970, 6, 26, 20, 3)})()
row = MockRow()  # Mocking the row object
tr = MockTrace()  # Mocking the trace object
fname = row.filename
starttime = tr.stats.starttime
# Iterate through detection times and compile them
detection_times = []
fnames = []
on_off = [[peak_idx]]  # Mocking the on_off list for demonstration; replace with your actual logic
for i in np.arange(0, len(on_off)):
    triggers = on_off[i]
    on_time = starttime + timedelta(seconds=tr_times_filt[triggers[0]])
# Fix applied here
    on_time_str = on_time.strftime('%Y-%m-%dT%H:%M:%S.%f')
    detection_times.append(on_time_str)
    fnames.append(fname)
# Compile DataFrame of detections
detect_df = pd.DataFrame(data={
    'filename': fnames,
    'time_abs(%Y-%m-%dT%H:%M:%S.%f)': detection_times,
    'time_rel(sec)': [tr_times_filt[triggers[0]]]
})
# Save the DataFrame to a CSV file with the specified filename
detect_df.to_csv(fname + '.csv', index=False)
# Display the DataFrame
print(detect_df
```

NASA SPACE APPS CHALLENGE > xa.s12.00.mhz.1970-01-09HR00_evid00007.csv > data

```
filename,time_abs(%Y-%m-%dT%H:%M:%S.%f),time_rel(sec)
xa.s12.00.mhz.1970-01-09HR00_evid00007,1970-06-26T22:13:38.641509,7838.641509433963
```



Example Shown Uses Lunar data set:
\S12_GradeB\xa.s12.00.mhz.1970-01-09HR00_evid00007.csv