Problem Set 2

```python
import numpy as np
from astropy.constants import c, G, M_sun, m_p
import astropy.units as u

def bondi_gamma_factor(gamma):
    """
    Calculate the factor involving gamma in the Bondi accretion rate.
    """
    factor = (2 / (5 - 3 * gamma)) ** ((5 - 3 * gamma) / (2 * (gamma - 1)))
    return factor

def bondi_accretion_rate(M_star, rho_inf, c_inf, gamma=1.4):
    """
    Calculate the Bondi accretion rate.

    Parameters:
    - M_star: Mass of the star in grams
    - rho_inf: Density of ISM in g/cm^3
    - c_inf: Sound speed in ISM in cm/s
    - gamma: Adiabatic index (default is 1.4 for ISM)

    Returns:
    - accretion rate in g/s
    """
    # First part of the equation
    factor1 = np.pi * (G**2) * (M_star**2) * rho_inf / (c_inf**3)

    # Second part: The factor involving gamma
    factor2 = bondi_gamma_factor(gamma)

    print(f"Factor 1: {factor1}")
    print(f"Factor 2: {factor2}")

    # Bondi accretion rate in grams per second
    accretion_rate = factor1 * factor2

    return accretion_rate

def time_to_double_mass(M_star, rho_inf, c_inf, gamma=1.4):
    """
    Estimate the time for a star to double its mass via accretion.

    Parameters:
```

```
    - M_star: Initial mass of the star in grams
    - rho_inf: Density of ISM in g/cm^3
    - c_inf: Sound speed in ISM in cm/s
    - gamma: Adiabatic index (default is 1.4 for ISM)

    Returns:
    - Time to double the mass in seconds
    """
    accretion_rate = bondi_accretion_rate(M_star, rho_inf, c_inf, gamma)
    print(f"Bondi accretion rate: {accretion_rate.to(u.M_sun/u.yr)}")

    # Time to double the mass (in seconds)
    time_double = M_star / accretion_rate

    return time_double
```

```
# Example: Solar mass star in typical ISM conditions
M_star = M_sun  # Solar mass in grams
rho_inf = 1e-24 * u.g / u.cm**3 # ISM density
c_inf = 10 * u.km / u.s # Sound speed

# Calculate time to double mass
time_double = time_to_double_mass(M_star, rho_inf, c_inf).to(u.yr)

# Print result
print(f"Time to double the mass of a solar-mass star: {time_double:.2e}")
```

```
Factor 1: 55331588792011.84 m6 g / (s cm3 km3)
Factor 2: 2.5000000000000004
Bondi accretion rate: 2.1953875961320245e-15 solMass / yr
Time to double the mass of a solar-mass star: 4.56e+14 yr
```

0.1 The Eddington Luminosity

```
print(G, c, m_p, M_sun)
```

```
  Name   = Gravitational constant
  Value  = 6.6743e-11
  Uncertainty  = 1.5e-15
  Unit  = m3 / (kg s2)
  Reference = CODATA 2018   Name   = Speed of light in vacuum
  Value  = 299792458.0
  Uncertainty  = 0.0
  Unit  = m / s
  Reference = CODATA 2018   Name   = Proton mass
```

```
  Value  = 1.67262192369e-27
  Uncertainty  = 5.1e-37
  Unit  = kg
  Reference = CODATA 2018   Name   = Solar mass
  Value  = 1.988409870698051e+30
  Uncertainty  = 4.468805426856864e+25
  Unit  = kg
  Reference = IAU 2015 Resolution B 3 + CODATA 2018
```

0.2 Bondi Accretion onto Black Holes?

```
M_bh = 1e6 * M_sun  # SMBH mass
rho_inf = 1.67e-24 * u.g / u.cm**3 # galactic medium density
c_inf = 200 * u.km / u.s # sound speed
gamma = 4/3  # adiabatic index
```

0.2.a Estimate the sonic radius

```
# import package related to the current problem
from astr145 import schwarzschild_radius

def sonic_radius(M_bh, c_inf, gamma):
    factor = (5 - 3 * gamma) / 4
    r_s = G * M_bh / (c_inf**2) * factor
    return r_s

# Calculate the sonic radius
r_s = sonic_radius(M_bh, c_inf, gamma).to(u.km)
accretion_rate = bondi_accretion_rate(M_bh, rho_inf, c_inf, gamma).to(u.M_sun/
u.yr)
R_sch = schwarzschild_radius(M_bh).to(u.km)
# Output the results
print(f"Sonic radius: {r_s:.2e}")
print(f"Schwarzschild radius: {R_sch:.2e}")
print(f"Sonic radius in Schwarzschild units: {r_s/R_sch:.2e}")
print(f"Bondi accretion rate: {accretion_rate:.2e}")
```

```
Factor 1: 1.1550469160332473e+22 m6 g / (s cm3 km3)
Factor 2: 2.828427124746191
Sonic radius: 8.29e+11 km
Schwarzschild radius: 2.95e+06 km
Sonic radius in Schwarzschild units: 2.81e+05
Bondi accretion rate: 5.18e-07 solMass / yr
```

```python
epsilon = 0.1  # Efficiency factor
# Function to calculate Eddington luminosity
def eddington_luminosity(M):
    L_edd = 4 * np.pi * G * M_bh * c
    return L_edd

def eddington_luminosity(M_bh):
    L_edd = 1.25e38 * (M_bh / M_sun) * u.erg / u.s  # Eddington luminosity in
erg/s
    return L_edd

# Function to calculate Eddington accretion rate
def eddington_accretion_rate(M_bh, epsilon=epsilon):
    L_edd = eddington_luminosity(M_bh)
    Mdot_edd = L_edd / (epsilon * c**2)  # Eddington accretion rate
    return Mdot_edd.to(u.M_sun/u.yr)
```

```python
# Example: For a SMBH with 1e6 solar masses
M_bh = 1e6 * M_sun
# Calculate the Eddington accretion rate
edd_accretion_rate = eddington_accretion_rate(M_bh)

# Output the result
print(f"Eddington accretion rate: {edd_accretion_rate:.2e}")
print(f"Bondi accretion rate in Eddington units: {accretion_rate/
edd_accretion_rate:.2e}")
```

```
Eddington accretion rate: 2.21e-02 solMass / yr
Bondi accretion rate in Eddington units: 2.35e-05
```

```python
σT = 6.65e-25 * u.cm**2  # Thomson cross-section

f1 = 4  * m_p * c_inf**3 / (epsilon * c  * σT * M_sun * G * rho_inf)
f1 = f1.to(u.dimensionless_unscaled)
f2 = bondi_gamma_factor(gamma) **(-1)
print(f"Factor: {f1 * f2}")
```

```
Factor: 42828445976.572334
```

0.3 Rotating Black Holes

Stationary black holes have a point singularity while rotating black holes possess a ring-shaped singularity. This distinction leads to the intriguing concept of naked singularities that could theoretically be observed. The concept of naked singularities presents a paradox in physics since they lack an event horizon, allowing for potential observation, although none have been detected so far.

Bibliography