Phys 610 Midterm

Problems 6-8

Sean Ericson

In [ ]:
```
# Imports
import matplotlib as mpl
import matplotlib.pyplot as plt
import numpy as np
from scipy.integrate import solve_ivp

mpl.rcParams['figure.dpi'] = 200
```

In [ ]:
```
def effective_potential(r, M, b, l):
    tmp = r*r + b*b
    return 0.5*((1 - 2*M/np.sqrt(tmp))*(1 + l*l/tmp) - 1)

# Equation to integrate (dr/dphi = r'(r; M, b, l, E))
def r_prime(r, M, b, l, E):
    return -1*(r*r + b*b) * np.sqrt(2 * (E - effective_potential(r, M, b, l))) / l
```

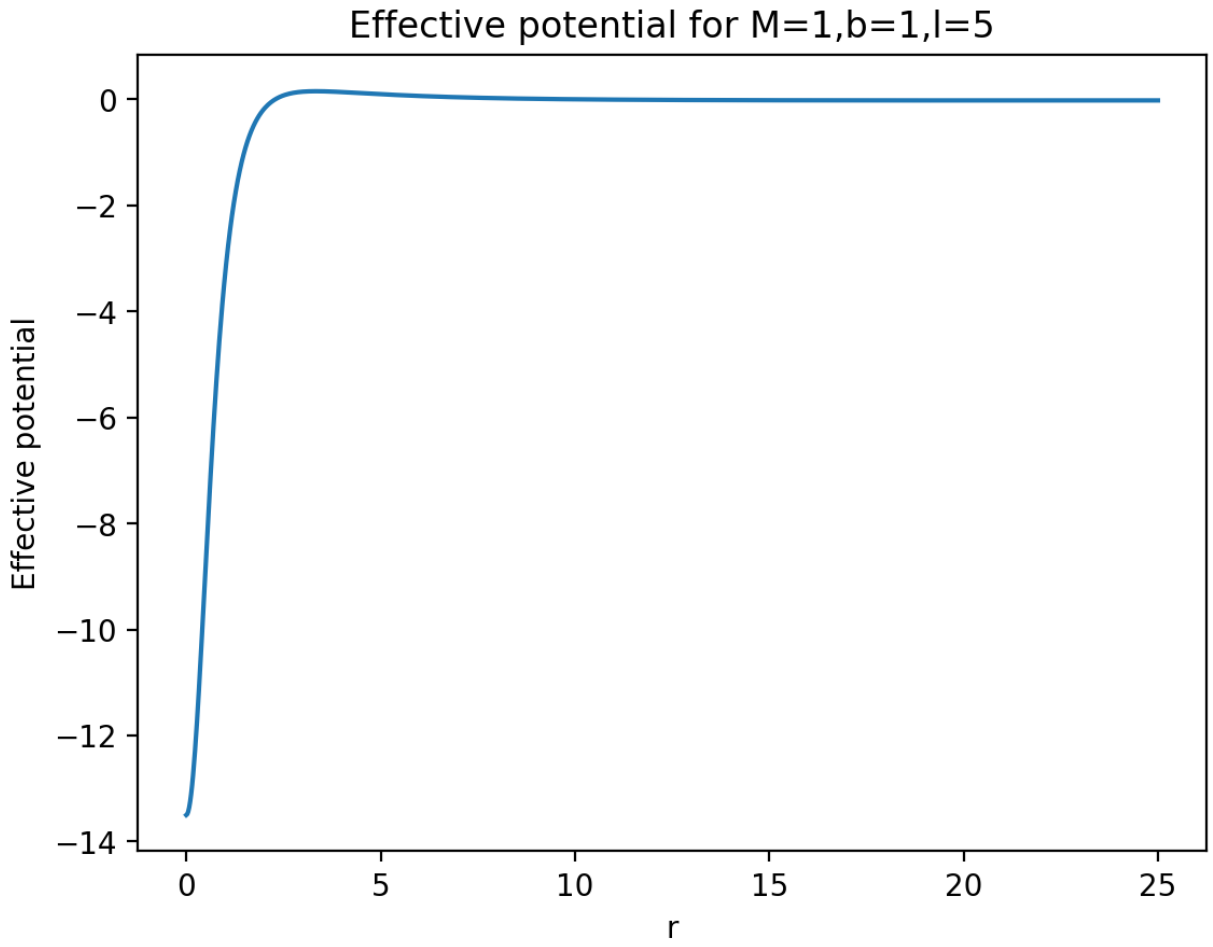
In [ ]:
```
def RungeKutta(f, y0, step_size, num_steps):
    ys = [y0]
    while len(ys) < num_steps:
        k1 = f(ys[-1])
        k2 = f(ys[-1] + step_size * k1 / 2)
        k3 = f(ys[-1] + step_size * k2 / 2)
        k4 = f(ys[-1] + step_size * k3)
        ys.append(ys[-1] + step_size * (k1 + 2*k2 + 2*k3 + k4) / 6)
    return ys
```

