# Analysis.py

```
import os
import errno
import math
import os
import sys
import re
import pickle
import h5py
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import csv
import pip
import plotly as pl
import pyspark as pys
import scipy as sci
import seaborn as sea
import statsmodels as stat
import sunpy as sun
import matplotlib.animation as animation

def get_vpic_info(dirs):
    """Get information of the VPIC simulation
    """
    with open(dirs+'info') as f:
        content = f.readlines()
    f.close()
    vpic_info = {}
    for line in content[1:]:
        if "=" in line:
            line_splits = line.split("=")
        elif ":" in line:
            line_splits = line.split(":")

        tail = line_splits[1].split("\n")
        vpic_info[line_splits[0].strip()] = float(tail[0])
    return vpic_info

def get_times(dirs):
    files = os.listdir(dirs + 'hydro_hdf5')
    if os.path.exists(dirs + 'hydro_hdf5/' + 'hydro-electron.xdmf'):
```

```
        files.remove('hydro-electron.xdmf')
    if os.path.exists(dirs + 'hydro_hdf5/' + 'hydro-ion.xdmf'):
        files.remove('hydro-ion.xdmf')
    if os.path.exists(dirs + 'hydro_hdf5/' + '.ipynb_checkpoints'):
        files.remove('.ipynb_checkpoints')
    c=0
    time = [None] * len(files)
    for file in files:
        time[c] = int(re.findall(r'\d+', files[c])[0])
        c = c+1
    time.sort()
    return time

def load_var(var, dirs, time_step, species):
    hydro_file = h5py.File(dirs+"hydro_hdf5/T."+str(time_step)+"/hydro_" +
species + "_"+str(time_step)+".h5", 'r')
    field_file = h5py.File(dirs+"field_hdf5/T."+str(time_step)+"/fields_" +
str(time_step)+".h5", 'r')
    jvec={}
    if var in field_file['Timestep_'+str(time_step)]:
        group=field_file['Timestep_'+str(time_step)]
        dset = group[var]
        jvec[var+str(time_step)] = np.zeros(dset.shape, dtype=dset.dtype)
        dset.read_direct(jvec[var+str(time_step)])
        globals()[var] = dset[:,:,0]
    else:
        group=hydro_file['Timestep_'+str(time_step)]
        dset = group[var]
        jvec[var+str(time_step)] = np.zeros(dset.shape, dtype=dset.dtype)
        dset.read_direct(jvec[var+str(time_step)])
        globals()[var] = dset[:,:,0]
    return globals()[var]

def load_vars(dirs, time_step, species):
    hydro_file = h5py.File(dirs+"hydro_hdf5/T."+str(time_step)+"/hydro_" +
species + "_"+str(time_step)+".h5", 'r')
    field_file = h5py.File(dirs+"field_hdf5/T."+str(time_step)+"/fields_" +
str(time_step)+".h5", 'r')

    jvec={}
    var = ['cbx', 'cby', 'cbz', 'ex', 'ey', 'ez']
    group=field_file['Timestep_'+str(time_step)]

    for i in var:
        dset = group[i]
        jvec[i+str(time_step)] = np.zeros(dset.shape, dtype=dset.dtype)
```

```
        dset.read_direct(jvec[i+str(time_step)])
        globals()[i] = dset[:,:,0]
    cbx = globals()['cbx']
    cby = globals()['cby']
    cbz = globals()['cbz']
    ex = globals()['ex']
    ey = globals()['ey']
    ez = globals()['ez']

    jvec={}
    var = ['jx', 'jy', 'jz', 'ke', 'px', 'py', 'pz', 'rho', 'txx', 'txy',
'tyy', 'tyz', 'tzx', 'tzz']
    group=hydro_file['Timestep_'+str(time_step)]

    for i in var:
        dset = group[i]
        jvec[i+str(time_step)] = np.zeros(dset.shape, dtype=dset.dtype)
        dset.read_direct(jvec[i+str(time_step)])
        globals()[i] = dset[:,:,0]

    jx = globals()['jx']
    jy = globals()['jy']
    jz = globals()['jz']
    ke = globals()['ke']
    px = globals()['px']
    py = globals()['py']
    pz = globals()['pz']
    rho = globals()['rho']
    txx = globals()['txx']
    txy = globals()['txy']
    tyy = globals()['tyy']
    tyz = globals()['tyz']
    tzx = globals()['tzx']
    tzz = globals()['tzz']

    return cbx, cby, cbz, ex, ey, ez, jx, jy, jz, ke, px, py, pz, rho, txx,
txy, tyy, tyz, tzx, tzz

def EbSpec2D(dbx, dby):
    nx =  np.shape(dbx)[0]
    ny =  np.shape(dbx)[0]

    dbx_k = np.fft.fftn(dbx)/(nx*ny)
    db2x_k = np.abs(dbx_k)**2

    dby_k = np.fft.fftn(dby)/(nx*ny)
```

```
    db2y_k = np.abs(dbx_k)**2

    db2k = db2x_k + db2y_k

    kx = np.fft.fftshift(np.fft.fftfreq(nx) * nx)

    knrm = np.sqrt(kx**2 + kx**2)

    dEbx_k = np.fft.fftshift(np.fft.fftn(dbx))/(nx*ny)

    dEb2x_k=np.abs(dEbx_k)**2
    dEb2x_k=dEb2x_k.sum(axis = 1)

    dEby_k = np.fft.fftshift(np.fft.fftn(dby))/(nx*ny)

    dEb2y_k=np.abs(dEby_k)**2
    dEb2y_k=dEb2y_k.sum(axis = 0)
    dEb2_k = dEb2x_k + dEb2y_k

    return knrm, dEb2_k


def pltpwrl(x0,y0,xi=1,xf=10,alpha=1.66667,stl=":",col='b', label =
r'$k^{-5/3}$'):
    x=np.linspace(xi,xf,50)
    plt.plot(x,(y0*x0**alpha)*x**-alpha,ls=stl,color=col, label = label)


def plot_spec(knrm, dv2):
    plt.loglog(knrm/5, dv2,  color = 'black')
    plt.grid(True, which="both")

    plt.ylabel(r'$E^b(k)$')
    plt.xlabel(r'$k(d_i)$')
    plt.axvline(10**0, label = r'$d_i$', color = 'red')
    plt.axvline(5*10**0, label = r'$d_e$', color = 'blue')
    plt.legend()

def make_gif(var, dirs, species):
    fig = plt.figure(figsize=(2,2),dpi=200)
    time = get_times(dirs)
    #fig = plt.figure()

    ims = []
    for it in time:
        x = load_var(var, dirs, it, species)
```

```
        im=plt.imshow(x)
        ims.append([im])

    ani = animation.ArtistAnimation(fig, ims)
    ani.save(var + '.gif')
    plt.close()


def pid_calc(dirs, timestep, species):
    vpic_info = get_vpic_info(dirs)
    dx = vpic_info['dx/de']
    dy = vpic_info['dy/de']
    dz = vpic_info['dz/de']
    mi_me = vpic_info['mi/me']

    pxx = np.array(txx - (jx/rho)*px)
    pxy = np.array(txy - (jx/rho)*py)
    pxz = np.array(tzx - (jx/rho)*pz)
    pyy = np.array(tyy - (jy/rho)*py)
