# Visualization of dendritic morphology

Due Wednesday, Sep 19, 2018

算法说明：通过查阅文献，现已有成熟的用于做 Sholl Analysis 的软件，可以直接分析神经元的图片。由于作业中突触的坐标和系数已给，故编程判断 branches 和 intersections 的算法简化了很多。只需寻找子节点数目大于等于 2 的数目即为分叉点；比较每个节点到胞体的距离即可得到 intersections 的数目。

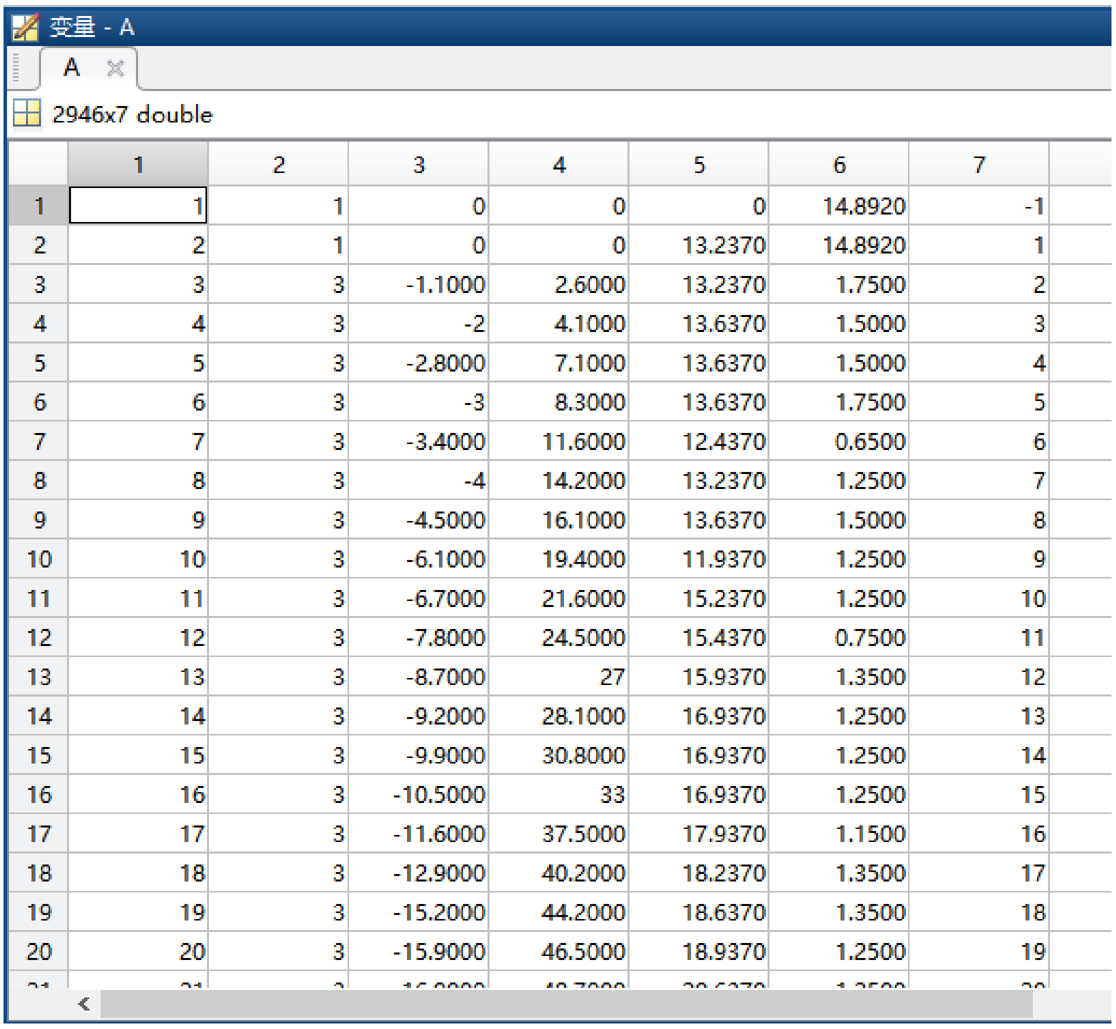
## Ⅰ. pyramidal dendrite

### 1. Load the data file

Code:

```
%Load the data file
filename = 'j8_L23pc.CNG.swc.txt';
delimiterIn = ' ';
headerlinesIn = 0;
A = importdata(filename,delimiterIn,headerlinesIn);

segmentindex=A(:,1);
segment_type=A(:,2);
x=A(:,3);
y=A(:,4);
z=A(:,5);
a=[x y z];
segment_diameter=A(:,6);
father_segment_index=A(:,7);
```

Result:


变量 - A

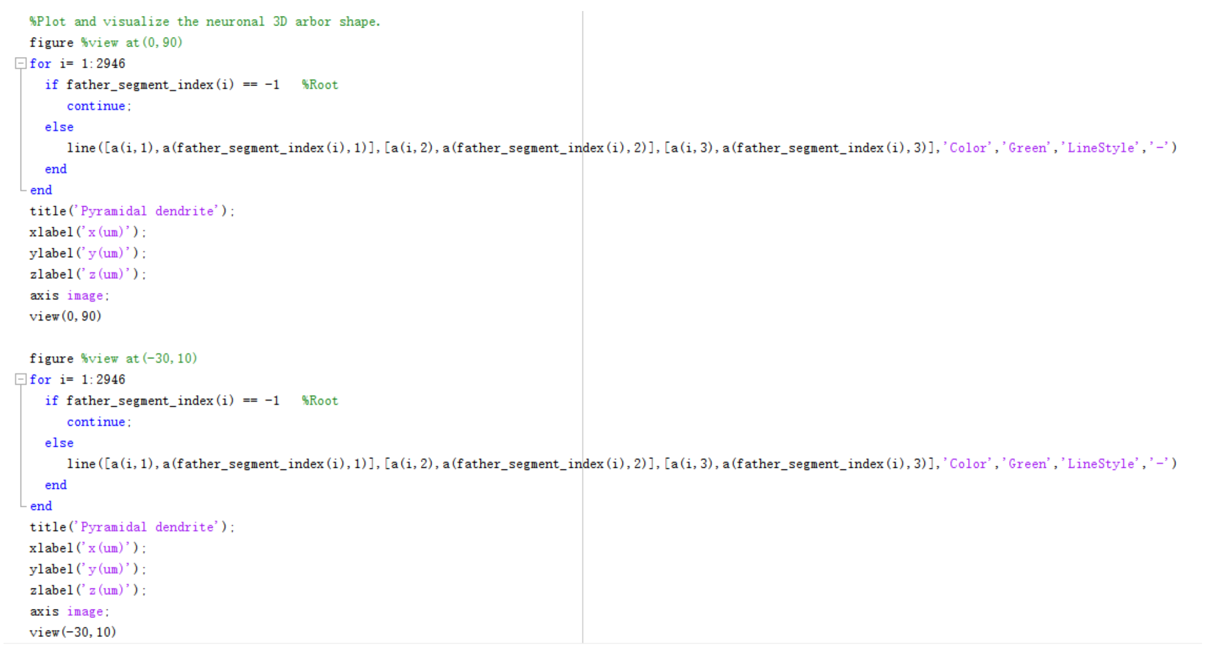
A

2946x7 double

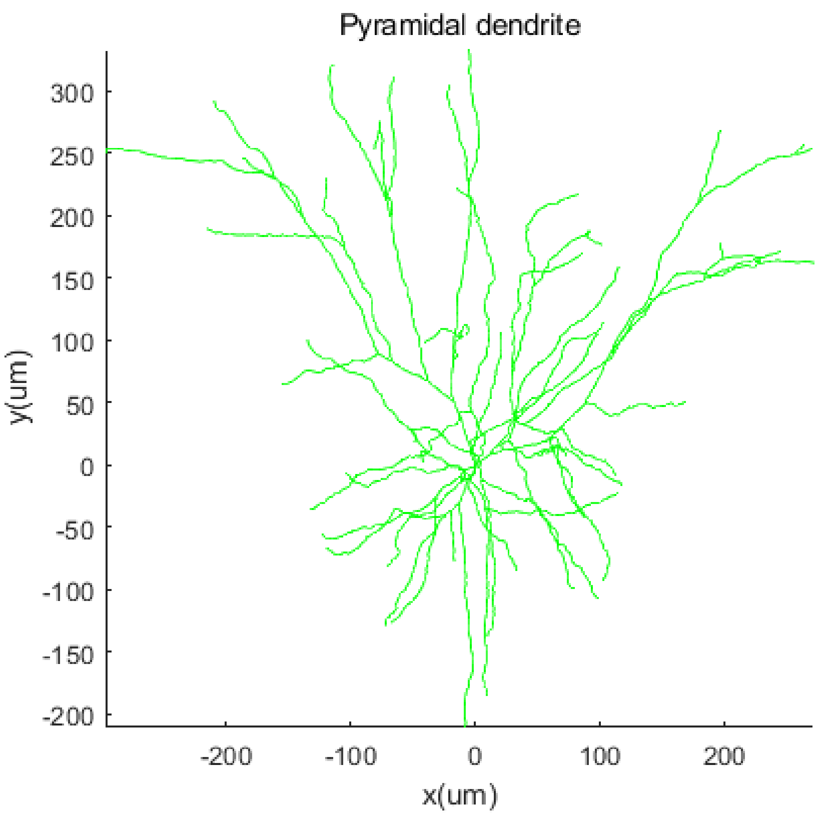
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 14.8920 | -1 |
| 2 | 2 | 1 | 0 | 0 | 13.2370 | 14.8920 | 1 |
| 3 | 3 | 3 | -1.1000 | 2.6000 | 13.2370 | 1.7500 | 2 |
| 4 | 4 | 3 | -2 | 4.1000 | 13.6370 | 1.5000 | 3 |
| 5 | 5 | 3 | -2.8000 | 7.1000 | 13.6370 | 1.5000 | 4 |
| 6 | 6 | 3 | -3 | 8.3000 | 13.6370 | 1.7500 | 5 |
| 7 | 7 | 3 | -3.4000 | 11.6000 | 12.4370 | 0.6500 | 6 |
| 8 | 8 | 3 | -4 | 14.2000 | 13.2370 | 1.2500 | 7 |
| 9 | 9 | 3 | -4.5000 | 16.1000 | 13.6370 | 1.5000 | 8 |
| 10 | 10 | 3 | -6.1000 | 19.4000 | 11.9370 | 1.2500 | 9 |
| 11 | 11 | 3 | -6.7000 | 21.6000 | 15.2370 | 1.2500 | 10 |
| 12 | 12 | 3 | -7.8000 | 24.5000 | 15.4370 | 0.7500 | 11 |
| 13 | 13 | 3 | -8.7000 | 27 | 15.9370 | 1.3500 | 12 |
| 14 | 14 | 3 | -9.2000 | 28.1000 | 16.9370 | 1.2500 | 13 |
| 15 | 15 | 3 | -9.9000 | 30.8000 | 16.9370 | 1.2500 | 14 |
| 16 | 16 | 3 | -10.5000 | 33 | 16.9370 | 1.2500 | 15 |
| 17 | 17 | 3 | -11.6000 | 37.5000 | 17.9370 | 1.1500 | 16 |
| 18 | 18 | 3 | -12.9000 | 40.2000 | 18.2370 | 1.3500 | 17 |
| 19 | 19 | 3 | -15.2000 | 44.2000 | 18.6370 | 1.3500 | 18 |
| 20 | 20 | 3 | -15.9000 | 46.5000 | 18.9370 | 1.2500 | 19 |