Problem 7

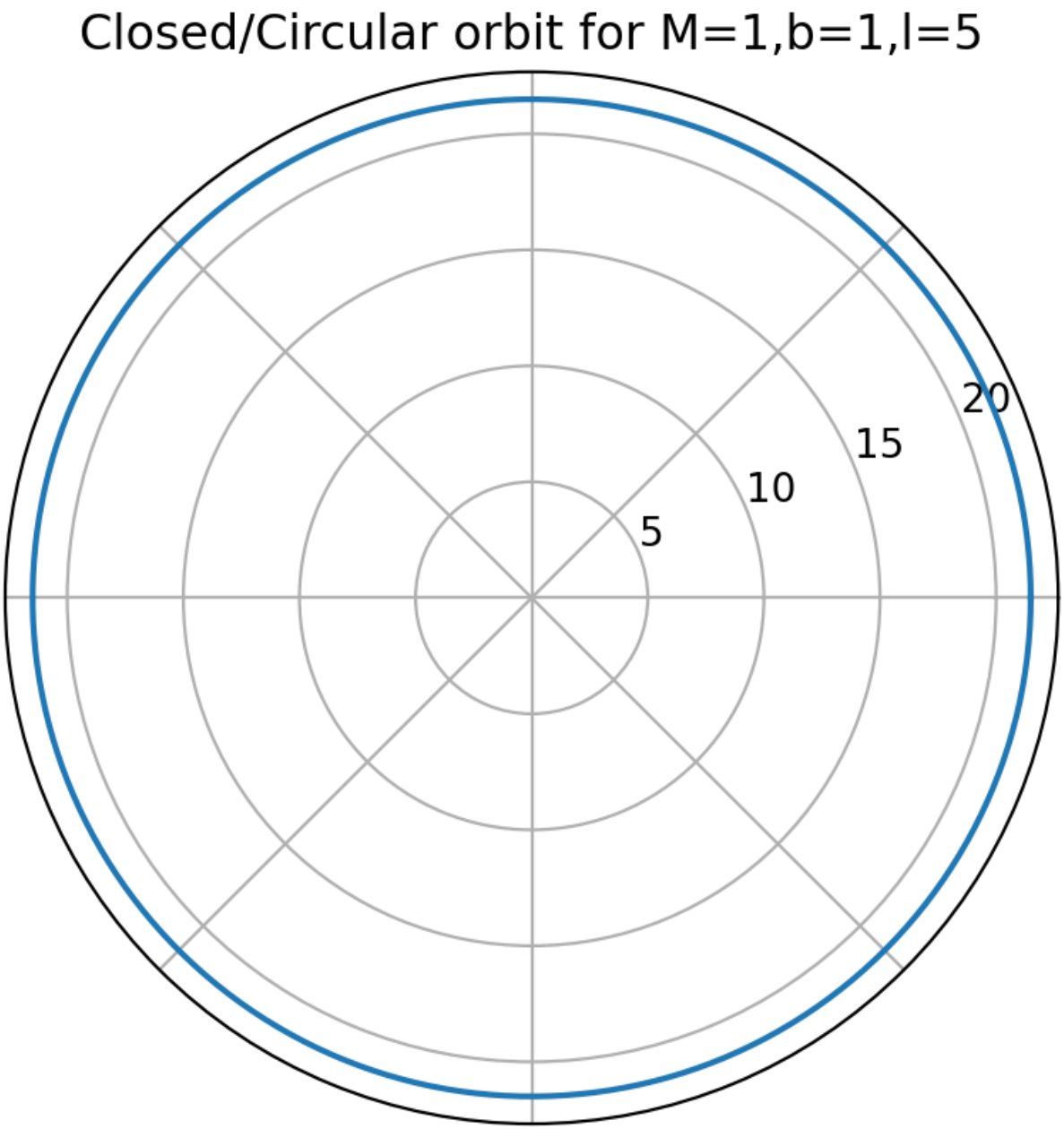
In [ ]:
```
M = 1
b = 1
l = 5
```

In [ ]:
```
rs = np.linspace(0, 25, 1000)
Vs = [effective_potential(r, M, b, l) for r in rs]
plt.plot(rs, Vs)
plt.title("Effective potential for M=1,b=1,l=5")
plt.xlabel("r")
plt.ylabel("Effective potential")
```

Out[ ]:
```
Text(0, 0.5, 'Effective potential')
```



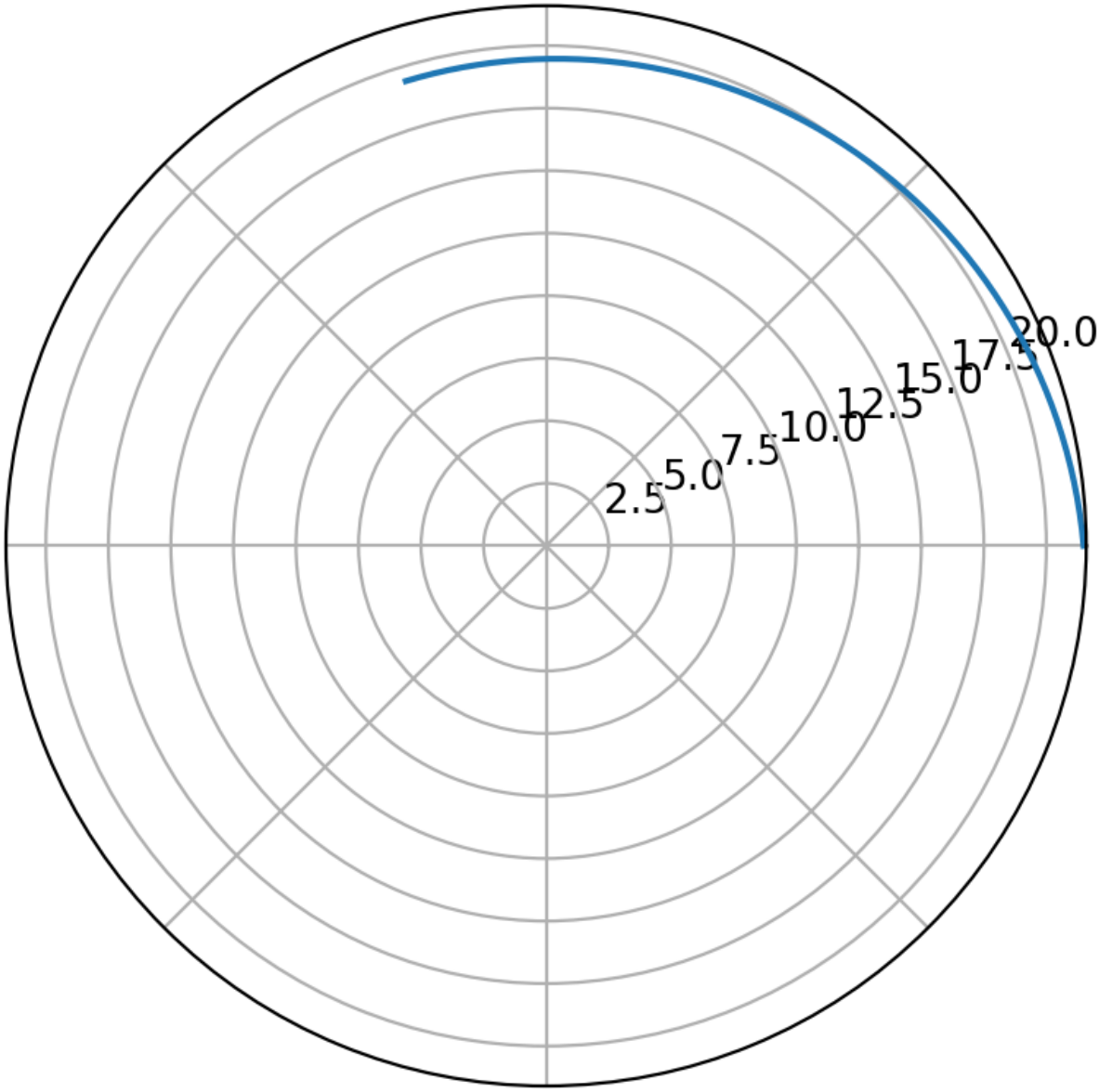
In [ ]:
```
r0 = 21.4906 # Start at local minimum calculated in Mathematica
E = effective_potential(r0, M, b, l) # Give test particle 0 kinetic energy
step_size = 0.01
final_angle = 2*np.pi
num_step = int(final_angle / step_size)
angles = np.linspace(0, final_angle, num_step)

func = lambda y: r_prime(y, M, b, l, E)
rs = RungeKutta(func, r0, step_size=step_size, num_steps=num_step)

#Make plt figure
fig = plt.figure()

#Make sub-plot with attribute "polar"
ax = fig.add_subplot(polar=True)

#Plot function
ax.plot(angles, rs)
ax.set_xticklabels([])

plt.title("Closed/Circular orbit for M=1,b=1,l=5")
plt.show()
```



