Modeling the Tension in a Spherical Bacterium with Two Layers

We consider a spherical bacterium with two layers: a peptidoglycan (PG) layer and a plasma membrane (PM). The bacterium is assumed to have an internal pressure P, and we model the mechanical response using tension-strain relations for each layer.

The total tension balance on a single diameter is given by:

$$T_{\mathrm{PG}} + T_{\mathrm{PM}} = \frac{PD}{4}.$$

We define:

$$T_{\mathrm{PG}} = K_{\mathrm{PG}}\epsilon, \quad T_{\mathrm{PM}} = K_{\mathrm{PM}}\epsilon^n,$$

where ϵ is the strain and n is the strain-hardening exponent.

The strain is defined via the current diameter D and a reference diameter D_0:

$$\epsilon = \left(\frac{D}{D_0}\right)^2 - 1 \implies D = D_0\sqrt{\epsilon + 1}.$$

Substitute D into the tension balance:

$$K_{\mathrm{PG}}\epsilon + K_{\mathrm{PM}}\epsilon^n = \frac{PD_0\sqrt{\epsilon+1}}{4}.$$

Squaring both sides:

$$(K_{\mathrm{PG}}\epsilon + K_{\mathrm{PM}}\epsilon^n)^2 = \left(\frac{PD_0\sqrt{\epsilon+1}}{4}\right)^2.$$

Expanding the left-hand side:

$$K_{\mathrm{PG}}^2\epsilon^2 + 2K_{\mathrm{PG}}K_{\mathrm{PM}}\epsilon^{n+1} + K_{\mathrm{PM}}^2\epsilon^{2n} = \frac{P^2D_0^2(\epsilon+1)}{16}.$$

Rearranging:

$$K_{\mathrm{PM}}^2\epsilon^{2n} + 2K_{\mathrm{PG}}K_{\mathrm{PM}}\epsilon^{n+1} + K_{\mathrm{PG}}^2\epsilon^2 - \frac{P^2D_0^2(\epsilon+1)}{16} = 0.$$

For a given P, this equation can be solved numerically for ϵ. Once $\epsilon(P)$ is known:

$$T_{\mathrm{PG}}(P) = K_{\mathrm{PG}}\epsilon(P), \quad T_{\mathrm{PM}}(P) = K_{\mathrm{PM}}[\epsilon(P)]^n.$$

Below is a Python code snippet that uses a numerical method to solve for ϵ and plot T_{PG} and T_{PM} as functions of P for $n = 4$. You can adjust the parameters as needed.

```
import numpy as np
from scipy.optimize import fsolve
import matplotlib.pyplot as plt

# Given parameters (example values)
K_PG = 1.0
K_PM = 10.0
D_0 = 1
n = 4  # n=4 in this example

def equation(epsilon, P, K_PG, K_PM, D_0):
    # Equation: K_PG*epsilon + K_PM*(epsilon^n) - (P*D0*sqrt(epsilon+1))/4 = 0
    return K_PG*epsilon + K_PM*(epsilon**n) - (P * D_0 * np.sqrt(epsilon + 1) / 4.0)

# Define a range of pressures:
P_values = np.linspace(0.1, 5.0, 100)  # from P=0.1 to 5.0

epsilon_values = []
T_PG_values = []
T_PM_values = []

# Initial epsilon
epsilon_g = 0.0

for P in P_values:
    # Solve for epsilon:
    epsilon_sol = fsolve(lambda eps: equation(eps, P, K_PG, K_PM, D_0), epsilon_g)
    epsilon_sol = epsilon_sol[0]

    # Update guess for next iteration
    epsilon_guess = epsilon_sol

    # Calculate tensions:
    T_PG_val = K_PG*epsilon_sol
    T_PM_val = K_PM*(epsilon_sol**n)

    epsilon_values.append(epsilon_sol)
    T_PG_values.append(T_PG_val)
    T_PM_values.append(T_PM_val)

# Plot T_PG and T_PM vs P
plt.figure(figsize=(8,6))
plt.plot(P_values, T_PG_values, label='T_PG(P)')
plt.plot(P_values, T_PM_values, label='T_PM(P)')
plt.xlabel('Pressure (P)')
```

```
plt.ylabel('Tension')
plt.title('Tensions vs Pressure for n='+str(n))
plt.legend()
plt.grid(True)
plt.show()
```

Below is a plot illustrating the tension (T) versus the pressure (P) in both the PG and PM layers: