Assignment

TOV

Name TN 1: Tim Peinkofer

tim.peinkofer@student.uni-tuebingen.de

Name TN 2: Fabian Kostow

fabian.kostow@student.uni-tuebingen.de

Tutor: Jose Carlos Olvera Meneses

Date: February 14, 2025

1 TOV

The kernel of the code is the rk4 solver for a system of n-coupled ODE's, which we already have from the Lorenz attractor. We just made a few changes.

```
def rk4(system, y0, t, h, K, Gamma):
"""Runge-Kutta 4. Ordnung (RK4) f r das System."""
n = len(t)
y = np.zeros((n, len(y0)))
y[0] = y0

for i in range(1, n):
    k1 = h * system(y[i - 1], t[i - 1], K, Gamma)
    k2 = h * system(y[i - 1] + k1 / 2, t[i - 1] + h / 2,
                               K, Gamma)
    k3 = h * system(y[i - 1] + k2 / 2, t[i - 1] + h / 2,
                               K, Gamma)
    k4 = h * system(y[i - 1] + k3, t[i - 1] + h, K, Gamma
                               )
    y[i] = y[i - 1] + (k1 + 2 * k2 + 2 * k3 + k4) / 6

    # Stop if pressure becomes negative
    if y[i, 0] <= 0:
        y[i:] = 0
        break

return y
```

Because we will vary the constants Gamma and K, we have to give them to the system every time by calling the function.

The system for the TOV equations is given by:

```python
def tov_equations(y, r, K, Gamma):
"""System der TOV-Gleichungen in geometrischen Einheiten.
"""
P, m = y
if P <= 0:  # Au erhalb des Sterns verschwindet der Druck
    return np.array([0, 0])

# rest density
rho_0 = (P / K) ** (1 / Gamma)

# relative energy contribution
epsilon = P / (rho_0 * (Gamma - 1))

# total density
rho = rho_0 * (1 + epsilon)

# precoutions for zero divition
if r < 1e-6:
    return np.array([0, 0])

denom = r**2 - 2 * m * r
if denom <= 1e-10:  #singularity save
    return np.array([0, 0])

P_prime = - (rho + P) * (m + 4 * np.pi * r**3 * P) / denom
m_prime = 4 * np.pi * r**2 * rho

return np.array([P_prime, m_prime])
```

The important point here is to take care not to divide by zero .

In the next segment of the code we defined the function, which takes the initial/ boundary conditions and solves the TOV equations.

```python
def make_star(rhoe, K, Gamma, r_max=10, dr=1e-3):
    """Erzeugung des Sterns in geometrischen Einheiten."""
    P_central = K * rhoe ** Gamma  # central preasure
    r = np.arange(dr, r_max, dr)
    y0 = np.array([P_central, 0])
    y = rk4(tov_equations, y0, r, dr, K, Gamma)
    P, m = y[:, 0], y[:, 1]

    surface_idx = np.argmax(P <= 0) - 1  # Last Index
    R, M = r[surface_idx], m[surface_idx]
    return r, P, m, R, M
```

We have to choose the maximum Radius. It's important to choose it so that we are at the outside of the star, to analyze the entire star.

Now we want to check the results for given Data in geometric units

$$\rho_0 = 8 \cdot 10^{-4}$$
$$\Gamma = 2.5$$
$$K = 3000$$

We get the following plots:

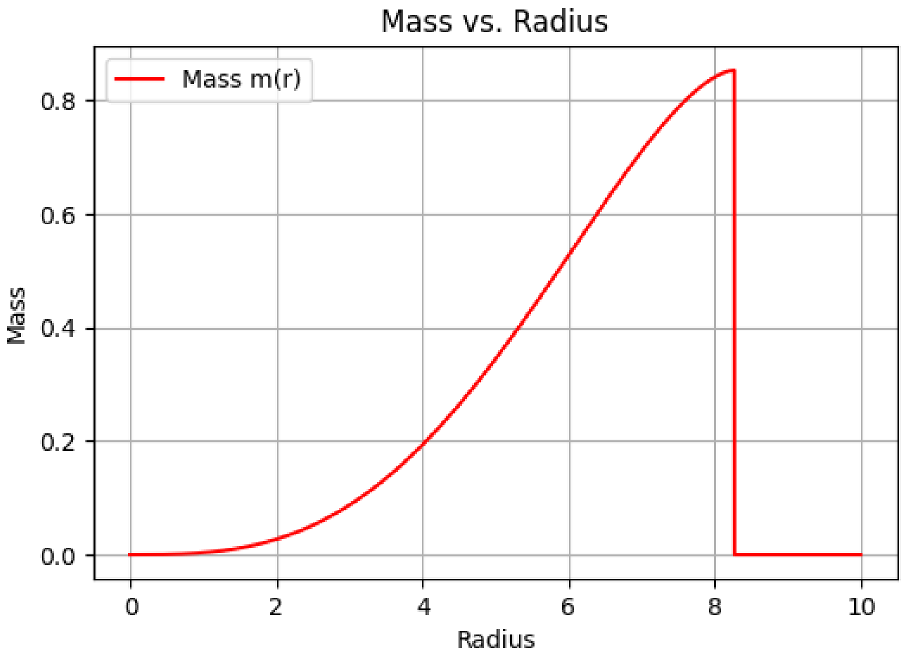

Figure 1: Mass-Radius

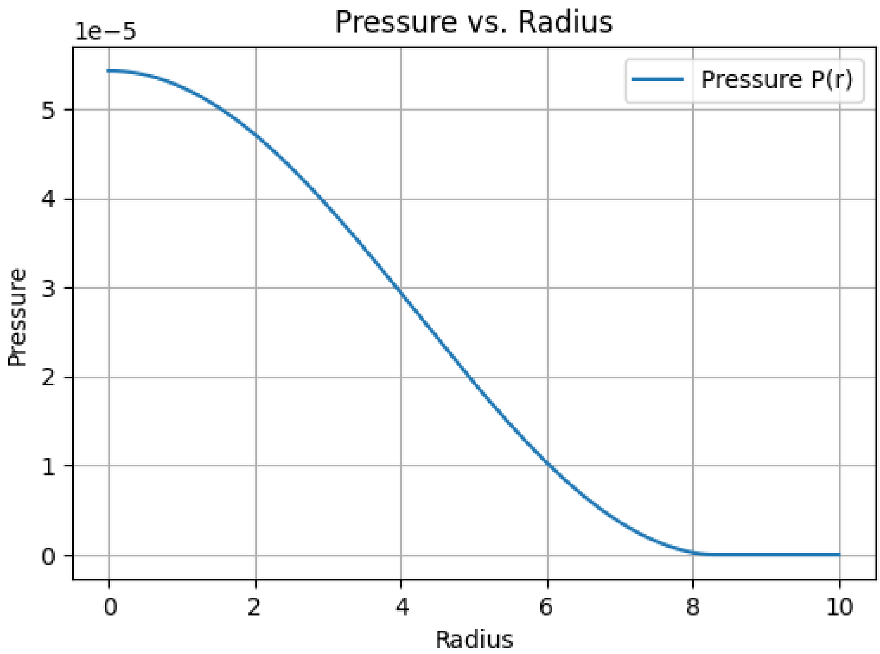

Figure 2: Pressure-Radius

For the total mass and the maximum radius in geometric units we get:

$$M = 0.8533$$
$$R = 8.2690$$

In the last step, we want to look at the total mass and the maximum radius for different K and Γ.

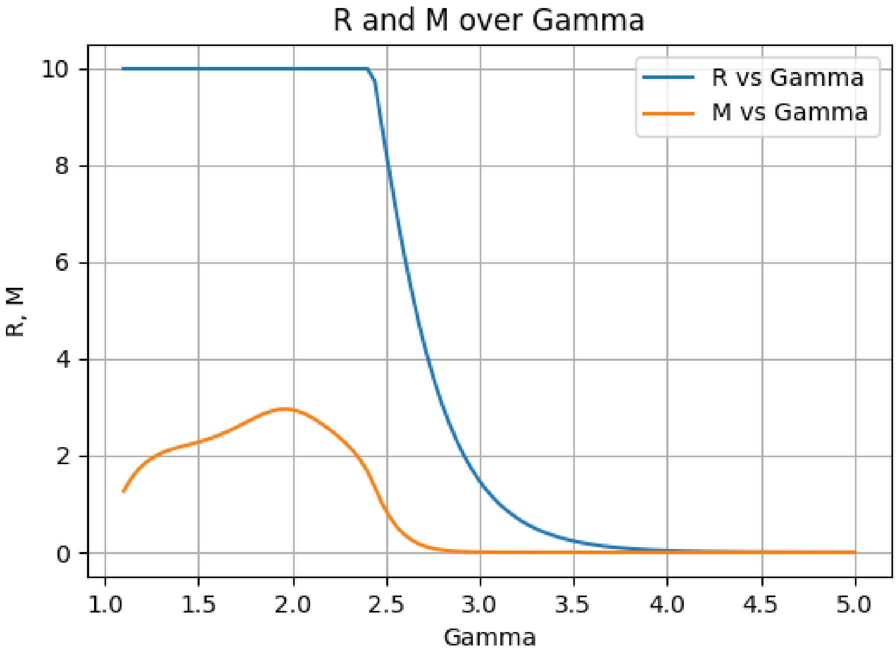

Figure 3: Maximum Mass-Radius for different Γ

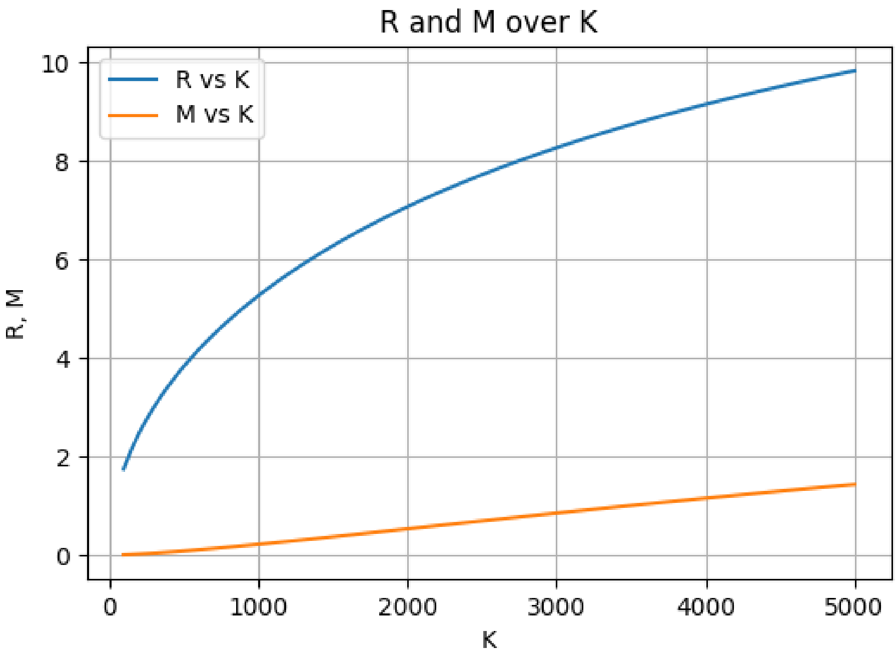

Figure 4: Maximum Mass-Radius for different K