    pyx = np.array(txy - (jy/rho)*px)
    pyz = np.array(tyz - (jy/rho)*pz)
    pzz = np.array(tzz - (jz/rho)*pz)
    pzx = np.array(tzx - (jz/rho)*px)
    pzy = np.array(tyz - (jz/rho)*py)

    p = (1/3)*(pxx + pyy + pzz)
    vx = px/rho
    vy = py/rho
    vz = pz/rho
    if species == 'ion':
        particle_mass = mi_me
    if species == 'electron':
        particle_mass = 1
    ux = px/rho/particle_mass
    uy = py/rho/particle_mass
    uz = pz/rho/particle_mass


    dux_dx = (np.roll(ux,1,axis=1)-np.roll(ux,-1,axis=1))/(2)
    duy_dx = (np.roll(uy,1,axis=1)-np.roll(uy,-1,axis=1))/(2)
    duz_dx = (np.roll(uz,1,axis=1)-np.roll(uz,-1,axis=1))/(2)

    dux_dy = (np.roll(ux,1,axis=0)-np.roll(ux,-1,axis=0))/(2)
    duy_dy = (np.roll(uy,1,axis=0)-np.roll(uy,-1,axis=0))/(2)
    duz_dy = (np.roll(uz,1,axis=0)-np.roll(uz,-1,axis=0))/(2)
```

```
    Sxx = np.array(dux_dx)
    Syy = np.array(duy_dy)
    Szz = 0

    Sxy = np.array((1/2)*(dux_dy + duy_dx))
    Sxz = np.array((1/2)*(duz_dx))
    Syz = np.array((1/2)*(duz_dy))

    theta = np.array(dux_dx + duy_dy)
    pthe = -p * theta

    Dxx = Sxx - (1/3)*theta
    Dyy = Syy - (1/3)*theta
    Dzz = Szz - (1/3)*theta

    Dxy = Sxy
    Dyz = Syz
    Dxz = Sxz

    PIxx = pxx - p
    PIyy = pyy - p
    PIzz = pzz - p


    pid = -(PIxx*Dxx+PIyy*Dyy+PIzz*Dzz+ 2.*(pxy*Dxy+pxz*Dxz+pyz*Dyz))
    return pid

def pth_calc(dirs, timestep, species):
    cbx, cby, cbz, ex, ey, ez, jx, jy, jz, ke, px, py, pz, rho, txx, txy,
tyy, tyz, tzx, tzz =load_vars(dirs,timestep,species)
    vpic_info = get_vpic_info(dirs)
    if species == 'ion':
        particle_mass = mi_me
    if species == 'electron':
        particle_mass = 1

    dx = vpic_info['dx/de']
    pxx = np.array(txx - (jx/rho)*px)
    pyy = np.array(tyy - (jy/rho)*py)
    pzz = np.array(tzz - (jz/rho)*pz)
    p = (1/3)*(pxx + pyy + pzz)

    ux = px/rho/particle_mass
    uy = py/rho/particle_mass
    uz = pz/rho/particle_mass
    dux_dx = (np.roll(ux,1,axis=1)-np.roll(ux,-1,axis=1))/(2)
```

```
    duy_dy = (np.roll(uy,1,axis=0)-np.roll(uy,-1,axis=0))/(2)

    theta = np.array(dux_dx + duy_dx)
    ptheta = p * theta

    return ptheta

def plt_var(var, dirs, lx = 'de', var_lab = 'var'):
    vpic_info = get_vpic_info(dirs)

    if lx == 'de':
        Lx = np.linspace(0, vpic_info['Lx/de'], int(vpic_info['nx']))
        Ly = np.linspace(0,vpic_info['Ly/de'], int(vpic_info['ny']))
        xlab = r'$X(d_e)$'
        ylab = r'$Y(d_e)$'
    elif lx == 'di':
        Lx = np.linspace(0, vpic_info['Lx/di'], int(vpic_info['nx']))
        Ly = np.linspace(0,vpic_info['Ly/di'], int(vpic_info['ny']))
        xlab = r'$X(d_i)$'
        ylab = r'$Y(d_i)$'
    plt.pcolormesh(Ly,Lx, var)
    plt.xlabel(xlab)
    plt.ylabel(ylab)
    plt.colorbar(label = var_lab)
```