```
1. # superbubbles.py (python3)
 2.
   from numpy import pi, sqrt, arccos, arcsin, exp, log
4. from tqdm import tqdm
 5. import scipy.integrate as integrate
   import numpy as np
    import os, plawt
 8.
    figdir = 'figures'
   if not os.path.exists(figdir):
        os.mkdir(figdir)
11.
12.
13. # Dimensionless Scaling
14. H = 1 \# \lceil L \rceil = H
15. gamma = 5/3
16. L_not = 1
17. \text{ rho\_not} = 1
18. P = 1
19.
20.
    def r(z,y):
21.
        """ Get the shape of the shockfront """
22.
        arg = 1 - y^**2/(4^*H^**2) + exp(-z/H)
        arg *= \exp(z/(2*H))/2
23.
24.
        return 2 * H * arccos(arg)
25.
26.
    def z12(y):
27.
        Get the edges of the shockfront
28.
29.
        returns tuple (z1, z2)
30.
        return (-2*H*log(1 - y/(2*H)), -2*H*log(1 + y/(2*H)))
31.
32. z12 = np.vectorize(z12)
33.
34.
    def rmax(y):
        """ Get max radius of the bubble """
35.
        return 2*H*arcsin(y/(2*H))
36.
37.
38.
    def shockfronts():
39.
40.
        import imageio
41.
```

```
42.
        z = np.arange(-2, 10, 0.00001)
43.
        y = [0.1, 0.5, 1, 1.4, 1.7, 1.9, 1.98, 2.0]
44.
        figure1 = {
            'ylabel': 'z/H', 'xlabel': 'r/H',
45.
46.
            'filename': 'shockfront.png',
47.
            'ylim': (-2, 10), 'xlim': (-6, 6),
            'figsize': (6/1.3, 6.5/1.3),
48.
49.
            'show': False,
            # 'legend': {'loc':4}
50.
51.
        }
52.
53.
        for i, yi in enumerate(tqdm(y)):
54.
            figure1[i] = \{'x': np.concatenate((r(z, y[i]), -r(z, y[i]))), 'y': np.concatenate((z,z)), 'label': '$y=$'+str(yi)}
55.
        plawt.plot(figure1)
56.
57.
        animation = {
58.
            'ylabel': 'z/H', 'xlabel': 'r/H',
59.
            'ylim': (-2, 10), 'xlim': (-6, 6),
60.
            'figsize': (6/1.3, 6.5/1.3),
61.
            'title': 'Likely how W4 expanded',
            'show': False,
62.
            'keepOpen': True,
63.
            'legend': {'loc':4}
64.
65.
66.
        y = np.arange(0.01, 2.05, 0.05)
67.
        with imageio.get_writer('blast.gif', mode='I', fps=24) as writer:
            for i, t in enumerate(tqdm(y)):
68.
                animation[0] = {'x': np.concatenate((r(z, y[i]), -r(z, y[i]))), 'y': np.concatenate((z,z)),
69.
                     'line':'k-', 'label':'$y=$'+str(y[i])}
70.
71.
                plt = plawt.plot(animation)
                fig = plt.gcf()
72.
73.
                fig.canvas.draw()
74.
                data = fig.canvas.tostring_rgb()
75.
                row, col = fig.canvas.get_width_height()
76.
                image = np.fromstring(data, dtype=np.uint8).reshape(col, row, 3)
77.
                writer.append_data(image)
78.
                plt.close()
79.
80.
    shockfronts()
81.
82. ### Math Helpers ###
```

```
83.
  84. # Derivatives of stuff
  85. dy = lambda Eth, Omega: sqrt((gamma**2 - 1)*Eth / 2 / (rho_not * Omega))
  86. drdy = lambda z, y: y / ( 2*sqrt(1 - 1/4*exp(z/H)*(1-y**2/(4*H**2)+exp(-z/H))**2) )
  87. 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)
  89.
  90. # Equations from paper
  91. PFunc = \frac{1}{2} PFunc =
  92. PFunc = np.vectorize(PFunc)
  93. 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
  95.
  96.
             dzsdt = lambda y, E, O: (dy(E, O)/(1-y/(2*H)), -dy(E, O)/(1+y/(2*H)))
              dzsdt = np.vectorize(dzsdt)
  98.
  99.
             ###
100.
101. # initial conditions
102. dt = 0.0001 \# only seems to work with this dt
103. time = np.arange(0.005, 10, dt)
104. yi = 0.01
105. Omegai = OmegaFunc(yi)
106. Ethi = P/(gamma-1)*Omegai
107.
108. initialstate = [yi, Omegai, Ethi]
109. ys = [yi]
110. Omegas = [Omegai]
111. Es = [Ethi]
112.
113. # Integrate
114. for t in tqdm(time):
115.
                        ynext = ys[-1] + dy(Es[-1], Omegas[-1])*dt
116.
                         omeganext = OmegaFunc(ynext)
117.
                        energynext = EnergyFunc(Omegas[-1], omeganext, Es[-1])
118.
119.
                        ys.append(ynext)
120.
                        Omegas.append(omeganext)
121.
                        Es.append(energynext)
122.
                        if ynext > 1.99999:
123.
                                   break
```

```
124.
125. # Calculate extras
126. z12s = z12(ys)
127. r = np.vectorize(r) # vectorize after we're done integrating because it makes it really slow otherwise
128. Ps = PFunc(Es, Omegas)
129. dz1sdt = dzsdt(ys, Es, Omegas)[0]
130.
131. # Plot
132. plawt.plot({
133.
         0: {'x': time[:len(ys)], 'y': ys, 'line':'k-'},
         'show':False,
134.
135.
         'filename': os.path.join(figdir, 'y.png'),
         'title': "(a) $y$ vs Time",
136.
137.
         'xlabel': '$\\tilde{t}$',
138.
         'ylabel': '$\\tilde{y}$',
139.
         'set_yscale': 'log', 'set_xscale': 'log',
         'xlim': (0.01, 10), 'ylim': (0.1, 10.0),
140.
         'grid':True
141.
142. })
143. plawt.plot({
         0: {'x': time[:len(ys)], 'y': Es, 'line':'k-'},
144.
         'show':False,
145.
         'filename': os.path.join(figdir, 'energy.png'),
146.
         'title': "Thermal Energy vs Time",
147.
148.
         'xlabel': '$\\tilde{t}$',
         'vlabel': '$\\tilde{E}_{th}$',
149.
150.
         'set_yscale': 'log', 'set_xscale': 'log',
         'xlim': (0.01, 10), 'ylim': (0.01, 10.0),
151.
         'grid':True
152.
153. })
154. plawt.plot({
         0: {'x': time[:len(ys)], 'y': z12s[0], 'label': '$\\tilde{z}_1$', 'line':'k-'},
155.
156.
         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--'},
157.
158.
         'filename': os.path.join(figdir, 'blastedges.png'),
159.
         'title': 'Blast Edges vs. Time',
         'xlabel': '$\\tilde{t}$',
160.
161.
         'ylabel': 'Distance',
162.
         'legend': {'loc': 4},
163.
         'set_yscale': 'log', 'set_xscale': 'log',
164.
         'xlim': (0.01, 10), 'ylim': (0.1, 10.0),
```

```
'grid':True
165.
166. })
167. plawt.plot({
         0: {'x': time[:len(ys)], 'y': Ps, 'line':'k-'},
168.
         'show':False,
169.
170.
         'filename': os.path.join(figdir, 'pressure.png'),
         'title': "Pressure vs. Time",
171.
         'xlabel': '$\\tilde{t}$',
172.
173.
        'ylabel': '$\\tilde{P}$',
        'set_yscale': 'log', 'set_xscale': 'log',
174.
175.
         'xlim': (0.01, 10), 'ylim': (0.01, 10.0),
         'grid':True
176.
177. })
178. plawt.plot({
         0: {'x': time[:len(ys)], 'y': dz1sdt, 'line':'k-'},
179.
         'show':False,
180.
181.
         'filename': os.path.join(figdir, 'blastedgespeed.png'),
182.
         'title': "Blast Edge Speed",
183.
         'xlabel': '$\\tilde{t}$',
184.
         'ylabel': '$d\\tilde{z}_1/dt$',
         'set_yscale': 'log', 'set_xscale': 'log',
185.
         'xlim': (0.01, 10), 'ylim': (0.01, 10.0),
186.
187.
         'grid':True
188. })
189. plawt.plot({
         0: {'x': z12s[0], 'y': dz1sdt, 'line':'k-'},
190.
191.
         'show':False,
192.
         'filename': os.path.join(figdir, 'blastedgeSpeedvsPos.png'),
         'title': "Blast Edge Speed vs. Position",
193.
         'xlabel': '$\\tilde{z}_1$',
194.
         'ylabel': '$d\\tilde{z}_1/dt$',
195.
196.
         'set_yscale': 'log', 'set_xscale': 'log',
         'xlim': (0.1, 10), 'ylim': (0.1, 10.0),
197.
         'grid':True
198.
199. })
```