IN SILICO STUDY OF THE ROLE OF CELL GROWTH FACTORS ON PHOTOSYNTHESIS USING A VIRTUAL LEAF TISSUE GENERATOR COUPLED TO A MICROSCALE PHOTOSYNTHESIS GAS EXCHANGE MODEL

M.A. Retta[1], M.K. Abera[1], H.N.C. Berghuijs[2,3,4], P. Verboven[1], P.C. Struik[2,3], B.M. Nicolaï[1,5*]

[1]Division BIOSYST-MeBioS, KU Leuven - University of Leuven, Willem de Croylaan 42, B-3001, Leuven, Belgium

[2]Centre for Crop Systems Analysis, Wageningen University, Droevendaalsesteeg 1, 6708 PB Wageningen, The Netherlands

[3]BioSolar Cells, P.O. Box 98, 6700 AB Wageningen, The Netherlands

[4] Department of Crop Production Ecology, Swedish University of Agricultural Sciences, Ulls väg 16, 75651 Uppsala, Sweden

[5]Flanders Centre of Postharvest Technology, Willem de Croylaan 42, B-3001, Leuven, Belgium

* Corresponding author:

Bart M. Nicolaï, Flanders Center of Postharvest Technology / BIOSYST-MeBioS, KU Leuven, Willem de Croylaan 42, B-3001 Leuven, Belgium

Email: bart.nicolai@kuleuven.be

Tel: +32 16 322375

Fax: +32 16 322955

Supplementary Data

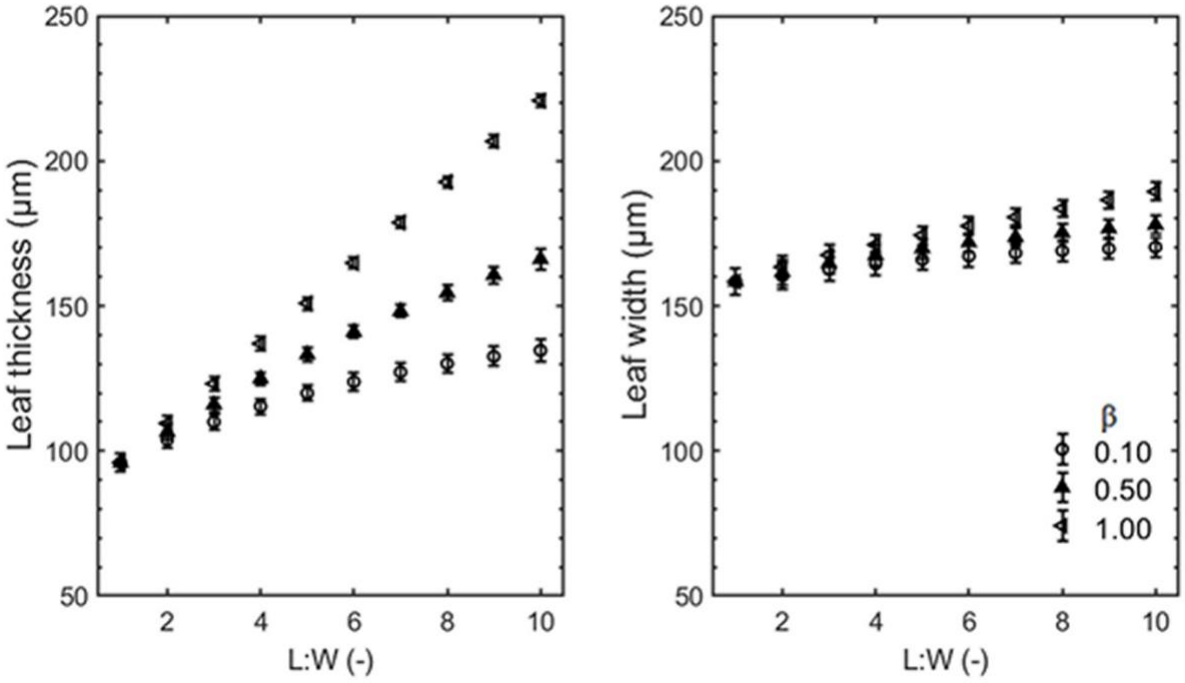

Fig. S1. Effect of palisade mesophyll length to width ratio (L:W ratio) on leaf width and leaf thickness. The extent of airspace formation in palisade mesophyll was medium. The anisotropies during growth (β) were 0.10 (○), 0.50 (▲) and 1.00 (Δ). Errors bars show standard deviation (n = 3).

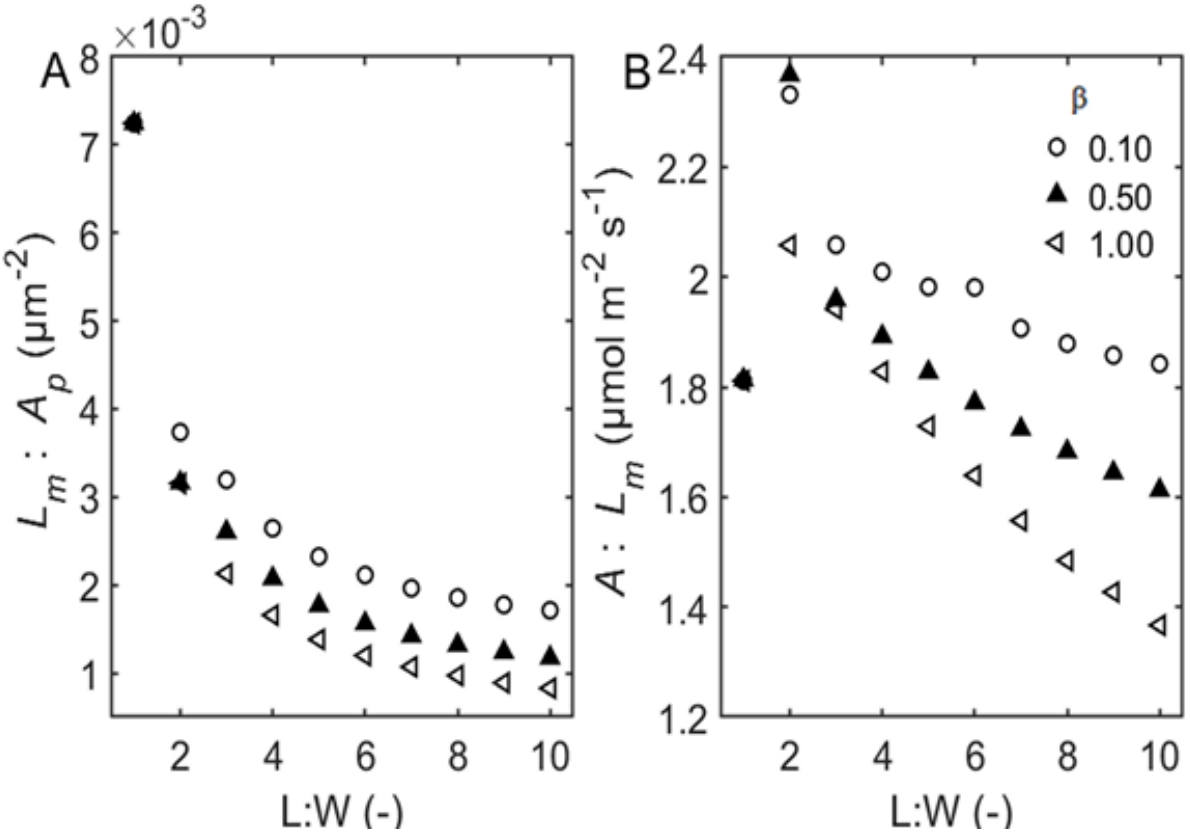

Fig. S2. Effect of growth anisotropy factor on L_m to A_p (A) and photosynthesis (A) to L_m (B). The extent of airspace formation in the palisade mesophyll was moderate. Photosynthesis was calculated at CO_2 concentration of 400 μmol mol^{-1}, stomatal conductance of 0.15 mol m^{-2} s^{-1} and irradiance of 1500 μmol m^{-2} s^{-1}. The anisotropies during growth (β) were 0.10 (○), 0.50 (▲) and 1.00 (Δ).

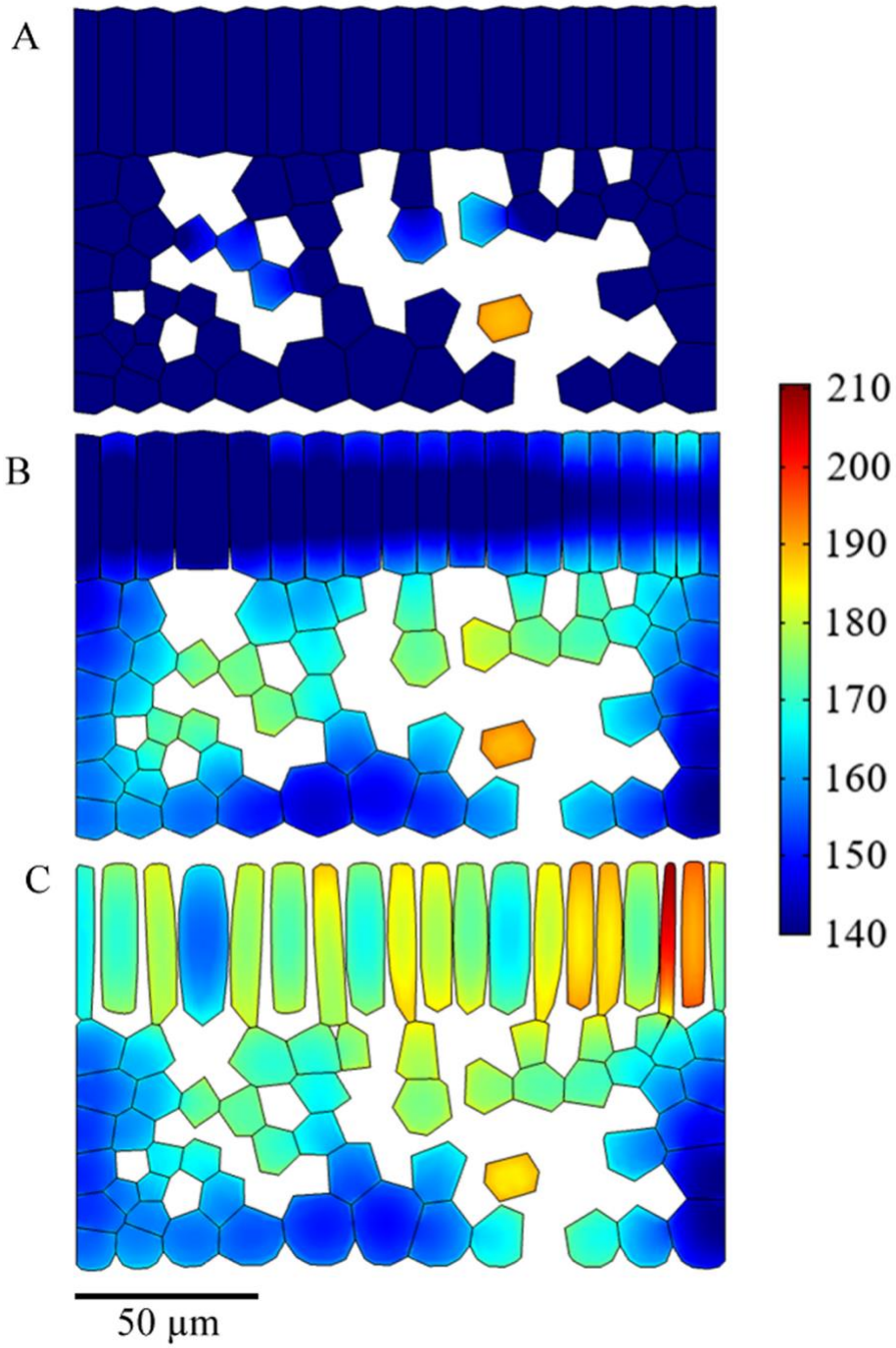

Fig. S3. Comparison of mean CO_2 concentration in mesophyll. Shown are for low airspace formation (A), medium extent of airspace formation (B) and high extent of airspace formation (C). L:W ratio of 3, anisotropy of 1. CO_2 concentration was expressed in equivalent gas phase concentration (µmol mol^{-1}). The CO_2 concentration profiles were computed at CO_2 concentration of 400 µmol mol^{-1}, stomatal conductance of 0.15 mol m^{-2} s^{-1} and irradiance of 1500 µmol m^{-2} s^{-1} at 21 % O_2. Scale bar 50 µm.

