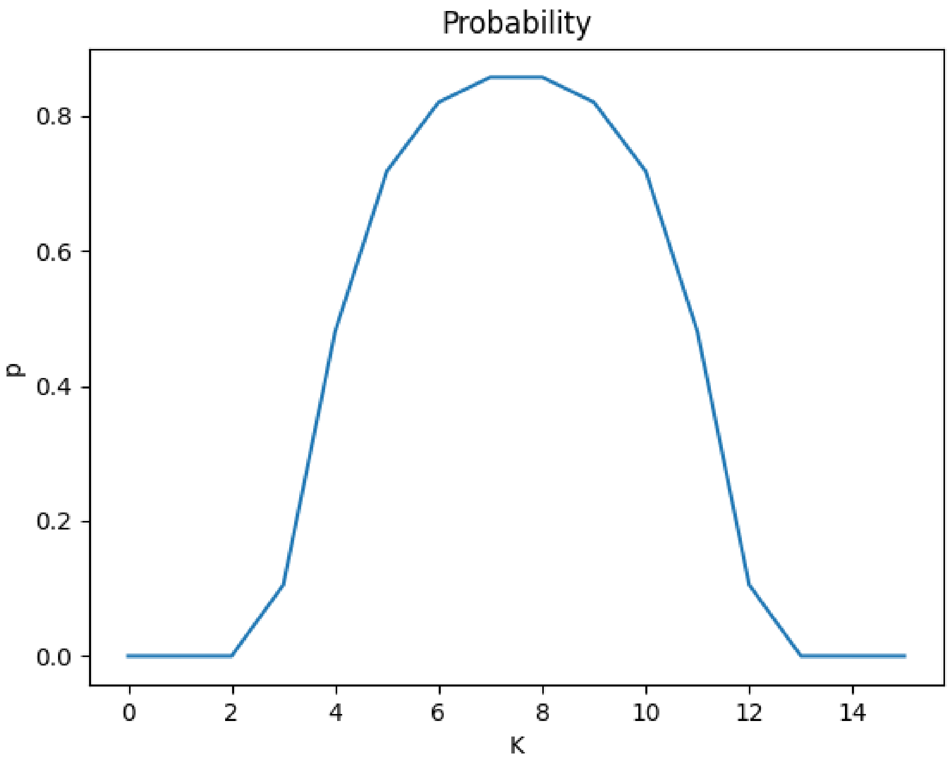
Problem Set I

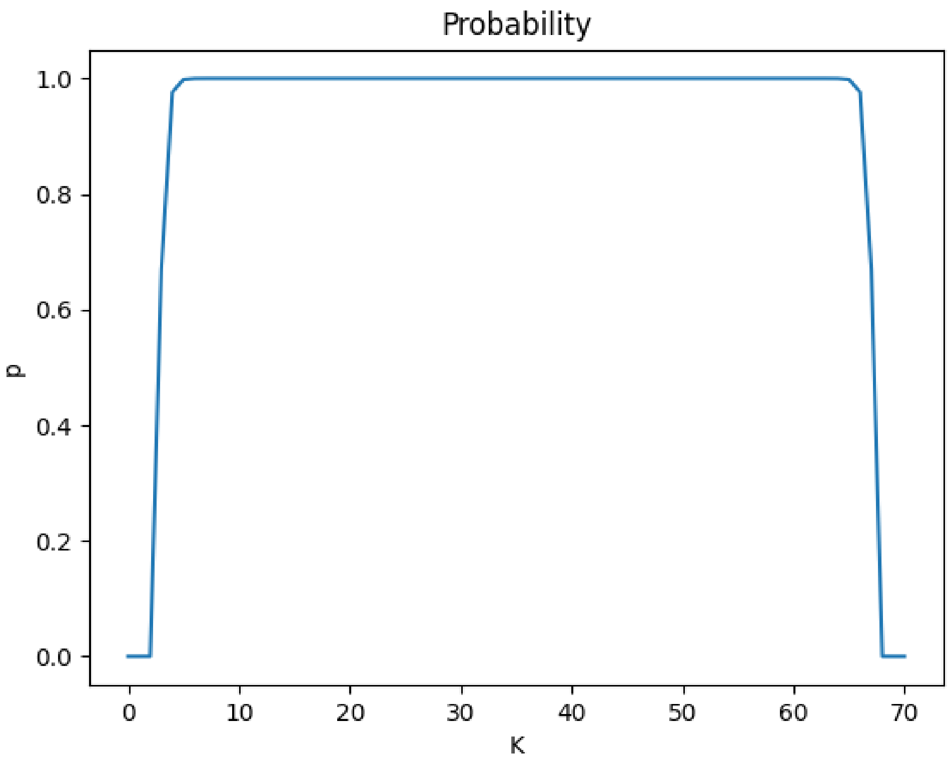
程玉锟 PB18020691

第一题

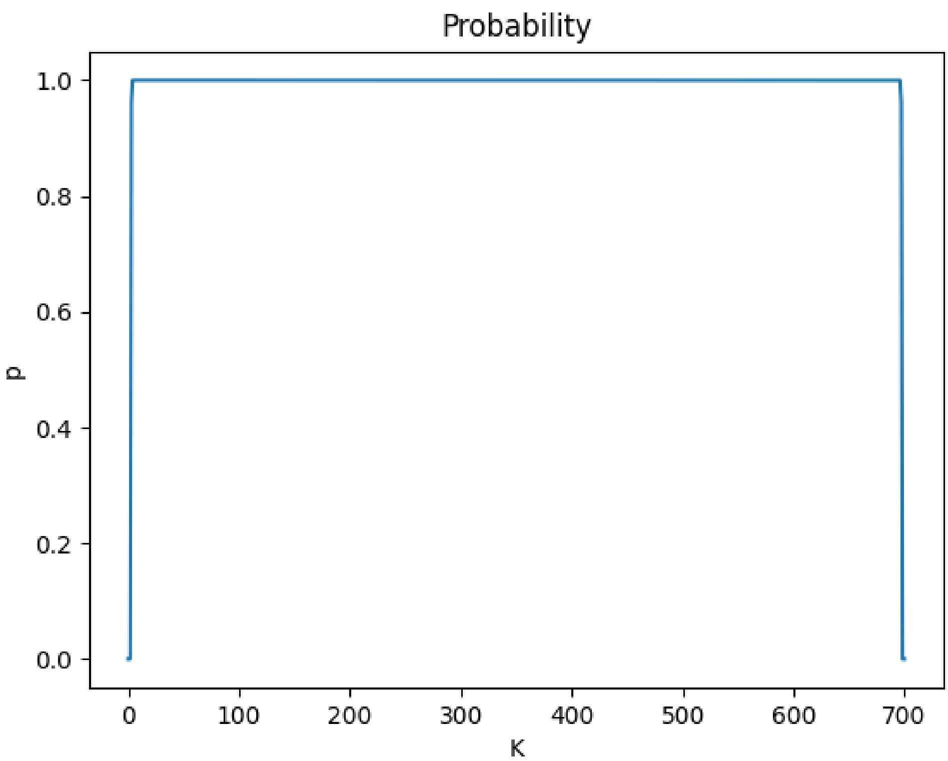
1. $p=\prod_{i=0}^{N-1}(1-\frac{i}{\binom{M}{K}})$
2. 取定N,M，绘制$K-p$关系图（画成折线图纯粹是便于观察变化趋势）
    - 取$N=45,M=15$



    - 取$N=210,M=70$



- 取$N=2100, M=700$



- N, M确定时，$\binom{M}{K}$越大，p越大，当$K=[\frac{M}{2}]$时（$M/2$向上向下两个取整都保留），有最大的$\binom{M}{K}$，也就有最大的p.
- 由于组合数是关于K具有对称关系的，p对于K也有对称关系.
- 也可以发现当M较大时，p会在K很小的时候迅速趋于1.

3. 将数据带入公式，

- $K=2$时，$p=0.00012298719078194986$
- $K=3$时，$p=0.99614882169200403$

可见，p将随K的增大迅速趋于1，p最大值的95%即为95%，$K=3, 6997$的时候p最接近95%.

4. M很大时，$\binom{M}{K}$随着K从0增大会迅速增加。K为一个小数字，可以使取得的输入的组合数较小，更容易取到所有的输入组合。此时K的变化也会引发组合数很大的变化，从而实现功能复杂度对K敏感的依赖关系。（K接近M的时候也可以有这样的效应，不过此时每个细胞获取输入的消耗过大，并不合理。）

第二题

求解问题 python 代码如下:

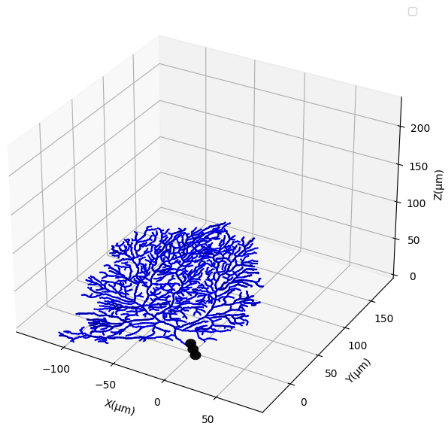
```
import matplotlib as mpl
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import LinearLocator, FormatStrFormatter


#导入文件模块

node = np.dtype([('index','i4'), ('type', 'i1'), ('x', 'f4'), ('y', 'f4'), ('z', 'f4'),
('diam', 'f4'),('father_index','i4')])

def NewNode(line_list):#将切分好的数据行转化为一个单元素np数组
    index=int(line_list[0])
    type=int(line_list[1])
    x=float(line_list[2])
    y=float(line_list[3])
    z=float(line_list[4])
    diam=float(line_list[5])
    father_index=int(line_list[6])
    return np.array([(index, type, x, y, z, diam, father_index)],node)

color = {0:'white',1:'black',2:'red',3:'blue',4:'purple'}

print("Loading *.swc File...")
path = input("Filepath:")
with open(path) as file:
    line = file.readline()
    while line[0]=="#":
        print(line)
        line = file.readline()
    NeurNode = np.empty([0], dtype = node, order = 'C')
    while (line):
        line_list = line.split()
        NeurNode=np.append(NeurNode,NewNode(line_list))
        line = file.readline()
print("File Loading Succeed!\n\n------------------------------------\n")
n=len(NeurNode)

#绘图模块
print("Ploting 3D Arbor Shape...")
mpl.rcParams['legend.fontsize'] = 10

fig1 = plt.figure()
ax = fig1.gca(projection='3d')
for i in range(n):
    pr=NeurNode[i]['diam']/2;
    theta = np.linspace(0 , np.pi, 20)
    phi = np.linspace(0 , 2 * np.pi, 20)
    theta,phi = np.meshgrid(theta,phi)
    px = NeurNode[i]['x']+pr*np.sin(theta)*np.cos(phi)
    py = NeurNode[i]['y']+pr*np.sin(theta)*np.sin(phi)
    pz = NeurNode[i]['z']+pr*np.cos(theta)
    surf = ax.plot_surface(px,py,pz,color=color[NeurNode[i]['type']])
    if(NeurNode[i]['father_index']!=(-1)):
        j=NeurNode[i]['father_index']-1
```