In [ ]:
```
r0 = 21.4906 # Start at local minimum calculated in Mathematica
E = effective_potential(r0, M, b, l)*0.99  # Give test particle a little kinetic energ
step_size = 0.01
final_angle = 2*np.pi
num_step = int(final_angle / step_size)
angles = np.linspace(0, final_angle, num_step)

func = lambda y: r_prime(y, M, b, l, E)
rs = RungeKutta(func, r0, step_size=step_size, num_steps=num_step)

#Make plt figure
fig = plt.figure()

#Make sub-plot with attribute "polar"
ax = fig.add_subplot(polar=True)

#Plot function
ax.plot(angles, rs)
```

```
ax.set_xticklabels([])

plt.title("")
plt.show()
```

```
C:\Users\Sean\AppData\Local\Temp\ipykernel_15476\4176942516.py:7: RuntimeWarning: inv
alid value encountered in sqrt
  return -1*(r*r + b*b) * np.sqrt(2 * (E - effective_potential(r, M, b, l))) / l
```



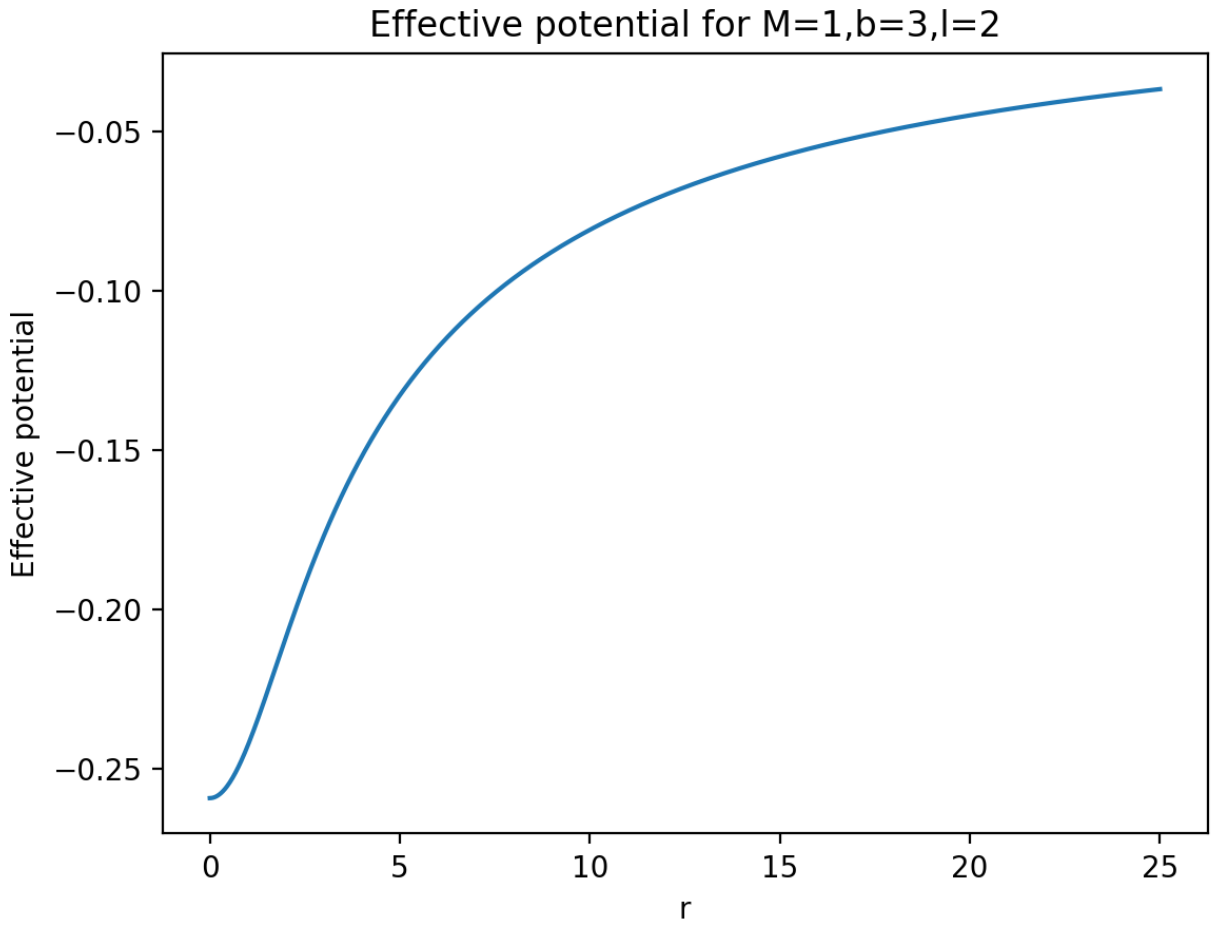
Integrator seems to break when I try a non-circular orbit

Problem 8

In [ ]:
```
M = 1
b = 3
l = 2
E = -0.05
```

In [ ]:
```
rs = np.linspace(0, 25, 1000)
Vs = [effective_potential(r, M, b, l) for r in rs]
```