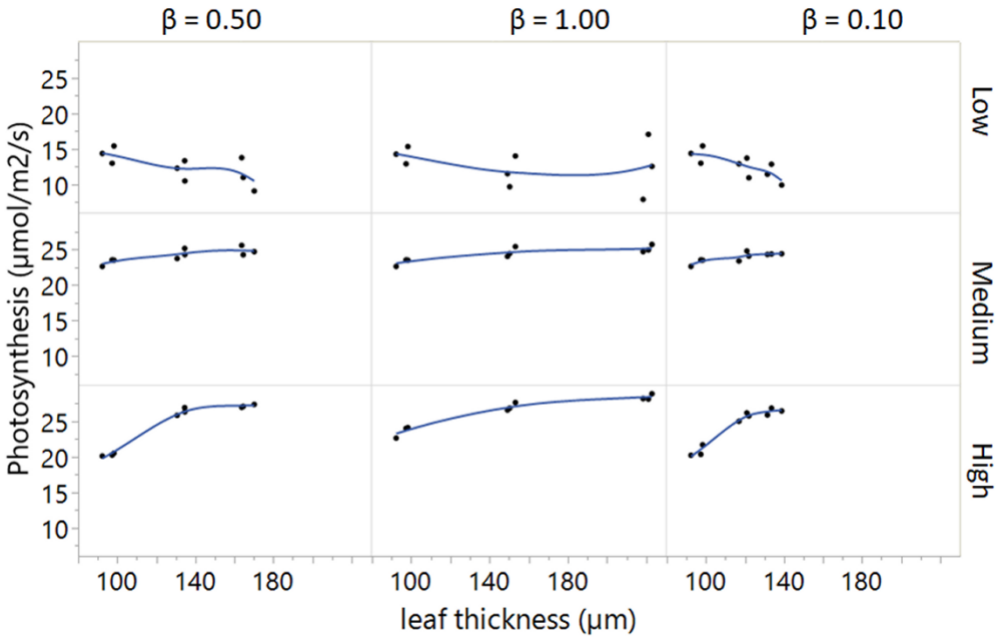

Fig. S4. The relation between photosynthesis and leaf thickness which is used a proxy for leaf mass per area. The growth anisotropy (β) was 0.10,0.50 and 1.00 while the extent of airspace formation was "low", "medium" and "High" (see main text). A smooth line was added to show trends.

Table S1. Parameters of the growth model

Variable	Symbol	Units	Value	Notes and references
Degree of anisotropy	a	-	0 0.9	Spongy cells, assumed Palisade cells, fitted
Damping factor	b	Ns/m	$3.5x10_6$	Abera *et al.*, 2013
Ratio of maximum resting length and initial resting lengths of walls	C	-	3 1	Vertical walls of Palisade cells, fitted Spongy cells, assumed
Spring constant of cells aligned along maximum growth direction	k_{min}	MN m^{-1}	Calculated	See Materials and Methods
Initial resting length of cell wall	$l_{n,0}$	µm	Calculated	See Materials and Methods
Maximum resting length of cell wall	$l_{n,max}$	µm	Calculated	See Materials and Methods
Applied turgor pressure	P_{max}	MPa	1	Beauzamy, Derr & Boudaoud, 2015
Time constant for uniform increase in turgor	t_c	s	100000	Assumed
Time constant for length to reach maximum	τ	s	200000	Assumed
Angle expressing directionality of cell wall growth	α	-	Calculated	See Materials and Methods

Protocol S1. Algorithm for virtual leaf tissue generator

The virtual tissue generator is based on a cell growth model explained in detail previously (Abera *et al.*, 2013). In brief, the cell is conceptualized as an elastic thin-walled structure sustained by turgor pressure. Initial cell wall network topologies are generated using Voronoi tessellation which is a partitioning of a plane into regions based on proximity to a set of initially generated points (Watson, 1993). Cell shapes are driven by a net force on the vertices of the network. Newton's first law and Hooke's law for springs are applied to simulate cell growth and expansion. The main governing equations of cell wall mechanics are given below. Parameters of the growth model are given in Table S1.

The velocity (**v**; m s^{-1}) and the position (**x**; m) of a vertex in the wall (*w*) network caused by the action of turgor forces ($\mathbf{F}_t$; N) and tension forces ($\mathbf{F}_s$; N) contributed by a set of walls (*W*) incident on the vertex were calculated by:

$$m \cdot \frac{d\mathbf{v}}{dt} = \sum_{w \in W} \left(\mathbf{F}_t + \mathbf{F}_s \right) - b\mathbf{v} \tag{S1}$$

$$\frac{d\mathbf{x}}{dt} = \mathbf{v} \tag{S2}$$

where m is the mass of a vertex (unity), the term $-b\mathbf{v}$ is the damping force which is the product of damping factor (*b*; N s m^{-1}) and **v**. It accounts for frictional forces.

$\mathbf{F}_t$ is given by:

$$\mathbf{F}_t = \frac{\Delta P_t l}{2} \cdot \mathbf{n} \tag{S3}$$

where ΔP_t is the difference in turgor pressure (P_t; Pa) of two adjacent cells sharing two incident vertices of a wall; l is the actual cell wall length (m) at a current time; and, **n** is the unit vector normal to the cell wall.

$\mathbf{F}_s$ is given by:

$$\mathbf{F}_s = -k\mathbf{u} \tag{S4}$$

where k (N m^{-1}) is the spring constant of the wall (see Supplementary Eq. S6) and **u** (m) is the net elongation of the vector of the wall whose norm is the difference between l and the current resting length (l_n; m) of the wall.

The change in time of l_n due to cell growth is given by:

$$\frac{dl_n}{dt} = \frac{1}{\tau}\left(l - l_n\right)\left(l_{n,\max} - l_n\right) \tag{S5}$$

where $l_{n,\max}$ (m) is the maximum resting length of the wall, and τ (s) is the time constant for length to reach maximum.

Anisotropic expansion of cell wall is modeled by varying k and $l_{n,\max}$ based on the orientation of cell walls (λ) as:

$$k = k_{\min} + k_{\min}(C-1)(1-\lambda) \tag{S6}$$

$$l_{n,\max} = \lambda(C-1)l_{n,0} + l_{n,0} \tag{S7}$$

where $k_{\min}$ is the spring constant of walls parallel to the maximum growth direction, C is the ratio of $l_{n,\max}$ and initial resting lengths ($l_{n,o}$; m) of walls.

λ is given by:

$$\lambda = 1 - a\cos^2\left(\alpha + \frac{\pi}{2}\right) \tag{S8}$$

where a is the degree of growth anisotropy ($0 \leq a \leq 1$) and α is the angle (radians) between a cell wall and the major axis of the cell. When $a = 0$ the growth is isotropic and when $a = 1$ it is an anisotropic in the direction of the major axis of the cell. Supplementary Eqs. 6-8 allow switching the growth from totally isotropic growth to any degree of anisotropy.

A uniform increase in turgor pressure P (Pa) in the cell network is given by:

$$\frac{dP}{dt} = \frac{P_{\max} - P}{t_c} \tag{S9}$$

where $P_{\max}$ (Pa) is the maximum turgor pressure; and, time t_c (s) is the time constant for turgor to reach a maximum.

The initial topology consists of a Voronoi tessellation of the cells in the spongy mesophyll layer (Fig. 1A, main text). Pores in the leaf tissue that result from death of cells (lysogenous origion) were simulated here using a procedure in Matlab (The Mathworks, Natick, MA, USA) that uses a convex envelope to select Voronoi cells that lie completely in the envelope. Some of these were then assigned to be pores to match porosity (Abera *et al.*, 2013). Layers of rectangular cells for palisade mesophyll and epidermis were then added (Fig. 1B, main text). Values of $l_{n,o}$ were obtained from this starting topology. $k_{\min}$ was calculated by assuming a Young's modulus of elasticity of the cell wall of 30 MPa (Abera *et al.*, 2014) and an average value of $l_{n,o}$ as the ratio of the two. The value of C for all the walls of palisade cells except those that are parallel to the major axis of each cell was set to 1. For walls of palisade cells that are parallel to the major axis of growth, λ (Supplementary Eq. S8) was calculated assuming an anisotropy of 0.9. Only for the

aforementioned walls, therefore, C was scaled using a fixed factor for length to width ratio of those walls. Consequently, $l_{n,\max}$ is calculated using (Supplementary Eq. S7). When a wall in a cell is perpendicular to the major axis of the cell, $l_{n,\max} = l_{n,o}$ and growth is zero (Supplementary Eq. S5). The initial value of turgor pressure in the cells (Fig. 1B, main text) was zero. At each time step, the turgor pressure in all the cells was increased according to Supplementary Eq. S9. Turgor pressure in the intercellular airspaces was always zero. Cell expansion results from turgor pressure acting on the yielding cell wall material. Velocities of vertices were then calculated from the net force due to the turgor and tension forces. The degree of growth anisotropy (Supplementary Eq. S8) was varied between spongy and palisade cells to mimic their contrasting topology. Intercellular airspaces among adjacent walls were formed by loosening the zero-resting length springs, simulating the separation of cells along middle lamella occurring during growth; this type of airspace formation is called schizogenous (Abera *et al.*, 2013). Supplementary Eqs. 1-9 were solved using an ordinary differential equation solver in Matlab (The Mathworks, Natick, MA, USA). The iteration was stopped when the velocities of the vertices fell below a set threshold value which was taken as an equilibrium state. A timespan of 5 million seconds with time step size of 5000 seconds was used for the simulation. The resulting topology is shown in Fig. 1C (main text). At the end of growth, larger intercellular airspaces in the palisade mesophyll were created in a separate step. Depending on the extent of airspace desired, a number of walls connecting the pores formed at the common vertex of three cells were removed, thereby connecting these pores (Abera *et al.*, 2013). For this, a random number generator in Matlab was used to determine the number of walls in the palisade cell to separate. When all the parallel walls of palisade mesophyll are separated, maximum porosity is achieved. The pressure in these pores was 90% of that of the cells and Supplementary Eqs. S1-S9 were solved again resulting in topologies shown in Fig. 2 (main

text). Formation of additional intercellular airspaces due to death of cells (lysigenous origin) was also simulated here by simple random sampling and removing some of the virtual cells.

The values of *C* and the growth anisotropy factor for palisade cells were optimized using a separate set of light microscopy images as described in Experimental data section. The optimization minimized the differences in the mean area of cells and aspect ratio between the images from light microscopy and virtual leaf tissue generator in Matlab (The Mathworks, Natick, MA, USA).

For a typical virtual leaf tissue containing 93 cells, a CPU time of 5128 seconds was required on an Intel(R) Core(TM) i7-3770 CPU X5650, 3.40 GHz processor, 16.00 GB RAM and Windows 7 professional, 64 bit operating system computer.

References

Abera MK, Fanta SW, Verboven P, Ho QT, Carmeliet J, Nicolai BM. 2013. Virtual fruit tissue generation based on cell growth modelling. Food and bioprocess technology **6**, 859–869.

Abera MK, Verboven P, Herremans E, Defraeye T, Fanta SW, Ho QT, Carmeliet J, Nicolai BM. 2014. 3D Virtual pome fruit tissue generation based on cell growth modeling. Food and bioprocess technology **7**, 542–555.

Beauzamy L, Derr J, Boudaoud A. 2015. Quantifying hydrostatic pressure in plant cells by using indentation with an atomic force microscope. Biophysical Journal **108**, 2448–2456.

Watson D. 1993. Spatial tessellations: concepts and applications of voronoi diagrams. Computers & Geosciences **19**, 1209–1210.

Protocol S2. Matlab code for the growth algorithm

The following subsections present the Matlab code for growth algorithm.

Running algorithm

This is the running file that calls for the functions GROWTH, Doublewalls, starttesselation, Nearestpoint and SPLIT.

```
% Running files for the tissue growth algorithm
% The algorithm is the extension of fruit growth model presented in
% Abera et al. 2013. Food and Bioprocess Technology.(6),4.pp.859-869
% The first step is to run the case GROWTH. The coordinates  and vertices
% of the generated geometries  and are stored in the cords folder along with
% other outputs essential for splitting phase. Next, the case 'SPLIT' is
% run and the associated vertices are stored in split folder.

%% Input parameters
ap=1;                % Turgor pressure (Mpa)
ar=1;                % The absolute tolerance of the solver is 10^-ar
dr=100;              % Damping coefficient(-)
l_n_max_2_l_n_0=1;  % Ratio of maximum resting length and initial resting
length of walls
P_h_2_w=5.0;         % Palisade height to width ratio
n=3;                 % Number of geometries you want to generate

steps= 'GROWTH';     % set this to GROWTH or SPLITING
switch steps

    case 'GROWTH'

    for geom=1:n

    string=['cords_',num2str(geom)]; % storing outputs for each geometry
    mkdir(string)
    currentFolder=pwd;


[coordinates,dimension,numberofcells,walls,wall2cell,l_n,Cellarea,CellvertexI
d,t,l_n_0,y,epiId,paliId,hypoId,KK]=GROWTH(ap,ar,dr,l_n_max_2_l_n_0,P_h_2_w,g
eom,string,currentFolder);

    coordinates(:,1)=coordinates(:,1)+abs(min(coordinates(:,1)));
    coordinates(:,2)=coordinates(:,2)+abs(min(coordinates(:,2)));

%  assign important matrices for each geometry
     cord{geom}=coordinates;
     CellId{geom}=CellvertexId;
     PID{geom}=paliId;
     EID{geom}=epiId;
     WALL{geom}=walls;
     W2C{geom}=wall2cell;
    end
```

```
%% save important matrices for to convert later into comsol geometry
    save cord cord
    save CellId CellId
    save PID PID
    save EID EID
    save WALL WALL
    save W2C W2C

    case 'SPLITING'

    currentFolder=pwd;

    for geom=1:n
    string=['cords_',num2str(geom)];


[coordinates,dimension,numberofcells,walls,wall2cell,l_n,Cellarea,CellvertexI
d,t,l_n_0,y,epiId,paliId,KK,geom,adhesive_wall,split_wall,split_turgor_1,spli
t_turgor_2,...

left_boundary,right_boundary,az_top,az_top_2,az_bottom,az_bottom_2,con_point,
con_point_2,con_point_3]=SPLIT(ap,ar,dr,l_n_max_2_l_n_0,P_h_2_w,geom,string,
currentFolder);

    coordinates(:,1)=coordinates(:,1)+abs(min(coordinates(:,1)));
    coordinates(:,2)=coordinates(:,2)+abs(min(coordinates(:,2)));

    % assign important matrices for each geometry
     cord{geom}=coordinates;
     CellId{geom}=CellvertexId;
     PID{geom}=paliId;
     EID{geom}=epiId;
     WALL{geom}=walls;
     W2C{geom}=wall2cell;
    end


%% save important matrices for to convert later into comsol geometry
    cd(currentFolder)
    mkdir('Splitted_geoms')
    cd('Splitted_geoms')

    save cord cord
    save CellId CellId
    save PID PID
    save EID EID
    save WALL WALL
    save W2C W2C

    cd(currentFolder)

end

%% visualizing geometries
```

```
figure;
hold on
for j=1:length(CellvertexId)

        if ismember(j,epiId)

patch(coordinates(CellvertexId{j},1),coordinates(CellvertexId{j},2),'r','line
width',0.01)

        elseif ismember(j,paliId)

patch(coordinates(CellvertexId{j}(:),1),coordinates(CellvertexId{j}(:),2),'m'
,'linewidth',1)
        else


patch(coordinates(CellvertexId{j},1),coordinates(CellvertexId{j},2),'g')
            hold on
        end
 end
axis('equal')
```

Growth

```
% This code solves the growth algorithm equations using ode15s solver after
% the initial voronoi tesselations are generated. The doublewalls function
% generates the intitial tesselation, defines wall to vertex (wall2vertex)
% relations, wall to cell (wall2cell) relations and cell to vertex
% (cell2vertex) relations. The resulting outputs are saved in separate
% folders specified by the path specified by string. The geometry is ready %
to be used in the next step of generating intercellular airspaces in
% palisade

function
[coordinates,dimension,numberofcells,walls,wall2cell,l_n,Cellarea,CellvertexI
d,t,l_n_0,y,epiId,paliId,hypoId,KK,geom]=GROWTH(ap,ar,dr,l_n_max_2_l_n_0,P_h_
2_w,geom,string,string11)

[coordinates,dimension,numberofcells,walls,wall2cell,Wallbnd,l_n,Cellarea,Cel
lvertexId,epiId,hypoId,paliId,stoneId,wallvepiuper,oldwall,adhesive_wall,adw,
WO_1,WO_2,...
ad_direction,old_cord,split_wall,split_turgor_1,split_turgor_2,con_point,con_
point_2,con_point_3]=doublewalls(geom,string, string11);

%% Parameters
E=30;        % Young's modulus (MPa)
tau1=200000; % Time constant for length to grow to maximum

%% assigning turgore pressure to the cells, spring constant to the walls  and
damping coefficient for the matrix

turgorpressure(numberofcells,1)=0;
A0=mean(Cellarea);                % average cell area of the initial cells
drag=dr;                          % drag coefficient
l_n_0=l_n;                        % initial resting length
```

```
l_nmax=l_n_max_2_l_n_0*l_n_0;      % setting the maximum resting length of
the walls
aniso_pali = 1;                    % Anisotropy of palisade mesophyll
%% initializing the variables
amk=[];  % The walls of palisade cells that are parallel to growth direction
akk=[];  % The walls of palisade cells that are perpendicular to growth
direction
aml=[];  % The walls of palisade cells that are in the initial tesselatin

anisotropy=zeros(length(walls),1); % anisotropy of spongy mesophyll is 0

for i=1:length(walls)

    ak(i)=E/mean(l_n_0(i));%(i);
    C_l_n(i)=l_n_max_2_l_n_0;

      if
(ismember(wall2cell(i,1),paliId)&ismember(wall2cell(i,2),paliId))|((wall2cell
(i,1)==numberofcells&ismember(wall2cell(i,2),paliId)...
              &abs(coordinates(walls(i,1),1)-
coordinates(walls(i,2),1))<0.00001)...

|(wall2cell(i,2)==numberofcells&ismember(wall2cell(i,1),paliId)&abs(coordinat
es(walls(i,1),1)-coordinates(walls(i,2),1))<0.00001))
                anisotropy(i)=aniso_pali;
                C_l_n(i)=P_h_2_w*l_n_max_2_l_n_0; %setting the maximum
resting length of the walls
                amk=[amk,i];
       elseif
(ismember(wall2cell(i,1),epiId)|ismember(wall2cell(i,2),epiId))...
                ak(i)=ak(i);
       end
end



for i=1:length(walls)
    if (ismember(wall2cell(i,1),paliId)|ismember(wall2cell(i,2),paliId))&
~ismember(i,amk)
        akk=[akk,i];
        ak(i)=6*ak(i);
    end
end

for i=1:length(walls)
    if ~ismember(i,[akk,amk])
        ak(i)=2*ak(i);
    end
end



for i=1:oldwall
    if (ismember(wall2cell(i,1),paliId)|ismember(wall2cell(i,2),paliId))&
~ismember(i,amk)
       aml=[aml,i];
       l_n_0(i)=1*l_n_0(i);
```

```
    end
end

for i=1:length(walls)

Degree1(i)= atan((coordinates(walls(i,2),2)-
coordinates(walls(i,1),2))/(coordinates(walls(i,2),1)-
coordinates(walls(i,1),1)));
lambda(i)=1-(1-anisotropy(i)*(cos(Degree1(i)+pi/2))^2);
l_nmax(i)=lambda(i)*(C_l_n(i)-1)*l_n_0(i)+C_l_n(i)*l_n_0(i);
end

tau=tau1;
%% Solving the growth equations
velocity=zeros(length(coordinates),dimension);
tstart=0;
tfinal=5000000;
at=10^(-ar);    % absolute tolerance
rt=10^(-(ar+1));%relative tolerance
pr=0;           % initialing turgore pressure
tc=100000;      % time constant for torgore pressure to reach maximum
pmax=ap;
y0=[coordinates(:,1);coordinates(:,2);l_n;pr];% intial conditions for the ode
solver

options =
odeset('RelTol',rt,'AbsTol',at,'events',@events,'stats','on','MaxOrder',1,'BD
F','on');

%% beginning of the solution
for j=1:1
%% identifing walls of boundary cell and assigning spring constant
for i=1:length(walls)
    if  ismember(i, amk)
       KK(i)=ak(i)*(C_l_n(i)-1)*(1-lambda(i))+ak(i);
    else
        KK(i)=ak(i);
    end
end

left_boundary=[];
right_boundary=[];
bottom_boundaryy=[];
top_boundaryy=[];

% Finding boundaries of the geometry
for i=1:length(coordinates)
    if abs(coordinates(i,1)-min(coordinates(:,1)))<0.00001
        left_boundary=[left_boundary,i];
    end
    if abs(coordinates(i,1)-max(coordinates(:,1)))<0.00001
        right_boundary=[right_boundary,i];
    end

    if abs(coordinates(i,2)-min(coordinates(:,2)))<0.00001
```

```
        bottom_boundaryy=[bottom_boundaryy,i];
    end
    if abs(coordinates(i,2)-max(coordinates(:,2)))<0.00001
        top_boundaryy=[top_boundaryy,i];
    end

end

Ab_bottom=bottom_boundaryy(find(bottom_boundaryy<old_cord+1));
Ab_top=top_boundaryy(find(top_boundaryy<old_cord+1));

az_top=[Ab_top(1:2:end),top_boundaryy(find(top_boundaryy>old_cord))];
az_bottom=[Ab_bottom(1:2:end),bottom_boundaryy(find(bottom_boundaryy>old_cord
))];
az_top_2=[Ab_top(2:2:end)];
az_bottom_2=[Ab_bottom(2:2:end)];
con_left=coordinates(left_boundary,1);
con_b1=coordinates(az_bottom,2);
con_b2=coordinates(az_bottom_2,2);

bottom_left=find(abs(coordinates(:,1)-
min(coordinates(:,1)))<0.0001&abs(coordinates(:,2)-
min(coordinates(:,2)))<0.0001);
bottom_right=find(abs(coordinates(:,1)-
max(coordinates(:,1)))<0.0001&abs(coordinates(:,2)-
min(coordinates(:,2)))<0.0001);
top_left=find(abs(coordinates(:,1)-
min(coordinates(:,1)))<0.0001&abs(coordinates(:,2)-
max(coordinates(:,2)))<0.0001);
top_right=find(abs(coordinates(:,1)-
max(coordinates(:,1)))<0.0001&abs(coordinates(:,2)-
max(coordinates(:,2)))<0.0001);
top_boundary=top_boundaryy;
bottom_boundary=bottom_boundaryy;
con_bottom=coordinates(bottom_boundary,2);

ABD=1:length(coordinates);
stopc=ABD(find(~ismember(ABD,[left_boundary,right_boundary,bottom_boundary,to
p_boundary])));

for i=1:length(walls)
    if (ismember(wall2cell(i,1),paliId)&
~ismember(wall2cell(i,2),paliId))|(ismember(wall2cell(i,2),paliId)&
~ismember(wall2cell(i,1),paliId));%,[225,246,247,252])

    lambda(i)=1;
    l_nmax(i)=lambda(i)*(C_l_n(i)-1)*l_n_0(i)+C_l_n(i)*l_n_0(i);

    end
end


savetime=tstart:5000:tfinal;

[t,y]=ode15s(@f,[savetime],y0,options); % the ODE solver
```

```
end

%% defining the ODE function to be solved

function dydt=f(t,y)

   %% initializing the parameters at every time step
 b=[];
 force=zeros(length(coordinates),dimension);
 pf=zeros(length(coordinates),dimension);
 Tf=zeros(length(coordinates),dimension);


coordinates=[y(1:1*length(coordinates)),y(1*length(coordinates)+1:2*length(co
ordinates))];% new coordinates
    coordinates(left_boundary,1)=con_left;
    velocity(left_boundary,1)=mean(velocity(left_boundary,1));
    velocity(right_boundary,1)=mean(velocity(right_boundary,1));

    velocity(az_top_2,2)=mean(velocity(az_top_2,2));
    velocity(az_bottom,2)=mean(velocity(az_bottom,2));
    velocity(az_top,2)=mean(velocity(az_top,2));
    velocity(az_bottom_2,2)=mean(velocity(az_bottom_2,2));
    velocity(az_bottom_2,2)=mean(velocity(az_bottom_2,2));

    l_n=y(2*length(coordinates)+1:length(y)-1);
    turgorpressure(1:numberofcells-1,1)=y(end);


%Calculate new Length and direction of walls

for i=1:length(walls)
    l(i)=norm(coordinates(walls(i,1),:)-coordinates(walls(i,2),:),2);
        direction(i,:)=(coordinates(walls(i,1),:)-
coordinates(walls(i,2),:))/l(i);
        normal_direction(i,1)=direction(i,2);
        normal_direction(i,2)=-direction(i,1);
    Degree1(i)= atan((coordinates(walls(i,2),2)-
coordinates(walls(i,1),2))/(coordinates(walls(i,2),1)-
coordinates(walls(i,1),1)));

    Fturgor=(turgorpressure(wall2cell(i,2))-
turgorpressure(wall2cell(i,1)))*l(i)*(normal_direction(i,:));

%% tension force

      b(i)=((l(i)-l_n(i)).*(l_nmax(i)-l_n(i)))/(tau);

          Ftension=KK(i)*(l(i)-l_n(i))*direction(i,:);

 %% net force

       pf(walls(i,1),:)=pf(walls(i,1),:)+0.5*Fturgor;
```

```
       pf(walls(i,2),:)=pf(walls(i,2),:)+0.5*Fturgor ;
       Tf(walls(i,1),:)=Tf(walls(i,1),:)-Ftension;
       Tf(walls(i,2),:)=Tf(walls(i,2),:)+Ftension;

    kr(i)=0;

end

for i=1:length(adhesive_wall)
     ll(i)=1+norm(coordinates(adhesive_wall(i,1),:)-
coordinates(adhesive_wall(i,2),:),2);
     Tf(adhesive_wall(i,1),:)=Tf(adhesive_wall(i,1),:)-
100*(coordinates(adhesive_wall(i,1),:)-
coordinates(adhesive_wall(i,2),:))/ll(i);

Tf(adhesive_wall(i,2),:)=Tf(adhesive_wall(i,2),:)+100*(coordinates(adhesive_w
all(i,1),:)-coordinates(adhesive_wall(i,2),:))/ll(i);

end

force=pf+Tf;
force(left_boundary,1)=0;
force(az_bottom,2)=0;

velocity(:,1)=force(:,1)/drag;
velocity(:,2)=force(:,2)/drag;



velocity(left_boundary,1)=mean(velocity(left_boundary,1));
velocity(right_boundary,1)=mean(velocity(right_boundary,1));

velocity(az_top_2,2)=mean(velocity(az_top_2,2));
velocity(az_bottom,2)=mean(velocity(az_bottom,2));
velocity(az_top,2)=mean(velocity(az_top,2));
velocity(az_bottom_2,2)=mean(velocity(az_bottom_2,2));
velocity(az_bottom_2,2)=mean(velocity(az_bottom_2,2));

    p=1/tc*(pmax-y(end));

    dydt=[velocity(:,1);velocity(:,2);b(:);p];

end
 %% saving the time history of forces on vertices
function [value,isterminal,direction] = events(t,y)
   for i=1:length( CellvertexId)

Cellarea(i)=polyarea(coordinates(CellvertexId{i}(:),1),coordinates(Cellvertex
Id{i}(:),2));
   end

areacheck=(mean(Cellarea)-5*A0);
NV=norm(velocity(stopc,:));
value=mean(NV)-0.000001;   % checking equilibrium velocities
t1=t;
```

```
isterminal = ones(1,1);
direction =-ones(1,1);
end

%% to be saved
cd(string)

save coordinates coordinates
save CellvertexId CellvertexId
save walls walls
save wall2cell wall2cell
save l_n l_n
save epiId epiId
save paliId paliId
save KK KK
save adhesive_wall adhesive_wall
save split_wall split_wall
save split_turgor_1 split_turgor_1
save split_turgor_2 split_turgor_2
save left_boundary left_boundary
save right_boundary right_boundary
save az_top az_top
save az_top_2 az_top_2
save az_bottom az_bottom
save az_bottom_2 az_bottom_2
save con_point con_point
save con_point_2 con_point_2
save con_point_3 con_point_2
cd(string11)

end
```

Doublewalls

```
function [
coordinates,dimension,numberofcells,walls,wall2cell,Wallbnd,l_n,Cellarea,Cell
vertexId,epiId,hypoId,paliId,stoneId,wallvepiuper,oldwall,adhesive_wall,adw,W
O_1,WO_2,...

ad_direction,old_cord,split_wall,split_turgor_1,split_turgor_2,con_point,con_
point_2,con_point_3]=doublewalls(geom,string, string11);

% geom=2 % uncomment this and select a number so that the initial tesselation
for spongy cells is kept the same among geometries

[
coordinates,dimension,numberofcells,walls,wall2cell,Wallbnd,l_n,Cellarea,Cell
vertexId,epiId,hypoId,paliId,stoneId,wallvepiuper,oldwall]=starttesselation(g
eom);

old_cord=length(coordinates);

left_boundary=[];
right_boundary=[];
```

```
bottom_boundaryy=[];
top_boundaryy=[];
for i=1:length(coordinates)
    if abs(coordinates(i,1)-min(coordinates(:,1)))<0.00001
        left_boundary=[left_boundary,i];
    end
    if abs(coordinates(i,1)-max(coordinates(:,1)))<0.00001
        right_boundary=[right_boundary,i];
    end

    if abs(coordinates(i,2)-min(coordinates(:,2)))<0.00001
        bottom_boundaryy=[bottom_boundaryy,i];
    end
    if abs(coordinates(i,2)-max(coordinates(:,2)))<0.00001
        top_boundaryy=[top_boundaryy,i];
    end
end

    %%
    cord=[];
    wall=[];
    w2c=[];
    CellID=[];
    wall=walls;
    w2c=wall2cell;
    CellID=CellvertexId;
    cord=coordinates;
    AA=[];
    BB=[];
    adhesive_wall=[];
    WO_1=[];
    WO_2=[];
    split_wall=[];
    con_point=[];
    con_point_2=[];
    con_point_3=[];
    split_turgor_1=[];
    split_turgor_2=[];

    con_point_pal=[];
    con_point_2_pal=[];
    con_point_3_pal=[];

    con_point_epi=[];
    con_point_2_epi=[];
    con_point_3_epi=[];

    con_point_spongy=[];
    con_point_2_spongy=[];
    con_point_3_spongy=[];

    con_point_pal_spongy=[];
    con_point_2_pal_spongy=[];
    con_point_3_pal_spongy=[];
```

```
     for i=1:length(walls)
        if
(~ismember(numberofcells,wall2cell(i,:)))%&(ismember(wall2cell(i,1),paliId)|i
smember(wall2cell(i,2),paliId))

          % update coordinates
          cord_before=length(cord);
           if ~ismember(walls(i,1),AA)
           cord(walls(i,1),:)=coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:)));
            end
          if ~ismember(walls(i,2),AA)
           cord(walls(i,2),:)=coordinates(walls(i,2),:)-
0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:)));
          end



          ab1=[];
          ad11=[];
          ad1=[];
          ad33=[];
          ad_33=[];
          an=[];
          ar=[];
          am1=[];
          am2=[];
              for j=1:length(cord)-length(coordinates)

                  ad1=coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:)));
                  ab1(j)=norm(ad1-cord(j+length(coordinates),:),2);
              end

              ad11=find(ab1<0.0001);
              if isempty(ad11)
             cord(length(cord)+1,:)=coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:)));
              end

           for j=1:length(cord)-length(coordinates)

                  ad2=coordinates(walls(i,2),:)-
0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:)));
                  ab2(j)=norm(ad2-cord(j+length(coordinates),:),2);
                  ad_2=coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:)));
              end
```

```
            ad22=find(ab2<0.01);

            for j=1:length(coordinates)
                ad3(j)=norm(ad2-cord(j,:),2);
            end
            ad33=find(ad3<0.01);

            if isempty(ad22)&isempty(ad33)
           cord(length(cord)+1,:)=coordinates(walls(i,2),:)-
0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:)));
            elseif isempty(ad22)& ~isempty(ad33)
           cord(length(cord)+1,:)=coordinates(walls(i,2),:)-
0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:)));
            end
            %% subdividing walls of the first cell

             cord(length(cord)+1,:)=0.5*((coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:)))).*...
                +(coordinates(walls(i,2),:)-0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:))));
             cord(length(cord)+1,:)=0.5*((coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:))))+cord(length(cord),:));
             cord(length(cord)+1,:)=0.5*((coordinates(walls(i,2),:)-
0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,1)},:))))+cord(length(cord)-1,:));
            %% walls of the second cell

            cord(length(cord)+1,:)=0.5*((coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:)))).*...
                +(coordinates(walls(i,2),:)-0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:))));
            cord(length(cord)+1,:)=0.5*((coordinates(walls(i,1),:)-
0.1*(coordinates(walls(i,1),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:))))+cord(length(cord),:));
             cord(length(cord)+1,:)=0.5*((coordinates(walls(i,2),:)-
0.1*(coordinates(walls(i,2),:)-
mean(coordinates(CellvertexId{wall2cell(i,2)},:))))+cord(length(cord)-1,:));
            cord_after=length(cord);
            new_cord=cord_after-cord_before;


            if new_cord==8
                adhesive_wall=[adhesive_wall;[walls(i,1),length(cord)-7]];
                adhesive_wall=[adhesive_wall;[length(cord)-4,length(cord)-
1]];
                adhesive_wall=[adhesive_wall;[length(cord)-5,length(cord)-
2]];
                adhesive_wall=[adhesive_wall;[length(cord)-
3,length(cord)]];
                adhesive_wall=[adhesive_wall;[walls(i,2),length(cord)-6]];
            elseif ~isempty(ad11)
```

```
                    arr=find(adhesive_wall(:,:)==walls(i,1));
                    if length(arr)==2


adhesive_wall=[adhesive_wall;[adhesive_wall(arr(1),2),adhesive_wall(arr(2),2)
]];
                    else


adhesive_wall=[adhesive_wall;[walls(i,1),length(coordinates)+ad11]];
                    end
                    adhesive_wall=[adhesive_wall;[length(cord)-4,length(cord)-
1]];
                    adhesive_wall=[adhesive_wall;[length(cord)-5,length(cord)-
2]];
                    adhesive_wall=[adhesive_wall;[length(cord)-
3,length(cord)]];
                    adhesive_wall=[adhesive_wall;[walls(i,2),length(cord)-6]];
                else
                    adhesive_wall=[adhesive_wall;[walls(i,1),length(cord)-6]];
                    adhesive_wall=[adhesive_wall;[length(cord)-4,length(cord)-
1]];
                    adhesive_wall=[adhesive_wall;[length(cord)-5,length(cord)-
2]];
                    adhesive_wall=[adhesive_wall;[length(cord)-
3,length(cord)]];

adhesive_wall=[adhesive_wall;[walls(i,2),length(coordinates)+ad22]];
                end
                if
ismember(wall2cell(i,1),paliId)&ismember(wall2cell(i,2),paliId)

                    split_wall=[split_wall;adhesive_wall(length(adhesive_wall)-
4:length(adhesive_wall),:)];

con_point_pal=[con_point_pal;adhesive_wall(length(adhesive_wall)-2,:)];

con_point_2_pal=[con_point_2_pal;adhesive_wall(length(adhesive_wall)-
1,:);adhesive_wall(length(adhesive_wall)-3,:)];

con_point_3_pal=[con_point_3_pal;adhesive_wall(length(adhesive_wall)-
4,:);adhesive_wall(length(adhesive_wall),:)];
                elseif
(ismember(wall2cell(i,1),epiId)&~ismember(wall2cell(i,2),epiId))|(~ismember(w
all2cell(i,1),epiId)&ismember(wall2cell(i,2),epiId))

                    split_wall=[split_wall;adhesive_wall(length(adhesive_wall)-
4:length(adhesive_wall),:)];

con_point_epi=[con_point_epi;adhesive_wall(length(adhesive_wall)-2,:)];

con_point_2_epi=[con_point_2_epi;adhesive_wall(length(adhesive_wall)-
1,:);adhesive_wall(length(adhesive_wall)-3,:)];

con_point_3_epi=[con_point_3_epi;adhesive_wall(length(adhesive_wall)-
4,:);adhesive_wall(length(adhesive_wall),:)];
```

```
            elseif
~ismember(wall2cell(i,1),[epiId;paliId])&~ismember(wall2cell(i,2),[epiId;pali
Id])

                split_wall=[split_wall;adhesive_wall(length(adhesive_wall)-
4:length(adhesive_wall),:)];

con_point_spongy=[con_point_spongy;adhesive_wall(length(adhesive_wall)-2,:)];

con_point_2_spongy=[con_point_2_spongy;adhesive_wall(length(adhesive_wall)-
1,:);adhesive_wall(length(adhesive_wall)-3,:)];

con_point_3_spongy=[con_point_3_spongy;adhesive_wall(length(adhesive_wall)-
4,:);adhesive_wall(length(adhesive_wall),:)];
            elseif
(ismember(wall2cell(i,1),[paliId])&~ismember(wall2cell(i,2),[epiId;paliId]))|
(ismember(wall2cell(i,2),[paliId])&~ismember(wall2cell(i,1),[epiId;paliId]))
                split_wall=[split_wall;adhesive_wall(length(adhesive_wall)-
4:length(adhesive_wall),:)];

con_point_pal_spongy=[con_point_pal_spongy;adhesive_wall(length(adhesive_wall
)-2,:)];

con_point_2_pal_spongy=[con_point_2_pal_spongy;adhesive_wall(length(adhesive_
wall)-1,:);adhesive_wall(length(adhesive_wall)-3,:)];

con_point_3_pal_spongy=[con_point_3_pal_spongy;adhesive_wall(length(adhesive_
wall)-4,:);adhesive_wall(length(adhesive_wall),:)];

            end

            adhesive_wall=unique(adhesive_wall(:,:),'rows');
          a11=find(CellvertexId{wall2cell(i,1)}==walls(i,1));
          a21=find(CellvertexId{wall2cell(i,1)}==walls(i,2));
          a12=find(CellvertexId{wall2cell(i,2)}==walls(i,1));
          a22=find(CellvertexId{wall2cell(i,2)}==walls(i,2));

          ab11=find(CellID{wall2cell(i,1)}==walls(i,1));
          ab21=find(CellID{wall2cell(i,1)}==walls(i,2));
          ab12=find(CellID{wall2cell(i,2)}==walls(i,1));
          ab22=find(CellID{wall2cell(i,2)}==walls(i,2));

          % update vertices of first cell
  as=CellID{wall2cell(i,1)};
    if ~isempty(ab11)&~isempty(ab21)
            if ab11(1)>ab21(1)& ab21(1)~=1
                if isempty(ad33)

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab21(1)),length(cord)-
4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab11(1):length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                  wall(i,:)=[walls(i,1),length(cord)-4];
                  wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                  wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                  wall(length(wall)+1,:)=[length(cord)-3,walls(i,2)];
```

```
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end
                  else
                    CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab21(1)-
1),length(cord)-4,length(cord)-5,length(cord)-3,length(cord)-
6,CellID{wall2cell(i,1)}(ab11(1):length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                    wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                end
            elseif ab11(1)>ab21(1)& ab21(1)==1
                if isempty(ad33)

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(ab21(1):length(CellID{wall2cel
l(i,1)})-1),length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab21(1))];
                    wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,walls(i,2)];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
```

```
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end
                else
                    CellID{wall2cell(i,1)}=[length(cord)-4,length(cord)-
5,length(cord)-3,length(cord)-
6,CellID{wall2cell(i,1)}(ab21(1)+1:length(CellID{wall2cell(i,1)})-
1),length(cord)-1,length(cord)-4];
                     wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                end
            elseif ab11(1)<ab21(1)& ab11(1)~=1
                if isempty(ad33)

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab11(1)),length(cord)-
4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab21(1):length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                    wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,walls(i,2)];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
```

```
                        end
                    end
                else

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab11(1)),length(cord)-
4,length(cord)-5,length(cord)-3,length(cord)-
6,CellID{wall2cell(i,1)}(ab21(1)+1:length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                     wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                end
            elseif ab11(1)<ab21(1)& ab11(1)==1
                if isempty(ad33)

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(ab11(1)),length(cord)-
4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab11(1)+1:length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(ab11(1))];
                    wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,walls(i,2)];
                    for j=1:length(walls)
                       if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            walls(i,:)
                            walls(j,:)
                            azr=3;
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end
```

```
                else

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(ab11(1):length(CellID{wall2cel
l(i,1)})-2),length(cord)-6,length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab11(1))];
                    wall(i,:)=[walls(i,1),length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=walls(i,1);
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                end
            end
    elseif (~isempty(ab11)&isempty(ab21)) & (~isempty(a11) &~isempty(a21))
                if a11(1)>a21(1)& a21(1)~=1
                    am1=CellID{wall2cell(i,1)}(length(CellID{wall2cell(i,1)})-
1);
                    am2=CellID{wall2cell(i,1)}(a21(1));
                   CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab11(1)-
1),length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab11(1):length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=am1;
                            elseif an==2
                            wall(j,ar)=am2;
                            end
                        end
                    end
                elseif a11(1)>a21(1)& a21(1)==1
                    am1=CellID{wall2cell(i,1)}(length(CellID{wall2cell(i,1)})-
1);
```

```
                am2=CellID{wall2cell(i,1)}(a21(1));

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(a21(1):length(CellID{wall2cell
(i,1)})-1),length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(a21(1))];
                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))

                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));

                                if an==1
                                wall(j,ar)=am1;
                                elseif an==2
                                wall(j,ar)=am2;
                                end
                                end

                    end
            elseif a11(1)<a21(1)& a11(1)~=1
                am1=CellID{wall2cell(i,1)}(ab11(1));
                am2=CellID{wall2cell(i,1)}(ab11(1)+1);

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab11(1)),length(cord)-
4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab11(1)+1:length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                    for j=1:length(walls)
                       if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=am1;
                                elseif an==2
                                wall(j,ar)=am2;
                                end
                            end
                    end
            elseif a11(1)<a21(1)& a11(1)==1
                am1=CellID{wall2cell(i,1)}(a11(1));
```

```
                am2=CellID{wall2cell(i,1)}(a11(1)+1);

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(a11(1)),length(cord)-
4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(a11(1)+1:length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(a11(1))];

                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=am1;
                            elseif an==2
                            wall(j,ar)=am2;
                            end
                        end
                    end
                end



    elseif (isempty(ab11)& ~isempty(ab21))& (~isempty(a11) &~isempty(a21))
              if a11(1)>a21(1)& a21(1)~=1
                  am1=CellID{wall2cell(i,1)}(ab21(1)+1);
                  am2=CellID{wall2cell(i,1)}(ab21(1));

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab21(1)),length(cord)-
4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab21(1)+1:length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=am1;
                            elseif an==2
                            wall(j,ar)=am2;
                            end
```

```
                        end
                    end
                elseif a11(1)>a21(1)& a21(1)==1
                    am1=CellID{wall2cell(i,1)}(ab21(1));
                    am2=CellID{wall2cell(i,1)}(length(CellID{wall2cell(i,1)})-
1);

CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(ab21(1):length(CellID{wall2cel
l(i,1)})-1),length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab21(1))];
                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=am1;
                            elseif an==2
                            wall(j,ar)=am2;
                            end
                        end
                    end
                elseif a11(1)<a21(1)& a11(1)~=1
                    if isempty(ad33)
                       am1=CellID{wall2cell(i,1)}(ab21(1)-1);
                    am2=CellID{wall2cell(i,1)}(ab21(1));
                   CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab21(1)-
1),length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab21(1):length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                    wall(i,:)=[am1,length(cord)-4];
                    wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                    wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                    wall(length(wall)+1,:)=[length(cord)-3,am2];
                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=am1;
                            elseif an==2
                            wall(j,ar)=am2;
                            end
                        end
                    end
                else
```

```
                    am1=CellID{wall2cell(i,1)}(ab21(1)-1);
                 am2=CellID{wall2cell(i,1)}(ab21(1)+1);
                  CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(1:ab21(1)-
1),length(cord)-4,length(cord)-5,length(cord)-3,length(cord)-
6,CellID{wall2cell(i,1)}(ab21(1)+1:length(CellID{wall2cell(i,1)})-
1),CellID{wall2cell(i,1)}(1)];
                  wall(i,:)=[am1,length(cord)-4];
                  wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                  wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                  wall(length(wall)+1,:)=[length(cord)-3,length(cord)-6];
                  for j=1:length(walls)
                      if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                  &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                              ar=find(ismember(walls(j,:),walls(i,:)));
                              an=find(ismember(walls(i,:),walls(j,:)));
                              if an==1
                              wall(j,ar)=am1;
                              elseif an==2
                              wall(j,ar)=am2;
                              end
                          end
                    end
                end
            elseif a11(1)<a21(1)& a11(1)==1
                am1=CellID{wall2cell(i,1)}(length(CellID{wall2cell(i,1)})-
1);
                am2=CellID{wall2cell(i,1)}(ab21(1)-1);
               CellID{wall2cell(i,1)}=[CellID{wall2cell(i,1)}(ab21(1)-
1),length(cord)-4,length(cord)-5,length(cord)-
3,CellID{wall2cell(i,1)}(ab21(1)),CellID{wall2cell(i,1)}(ab21(1)-1)];
                  wall(i,:)=[am1,length(cord)-4];
                  wall(length(wall)+1,:)=[length(cord)-4,length(cord)-5];
                  wall(length(wall)+1,:)=[length(cord)-5,length(cord)-3];
                  wall(length(wall)+1,:)=[length(cord)-3,am2];
                  abe=2;
                  for j=1:length(walls)
                     if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,1),wall2cell(j,:)))...
                                  &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,1)}))))
                              ar=find(ismember(walls(j,:),walls(i,:)));
                              an=find(ismember(walls(i,:),walls(j,:)));
                              if an==1
                              wall(j,ar)=am1;
                              elseif an==2
                              wall(j,ar)=am2;
                              end
                          end
                    end
            end

    end
```

```
      as=CellID{wall2cell(i,1)};

      % update vertices of second cell
        az=CellID{wall2cell(i,2)};
    if ~isempty(ab12)&~isempty(ab22)

          if ab12(1)>ab22(1)& ab22(1)~=1

                    if isempty(ad33)

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-1),[length(cord)-
6,length(cord),length(cord)-2,length(cord)-1,length(cord)-
7],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                  &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                    ad=[];
                  for ii=1:length(walls)
            if ismember(wall2cell(i,2),wall2cell(ii,:))
                ad=find(~ismember(wall(ii,:),CellID{wall2cell(i,2)}));
                if ~isempty(ad)& length(ad)==1
                  if ad==1
                wall(ii,ad)=length(cord)-7;
                  else
                 wall(ii,ad)=length(cord)-6;
                  end
                end
             end
         end

                    else

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-
1),[walls(i,2),length(cord),length(cord)-2,length(cord)-1,length(cord)-
7],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                      wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-
1]];
```

```
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),walls(i,2)];

                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end
                    end


             elseif ab12(1)>ab22(1)& ab22(1)==1
                    if isempty(ad33)
                    CellID{wall2cell(i,2)}=[[length(cord)-
6,length(cord),length(cord)-2,length(cord)-1,length(cord)-
7],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),length(cord)-6];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end

                    else

CellID{wall2cell(i,2)}=[[walls(i,2),length(cord),length(cord)-2,length(cord)-
1,length(cord)-
7],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),walls(i,2)];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
```

```
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),walls(i,2)];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end

                    end

           elseif ab12(1)<ab22(1)& ab12(1)~=1
                    if isempty(ad33)

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab12(1)-1),[length(cord)-
7,length(cord)-1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                       wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-
1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                    else

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab12(1)-1),[length(cord)-
7,length(cord)-1,length(cord)-
2,length(cord),walls(i,2)],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wal
l2cell(i,2)})-1),CellID{wall2cell(i,2)}(1)];
                       wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-
1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
```

```
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),walls(i,2)];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end
                    end


            elseif ab12(1)<ab22(1)& ab12(1)==1
                    if isempty(ad33)
                    CellID{wall2cell(i,2)}=[[length(cord)-7,length(cord)-
1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wall2cell(i,2)})-
1),length(cord)-7];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                    else
                        CellID{wall2cell(i,2)}=[[length(cord)-7,length(cord)-
1,length(cord)-
2,length(cord),walls(i,2)],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wal
l2cell(i,2)})-1),length(cord)-7];
                        wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-
1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),walls(i,2)];
                    for j=1:length(walls)
```

```
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=walls(i,2);
                            end
                        end
                    end
                    end

           end

    elseif (~isempty(ab12)&isempty(ab22))& (~isempty(a12) &~isempty(a22))

           if a12(1)>a22(1)& a22(1)~=1

                    if new_cord==8,


CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab12(1)-2),[length(cord)-
6,length(cord),length(cord)-2,length(cord)-1,length(cord)-
7],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                    else

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab12(1)-
2),[length(coordinates)+ad22,length(cord),length(cord)-2,length(cord)-
1,length(cord)-
6],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
```

```
                     wall(length(wall)+1,:)=[[length(cord)-6,length(cord)-
1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];

wall(length(wall)+1,:)=[length(cord),length(coordinates)+ad22];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-6;
                            elseif an==2
                            wall(j,ar)=length(coordinates)+ad22;
                            end
                        end
                    end
                    end


             elseif a12(1)>a22(1)& a22(1)==1
                if new_cord==8
                    CellID{wall2cell(i,2)}=[[length(cord)-
6,length(cord),length(cord)-2,length(cord)-1,length(cord)-
7],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),length(cord)-6];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                else

CellID{wall2cell(i,2)}=[[length(coordinates)+ad22,length(cord),length(cord)-
2,length(cord)-1,length(cord)-
6],CellID{wall2cell(i,2)}(ab12(1)+1:length(CellID{wall2cell(i,2)})-
1),length(coordinates)+ad22];
                    wall(length(wall)+1,:)=[[length(cord)-6,length(cord)-1]];
```

```
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];

wall(length(wall)+1,:)=[length(cord),length(coordinates)+ad22];
                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(cord)-6;
                                elseif an==2
                                wall(j,ar)=length(coordinates)+ad22;
                                end
                        end
                    end
                end

            elseif a12(1)<a22(1)& a12(1)~=1

                     if new_cord==8


CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab12(1)-1),[length(cord)-
7,length(cord)-1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab12(1)+2:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                      wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-
1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(cord)-7;
                                elseif an==2
                                wall(j,ar)=length(cord)-6;
                                end
                        end
                    end
                     else

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab12(1)-1),[length(cord)-
6,length(cord)-1,length(cord)-
2,length(cord),length(coordinates)+ad22],CellID{wall2cell(i,2)}(ab12(1)+2:len
gth(CellID{wall2cell(i,2)})-1),CellID{wall2cell(i,2)}(1)];
```

```
                      wall(length(wall)+1,:)=[[length(cord)-6,length(cord)-
1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];

wall(length(wall)+1,:)=[length(cord),length(coordinates)+ad22];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-6;