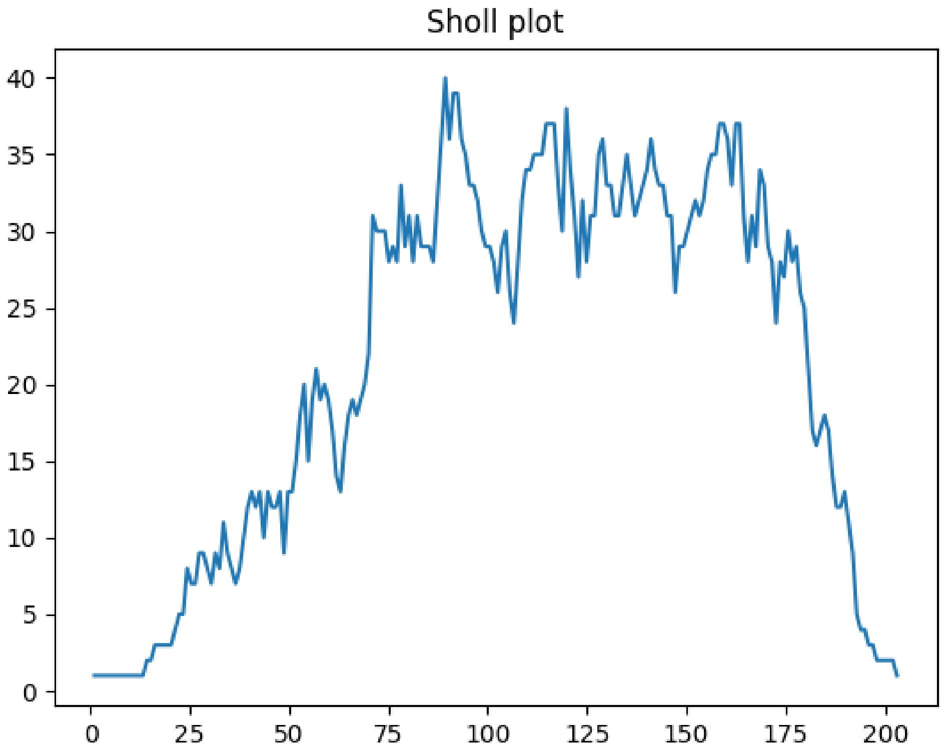
```
        ax.plot([NeurNode[i]['x'],NeurNode[j]['x']],[NeurNode[i]['y'],NeurNode[j]['y']],
[NeurNode[i]['z'],NeurNode[j]['z']],c=color[NeurNode[i]['type']])


coor=np.zeros([3,n],dtype='f4')
for i in range(n):
    coor[0,i]=NeurNode[i]['x']
    coor[1,i]=NeurNode[i]['y']
    coor[2,i]=NeurNode[i]['z']

cdis = np.zeros([3],dtype='f4')
cmax = np.zeros([3],dtype='f4')
cmin = np.zeros([3],dtype='f4')
for i in range(3):
    cmax[i]=np.amax(coor[i])
    cmin[i]=np.amin(coor[i])
    cdis[i]=cmax[i]-cmin[i]
dismax=np.amax(cdis)

ax.set_xlabel('X(μm)')
ax.set_xlim3d(cmin[0],cmin[0]+dismax)
ax.set_ylabel('Y(μm)')
ax.set_ylim3d(cmin[1],cmin[1]+dismax)
ax.set_zlabel('Z(μm)')
ax.set_zlim3d(cmin[2],cmin[2]+dismax)
ax.legend()

print("Ploting succeed! Please Close the \'Figure1\' Window to Continue...\n\n---------------
---------------------\n")

plt.show()

#计算分支点模块
#思路：遍历一遍所有的索引数，统计它在father_index中出现过几次，大于等于2次就算一个分支
print("Caculating Branching Points...")
branch_point = 0
for i in range(1,n+1,1):
    flag = 0
    for j in range(n):
        if(NeurNode[j]['type']==3):
            if (NeurNode[j]['father_index']==i):
                if (flag==0):
                    flag = 1
                elif (flag == 1):
                    branch_point += 1

print("branching points = ",branch_point)
print("\n------------------------------------\n")

#Sholl 分析
#思路：计算所有点到胞体点（father_index==-1）的距离，写入数组distance中，取一组离散的r，判断每一个点和它的
父点的（distance-r）之积是否小于等于0，如果是，记为一个交点
print("Sholl Analysising...")
for root in range(n):
    if(NeurNode[root]['father_index']==-1):
        break
distance = np.zeros([n], dtype = 'f4', order = 'C')
for i in range (n):
    dis = ((NeurNode[i]['x']-NeurNode[root]['x'])**2+(NeurNode[i]['y']-NeurNode[root]
['y'])**2+(NeurNode[i]['z']-NeurNode[root]['z'])**2)**0.5
    distance[i]=dis

def intersections(_r,_distance,_NeurNode):

    _lenthd = len(_distance)
```

