```python
# superbubbles.py (python3)

from numpy import pi, sqrt, arccos, arcsin, exp, log
from tqdm import tqdm
import scipy.integrate as integrate
import numpy as np
import os, plawt

figdir = 'figures'
if not os.path.exists(figdir):
    os.mkdir(figdir)

# Dimensionless Scaling
H = 1 # [L] = H
gamma = 5/3
L_not = 1
rho_not = 1
P = 1

def r(z,y):
    """ Get the shape of the shockfront """
    arg = 1 - y**2/(4*H**2) + exp(-z/H)
    arg *= exp(z/(2*H))/2
    return 2 * H * arccos(arg)

def z12(y):
    """
    Get the edges of the shockfront
    returns tuple (z1, z2)
    """
    return (-2*H*log(1 - y/(2*H)), -2*H*log(1 + y/(2*H)))
z12 = np.vectorize(z12)

def rmax(y):
    """ Get max radius of the bubble """
    return 2*H*arcsin(y/(2*H))


def shockfronts():
    import imageio

    z = np.arange(-2, 10, 0.00001)
    y = [0.1, 0.5, 1, 1.4, 1.7, 1.9, 1.98, 2.0]
    figure1 = {
        'ylabel': 'z/H', 'xlabel': 'r/H',
        'filename': 'shockfront.png',
        'ylim': (-2, 10), 'xlim': (-6, 6),
        'figsize': (6/1.3, 6.5/1.3),
        'show': False,
        # 'legend': {'loc':4}
    }

    for i, yi in enumerate(tqdm(y)):
        figure1[i] = {'x': np.concatenate((r(z, y[i]), -r(z, y[i]))), 'y': np.concatenate((z,z)), 'label': '$y=$'+str(yi)}
    plawt.plot(figure1)

    animation = {
        'ylabel': 'z/H', 'xlabel': 'r/H',
        'ylim': (-2, 10), 'xlim': (-6, 6),
        'figsize': (6/1.3, 6.5/1.3),
        'title': 'Likely how W4 expanded',
        'show': False,
        'keepOpen': True,
        'legend': {'loc':4}
    }
    y = np.arange(0.01, 2.05, 0.05)
    with imageio.get_writer('blast.gif', mode='I', fps=24) as writer:
        for i, t in enumerate(tqdm(y)):
            animation[0] = {'x': np.concatenate((r(z, y[i]), -r(z, y[i]))), 'y': np.concatenate((z,z)),
                'line':'k-', 'label':'$y=$'+str(y[i])}
            plt = plawt.plot(animation)
            fig = plt.gcf()
            fig.canvas.draw()
            data = fig.canvas.tostring_rgb()
            row, col = fig.canvas.get_width_height()
            image = np.fromstring(data, dtype=np.uint8).reshape(col, row, 3)
            writer.append_data(image)
            plt.close()

shockfronts()

### Math Helpers ###
```

```python

# Derivatives of stuff
dy = lambda Eth, Omega: sqrt((gamma**2 - 1)*Eth / 2  / (rho_not * Omega))
drdy = lambda z, y: y / ( 2*sqrt(1 - 1/4*exp(z/H)*(1-y**2/(4*H**2)+exp(-z/H))**2) )
dOmega = lambda y, dy: 2 * pi * integrate.quad(lambda z: r(z, y) * drdy(z, y) * dy, z12(y)[1], z12(y)[0])[0]
dEth = lambda y, dy, P: L_not - P * dOmega(y, dy)

# Equations from paper
PFunc = lambda E, O: (gamma - 1)*E/O
PFunc = np.vectorize(PFunc)
OmegaFunc = lambda y: pi * integrate.quad(lambda z: r(z, y)**2, z12(y)[1], z12(y)[0])[0]
EnergyFunc = lambda oprev, onext, E: L_not*dt - (gamma-1)*E*(onext-oprev)/oprev+E

dzsdt = lambda y, E, O: ( dy(E, O)/(1-y/(2*H)), -dy(E, O)/(1+y/(2*H)) )
dzsdt = np.vectorize(dzsdt)

###

# initial conditions
dt = 0.0001 # only seems to work with this dt
time = np.arange(0.005, 10, dt)
yi = 0.01
Omegai = OmegaFunc(yi)
Ethi = P/(gamma-1)*Omegai

initialstate = [yi, Omegai, Ethi]
ys = [yi]
Omegas = [Omegai]
Es = [Ethi]

# Integrate
for t in tqdm(time):
    ynext = ys[-1] + dy(Es[-1], Omegas[-1])*dt
    omeganext = OmegaFunc(ynext)
    energynext = EnergyFunc(Omegas[-1], omeganext, Es[-1])

    ys.append(ynext)
    Omegas.append(omeganext)
    Es.append(energynext)
    if ynext > 1.99999:
        break
```

```python

# Calculate extras
z12s = z12(ys)
r = np.vectorize(r) # vectorize after we're done integrating because it makes it really slow otherwise
Ps = PFunc(Es, Omegas)
dz1sdt = dzsdt(ys, Es, Omegas)[0]

# Plot
plawt.plot({
    0: {'x': time[:len(ys)], 'y': ys, 'line':'k-'},
    'show':False,
    'filename': os.path.join(figdir,'y.png'),
    'title': "(a) $y$ vs Time",
    'xlabel': '$\\tilde{t}$',
    'ylabel': '$\\tilde{y}$',
    'set_yscale': 'log', 'set_xscale': 'log',
    'xlim': (0.01, 10), 'ylim': (0.1, 10.0),
    'grid':True
})
plawt.plot({
    0: {'x': time[:len(ys)], 'y': Es, 'line':'k-'},
    'show':False,
    'filename': os.path.join(figdir,'energy.png'),
    'title': "Thermal Energy vs Time",
    'xlabel': '$\\tilde{t}$',
    'ylabel': '$\\tilde{E}_{th}$',
    'set_yscale': 'log', 'set_xscale': 'log',
    'xlim': (0.01, 10), 'ylim': (0.01, 10.0),
    'grid':True
})
plawt.plot({
    0: {'x': time[:len(ys)], 'y': z12s[0], 'label': '$\\tilde{z}_1$', 'line':'k-'},
    1: {'x': time[:len(ys)], 'y': -z12s[1], 'label': '$\\tilde{z}_2$'},
    2: {'x': time[:len(ys)], 'y': r(0, ys), 'label': '$\\tilde{r}(z=0,y)$', 'line': 'k--'},
    'filename': os.path.join(figdir,'blastedges.png'),
    'title': 'Blast Edges vs. Time',
    'xlabel': '$\\tilde{t}$',
    'ylabel': 'Distance',
    'legend': {'loc': 4},
    'set_yscale': 'log', 'set_xscale': 'log',
    'xlim': (0.01, 10), 'ylim': (0.1, 10.0),
    'grid':True
})
plawt.plot({
    0: {'x': time[:len(ys)], 'y': Ps, 'line':'k-'},
    'show':False,
    'filename': os.path.join(figdir,'pressure.png'),
    'title': "Pressure vs. Time",
    'xlabel': '$\\tilde{t}$',
    'ylabel': '$\\tilde{P}$',
    'set_yscale': 'log', 'set_xscale': 'log',
    'xlim': (0.01, 10), 'ylim': (0.01, 10.0),
    'grid':True
})
plawt.plot({
    0: {'x': time[:len(ys)], 'y': dz1sdt, 'line':'k-'},
    'show':False,
    'filename': os.path.join(figdir,'blastedgespeed.png'),
    'title': "Blast Edge Speed",
    'xlabel': '$\\tilde{t}$',
    'ylabel': '$d\\tilde{z}_1/dt$',
    'set_yscale': 'log', 'set_xscale': 'log',
    'xlim': (0.01, 10), 'ylim': (0.01, 10.0),
    'grid':True
})
plawt.plot({
    0: {'x': z12s[0], 'y': dz1sdt, 'line':'k-'},
    'show':False,
    'filename': os.path.join(figdir,'blastedgeSpeedvsPos.png'),
    'title': "Blast Edge Speed vs. Position",
    'xlabel': '$\\tilde{z}_1$',
    'ylabel': '$d\\tilde{z}_1/dt$',
    'set_yscale': 'log', 'set_xscale': 'log',
    'xlim': (0.1, 10), 'ylim': (0.1, 10.0),
    'grid':True
})
```