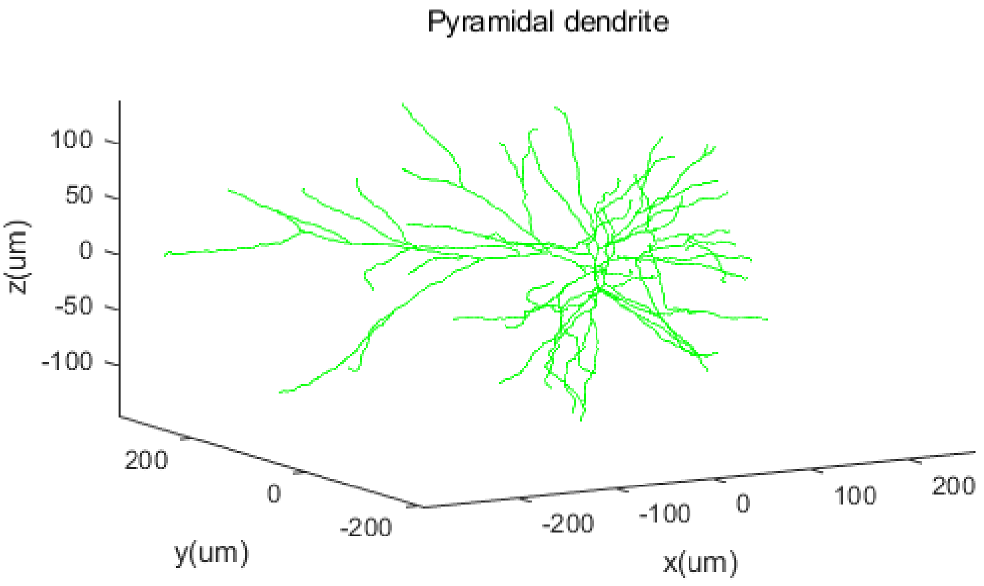
2. Plot and visualize the neuronal 3D arbor shape.

Code:


```
%Plot and visualize the neuronal 3D arbor shape.
figure %view at(0,90)
for i= 1:2946
    if father_segment_index(i) == -1   %Root
        continue;
    else
        line([a(i,1),a(father_segment_index(i),1)],[a(i,2),a(father_segment_index(i),2)],[a(i,3),a(father_segment_index(i),3)],'Color','Green','LineStyle','-')
    end
end
title('Pyramidal dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
axis image;
view(0,90)

figure %view at(-30,10)
for i= 1:2946
    if father_segment_index(i) == -1   %Root
        continue;
    else
        line([a(i,1),a(father_segment_index(i),1)],[a(i,2),a(father_segment_index(i),2)],[a(i,3),a(father_segment_index(i),3)],'Color','Green','LineStyle','-')
    end
end
title('Pyramidal dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
axis image;
view(-30,10)
```

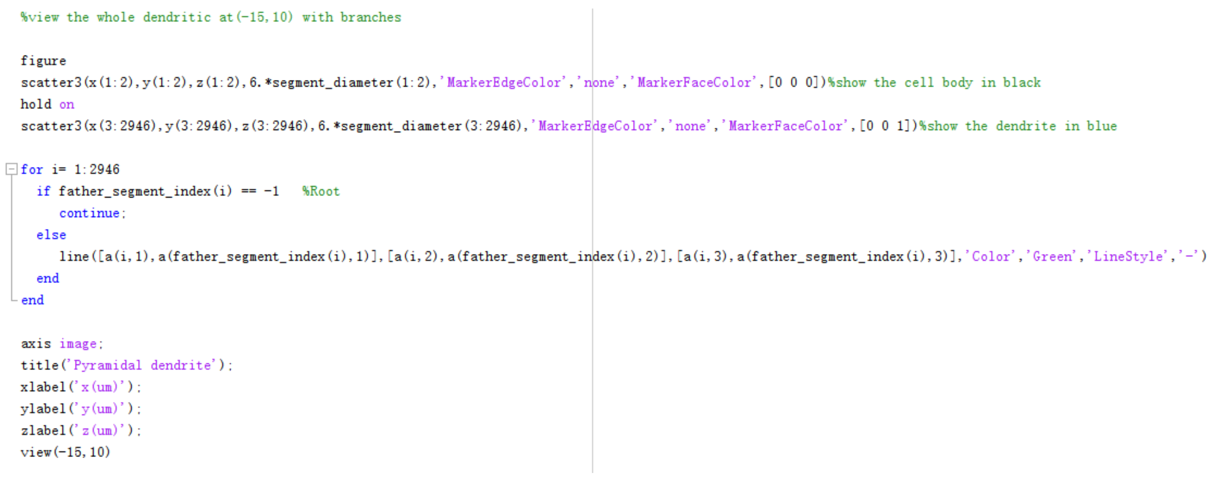
Result:


Pyramidal dendrite
300
250
200
150
100
50
0
-50
-100
-150
-200
y(um)
-200
-100
0
100
200
x(um)

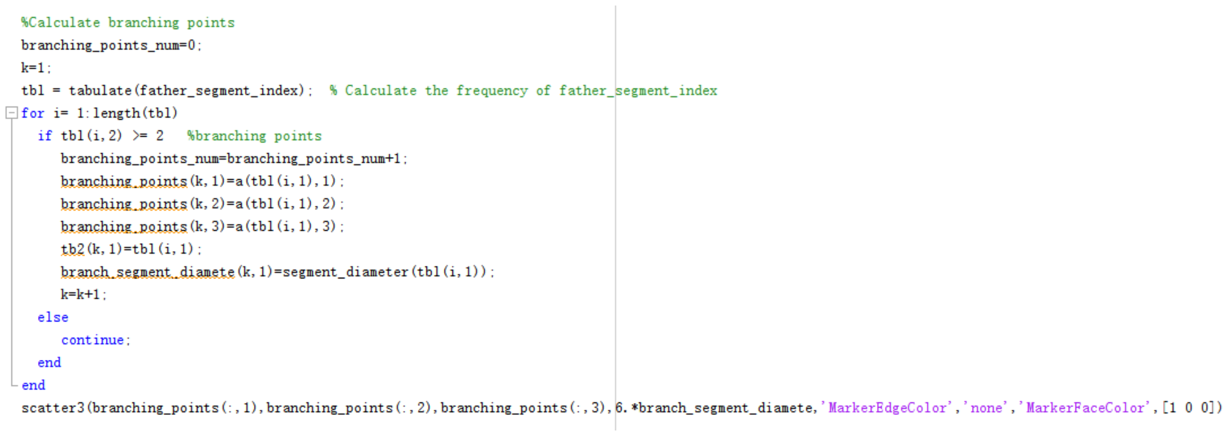

Pyramidal dendrite
100
50
0
-50
-100
z(um)
200
0
-200
y(um)
-200
-100
0
100
200
x(um)

3. Calculate how many branching points on both the pyramidal and Purkinjie dendrite.

Code:

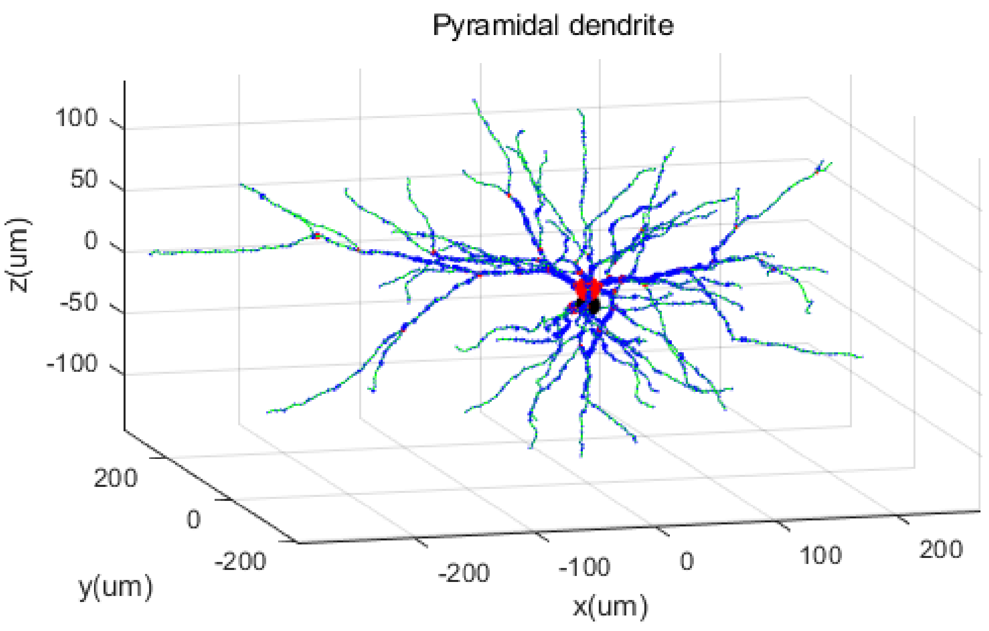


```
%view the whole dendritic at(-15,10) with branches

figure
scatter3(x(1:2),y(1:2),z(1:2),6.*segment_diameter(1:2),'MarkerEdgeColor','none','MarkerFaceColor',[0 0 0])%show the cell body in black
hold on
scatter3(x(3:2946),y(3:2946),z(3:2946),6.*segment_diameter(3:2946),'MarkerEdgeColor','none','MarkerFaceColor',[0 0 1])%show the dendrite in blue

for i= 1:2946
  if father_segment_index(i) == -1   %Root
     continue;
  else
     line([a(i,1),a(father_segment_index(i),1)],[a(i,2),a(father_segment_index(i),2)],[a(i,3),a(father_segment_index(i),3)],'Color','Green','LineStyle','-')
  end
end

axis image;
title('Pyramidal dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
view(-15,10)
```



```
%Calculate branching points
branching_points_num=0;
k=1;
tbl = tabulate(father_segment_index);  % Calculate the frequency of father_segment_index
for i= 1:length(tbl)
  if tbl(i,2) >= 2   %branching points
     branching_points_num=branching_points_num+1;
     branching_points(k,1)=a(tbl(i,1),1);
     branching_points(k,2)=a(tbl(i,1),2);
     branching_points(k,3)=a(tbl(i,1),3);
     tb2(k,1)=tbl(i,1);
     branch_segment_diamete(k,1)=segment_diameter(tbl(i,1));
     k=k+1;
  else
     continue;
  end
end
scatter3(branching_points(:,1),branching_points(:,2),branching_points(:,3),6.*branch_segment_diamete,'MarkerEdgeColor','none','MarkerFaceColor',[1 0 0])
```

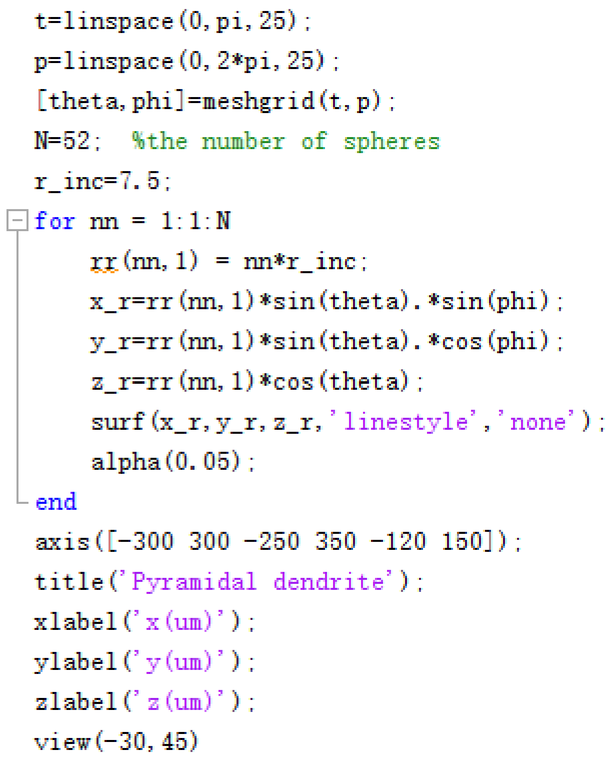
Result:



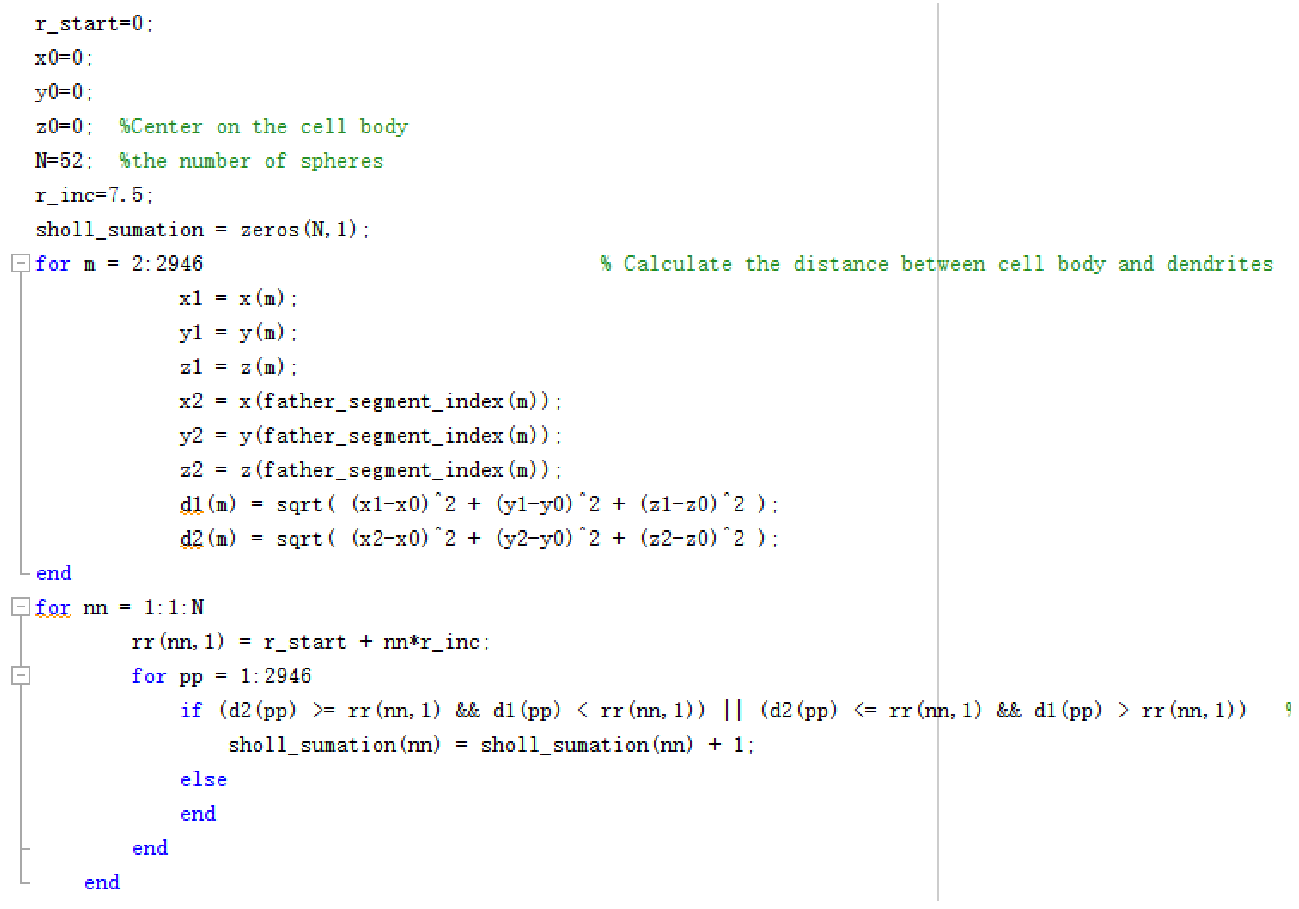
```
branching_points_num =

    51
```

4. Perform a Sholl plot. Center on the cell body and draw spheres, and plot the number of intersections between sphere and dendrites as a function of sphere radius.