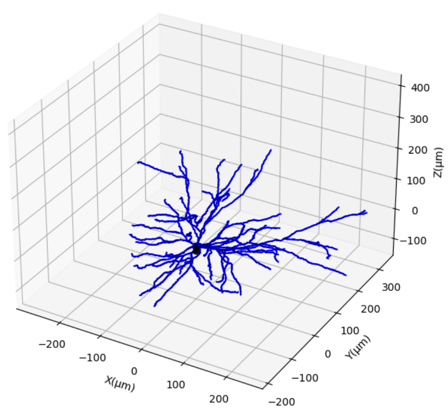
```
    _lenthr = len(_r)
    _inter = np.zeros([_lenthr],'i4','C')
    for j in range (_lenthr):
        for i in range (_lenthd):
            if(_NeurNode[i]['type']==3):
                if(((_distance[i]-_r[j])*(_distance[_NeurNode[i]['father_index']-1]-_r[j]))
<= 0.0):
                    _inter[j]+=1;
    return _inter;

dmax = np.amax(distance)
cut = 200

r = np.arange(dmax/cut,dmax+dmax/cut,dmax/cut)
y = intersections(r,distance,NeurNode)
plt.title("Sholl plot")
plt.plot(r, y)
print("Succeed!")
plt.show()
```

