import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from datetime import datetime, timedelta

from scipy.interpolate import interp1d

from scipy.signal import savgol\_filter

from astropy.io import fits

from matplotlib.dates import DateFormatter

# --- Functions for GOES flux ---

def calculate\_soft\_xray(fits\_file, start\_time, end\_time, dx=0.1):

    with fits.open(fits\_file) as hdu:

        time = hdu[2].data['TIME']

        flux = hdu[2].data['FLUX']

    mask = (time >= start\_time) & (time <= end\_time)

    t = time[mask]

    f = flux[mask]

    interp1 = interp1d(t, f[:, 0], kind='linear', fill\_value='extrapolate')

    interp2 = interp1d(t, f[:, 1], kind='linear', fill\_value='extrapolate')

    new\_time = np.arange(start\_time, end\_time, dx)

    f1 = interp1(new\_time)

    f2 = interp2(new\_time)

    base = datetime(2013, 10, 28)

    time\_dt = [base + timedelta(seconds=sec) for sec in new\_time]

    return time\_dt, f1, f2

# --- Functions for derived HXR ---

def derive\_hard\_xray(fits\_file, start\_time, end\_time, dx=0.1):

    with fits.open(fits\_file) as hdu:

        time = hdu[2].data['TIME']

        flux = hdu[2].data['FLUX']

    mask = (time >= start\_time) & (time <= end\_time)

    t = time[mask]

    f = flux[mask]

    interp1 = interp1d(t, f[:, 0], kind='linear', fill\_value='extrapolate')

    interp2 = interp1d(t, f[:, 1], kind='linear', fill\_value='extrapolate')

    new\_time = np.arange(start\_time, end\_time, dx)

    f1 = interp1(new\_time)

    f2 = interp2(new\_time)

    df1 = np.gradient(f1, dx)

    df2 = np.gradient(f2, dx)

    s1 = savgol\_filter(df1, window\_length=101, polyorder=3)

    s2 = savgol\_filter(df2, window\_length=101, polyorder=3)

    base = datetime(2013, 10, 28)

    time\_dt = [base + timedelta(seconds=sec) for sec in new\_time]

    return time\_dt, s1, s2

# --- Load RHESSI data ---

def load\_rhessi\_data(fits\_file, t\_start, t\_end):

    hdul = fits.open(fits\_file)

    hdr = hdul[0].header

    t0 = datetime.strptime(hdr['DATE\_OBS'], '%Y-%m-%dT%H:%M:%S.%f')

    rate = hdul['RATE'].data

    eb = hdul['ENEBAND'].data

    times = rate['TIME']

    rates = rate['RATE']

    rel\_times = [t0 + timedelta(seconds=(ts - times[0])) for ts in times]

    mask = [(rt >= t\_start) and (rt <= t\_end) for rt in rel\_times]

    filt\_times = np.array(rel\_times)[mask]

    filt\_rates = rates[mask, :]

    groups = {

        '25-50 keV': [i for i, (emin, emax) in enumerate(zip(eb['E\_MIN'], eb['E\_MAX'])) if 25 <= emin < 50],

        '50-100 keV': [i for i, (emin, emax) in enumerate(zip(eb['E\_MIN'], eb['E\_MAX'])) if 50 <= emin < 100],

        '100-300 keV': [i for i, (emin, emax) in enumerate(zip(eb['E\_MIN'], eb['E\_MAX'])) if 100 <= emin < 300]

    }

    flux = {}

    for name, idxs in groups.items():

        if idxs:

            fl = np.sum(filt\_rates[:, idxs], axis=1) \* 1e-3

            flux[name] = (filt\_times, fl)

    hdul.close()

    return flux

# --- Load radio data ---

def load\_radio\_data(rstn\_file, poemas\_file):

    rstn = pd.read\_csv(rstn\_file)

    rstn['Time'] = pd.to\_datetime(rstn['Time'].str.strip(), format='%H:%M:%S')

    base\_date = datetime(2013, 10, 28)

    rstn['Time'] = rstn['Time'].apply(lambda t: t.replace(

        year=base\_date.year, month=base\_date.month, day=base\_date.day

    ))

    poemas = pd.read\_csv(poemas\_file)

    poemas['Unnamed: 0'] = poemas['Unnamed: 0'].str.strip()

    poemas['Time'] = pd.to\_datetime(poemas['Unnamed: 0'], format='%H:%M:%S')

    poemas['Time'] = poemas['Time'].apply(lambda t: t.replace(

        year=base\_date.year, month=base\_date.month, day=base\_date.day

    ))

    return rstn, poemas

# --- Background subtraction ---

def subtract\_background(poemas, flare\_start, duration=300):

    base\_date = poemas['Time'].dt.date.iloc[0]

    flare\_time = datetime.strptime(flare\_start, '%H:%M:%S').time()

    fs = datetime.combine(base\_date, flare\_time)

    bg\_window = poemas[(poemas['Time'] >= fs - timedelta(seconds=duration)) &

                       (poemas['Time'] < fs)]

    mean\_bg = bg\_window['45 GHz'].mean() if not bg\_window.empty else 0.0

    poemas['45 GHz (Excess)'] = poemas['45 GHz'] - mean\_bg

    return poemas

# --- Combined plotting ---

def plot\_combined\_data(goes\_fits, rhessi\_fits, rstn\_file, poemas\_file,

                       start\_time, end\_time, flare\_start):

