A computational model of a Deep Brain Stimulation (DBS) electrode implanted in the superior cerebellar peduncle (SCP) of a rat involves several steps, including defining the geometry, material properties, and electrical properties of the electrode and surrounding tissue. The goal is to determine the optimum current that maximizes the stimulation effect while minimizing unwanted side effects.

Below is a simplified Python code using the NEURON simulator, which is commonly used for modelling neural systems.

Step 1: Install Required Libraries

Make sure you have NEURON installed. You can install it using:

```
pip install neuron
```

Step 2: Define the Geometry and Properties

```
from neuron import h, gui

import numpy as np


# Load NEURON's GUI

h.load_file("stdrun.hoc")


# Define the geometry of the electrode

electrode_diameter = 0.1  # in mm

electrode_length = 1.0    # in mm

electrode_tip = h.Section(name='electrode_tip')

electrode_tip.nseg = 1

electrode_tip.diam = electrode_diameter

electrode_tip.L = electrode_length


# Define the material properties

electrode_tip.Ra = 100  # Axial resistance in Ohm*cm

electrode_tip.cm = 1    # Membrane capacitance in uF/cm^2


# Define the surrounding tissue properties
```

```
tissue_resistivity = 300  # Ohm*cm
```

Step 3: Define the Stimulation Protocol

```
# Define the stimulation protocol
stim = h.IClamp(electrode_tip(0.5))  # Place the electrode in the middle of the section
stim.dur = 1.0  # Duration of the stimulus in ms
stim.delay = 1.0  # Delay before the stimulus starts in ms

# Define a range of currents to test
currents = np.linspace(0.1, 1.0, 10)  # Currents from 0.1 to 1.0 mA in 10 steps
```

Step 4: Run the Simulation and Record Results

```
# Define a vector to record the membrane potential
v_vec = h.Vector()
t_vec = h.Vector()
v_vec.record(electrode_tip(0.5)._ref_v)
t_vec.record(h._ref_t)

# Run the simulation for each current
results = []
for i, current in enumerate(currents):
    stim.amp = current  # Set the current amplitude
    h.tstop = 10.0  # Set the simulation duration in ms
    h.run()

    # Store the results
    results.append((current, np.array(v_vec), np.array(t_vec)))

    # Reset the simulation
    h.finitialize(-65)
```

Step 5: Analyze the Results

```
import matplotlib.pyplot as plt


# Plot the results
plt.figure(figsize=(10, 6))
for current, v, t in results:
    plt.plot(t, v, label=f'{current} mA')


plt.xlabel('Time (ms)')
plt.ylabel('Membrane Potential (mV)')
plt.title('Membrane Potential Response to DBS Stimulation')
plt.legend()
plt.grid(True)
plt.show()


# Determine the optimum current based on the response
# For example, the current that produces the maximum depolarization without causing
excessive hyperpolarization
optimum_current = currents[np.argmax([np.max(v) for _, v, _ in results])]
print(f'Optimum current: {optimum_current} mA')
```

Explanation:

- **Geometry and Properties**: The electrode is modeled as a cylindrical section with specific dimensions and material properties.
- **Stimulation Protocol**: A current clamp is used to simulate the DBS stimulation. The current is varied to find the optimum value.
- **Simulation**: The simulation is run for each current value, and the membrane potential is recorded.
- **Analysis**: The results are plotted, and the optimum current is determined based on the membrane potential response.