可以获得如下结果：

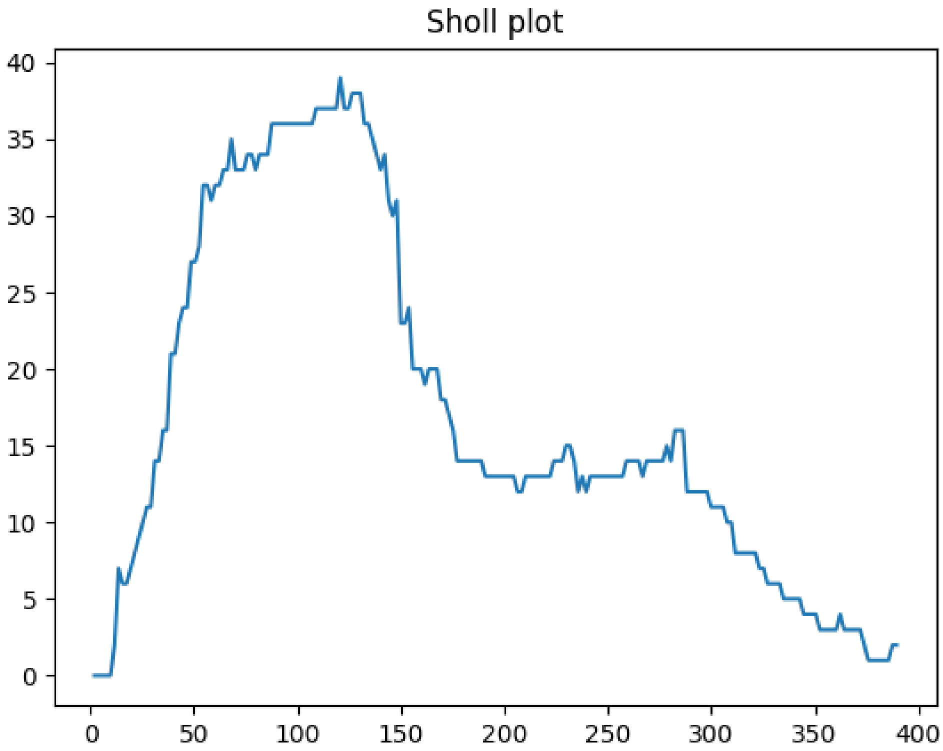
- Purkinjie cell: 树突分支节点数为378



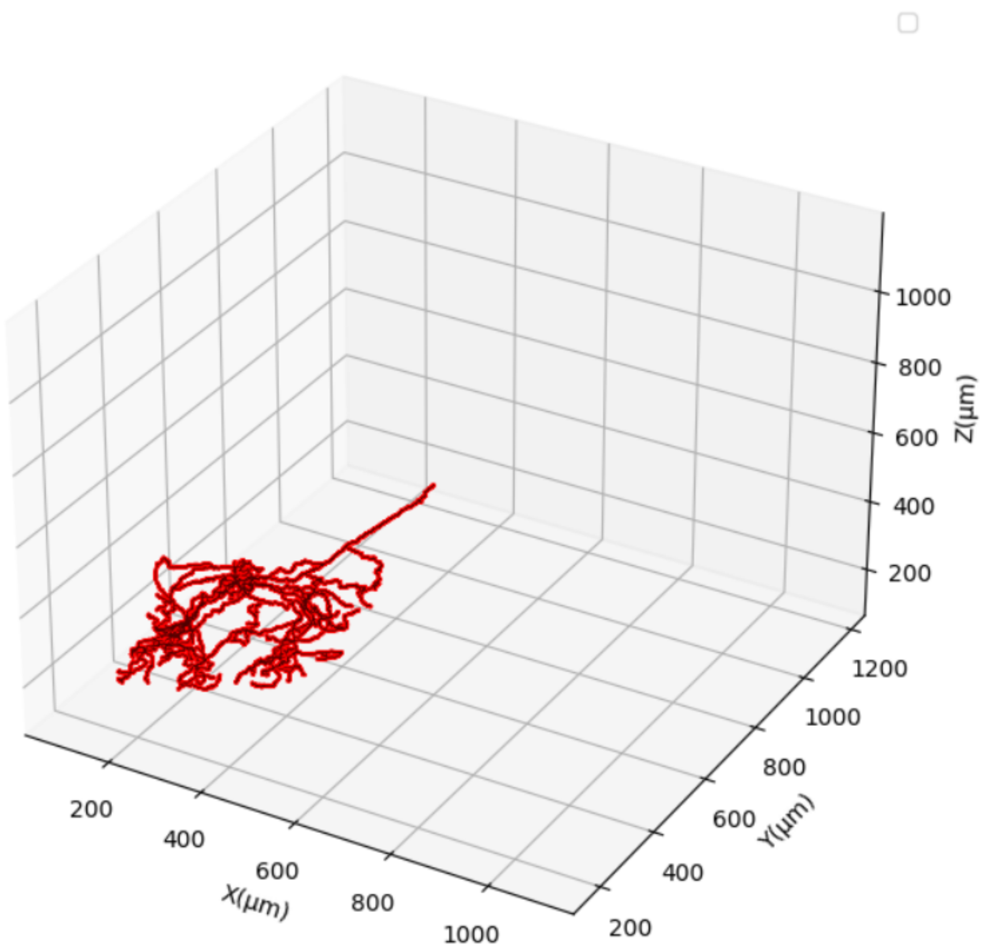


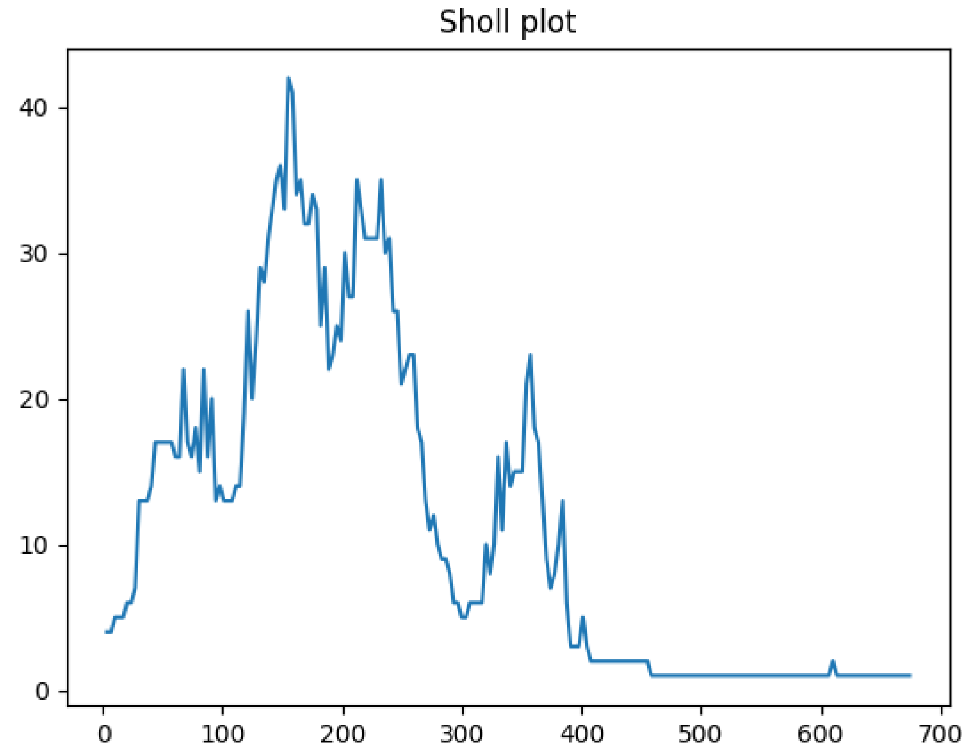
- pyramidal dendrite: 树突分支节点数53
- 

-



- one arbor from larval zebrafish: 分支节点数131，由于该细胞的文件中type一栏均为零，故未对节点类型进行区分。（上述代码片段不直接适用于本文件，需要修改一些参数方可使用。）





第三题

1. 设膜电位稳定在V_m，则此时膜两侧正离子净电荷流入与负离子净电荷流入代数和为零（取正电流入为正）。仅考虑带一个单位元电荷的离子，则有如下方程：
   - $\sum_{i=1}^{N} P_{M_i^+}[M_i^+]_{out} + \sum_{j=1}^{N} P_{A_j^-}[A_j^-]_{in} - \sum_{i=1}^{N} P_{M_i^+}[M_i^+]_{in}p(E \geq -eV_m) - \sum_{j=1}^{N} P_{A_j^-}[A_j^-]_{out}p(E \geq -eV_m) = 0$
2. 上述方程中概率$p(E \geq -eV_m)$可以由玻尔兹曼分布给出：
   - $p(E \geq -eV_m) = \frac{1}{k_BT}\int_{-eV_m}^{+\infty} e^{\frac{-E}{k_BT}} dE = e^{\frac{eV_m}{k_BT}}$
3. 方程可化为
   - $e^{\frac{eV_m}{k_BT}} = \frac{\sum_{i=1}^{N} P_{M_i^+}[M_i^+]_{out} + \sum_{j=1}^{N} P_{A_j^-}[A_j^-]_{in}}{\sum_{i=1}^{N} P_{M_i^+}[M_i^+]_{in} + \sum_{j=1}^{N} P_{A_j^-}[A_j^-]_{out}}$