    # 1) Compute all series

    sx\_t, sx1, sx2 = calculate\_soft\_xray(goes\_fits, start\_time, end\_time)

    hxr\_t, hxr1, hxr2 = derive\_hard\_xray(goes\_fits, start\_time, end\_time)

    r\_flux = load\_rhessi\_data(

        rhessi\_fits,

        datetime(2013, 10, 28, 20, 50, 0),

        datetime(2013, 10, 28, 21, 5, 0)

    )

    rstn, poemas = load\_radio\_data(rstn\_file, poemas\_file)

    poemas = subtract\_background(poemas, flare\_start)

    # 2) Prepare highlight for GOES SXR peak

    #    e.g. 20:57:00 UT on that date

    highlight\_time\_sxr = datetime(2013, 10, 28, 20, 57, 0)

    # 3) Create figure

    fig, axes = plt.subplots(4, 1, figsize=(10, 12), sharex=True,

                             gridspec\_kw={'hspace': 0.1})

    # Plot 1: GOES SXR

    axes[0].plot(sx\_t, sx1, label='GOES 1–8 Å')

    axes[0].plot(sx\_t, sx2, label='GOES 0.5–4 Å')

    axes[0].axvline(highlight\_time\_sxr,

                    color='red', linestyle='--', linewidth=1.5,

                    label=f'SXR Peak: {highlight\_time\_sxr.strftime("%H:%M:%S")}')

    axes[0].set\_yscale('log')

    axes[0].legend(loc='lower right')

    axes[0].set\_ylabel('W m⁻²')

    axes[0].xaxis.set\_major\_formatter(DateFormatter('%H:%M'))

    # Plot 2: RHESSI photon flux

    for name, (times, fl) in r\_flux.items():

        axes[1].plot(times, fl, label=name)

        idx = np.argmax(fl)

        tpk = times[idx]

        axes[1].axvline(tpk, linestyle='--',

                        label=f'{name} Peak: {tpk.strftime("%H:%M:%S")}')

    axes[1].legend(loc='upper right')

    axes[1].set\_ylabel('Counts s⁻¹ det⁻¹')

    axes[1].xaxis.set\_major\_formatter(DateFormatter('%H:%M'))

    # Plot 3: Radio bursts

    for key, lbl in [('4.9 GHz', 'RSTN 4.9'), ('8.8 GHz', 'RSTN 8.8'), ('15.4 GHz', 'RSTN 15.4')]:

        axes[2].plot(rstn['Time'], rstn[key], label=lbl)

        idx = rstn[key].idxmax()

        tpk = rstn.loc[idx, 'Time']

        axes[2].axvline(tpk, linestyle='--', label=f'{lbl} Peak: {tpk.strftime("%H:%M:%S")}')

    axes[2].plot(poemas['Time'], poemas['45 GHz (Excess)'], label='POEMAS 45 Ex')

    idx2 = poemas['45 GHz (Excess)'].idxmax()

    tpk2 = poemas.loc[idx2, 'Time']

    axes[2].axvline(tpk2, linestyle='--', label=f'POEMAS Peak: {tpk2.strftime("%H:%M:%S")}')

    axes[2].legend(loc='upper right')

    axes[2].set\_ylabel('SFU (excess)')

    # Plot 4: Derived HXR dF/dt

    factor = 1e-7

    axes[3].plot(hxr\_t, hxr1/factor, label='dF/dt chan1')

    axes[3].plot(hxr\_t, hxr2/factor, label='dF/dt chan2')

    combo = hxr1 + hxr2

    idx3 = np.argmax(combo)

    tpk3 = hxr\_t[idx3]

    axes[3].axvline(tpk3, color='k', linestyle='--',

                    label=f'HXR Peak: {tpk3.strftime("%H:%M:%S")}')

    axes[3].legend(loc='upper right')

    axes[3].set\_ylabel('dF/dt (W m⁻²)×1e-7')

    axes[3].set\_xlabel('UT')

    axes[3].xaxis.set\_major\_formatter(DateFormatter('%H:%M'))

    plt.tight\_layout()

    plt.show()

# --- Execute plotting (adjust paths/times as needed) ---

plot\_combined\_data(

    '/content/go1520131028 (1).fits',

    '/content/hsi\_spectrum\_20131028\_200000.fits',

    '/content/rstn 28.csv',

    '/content/28.csv',

    75000,

    75900,

    '20:54:00'

)