Code:



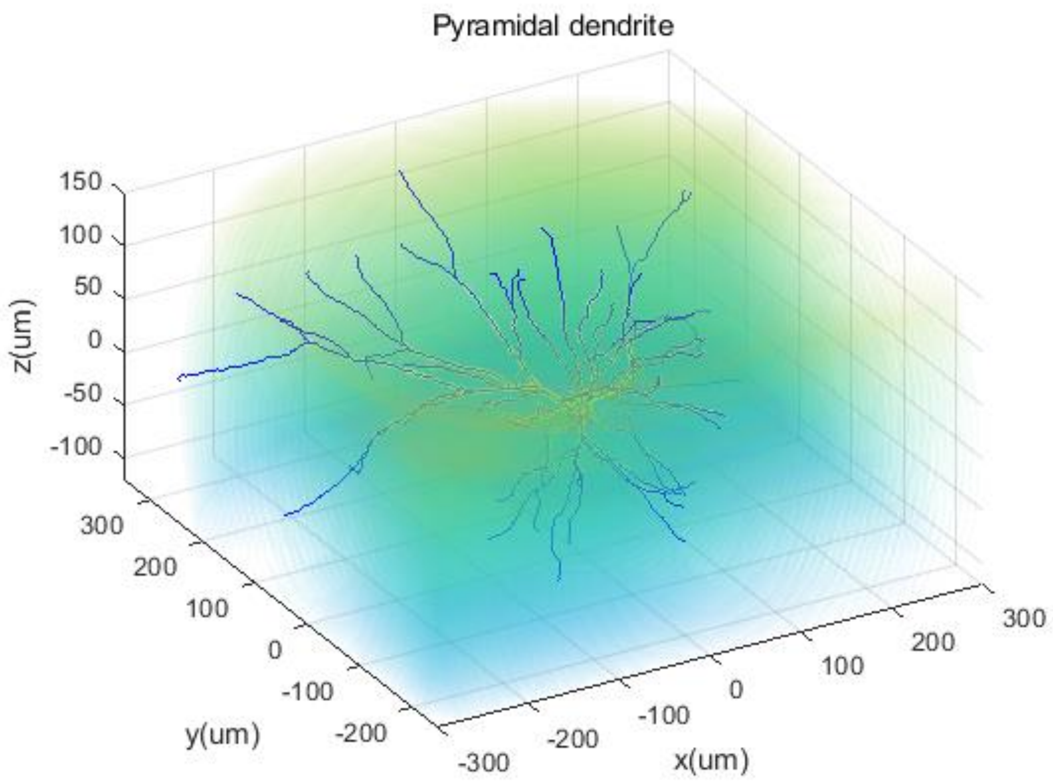
```
t=linspace(0,pi,25);
p=linspace(0,2*pi,25);
[theta,phi]=meshgrid(t,p);
N=52;  %the number of spheres
r_inc=7.5;
for nn = 1:1:N
    rr(nn,1) = nn*r_inc;
    x_r=rr(nn,1)*sin(theta).*sin(phi);
    y_r=rr(nn,1)*sin(theta).*cos(phi);
    z_r=rr(nn,1)*cos(theta);
    surf(x_r,y_r,z_r,'linestyle','none');
    alpha(0.05);
end
axis([-300 300 -250 350 -120 150]);
title('Pyramidal dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
view(-30,45)
```



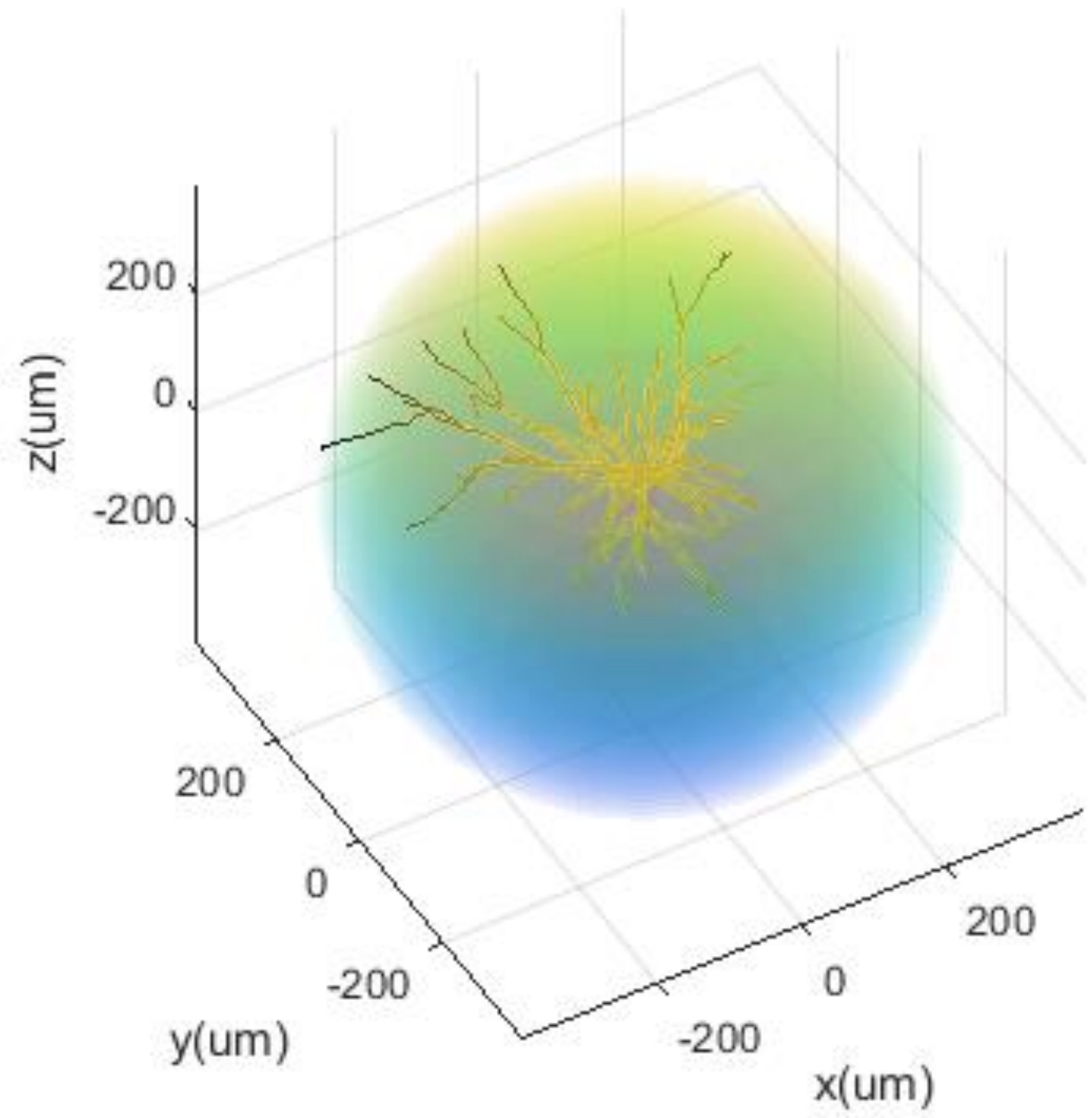
```
r_start=0;
x0=0;
y0=0;
z0=0;  %Center on the cell body
N=52;  %the number of spheres
r_inc=7.5;
sholl_sumation = zeros(N,1);
for m = 2:2946                                % Calculate the distance between cell body and dendrites
            x1 = x(m);
            y1 = y(m);
            z1 = z(m);
            x2 = x(father_segment_index(m));
            y2 = y(father_segment_index(m));
            z2 = z(father_segment_index(m));
            d1(m) = sqrt( (x1-x0)^2 + (y1-y0)^2 + (z1-z0)^2 );
            d2(m) = sqrt( (x2-x0)^2 + (y2-y0)^2 + (z2-z0)^2 );
end
for nn = 1:1:N
        rr(nn,1) = r_start + nn*r_inc;
        for pp = 1:2946
            if (d2(pp) >= rr(nn,1) && d1(pp) < rr(nn,1)) || (d2(pp) <= rr(nn,1) && d1(pp) > rr(nn,1))   %
                sholl_sumation(nn) = sholl_sumation(nn) + 1;
            else
            end
        end
    end
```

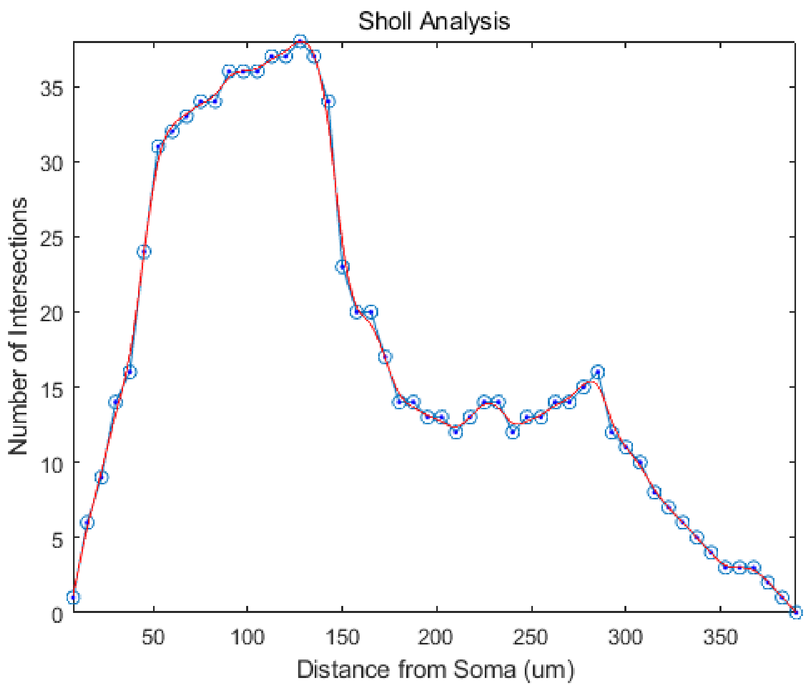
```
figure;
plot(rr,sholl_sumation,'o-');
hold on;
f=fit(rr,sholl_sumation,'smoothingspline')
plot(f,rr,sholl_sumation)
legend 'off';
title('Sholl Analysis');
ylabel('Number of Intersections'); xlabel('Distance from Soma (um)');
axis tight;

figure;
plot(rr(1:51,1),log10(sholl_sumation(1:51,1)./(4*pi*rr(1:51,1).^2)),'o-');
hold on;
f=fit(rr(1:51,1),log10(sholl_sumation(1:51,1)./(4*pi*rr(1:51,1).^2)),'poly1')
plot(f,rr(1:51,1),log10(sholl_sumation(1:51,1)./(4*pi*rr(1:51,1).^2)))
legend 'off';
title('Sholl Analysis');
ylabel('log10(N/S)'); xlabel('Distance from Soma (um)');
axis tight;
```