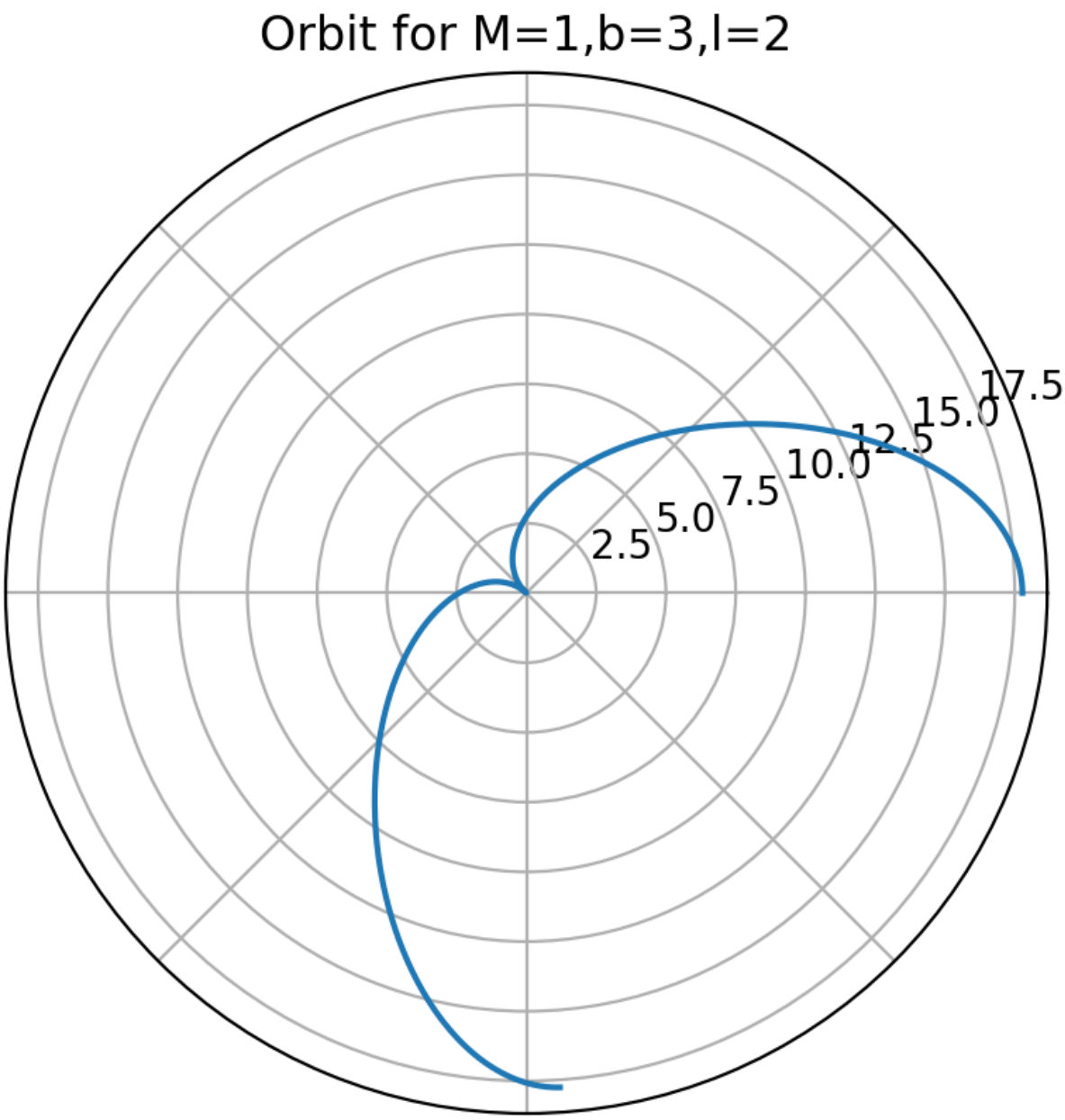
```
plt.plot(rs, Vs)
plt.title("Effective potential for M=1,b=3,l=2")
plt.xlabel("r")
plt.ylabel("Effective potential")
plt.show()
```



In [ ]:

```
r0 = 17.7759
step_size = 0.01
final_angle = 2*np.pi
num_step = int(final_angle / step_size)
angles = np.linspace(0, final_angle, num_step)

func = lambda y: r_prime(y, M, b, l, E)
rs = RungeKutta(func, r0, step_size=step_size, num_steps=num_step)
rs = [abs(r) for r in rs]

#Make plt figure
fig = plt.figure()

#Make sub-plot with attribute "polar"
ax = fig.add_subplot(polar=True)

#Plot function
ax.plot(angles, rs)
ax.set_xticklabels([])

plt.title("Orbit for M=1,b=3,l=2")
plt.show()
```

```
C:\Users\Sean\AppData\Local\Temp\ipykernel_15476\4176942516.py:7: RuntimeWarning: inv
alid value encountered in sqrt
  return -1*(r*r + b*b) * np.sqrt(2 * (E - effective_potential(r, M, b, l))) / l
```



In [ ]: