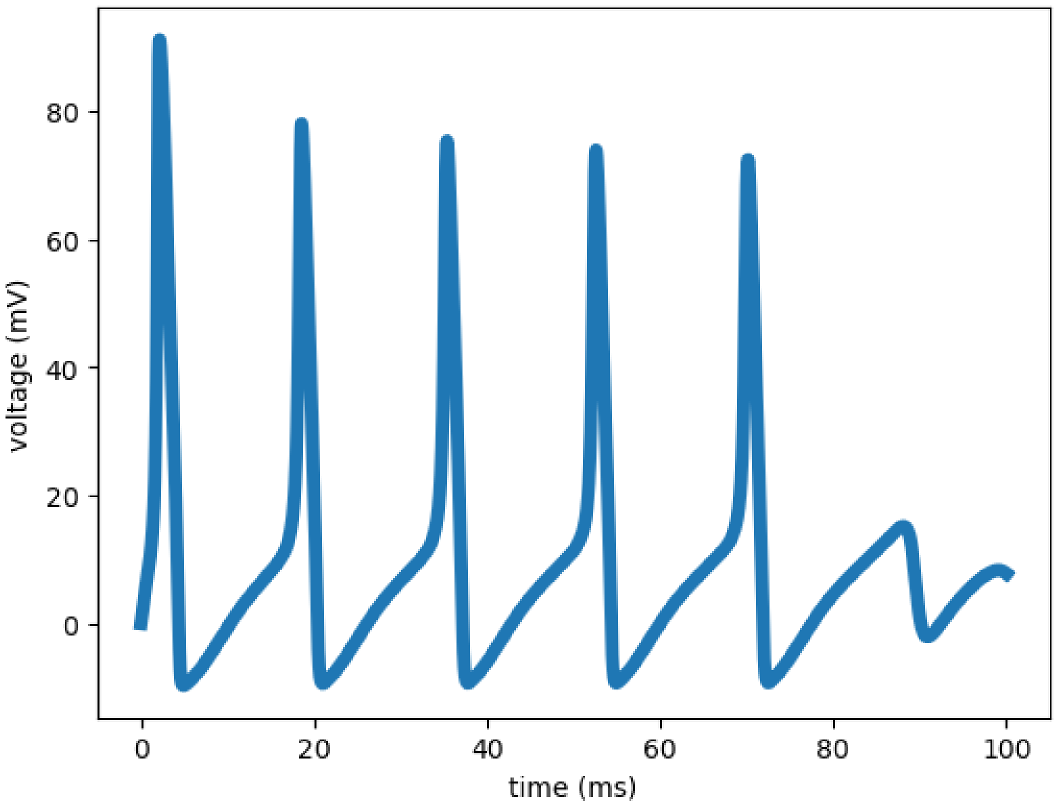
HH_Model_Homework-1

September 2, 2021

[69]:
```python
"""
hh_sim.py

simulates hodgkin-huxley model using scipy ode-integration package
"""

import matplotlib.pyplot as plt
import numpy as np

from scipy.integrate import odeint

T = 100.0     # end time (in milliseconds)
gK = 36.0    # average potassium channel conductance per unit area (mS/cm^2)
gNa = 120.0 # average sodium channel conductance per unit area (mS/cm^2)
gL = 0.3     # average leak channel conductance per unit area (mS/cm^2)
Cm = 1.0     # membrane capacitance per unit area (uF/cm^2)
EK = -12.0  # potassium potential (mV)
ENa = 100.0 # Sodium potential (mV)
EL = 10.6    # leak potential (mV)
Id = 11    # input current (mA)

# time vector
tvec = np.linspace(0, T, 10000)

# potassium ion-channel rate functions
def alpha_n(Vm):
    return (0.1-0.01*Vm)/(np.exp(1-0.1*Vm)-1)

def beta_n(Vm):
    return 0.125*np.exp(-Vm/80)

# sodium ion-channel rate functions
def alpha_m(Vm):
    return (2.5-0.1*Vm)/(np.exp(2.5-0.1*Vm)-1)

def beta_m(Vm):
    return 4*np.exp(-Vm/18)
```

```python
def alpha_h(Vm):
    return 0.07*np.exp(-Vm/20)

def beta_h(Vm):
    return 1/(np.exp(3-0.1*Vm)+1)

# n, m, and h steady-state values
def n_inf(Vm=0.0):
    return alpha_n(Vm) / (alpha_n(Vm) + beta_n(Vm))

def m_inf(Vm=0.0):
    return alpha_m(Vm) / (alpha_m(Vm) + beta_m(Vm))

def h_inf(Vm=0.0):
    return alpha_h(Vm) / (alpha_h(Vm) + beta_h(Vm))

# compute derivatives
def compute_derivatives(y, t0):
    dy = np.zeros((4,))

    Vm = y[0]
    n = y[1]
    m = y[2]
    h = y[3]

    # dVm/dt
    GK = (gK/Cm)*np.power(n,4.0)
    GNa = (gNa/Cm)*np.power(m,3.0)*h
    GL = gL/Cm

    dy[0] = (Id/Cm)-(GK*(Vm-EK))-(GNa*(Vm-ENa))-(GL*(Vm-EL))

    # dn/dt
    dy[1] = (alpha_n(Vm)*(1-n))-(beta_n(Vm)*n)

    # dm/dt
    dy[2] = (alpha_m(Vm)*(1-m))-(beta_m(Vm)*m)

    # dh/dt
    dy[3] = (alpha_h(Vm)*(1-h))-(beta_h(Vm)*h)

    return dy

# state (Vm, n, m, h)
Y = np.array([0.0, n_inf(), m_inf(), h_inf()])
```

```
# solve ODE system
# vy = (Vm[t0:tmax], n[t0:tmax], m[t0:tmax], h[t0:tmax])
Vy = odeint(compute_derivatives, Y, tvec)

# plot neuron potential
fig = plt.figure()
plt.plot(tvec, Vy[:, 0],linewidth=5)
plt.xlabel('time (ms)')
plt.ylabel('voltage (mV)')
plt.show()
```



4a - The minimal amount current to generate repetitive spikes is: 6.25

4b - The rheobase for $gk = 30$ is 2.76. The rheobase decreased because the potassium channel for the neuron is less than sodiums channel allowing sodium to flow easier into the neuron.

4c - The rheobase is now 11.9. The rheobase increased because the sodium channel decreased for the neuron therefore a higher Input is needed to create repetitive spikes.

4d - The rheobase is lower than part a because the potassium potential in the neuron is now 0 therefore a minimal amount of sodium is required to create a rheobase.

4e- The rheobase is higher than part a becuase the sodium potential in the neuron has decreased needing more Input to flow more sodium and potential into the neuron.

```
[ ]:
```