Result:



Pyramidal dendrite





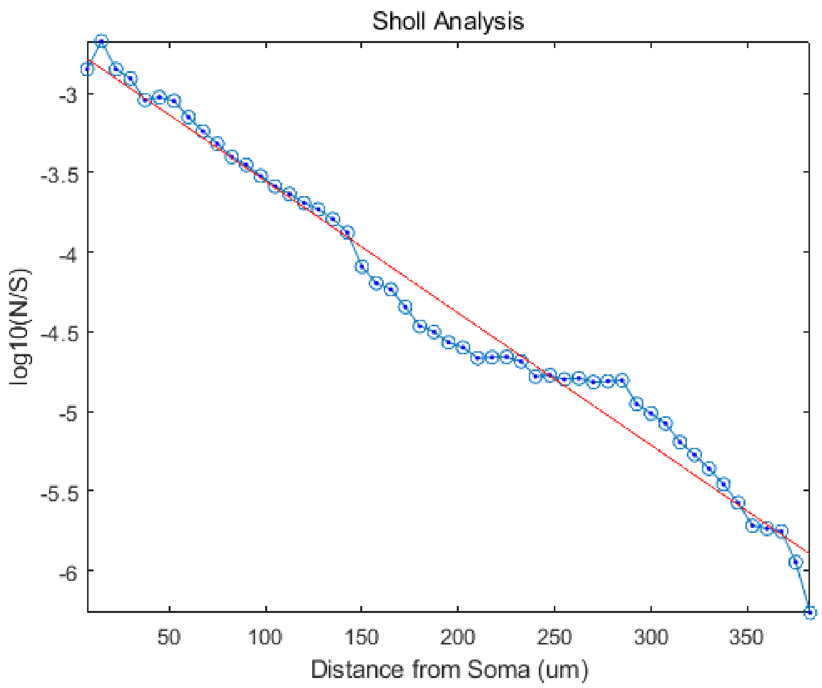
## Semi-Log Method [ edit ]

Somewhat more complicated than the Linear Method, the Semi-Log Method begins by calculating the function Y(r) = N/S where N is the number of dendrite crossings for a circle of radius r, and S is the area of that same circle. The base 10 logarithm is taken of this function, and a first order linear regression, linear fit, is performed on the resulting data set, that is

$$\log_{10}\left(\frac{N}{S}\right) = -k \cdot r + m.$$

where **k** is Sholl's Regression Coefficient.[1]

Sholl's Regression Coefficient is the measure of the change in density of dendrites as a function of distance from the cell body.[5] This method has been shown to have good discrimination value between various neuron types, and even similar types in different regions of the body.



```
Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
  p1 =   -0.008303  (-0.008653, -0.007953)
  p2 =       -2.72  (-2.799, -2.642)
```

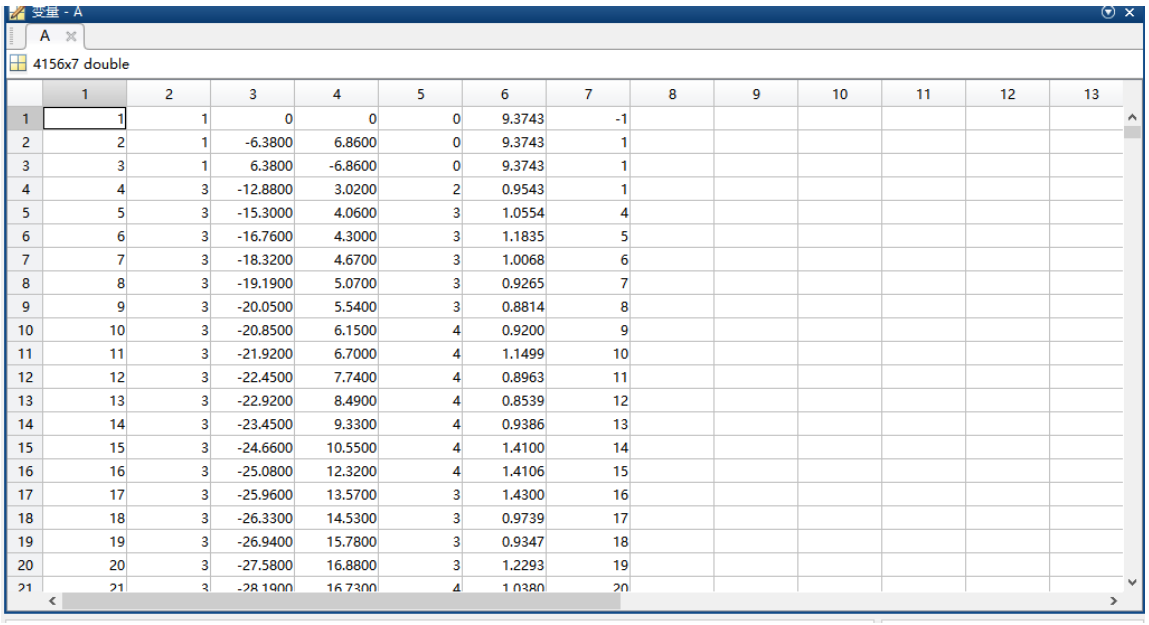
## Ⅱ. Purkinjie dendrite

### 1.Load the data file

Code:

```
%Load the data file
filename = 'Purkinje-slice-ageP43-6.CNG.swc.txt';
delimiterIn = ' ';
headerlinesIn = 0;
A = importdata(filename,delimiterIn,headerlinesIn);

segmentindex=A(:,1);
segment_type=A(:,2);
x=A(:,3);
y=A(:,4);
z=A(:,5);
a=[x y z];
segment_diameter=A(:,6);
father_segment_index=A(:,7);
```

Result:



变量 - A

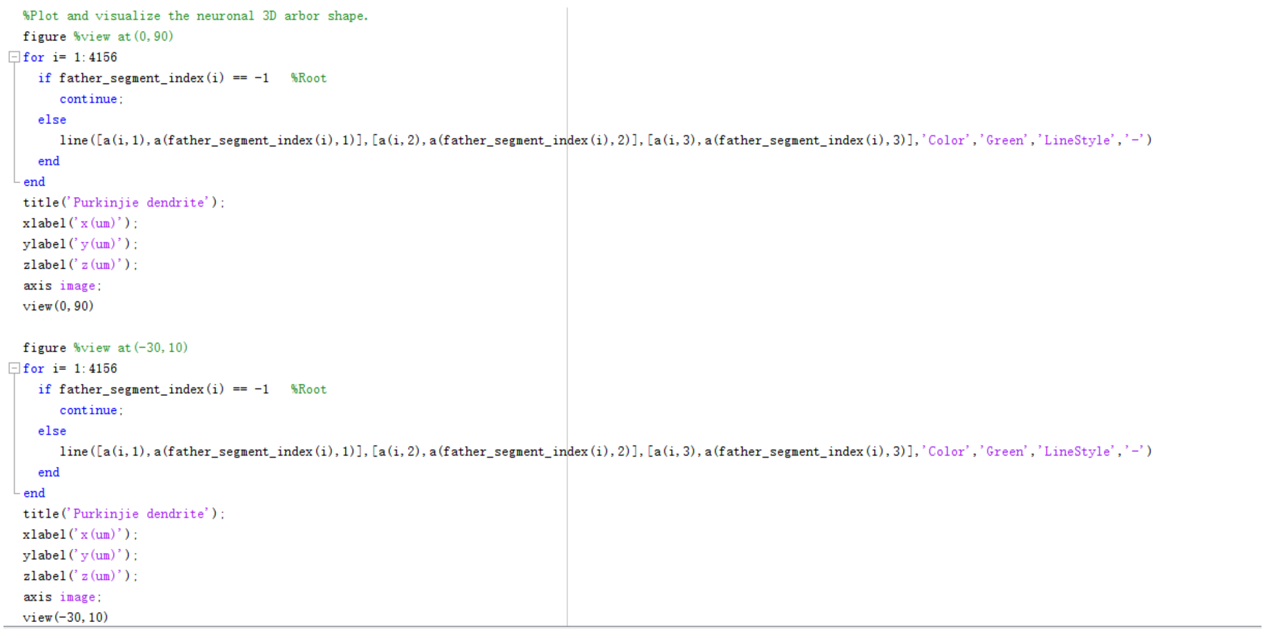
A

4156x7 double

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 | 0 | 9.3743 | -1 | | | | | | |
| 2 | 2 | 1 | -6.3800 | 6.8600 | 0 | 9.3743 | 1 | | | | | | |
| 3 | 3 | 1 | 6.3800 | -6.8600 | 0 | 9.3743 | 1 | | | | | | |
| 4 | 4 | 3 | -12.8800 | 3.0200 | 2 | 0.9543 | 1 | | | | | | |
| 5 | 5 | 3 | -15.3000 | 4.0600 | 3 | 1.0554 | 4 | | | | | | |
| 6 | 6 | 3 | -16.7600 | 4.3000 | 3 | 1.1835 | 5 | | | | | | |
| 7 | 7 | 3 | -18.3200 | 4.6700 | 3 | 1.0068 | 6 | | | | | | |
| 8 | 8 | 3 | -19.1900 | 5.0700 | 3 | 0.9265 | 7 | | | | | | |
| 9 | 9 | 3 | -20.0500 | 5.5400 | 3 | 0.8814 | 8 | | | | | | |
| 10 | 10 | 3 | -20.8500 | 6.1500 | 4 | 0.9200 | 9 | | | | | | |
| 11 | 11 | 3 | -21.9200 | 6.7000 | 4 | 1.1499 | 10 | | | | | | |
| 12 | 12 | 3 | -22.4500 | 7.7400 | 4 | 0.8963 | 11 | | | | | | |
| 13 | 13 | 3 | -22.9200 | 8.4900 | 4 | 0.8539 | 12 | | | | | | |
| 14 | 14 | 3 | -23.4500 | 9.3300 | 4 | 0.9386 | 13 | | | | | | |
| 15 | 15 | 3 | -24.6600 | 10.5500 | 4 | 1.4100 | 14 | | | | | | |
| 16 | 16 | 3 | -25.0800 | 12.3200 | 4 | 1.4106 | 15 | | | | | | |
| 17 | 17 | 3 | -25.9600 | 13.5700 | 3 | 1.4300 | 16 | | | | | | |
| 18 | 18 | 3 | -26.3300 | 14.5300 | 3 | 0.9739 | 17 | | | | | | |
| 19 | 19 | 3 | -26.9400 | 15.7800 | 3 | 0.9347 | 18 | | | | | | |
| 20 | 20 | 3 | -27.5800 | 16.8800 | 3 | 1.2293 | 19 | | | | | | |
| 21 | 21 | 3 | -28.1900 | 16.7300 | 4 | 1.0380 | 20 | | | | | | |

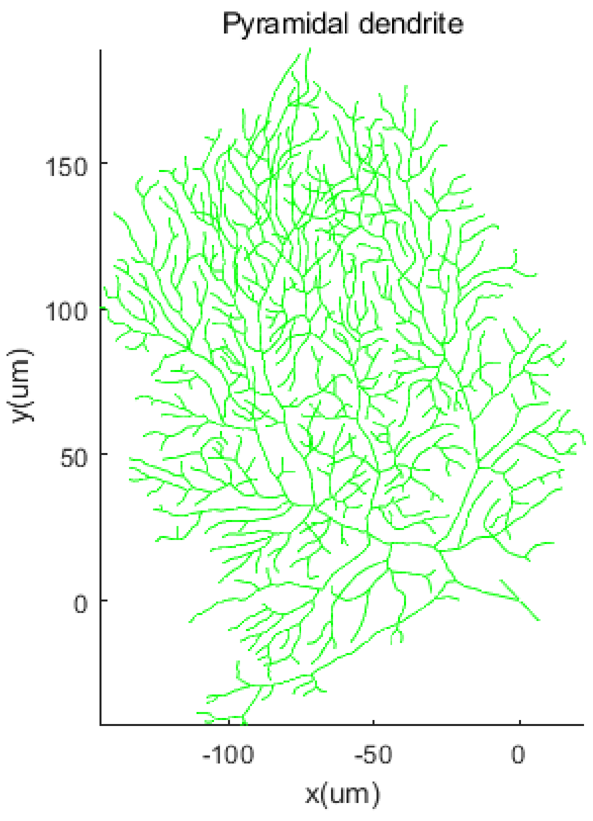
## 2.Plot and visualize the neuronal 3D arbor shape.

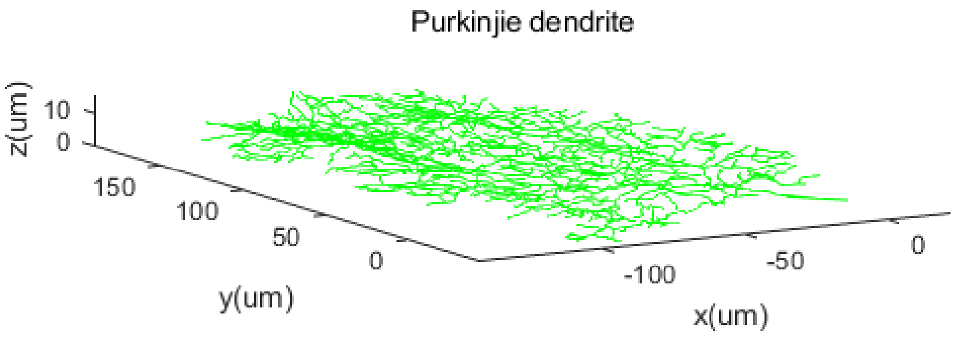
## Code:



```
%Plot and visualize the neuronal 3D arbor shape.
figure %view at(0,90)
for i= 1:4156
  if father_segment_index(i) == -1   %Root
      continue;
  else
      line([a(i,1),a(father_segment_index(i),1)],[a(i,2),a(father_segment_index(i),2)],[a(i,3),a(father_segment_index(i),3)],'Color','Green','LineStyle','-')
  end
end
title('Purkinjie dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
axis image;
view(0,90)

figure %view at(-30,10)
for i= 1:4156
  if father_segment_index(i) == -1   %Root
      continue;
  else
      line([a(i,1),a(father_segment_index(i),1)],[a(i,2),a(father_segment_index(i),2)],[a(i,3),a(father_segment_index(i),3)],'Color','Green','LineStyle','-')
  end
end
title('Purkinjie dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
axis image;
view(-30,10)
```

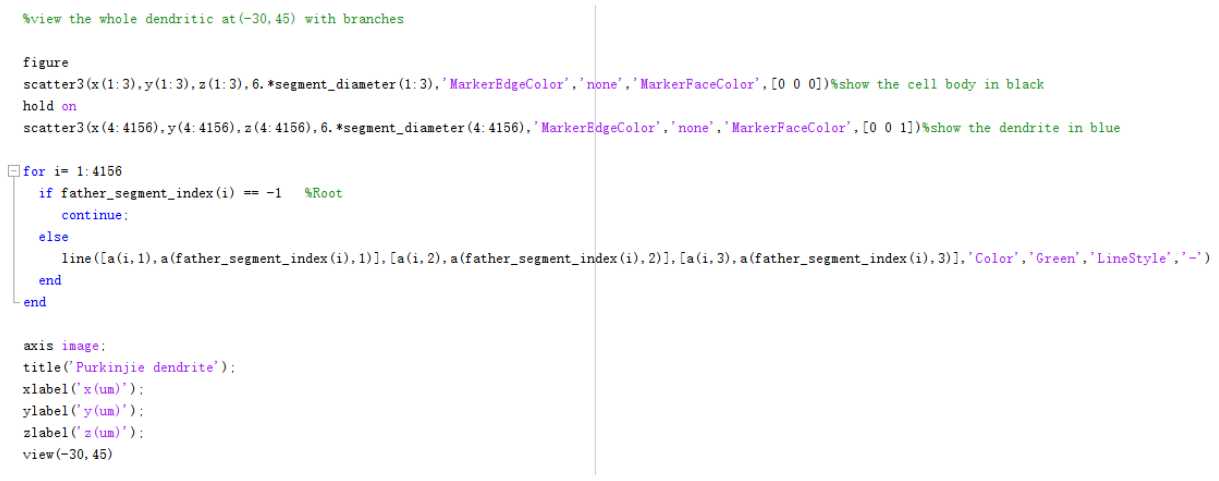
## Result:



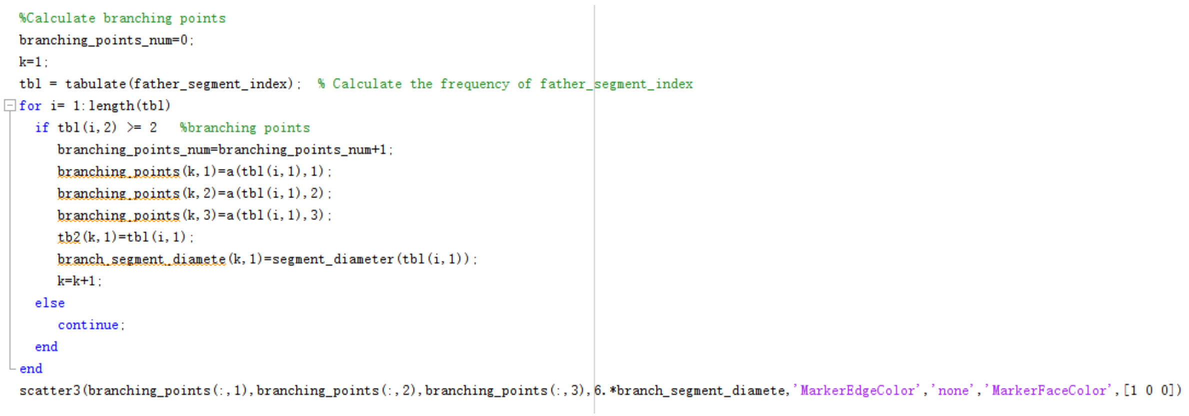


3. Calculate how many branching points on both the pyramidal and Purkinjie dendrite.

Code:

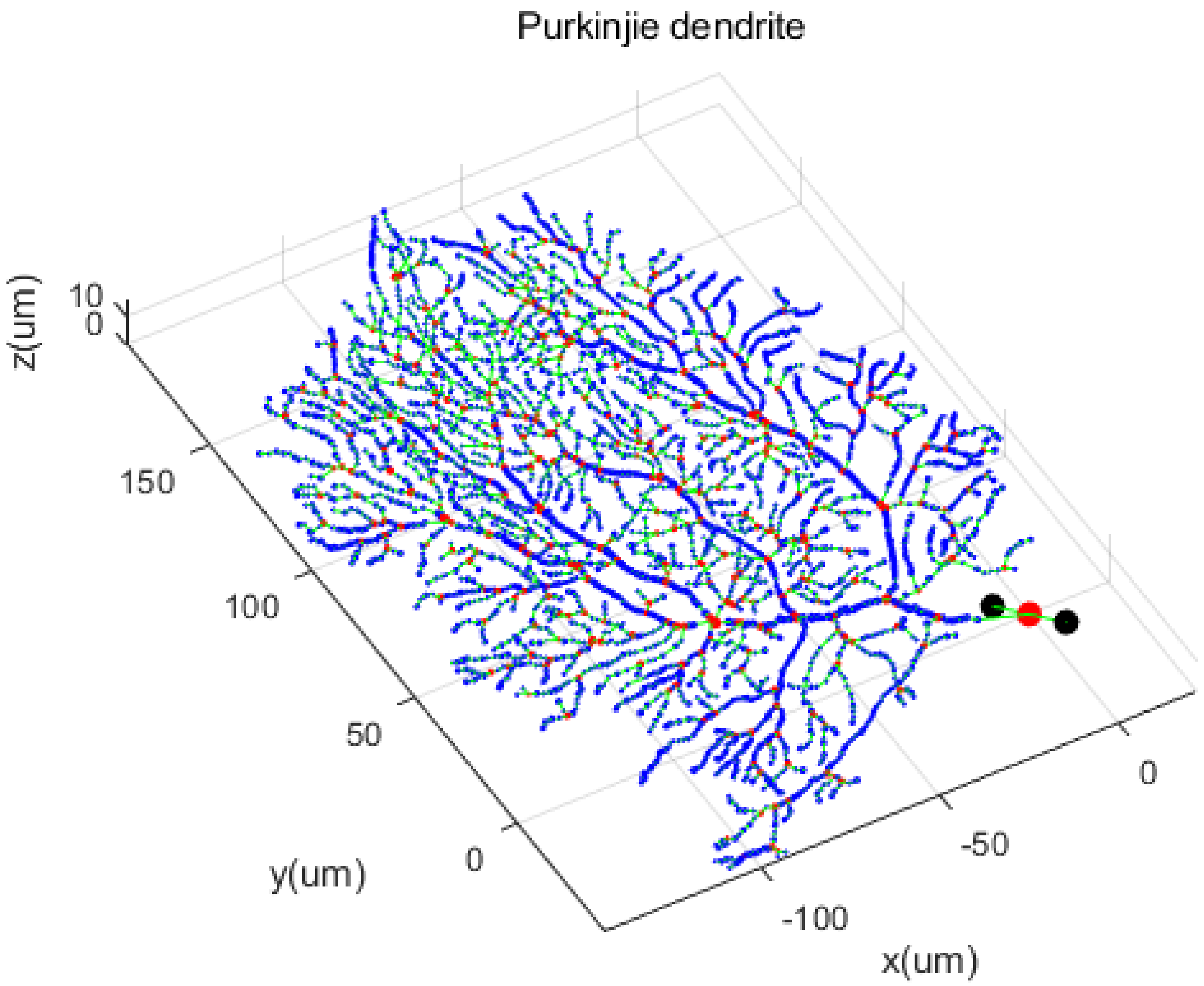


```
%view the whole dendritic at(-30,45) with branches

figure
scatter3(x(1:3),y(1:3),z(1:3),6.*segment_diameter(1:3),'MarkerEdgeColor','none','MarkerFaceColor',[0 0 0])%show the cell body in black
hold on
scatter3(x(4:4156),y(4:4156),z(4:4156),6.*segment_diameter(4:4156),'MarkerEdgeColor','none','MarkerFaceColor',[0 0 1])%show the dendrite in blue

for i= 1:4156
  if father_segment_index(i) == -1   %Root
     continue;
  else
     line([a(i,1),a(father_segment_index(i),1)],[a(i,2),a(father_segment_index(i),2)],[a(i,3),a(father_segment_index(i),3)],'Color','Green','LineStyle','-')
  end
end

axis image;
title('Purkinjie dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
view(-30,45)
```



```
%Calculate branching points
branching_points_num=0;
k=1;
tbl = tabulate(father_segment_index);  % Calculate the frequency of father_segment_index
for i= 1:length(tbl)
  if tbl(i,2) >= 2   %branching points
     branching_points_num=branching_points_num+1;
     branching_points(k,1)=a(tbl(i,1),1);
     branching_points(k,2)=a(tbl(i,1),2);
     branching_points(k,3)=a(tbl(i,1),3);
     tb2(k,1)=tbl(i,1);
     branch_segment_diamete(k,1)=segment_diameter(tbl(i,1));
     k=k+1;
  else
     continue;
  end
end
scatter3(branching_points(:,1),branching_points(:,2),branching_points(:,3),6.*branch_segment_diamete,'MarkerEdgeColor','none','MarkerFaceColor',[1 0 0])
```

Result:



```
branching_points_num =

   379
```

4.Perform a Sholl plot. Center on the cell body and draw spheres, and plot the number of intersections between sphere and dendrites as a function of sphere radius.

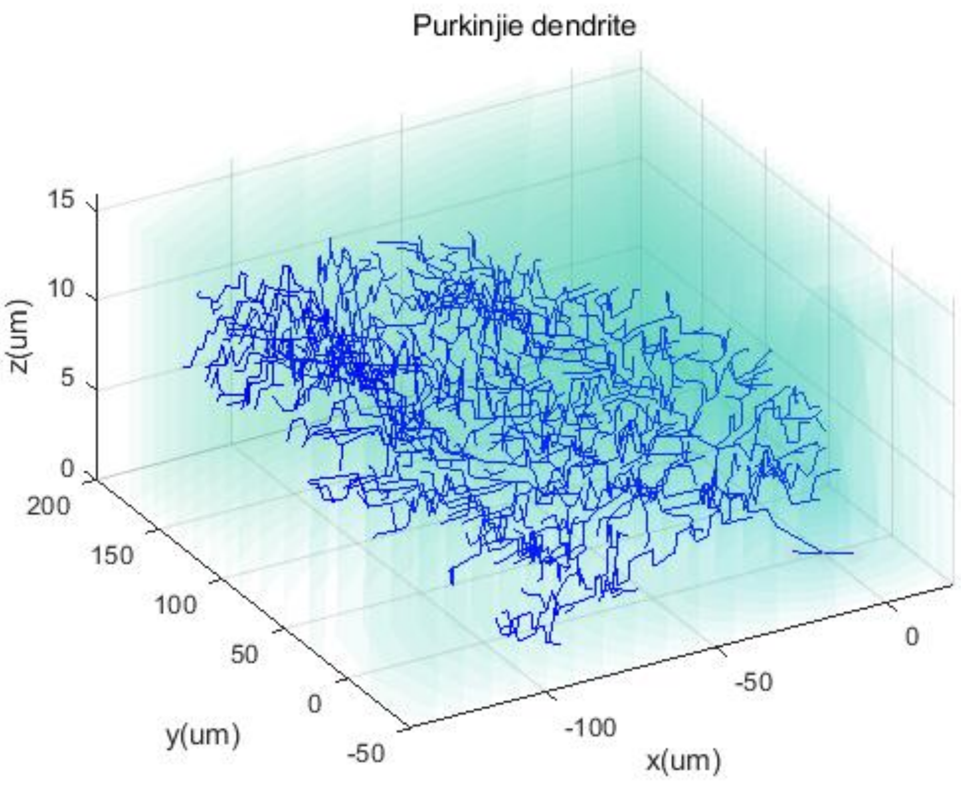
Code:

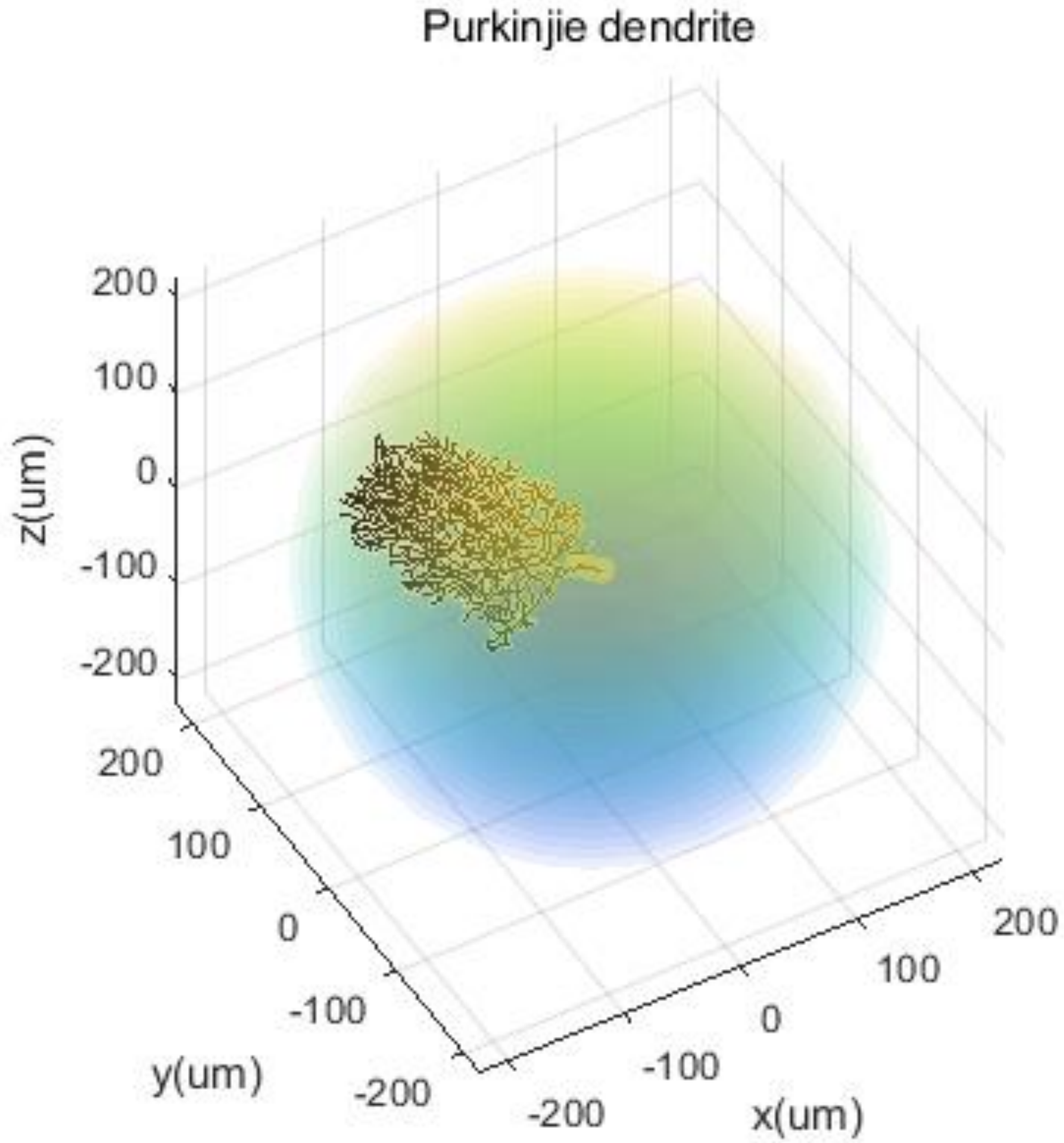
```
t=linspace(0,pi,25);
p=linspace(0,2*pi,25);
[theta,phi]=meshgrid(t,p);
N=30;  %the number of spheres
r_inc=7.5;
for nn = 1:1:N
    rr(nn,1) = nn*r_inc;
    x_r=rr(nn,1)*sin(theta).*sin(phi);
    y_r=rr(nn,1)*sin(theta).*cos(phi);
    z_r=rr(nn,1)*cos(theta);
    surf(x_r,y_r,z_r,'linestyle','none');
    alpha(0.05);
end
axis([-140 20 -50 200 0 16]);
title('Purkinjie dendrite');
xlabel('x(um)');
ylabel('y(um)');
zlabel('z(um)');
view(-30,45)
```

```
r_start=0;
x0=0;
y0=0;
z0=0;  %Center on the cell body
N=30;  %the number of spheres
r_inc=7.5;
sholl_sumation = zeros(N,1);
for m = 2:4156                          % Calculate the distance between cell body and dendrites
            x1 = x(m);
            y1 = y(m);
            z1 = z(m);
            x2 = x(father_segment_index(m));
            y2 = y(father_segment_index(m));
            z2 = z(father_segment_index(m));
            d1(m) = sqrt( (x1-x0)^2 + (y1-y0)^2 + (z1-z0)^2 );
            d2(m) = sqrt( (x2-x0)^2 + (y2-y0)^2 + (z2-z0)^2 );
end
for nn = 1:1:N
        rr(nn,1) = r_start + nn*r_inc;
        for pp = 1:4156
            if (d2(pp) >= rr(nn,1) && d1(pp) < rr(nn,1)) || (d2(pp) <= rr(nn,1) && d1(pp) > rr(nn,1))   % Calculate the number of intersections between sphere and dendrites
                sholl_sumation(nn) = sholl_sumation(nn) + 1;
            else
            end
        end
    end
```

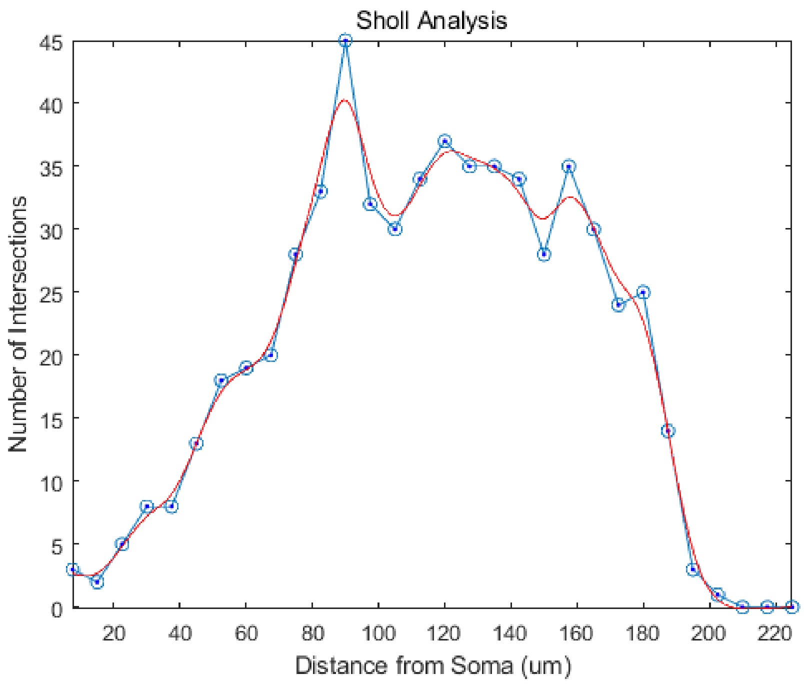
```
figure;
plot(rr,sholl_sumation,'o-');
hold on;
f=fit(rr,sholl_sumation,'smoothingspline')
plot(f,rr,sholl_sumation)
legend 'off';
title('Sholl Analysis');
ylabel('Number of Intersections'); xlabel('Distance from Soma (um)');
axis tight;

figure;
plot(rr(1:26,1),log10(sholl_sumation(1:26,1)./(4*pi*rr(1:26,1).^2)),'o-');
hold on;
f=fit(rr(1:26,1),log10(sholl_sumation(1:26,1)./(4*pi*rr(1:26,1).^2)),'poly1')
plot(f,rr(1:26,1),log10(sholl_sumation(1:26,1)./(4*pi*rr(1:26,1).^2)))
legend 'off';
title('Sholl Analysis');
ylabel('log10(N/S)'); xlabel('Distance from Soma (um)');
axis tight;
```

Result:





Purkinjie dendrite
z(um)
200
100
0
-100
-200
y(um)
200
100
0
-100
-200
x(um)
-200
-100
0
100
200



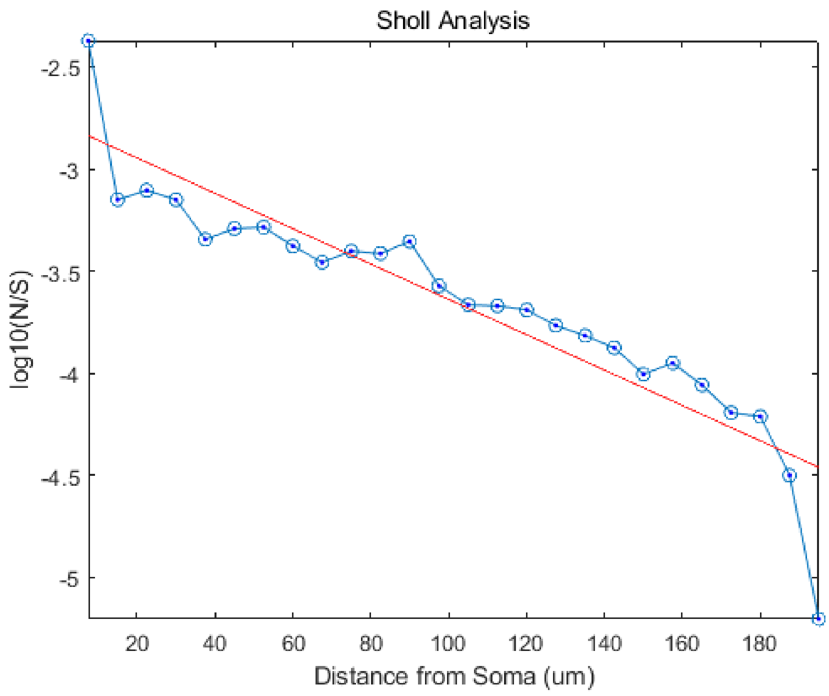
## Semi-Log Method [ edit ]

Somewhat more complicated than the Linear Method, the Semi-Log Method begins by calculating the function Y(r) = N/S where N is the number of dendrite crossings for a circle of radius r, and S is the area of that same circle. The base 10 logarithm is taken of this function, and a first order linear regression, linear fit, is performed on the resulting data set, that is

$$\log_{10}\left(\frac{N}{S}\right) = -k \cdot r + m.$$

where **k** is Sholl's Regression Coefficient.[1]

Sholl's Regression Coefficient is the measure of the change in density of dendrites as a function of distance from the cell body.[5] This method has been shown to have good discrimination value between various neuron types, and even similar types in different regions of the body.



```
Linear model Poly1:
f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
  p1 =   -0.008654  (-0.01024, -0.007071)
  p2 =      -2.773  (-2.956, -2.589)
```