   即为
   - $V_m = \frac{k_BT}{e}\ln\left(\frac{\sum_{i=1}^{N} P_{M_i^+}[M_i^+]_{out} + \sum_{j=1}^{N} P_{A_j^-}[A_j^-]_{in}}{\sum_{i=1}^{N} P_{M_i^+}[M_i^+]_{in} + \sum_{j=1}^{N} P_{A_j^-}[A_j^-]_{out}}\right)$
4. 对于带多个单位电荷的离子，有：
   - $\sum_{i=1}^{N} q_i P_{M_i^{q_i+}}[M_i^{q_i+}]_{out} + \sum_{j=1}^{N} q_j P_{A_j^{q_j-}}[A_j^{q_j-}]_{in} - \sum_{i=1}^{N} q_i P_{M_i^{q_i+}}[M_i^{q_i+}]_{in} e^{\frac{q_iV_m}{k_BT}} - \sum_{j=1}^{N} q_j P_{A_j^{q_j-}}[A_j^{q_j-}]_{out} e^{\frac{q_jV_m}{k_BT}} = 0$

   上式不易继续化简。可以发现，$Goldman - Hodgkin - Katz$方程只适用于一价离子。

   （猜想：如果离子通道对电荷的选择性较强，几乎不会让带电量不同的离子通过，那么$Goldman - Hodgkin - Katz$方程也可以简单地推广到高价离子的情形，只需将式中的e换成q即可。）

第四题

1. 先不考虑阈值问题，单纯对如下微分方程进行求解
   - $C\frac{dV}{dt} = -\frac{V}{R} + I(t)$，其中$I(t) = Q\sum_{k=-\infty}^{\infty}\delta(t-kT)$

   可以解得：
   - $V(t) = (\frac{Q}{C}\sum_{k=-\infty}^{\infty} e^{\frac{kT}{RC}} H(t-kT) + A)e^{\frac{-t}{RC}}$

其中，$H(x)$为单位阶跃函数，A为待定常数。

2. 考察$V(t)$，可以发现其具有如下性质：
    - 平移不变性：将原点平移kT个单位长度，即令$\tau = t - kT$，$V(\tau)$的形式不会改变。
    - 稳定性：t足够大时，$V(t)$重复某个固定的周期进行振动。
        - 设当$t = kT$时$V(kT) = V$，则经过T之后：
            - $V((k+1)T) = Ve^{\frac{-T}{RC}} + \frac{Q}{C}$
        - 一个T后V改变量$\Delta V = \frac{Q}{C} - V(1 - e^{\frac{-T}{RC}})$，定义$V_\infty = \frac{Q}{C}(1 - e^{\frac{-T}{RC}})^{-1}$
            - 当$V > V_\infty$时，$\Delta V < 0$;
            - 当$V < V_\infty$时，$\Delta V > 0$;
            - 当$V = V_\infty$时，$\Delta V = 0$.
        - 可见k足够大时，$V(kT)$会稳定在V_∞附近。
3. 由$V(t)$的性质，可以进一步考察考虑阈值电位V_{th}后的情况：
    - 如果$V_{th} \geq V_\infty$，至多在k为负无穷时触发过一次阈值。此处考虑0时刻已经经过了无限长的时间，$V(0) = V_\infty$，之后一直保持稳定振动，始终未触发阈值。
    - 如果$\frac{Q}{C} \leq V_{th} \leq V_\infty$，将多次触发阈值。此时总可以取$V(0) = 0$，之后电位将经过多次震荡式的增长直到触发阈值，电位归零，开始重复循环。触发频率为$1/nT$，其中n可以通过ΔV数列求和求得。
    - 如果$\frac{Q}{C} \geq V_{th}$，每个kT都将触发阈值，随后电位归零，$V(t)$表现为形如$\{x = kT, y > 0\}$的一族直线。触发频率为$1/T$。