                            elseif an==2
                            wall(j,ar)=length(coordinates)+ad22;
                            end
                        end
                    end
                     end

             elseif a12(1)<a22(1)& a12(1)==1
                 if new_cord==8
                    CellID{wall2cell(i,2)}=[[length(cord)-7,length(cord)-
1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab12(1)+2:length(CellID{wall2cell(i,2)})-
1),length(cord)-7];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
                 else
                   CellID{wall2cell(i,2)}=[[length(cord)-6,length(cord)-
1,length(cord)-
2,length(cord),length(coordinates)+ad22],CellID{wall2cell(i,2)}(ab12(1)+2:len
gth(CellID{wall2cell(i,2)})-1),length(cord)-6];
                   wall(length(wall)+1,:)=[[length(cord)-6,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
```

```
wall(length(wall)+1,:)=[length(cord),length(coordinates)+ad22];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(cord)-6;
                                elseif an==2
                                wall(j,ar)=length(coordinates)+ad22;
                                end
                            end
                        end
                    end

           end

   elseif (isempty(ab12)&~isempty(ab22))& (~isempty(a12) &~isempty(a22))

           if a12(1)>a22(1)& a22(1)~=1

                    if new_cord==8,


CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-1),[length(cord)-
6,length(cord),length(cord)-2,length(cord)-1,length(cord)-
7],CellID{wall2cell(i,2)}(ab22(1)+2:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                    wall(length(wall)+1,:)=[[length(cord)-7,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(cord)-7;
                                elseif an==2
                                wall(j,ar)=length(cord)-6;
                                end
                            end
                    end

                    elseif isempty(ad33)

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-1),[length(cord)-
6,length(cord),length(cord)-2,length(cord)-
```

```
1,length(coordinates)+ad11],CellID{wall2cell(i,2)}(ab22(1)+2:length(CellID{wa
ll2cell(i,2)})-1),CellID{wall2cell(i,2)}(1)];

wall(length(wall)+1,:)=[[length(coordinates)+ad11,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(coordinates)+ad11;
                                elseif an==2
                                wall(j,ar)=length(cord)-6;
                                end
                            end
                    end
                    else

CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-
1),[ad33,length(cord),length(cord)-2,length(cord)-
1,length(coordinates)+ad11],CellID{wall2cell(i,2)}(ab22(1)+2:length(CellID{wa
ll2cell(i,2)})-1),CellID{wall2cell(i,2)}(1)];

wall(length(wall)+1,:)=[[length(coordinates)+ad11,length(cord)-1]];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),ad33];
                    for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(coordinates)+ad11;
                                elseif an==2
                                wall(j,ar)=ad33;
                                end
                            end
                    end

                    end

            elseif a12(1)>a22(1)& a22(1)==1
                if new_cord==8,
                   CellID{wall2cell(i,2)}=[[length(cord)-
6,length(cord),length(cord)-2,length(cord)-1,length(cord)-
```

```
7],CellID{wall2cell(i,2)}(ab22(1)+2:length(CellID{wall2cell(i,2)})-
1),length(cord)-6];
                    wall(length(wall)+1,:)=[length(cord)-7,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(cord)-7;
                                elseif an==2
                                wall(j,ar)=length(cord)-6;
                                end
                        end
                    end
                elseif isempty(ad33)
                  CellID{wall2cell(i,2)}=[[length(cord)-
6,length(cord),length(cord)-2,length(cord)-
1,length(coordinates)+ad11],CellID{wall2cell(i,2)}(ab22(1)+2:length(CellID{wa
ll2cell(i,2)})-1),length(cord)-6];

wall(length(wall)+1,:)=[length(coordinates)+ad11,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                    &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                                ar=find(ismember(walls(j,:),walls(i,:)));
                                an=find(ismember(walls(i,:),walls(j,:)));
                                if an==1
                                wall(j,ar)=length(coordinates)+ad11;
                                elseif an==2
                                wall(j,ar)=length(cord)-6;
                                end
                        end
                    end
                else
                  CellID{wall2cell(i,2)}=[[ad33,length(cord),length(cord)-
2,length(cord)-
1,length(coordinates)+ad11],CellID{wall2cell(i,2)}(ab22(1)+2:length(CellID{wa
ll2cell(i,2)})-1),ad33];

wall(length(wall)+1,:)=[length(coordinates)+ad11,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),ad33];
                     for j=1:length(walls)
```

```
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(coordinates)+ad11;
                            elseif an==2
                            wall(j,ar)=ad33;
                            end
                        end
                    end
                end



           elseif a12(1)<a22(1)& a12(1)~=1
               if new_cord==8,


CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-2),[length(cord)-
7,length(cord)-1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];
                     wall(length(wall)+1,:)=[length(cord)-7,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                     for j=1:length(walls)
                        if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                            ar=find(ismember(walls(j,:),walls(i,:)));
                            an=find(ismember(walls(i,:),walls(j,:)));
                            if an==1
                            wall(j,ar)=length(cord)-7;
                            elseif an==2
                            wall(j,ar)=length(cord)-6;
                            end
                        end
                    end
               else
                   CellID{wall2cell(i,2)}=[CellID{wall2cell(i,2)}(1:ab22(1)-
2),[length(coordinates)+ad11,length(cord)-1,length(cord)-
2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wall2cell(i,2)})-
1),CellID{wall2cell(i,2)}(1)];

wall(length(wall)+1,:)=[length(coordinates)+ad11,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
```

```
                    for j=1:length(walls)
                       if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                  &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                           ar=find(ismember(walls(j,:),walls(i,:)));
                           an=find(ismember(walls(i,:),walls(j,:)));
                           if an==1
                           wall(j,ar)=length(coordinates)+ad11;
                           elseif an==2
                           wall(j,ar)=length(cord)-6;
                           end
                       end
                   end
               end




            elseif a12(1)<a22(1)& a12(1)==1
                 if new_cord==8,
                   CellID{wall2cell(i,2)}=[[length(cord)-7,length(cord)-
1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wall2cell(i,2)})-
1),length(cord)-7];
                    wall(length(wall)+1,:)=[length(cord)-7,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
                     for j=1:length(walls)
                       if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                  &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                           ar=find(ismember(walls(j,:),walls(i,:)));
                           an=find(ismember(walls(i,:),walls(j,:)));
                           if an==1
                           wall(j,ar)=length(cord)-7;
                           elseif an==2
                           wall(j,ar)=length(cord)-6;
                           end
                       end
                   end
                 else

CellID{wall2cell(i,2)}=[[length(coordinates)+ad11,length(cord)-
1,length(cord)-2,length(cord),length(cord)-
6],CellID{wall2cell(i,2)}(ab22(1)+1:length(CellID{wall2cell(i,2)})-
1),length(coordinates)+ad11];

wall(length(wall)+1,:)=[length(coordinates)+ad11,length(cord)-1];
                    wall(length(wall)+1,:)=[length(cord)-1,length(cord)-2];
                    wall(length(wall)+1,:)=[length(cord)-2,length(cord)];
                    wall(length(wall)+1,:)=[length(cord),length(cord)-6];
```

```
                    for j=1:length(walls)
                       if (ismember(numberofcells,wall2cell(j,:))&
ismember(wall2cell(i,2),wall2cell(j,:)))...
                                  &
(~isempty(find(ismember(walls(j,:),walls(i,:))))&~isempty(find(~ismember(wall
s(j,:),CellID{wall2cell(i,2)}))))
                           ar=find(ismember(walls(j,:),walls(i,:)));
                           an=find(ismember(walls(i,:),walls(j,:)));
                           if an==1
                           wall(j,ar)=length(coordinates)+ad11;
                           elseif an==2
                           wall(j,ar)=length(cord)-6;
                           end
                       end
                    end
                 end

          end

       az=CellID{wall2cell(i,2)};
    end

          AA=[AA;walls(i,:)];
          BB=[BB;wall2cell(i,:)];
          w2c(length(w2c)+1:length(wall),1)=wall2cell(i,1);
           w2c(length(w2c)-6:length(w2c),2)=wall2cell(i,2);
           WO_1=[WO_1,i,length(wall)-6:length(wall)-4];
           WO_2=[WO_2,length(wall)-3:length(wall)];

                 split_turgor_1=[split_turgor_1,[i,length(wall)-
6:length(wall)-4]];
                 split_turgor_2=[split_turgor_2,[length(wall)-
3:length(wall)]];

        end


     end

          ADH=adhesive_wall;
        for i=1:length(adhesive_wall)
          add=[];
          for j=1:length(adhesive_wall)
              if ismember(adhesive_wall(i,1),adhesive_wall(j,:))
                  add=[add,adhesive_wall(j,:)];
              end
          end
          adw{i}=unique([adhesive_wall(i,:),add]);
         if length(adw{i})==3

            ADH(length(ADH)+1,:)=[adw{i}(2),adw{i}(3)];


          end
```

```
        end
        adhesive_wall= unique(ADH,'rows');

        for i=1:length(adhesive_wall)

        ll(i)=norm(cord(adhesive_wall(i,1),:)-cord(adhesive_wall(i,2),:),2);

          ad_direction(i,:)=(cord(adhesive_wall(i,1),:)-
cord(adhesive_wall(i,2),:))/ll(i);
        end

          codd=[];
            codd=cord;

        for i=1:length(adhesive_wall)
            add=[];
            for j=1:length(adhesive_wall)
                if ismember(adhesive_wall(i,1),adhesive_wall(j,:))
                    add=[add,adhesive_wall(j,:)];
                end
            end
            adw{i}=unique([adhesive_wall(i,:),add]);
           if length(adw{i})==3

cord(adw{i}(1),:)=mean(codd([adw{i}(1);adw{i}(2);adw{i}(3)],:));

cord(adw{i}(2),:)=mean(codd([adw{i}(1);adw{i}(2);adw{i}(3)],:));

cord(adw{i}(3),:)=mean(codd([adw{i}(1);adw{i}(2);adw{i}(3)],:));

           elseif length(adw{i})==2
               cord(adw{i}(1),:)=mean(codd([adw{i}(1);adw{i}(2)],:));
               cord(adw{i}(2),:)=mean(codd([adw{i}(1);adw{i}(2)],:));

           end

        end
        for i=1:length(adhesive_wall)
            if ismember(adhesive_wall(i,1),left_boundary)|
ismember(adhesive_wall(i,2),left_boundary)
                cord(adhesive_wall(i,:),1)=min(cord(:,1));

            end
            if ismember(adhesive_wall(i,1),right_boundary)|
ismember(adhesive_wall(i,2),right_boundary)
                cord(adhesive_wall(i,:),1)=max(cord(:,1));
            end
            if ismember(adhesive_wall(i,1),bottom_boundaryy)|
ismember(adhesive_wall(i,2),bottom_boundaryy)
                cord(adhesive_wall(i,:),2)=min(cord(:,2));

            end
            if ismember(adhesive_wall(i,1),top_boundaryy)|
ismember(adhesive_wall(i,2),top_boundaryy)
```

```
                cord(adhesive_wall(i,:),2)=max(cord(:,2));
            end
        end
       coordinates=cord;
        CellvertexId=CellID;
        walls=wall;
        wall2cell=w2c;


coordinates=cord;


    counter=0;

   for i=1:length(CellvertexId)
       for j=1:length(CellvertexId{i})-1
           counter=counter+1;
           wall1(counter,:)=[CellvertexId{i}(j),CellvertexId{i}(j+1)];
           end

   end
w2c1=numberofcells*ones(length(wall1),2);


   for i=1:length(wall1)
       index=[];
      for j=1:length(CellvertexId)
          if
~isempty(intersect(coordinates(wall1(i,1),:),coordinates(CellvertexId{j},:),'
rows'))&
~isempty(intersect(coordinates(wall1(i,2),:),coordinates(CellvertexId{j},:),'
rows'))
       index=[index,j];
          end
      end
       if length(index)==2
       w2c1(i,:)=index;
       else
           w2c1(i,1)=index;
       end
   end

    for i=1:length(walls)
           l_n(i)=norm(coordinates(walls(i,1),:)-
coordinates(walls(i,2),:),2);
        end
        for i=1:length(CellvertexId)

Cellarea(i)=polyarea(coordinates(CellvertexId{i},1),coordinates(CellvertexId{
i},2));
        end
cd(string)
      save con_point_pal con_point_pal;
      save con_point_2_pal con_point_2_pal;
      save con_point_3_pal con_point_3_pal;
```

```
save con_point_epi con_point_epi;
save con_point_2_epi con_point_2_epi;
save con_point_3_epi con_point_3_epi;

save con_point_spongy con_point_spongy;
save con_point_2_spongy con_point_2_spongy;
save con_point_3_spongy con_point_3_spongy;

save con_point_pal_spongy con_point_pal_spongy;
save con_point_2_pal_spongy con_point_2_pal_spongy;
save con_point_3_pal_spongy con_point_3_pal_spongy;
cd(string11)
```

Starttesselation

```
function [
coordinates,dimension,numberofcells,walls,wall2cell,Wallbnd,l_n,Cellarea,Cell
vertexId,epiId,hypoId,paliId,stoneId,wallvepiuper,oldwall]=starttesselation(g
eom)

% Defining the size of the geometry and starting tessellations
close all
% geom=2
rand('state',geom);

numcell=17;
x = 1*sqrt(100*numcell)*rand(numcell,1);
y = 1*sqrt(100*numcell)*rand(numcell,1);
centroids=[x,y];

% figure(1)
% voronoi(centroids(:,1),centroids(:,2))
% xmax=max(x);
% ymax=max(y);

xxmin=0;
xxmax=1*sqrt(100*numcell);

centleaf=centroids(centroids(:,2)<0.7*mean(centroids(:,2)),:);
centroids=centleaf;
n=length(centroids);
Bc=zeros(n,2);

yymin=0;
yymax=max(centroids(:,2))+1;

%%
iteration=1;
%% iteration for generating the centroidal Voronoi cells

while sum(((centroids(:,1)-Bc(:,1)).*(centroids(:,1)-
Bc(:,1)))/n)+sum(((centroids(:,2)-Bc(:,2)).*(centroids(:,2)-Bc(:,2)))/n)>
0.01
```

```
    BB=[centroids(:,1),centroids(:,2);-centroids(:,1),centroids(:,2)...
    ;centroids(:,1),-centroids(:,2);xxmax+(xxmax-
centroids(:,1)),centroids(:,2)...
    ; centroids(:,1),yymax+(yymax-centroids(:,2))];

centroids2=[];
%  Voronoi diagram with centroids C and vertices V
[V,C]=voronoin(BB);

   for j=1:length(C)

        X = V(C{j},:);
        if ~isinf(X)
            % calculating  area and centroid of the voronoi region
            x0=mean(X(:,1));
            y0=mean(X(:,2));

            centroids2=[centroids2; x0 y0];

            % However, we are not interested in the centroids here any
more;we use
            % random generators

        end

    end

%% selecting properties in the range of interest
B=centroids2;

% centroidal points in the region of interest

[Id,~]=nearestpoint(centroids(:,1),BB(:,1));
size(BB)
size(B)
B0=B(Id,1:2);

% area of voronoi regions of interest

    Bc=centroids;
    centroids= B0;
    clc
    iteration=iteration+1;

end
%%
Centroids=[centroids(:,1),centroids(:,2);xxmin-
centroids(:,1),centroids(:,2)...
    ;centroids(:,1),yymin-centroids(:,2);xxmax+(xxmax-
centroids(:,1)),centroids(:,2)...
    ; centroids(:,1),yymax+(yymax-centroids(:,2))];

%% constructing voronoi diagrams
```

```
[V,C]=voronoin(Centroids);
[~,~]=voronoi(Centroids(:,1),Centroids(:,2));

X1=[];

for j=1:length(C)
   if V(C{j},1)~=inf

az=find(inpolygon(centroids(:,1),centroids(:,2),V(C{j},1),V(C{j},2))==1);

       if ~isempty(az)&V(C{j},1)~=inf
         X =(V(C{j},:));
        if ~isinf(X)
        X=[[X(:,1) ; X(1,1)],[X(:,2) ; X(1,2)]];
        end

        X1=[X1;X];
        V0{j}=X;

       end
   end
end



dis_x=[];
dis_y=[];
index=[];
index11=[];
COcel=[];

for i=1:length(V0)

    index1=[];
    coord_cell=V0{i};
    COcel=[COcel;coord_cell];

    for j=1:length(V0)
        if ~isequal(i,j)
            neighbor=intersect(coord_cell, V0{j},'rows');
            if ~isempty(neighbor)
                index1=[index1; i j];                          % pair of
neighbor cells
                dis_x1= sqrt((centroids(i,1)-centroids(j,1))^2 );
                dis_y1=sqrt((centroids(i,2)-centroids(j,2))^2);
                dis_x=[dis_x;dis_x1];
                dis_y=[dis_y;dis_y1];
                index=[index; j];                              % neighbour
cells to a given cell i
                index11=[index11;index1];
                NN{i}=index1;
            end
        end
    end

end
```

```
xk=COcel(10:length(COcel)-10,1);
yk=COcel(10:length(COcel)-10,2);

b=0.01;
a0=xk(:)<b+min(xk)==0;
xk(a0==0)=b+min(xk);
b0=xk(:)>-b+max(xk)==0;
xk(b0==0)=-b+max(xk);
c0=yk(:)<b+min(yk)==0;
yk(c0==0)=b+min(yk);
d0=yk(:)>-b+max(yk)==0;
yk(d0==0)=-b+max(yk);
k1=convhull(xk,yk);

AML=[];
for i=1:length(V0)
    [in , ~]=inpolygon(V0{i}(:,1),V0{i}(:,2),xk(k1),yk(k1));
    if isempty(find(in==0, 1))
    AML=[AML;i];
    end
end
%% Identifying the hypodermis regions to avoid^pores in this region
hypoId=[];
  for i=1:length(V0)
      if max(V0{i}(:,2))>1.5*mean(COcel(:,2))
          hypoId=[hypoId,i];
      end
  end

  hypoId=[];
  %% making pores
poreId=[];
for i=1:length(AML)
   Ind1=[];
    for j=1:length(AML)

        if ~isequal(i,j)
            neb=intersect(V0{AML(i)},V0{AML(j)},'rows');
            if ~isempty(neb)
                Ind1=[Ind1;AML(i) AML(j)];

            end
        end
    end
    if (size(Ind1,1)<1|~ismember(Ind1,poreId))&~ismember(AML(i),hypoId)
        poreId=[poreId;AML(i)];
    end

end

porId=[];

for i=1:length(poreId)
    porId=[porId;poreId(i),NN{poreId(i)}(1,2)];
```

```
end


for i=1:4:size(porId,1)
    if ismember(porId(i,2),AML)& ~ismember(NN{porId(i,1)}(:,2),poreId)
    poreId=[poreId;porId(i,2)];
    end

end

PP=[];

for i=1:length(poreId)
    abf=(V0{poreId(i)});
    if abs(min(abf(:,1))-min(COcel(:,1)))>40||abs(max(abf(:,1))-
max(COcel(:,1)))>40
        PP=[PP;poreId(i)];
    end
end

poreId=PP;
%% making unique coordinates of the voronoi tesselation
[Id,~]=nearestpoint(centroids(:,1),Centroids(:,1));

for i=1:length(Id)

    counter=0;
    for k=1:length(V0)
        for h=1:length(V0{k})
             if ~ismember(k,poreId)
            counter=counter+1;

            cordinates(counter,1)=abs(V0{k}(h,1));
            cordinates(counter,2)=abs(V0{k}(h,2));
             end

        end

    end
end

[coordinates,~,J]=unique(cordinates,'rows');
Coordinates=coordinates;

%% updating the vornoi cell numbers as some of the voronoi cells are made
pores
counter=0;
for k=1:length(V0)

   if ~ismember(k,poreId)
       counter=counter+1;
       V1{counter}=V0{k};
   end
```

```
end

V0=V1;

    %% updating the hypodermal cell Id cell Id's are updated
    hypoId=[];
  for i=1:length(V0)
      if max(V0{i}(:,2))>1.5*mean(COcel(:,2))
          hypoId=[hypoId,i];
      end
  end

  hypoId=[];% if u are making cortex tissue

%% making wall vertices
counter=0;
for i=1:length(V0)
for j=1:length(V0{i})
counter=counter+1;
if j<length(V0{i})
Walls(counter,1)=J(counter);
Walls(counter,2)=J(counter+1);
end
end
end
for i=1:length(Walls)
for j=2:length(Walls)

    if Walls(j,1)==Walls(i,2)&&Walls(j,2)==Walls(i,1)
Walls(j,:)=0;
    end
end

end

[~,~,WAll1]=find(Walls(:,1));
[~,~,WAll2]=find(Walls(:,2));
Wall=[WAll1,WAll2];

Walls= unique(Wall,'rows');
for i=1:length(Walls)
    if ((abs((coordinates(Walls(i,2),1)-
coordinates(Walls(i,1),1)))>abs((coordinates(Walls(i,2),2)-
coordinates(Walls(i,1),2))))&&(coordinates(Walls(i,1),1))>coordinates(Walls(i
,2),1))...
            ||((abs((coordinates(Walls(i,2),2)-
coordinates(Walls(i,1),2)))>abs((coordinates(Walls(i,2),1)-
coordinates(Walls(i,1),1))))&&(coordinates(Walls(i,1),2))>coordinates(Walls(i
,2),2))
        wall(i,1)=Walls(i,2);
        wall(i,2)=Walls(i,1);
    else
        wall(i,1)=Walls(i,1);
        wall(i,2)=Walls(i,2);
    end
```

```
end

 walls=wall;

%% palisade cells
for i=1:length(walls)
   l(i)=norm(coordinates(walls(i,1),:)-coordinates(walls(i,2),:),2);
end

ab_epi=mean(l);
epiuper=[];
for i=1:length(coordinates)
    if (abs(coordinates(i,2)-max(coordinates(:,2))))<0.01
        epiuper=[epiuper,i];
    end
end

for i=1:length(epiuper)-1

coordepiuper2(i,:)=[0.5*(coordinates(epiuper(i),1)+coordinates(epiuper(i+1),1
)), coordinates(epiuper(i),2)];
end

coordepiuper2(length(coordepiuper2)+1,:)=[coordinates(epiuper(1),1),
coordinates(epiuper(1),2)+ab_epi];
for i=1:length(epiuper)-1
   coordepiuper2(length(coordepiuper2)+1,:)=[coordepiuper2(i,1),
coordepiuper2(i,2)+ab_epi];
end

coordepiuper2(length(coordepiuper2)+1,:)=[coordinates(epiuper(length(epiuper)
),1),coordinates(epiuper(length(epiuper)),2)+ab_epi];

wallvepiuper(1,1)=epiuper(1);
for i=1:length(epiuper)-1
   wallvepiuper(length(wallvepiuper)+1,1)=length(coordinates)+i;
end
   wallvepiuper(length(wallvepiuper)+1,1)=epiuper(length(epiuper));
   wallvepiuper(1,2)=length(coordinates)+length(epiuper);

for i=2:length(epiuper)
     wallvepiuper(i,2)=length(coordinates)+length(epiuper)+i-1;
end

wallvepiuper(length(wallvepiuper),2)=(length(coordinates)+2*(length(epiuper))
);

for i=1:length(epiuper)-1
     wallHepiuper(i,1)=length(coordinates)+i;
     wallHepiuper(i,2)=epiuper(i+1);
     for j=1:length(walls)
         if ismember(walls(j,:),[epiuper(i),epiuper(i+1)])
             walls(j,2)=length(coordinates)+i;
         end
     end
```

```
end

for i=length(epiuper):length(coordepiuper2)-1
     wallHepiuper(i,1)=length(coordinates)+i;
     wallHepiuper(i,2)=length(coordinates)+i+1;
end

epiuperId{1}=[epiuper(1),length(coordinates)+1,length(coordinates)+length(epi
uper)+1,length(coordinates)+length(epiuper),epiuper(1)];

for i=2:length(epiuper)-1
   epiuperId{i}=[length(coordinates)+i-
1,epiuper(i),length(coordinates)+i,length(coordinates)+length(epiuper)+i,...
   length(coordinates)+length(epiuper)+i-1,length(coordinates)+i];
end


epiuperId{length(epiuperId)+1}=[length(coordinates)+i,epiuper(length(epiuper)
),length(coordinates)+length(coordepiuper2),...
 length(coordinates)+length(coordepiuper2)-1,length(coordinates)+i];

abc1=abs(coordinates(:,1))<0.001;
abc2=abs(coordinates(:,2))<0.001;

coordinates(abc1,1)=0;
coordinates(abc2,2)=0;

 for i=1:length(V0)
     abv1=abs(V0{i}(:,1))<0.001;
      abv2=abs(V0{i}(:,2))<0.001;
      V0{i}(abv1,1)=0;
       V0{i}(abv2,2)=0;

       [~, ia, ib] = intersect(V0{i}(1:length(V0{i})-1,:),
coordinates(epiuper,:),'rows');


    if length(ia)==2 && (min(ia)==1 && max(ia)~=length(V0{i})-1)

V0{i}=[V0{i}(min(ia),:);coordepiuper2(min(ib),:);V0{i}(max(ia):length(V0{i}),
:)];
    elseif length(ia)==2 && (min(ia)==1&&max(ia)==length(V0{i})-1)
        V0{i}=[V0{i}(1:length(V0{i})-
1,:);coordepiuper2(min(ib),:);V0{i}(length(V0{i}),:)];
    elseif length(ia)==2 && min(ia)~=1

V0{i}=[V0{i}(min(ia),:);coordepiuper2(min(ib),:);V0{i}(max(ia):length(V0{i})-
1,:);V0{i}(1:min(ia),:)];
    end

 end

epiuper2=[];
```

```
for i=length(epiuper):length(coordepiuper2)
    if (abs(coordepiuper2(i,2)-max(coordepiuper2(:,2))))<0.01

        epiuper2=[epiuper2,length(coordinates)+i];
    end
end

for i=length(epiuper):length(coordepiuper2)-1
    coordepiuper22(i-
length(epiuper)+1,:)=[0.5*(coordepiuper2(i,1)+coordepiuper2(i+1,1)),
coordepiuper2(i,2)];
end

coordepiuper22(length(coordepiuper22)+1,:)=[coordepiuper2(length(epiuper),1),
coordepiuper2(length(epiuper),2)+ab_epi];

for i=1:length(epiuper2)-1
   coordepiuper22(length(coordepiuper22)+1,:)=[coordepiuper22(i,1),
coordepiuper22(i,2)+ab_epi];
end

coordepiuper22(length(coordepiuper22)+1,:)=[coordepiuper2(length(coordepiuper
2),1),coordepiuper2(length(coordepiuper2),2)+ab_epi];
wallvepiuper2(1,1)=epiuper2(1);
wallvepiuper2(1,2)=length(coordinates)+length(coordepiuper2)+length(epiuper2)
;

for i=1:length(epiuper2)-1
     wallvepiuper2(i+1,1)=length(coordinates)+2*(length(epiuper2)-1)+i;
     wallvepiuper2(i+1,2)=length(coordinates)+2*(length(epiuper2)-
1)+i+length(epiuper2);
end

 wallvepiuper2(length(wallvepiuper2)+1,1)=epiuper2(length(epiuper2));

wallvepiuper2(length(wallvepiuper2),2)=length(coordinates)+length(coordepiupe
r2)+length(coordepiuper22);

for i=1:length(epiuper2)-1
     wallHepiuper2(i,1)=length(coordinates)+length(coordepiuper2)+i;
     wallHepiuper2(i,2)=epiuper2(i+1);

   for j=1:length(wallHepiuper)
         if ismember(wallHepiuper(j,:),[epiuper2(i),epiuper2(i+1)])
             wallHepiuper(j,2)=length(coordinates)+length(coordepiuper2)+i;
         end
   end
 end

for i=length(epiuper2):length(coordepiuper22)-1
     wallHepiuper2(i,1)=length(coordinates)+length(coordepiuper2)+i;
     wallHepiuper2(i,2)=length(coordinates)+length(coordepiuper2)+i+1;
end
```

```
epiuperId2{1}=[epiuper2(1),length(coordinates)+length(coordepiuper2)+1,length
(coordinates)+length(coordepiuper2)+length(epiuper2)+1,...
length(coordinates)+length(coordepiuper2)+length(epiuper2),epiuper2(1)];

for i=2:length(epiuper2)-1
      epiuperId2{i}=[length(coordinates)+length(coordepiuper2)+i-
1,epiuper2(i),length(coordinates)+length(coordepiuper2)+i,...

length(coordinates)+length(coordepiuper2)+length(epiuper2)+i,length(coordinat
es)+length(coordepiuper2)+length(epiuper2)+i-1,...
      length(coordinates)+length(coordepiuper2)+i-1];
end



epiuperId2{length(epiuperId)+1}=[length(coordinates)+length(coordepiuper2)+i,
epiuper2(length(epiuper2)),length(coordinates)+length(coordepiuper2)+length(c
oordepiuper22),...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)-
1,length(coordinates)+length(coordepiuper2)+i];

for i=1:length(epiuperId)
    epiupercellveretex=[];
    epiupercellveretex=epiuperId{i};
    [c]=
find(ismember(epiupercellveretex(1,1:length(epiupercellveretex(1,:))-
1),epiuper2));

epiuperId{i}=[epiupercellveretex(1,1:c(1)),length(coordinates)+length(coordep
iuper2)+i,epiupercellveretex(1,c(2)),epiupercellveretex(1,1)];
 end

epilower=[];

for i=1:length(coordinates)
    if (abs(coordinates(i,2)-min(coordinates(:,2))))<0.01
        epilower=[epilower,i];
    end
end

for i=1:length(epilower)-1

coordepilower2(i,:)=[0.5*(coordinates(epilower(i),1)+coordinates(epilower(i+1
),1)), coordinates(epilower(i),2)];
end

coordepilower2(length(coordepilower2)+1,:)=[coordinates(epilower(1),1),
coordinates(epilower(1),2)-ab_epi];

for i=1:length(epilower)-1
   coordepilower2(length(coordepilower2)+1,:)=[coordepilower2(i,1),
coordepilower2(i,2)-ab_epi];
end
```

```
coordepilower2(length(coordepilower2)+1,:)=[coordinates(epilower(length(epilo
wer)),1),coordinates(epilower(length(epilower)),2)-ab_epi];
wallvepilower(1,1)=epilower(1);
for i=1:length(epilower)-1

wallvepilower(length(wallvepilower)+1,1)=length(coordinates)+length(coordepiu
per2)+length(coordepiuper22)+i;
end

wallvepilower(length(wallvepilower)+1,1)=epilower(length(epilower));
wallvepilower(1,2)=length(coordinates)+length(coordepiuper2)+length(coordepiu
per22)+length(epilower);

for i=2:length(epilower)

wallvepilower(i,2)=length(coordinates)+length(coordepiuper2)+length(coordepiu
per22)+length(epilower)+i-1;
end
wallvepilower(length(wallvepilower),2)=length(coordinates)+length(coordepiupe
r2)+length(coordepiuper22)+2*(length(epilower));

for i=1:length(epilower)-1

wallHepilower(i,1)=length(coordinates)+length(coordepiuper2)+length(coordepiu
per22)+i;
     wallHepilower(i,2)=epilower(i+1);
     for j=1:length(walls)
         if ismember(walls(j,:),[epilower(i),epilower(i+1)])

walls(j,2)=length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+i
;
         end
     end

end

for i=length(epilower):length(coordepilower2)-1

wallHepilower(i,1)=length(coordinates)+length(coordepiuper2)+length(coordepiu
per22)+i;

wallHepilower(i,2)=length(coordinates)+length(coordepiuper2)+length(coordepiu
per22)+i+1;
end
epilowerId{1}=[epilower(1),length(coordinates)+length(coordepiuper2)+length(c
oordepiuper22)+length(epilower),...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+length(epilo
wer)+1,length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+1,...
epilower(1)];

for i=2:length(epilower)-1

epilowerId{i}=[length(coordinates)+length(coordepiuper2)+length(coordepiuper2
2)+i-
```

```
1,length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+length(epi
lower)+i-1,...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+length(epilo
wer)+i,...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+i,epilower(i
)...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+i-1];
end

epilowerId{length(epilowerId)+1}=[length(coordinates)+length(coordepiuper2)+l
ength(coordepiuper22)+i,...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+length(epilo
wer)+i,...
length(coordinates)+length(coordepiuper2)+length(coordepiuper22)+length(epilo
wer)+i+1,...
epilower(length(epilower)),length(coordinates)+length(coordepiuper2)+length(c
oordepiuper22)+i];
abc1=abs(coordinates(:,1))<0.001;
abc2=abs(coordinates(:,2))<0.001;
coordinates(abc1,1)=0;
coordinates(abc2,2)=0;

 for i=1:length(V0)
     abv1=abs(V0{i}(:,1))<0.001;
     abv2=abs(V0{i}(:,2))<0.001;
     V0{i}(abv1,1)=0;
     V0{i}(abv2,2)=0;
     [~, ia, ib] = intersect(V0{i}(1:length(V0{i})-1,:),
coordinates(epilower,:),'rows');

    if length(ia)==2 && (min(ia)==1 && max(ia)~=length(V0{i})-1)

V0{i}=[V0{i}(min(ia),:);coordepilower2(min(ib),:);V0{i}(max(ia):length(V0{i})
,:)];
    elseif length(ia)==2 & (min(ia)==1& max(ia)==length(V0{i})-1)
    V0{i}=[V0{i}(1:length(V0{i})-
1,:);coordepilower2(min(ib),:);V0{i}(length(V0{i}),:)];
    elseif length(ia)==2 & min(ia)~=1

V0{i}=[V0{i}(min(ia),:);coordepilower2(min(ib),:);V0{i}(max(ia):length(V0{i})
-1,:);V0{i}(1:min(ia),:)];
    end
end

for i=1:length(epiuper2)
      coordepiuper222(i,:)=[0.5*(coordepiuper22(length(coordepiuper22)-
length(epiuper2)+i-1,1)+coordepiuper22(length(coordepiuper22)-
length(epiuper2)+i,1)),...
      coordepiuper22(length(coordepiuper22),2)];
      epiuperId2{i}=[epiuperId2{i}(1:length(epiuperId2{i})-
2),length(coordinates)+length(coordepiuper2)+length(coordepilower2)+length(co
ordepiuper22)+i,...
      epiuperId2{i}(length(epiuperId2{i})-
1),epiuperId2{i}(length(epiuperId2{i}))];
```

```
abw1=wallHepiuper2(:,1)==length(coordinates)+length(coordepiuper2)+length(coo
rdepiuper22)-length(epiuper2)+i-1&...

wallHepiuper2(:,2)==length(coordinates)+length(coordepiuper2)+length(coordepi
uper22)-length(epiuper2)+i;

wallHepiuper2(abw1,2)=length(coordinates)+length(coordepiuper2)+length(coorde
pilower2)+length(coordepiuper22)+i;

wallHepiuper2(length(wallHepiuper2)+1,:)=[length(coordinates)+length(coordepi
uper2)+length(coordepilower2)+length(coordepiuper22)+i,...
      length(coordinates)+length(coordepiuper2)+length(coordepiuper22)-
length(epiuper2)+i];
end

for i=1:length(epilower)
      coordepilower222(i,:)=[0.5*(coordepilower2(length(coordepilower2)-
length(epilower)+i-1,1)+coordepilower2(length(coordepilower2)-
length(epilower)+i,1)),...
      coordepilower2(length(coordepilower2),2)];

epilowerId{i}=[epilowerId{i}(1:2),length(coordinates)+length(coordepiuper2)+l
ength(coordepilower2)+length(coordepiuper22)+length(coordepiuper222)+i,...
      epilowerId{i}(3:length(epilowerId{i})-1),epilowerId{i}(1)];

abw2=wallHepilower(:,1)==length(coordinates)+length(coordepiuper2)+length(coo
rdepiuper22)+length(coordepilower2)-length(epilower)+i-1&...

wallHepilower(:,2)==length(coordinates)+length(coordepiuper2)+length(coordepi
uper22)+length(coordepilower2)-length(epilower)+i;

wallHepilower(abw2,2)=length(coordinates)+length(coordepiuper2)+length(coorde
piuper22)+length(coordepilower2)+length(coordepiuper222)+i;

wallHepilower(length(wallHepilower)+1,:)=[epilowerId{i}(3),epilowerId{i}(4)];
end

epiId=[];
counter=0;
for i=1:length(epiuperId)
    counter=counter+1;
    epiId{counter}=epiuperId{i};
end

adf1=epiuperId2{1};
adf2=epiuperId2{2};
adf3=epiuperId2{length(epiuperId2)};
adf4=epiuperId2{length(epiuperId2)-1};
epiuperId2{2}=[adf1(1),adf1(length(adf1)-1),adf2(1:5),adf1(1)];
epiuperId2{length(epiuperId2)-1}=[adf3(2:3),adf4(1,[1,2,3,4,6]),adf4(1)];
for i=2:length(epiuperId2)-1
    counter=counter+1;
    epiId{counter}=epiuperId2{i};
end
```

```
for i=1:length(epilowerId)
    counter=counter+1;
    epiId{counter}=epilowerId{i};
end


wallHepi=[wallHepiuper;wallHepiuper2;wallHepilower];
wallvepi=[wallvepiuper;wallvepiuper2;wallvepilower];
coordepi2=[coordepiuper2;coordepiuper22;coordepilower2;coordepiuper222;coorde
pilower222];
pali_lim=[min(coordepiuper2(:,2)),max(coordepiuper2(:,2))];
epiup_lim=[min(coordepiuper22(:,2)),max(coordepiuper22(:,2))];
epilow_lim=[min(coordepilower2(:,2)),max(coordepilower2(:,2))];

%%updating the coordinates and the walls after making the epidermal cells

coordinates=[coordinates;coordepi2];
coordinates(epiuperId2{2}(7),:)=0.5*(coordinates(epiuperId2{1}(5),:)+coordina
tes(epiuperId2{2}(6),:));
coordinates(epiuperId2{length(epiuperId2)-
1}(6),:)=0.5*(coordinates(epiuperId2{length(epiuperId2)-
1}(7),:)+coordinates(epiuperId2{length(epiuperId2)}(3),:));
walls=[walls;wallvepi;wallHepi];
numberofcells=length(V0)+length(epiId)+1; % with out epidermal cells

% figure(7)
% hold on
% axis equal
%
% for i=1:length(epiuperId2)
%     plot(coordinates(epiuperId2{i},1),coordinates(epiuperId2{i},2),'r')
% end
%   for i=1:length(coordinates)
%
text(coordinates(i,1),coordinates(i,2),[strcat(num2str(i))],'FontSize',10)
% end

%% cellvertexId specification before making the corner pores,
abb=abs(coordinates(:,1))<0.00001;
abby=abs(coordinates(:,2))<0.00001;
coordinates(abb,1)=0;
coordinates(abby,2)=0;
for i=1:length(V0)
    abbv=abs(V0{i}(:,1))<0.00001;
    V0{i}(abbv,1)=0;
    abbvy=abs(V0{i}(:,2))<0.00001;
    V0{i}(abbvy,2)=0;

    for j=1:length(V0{i})-1
    [~, ~, ib] = intersect(V0{i}(j,:), coordinates(:,:),'rows');
    CellvertexId{i}(j)=ib;
    end

    CellvertexId{i}(length(CellvertexId{i})+1)=CellvertexId{i}(1);
end
```

```
  CellvertexId=[CellvertexId,epiId];
  PID1=[];

  %%
PID=[];

for i=1:(length(CellvertexId)-length(epiId))
    if rem(i,3)==0
        PID=[PID,i];
    end
end

for i=1:length(CellvertexId)
    ab_mean(i,:)=mean(coordinates(CellvertexId{i},:));
end

PP=[];
for i=1:length(PID)
    abf=mean(coordinates(CellvertexId{PID(i)},:));
    if abs(abf(1)-min(ab_mean(:,1)))>20&&abs(abf(1)-max(ab_mean(:,1)))>20
        PP=[PP;PID(i)];
    end
end

PID=[PP;PID1'];
CID1=[];
counter=0;
for i=1:length(CellvertexId)
    c0=mean(coordinates(CellvertexId{i},:));
    if ~ismember(i,PID)
    counter=counter+1;
    CID1{counter}=CellvertexId{i};
    end
end

poreId=[poreId;PID];
PP=[];



CellvertexId=[];
CellvertexId=CID1;
numberofcells=length(CID1)+1;
%%
V0=[];

for i=1:length(CellvertexId)

    if CellvertexId{i}(1)~=CellvertexId{i}(length(CellvertexId{i}))
        CellvertexId{i}=[CellvertexId{i},CellvertexId{i}(1)];
    end
```

```
    V0{i}=coordinates(CellvertexId{i}(1:length(CellvertexId{i})),:);

end

cordinates=[];

counter=0;
for k=1:length(V0)

    for h=1:length(V0{k})
        counter=counter+1;
        cordinates(counter,1)=(V0{k}(h,1));
        cordinates(counter,2)=(V0{k}(h,2));
    end

end


[coordinates,~,~]=unique(cordinates,'rows');

CellvertexId=[];
for i=1:length(V0)
    for j=1:length(V0{i})
    [~, ~, ib] = intersect(V0{i}(j,:), coordinates(:,:),'rows');
    CellvertexId{i}(j)=ib;
    end
end
% figure(8)
% for i=1:length(CellvertexId)
%     plot(coordinates(CellvertexId{i},1),coordinates(CellvertexId{i},2))
%     hold on
% end
% for i=1:length(coordinates)
% text(coordinates(i,1),coordinates(i,2),[strcat(num2str(i))],'FontSize',10)
% end


%% making wall vertices

Walls=[];
counter=0;
for i=1:length(CellvertexId)
for j=1:length(CellvertexId{i})
counter=counter+1;
if j<length(CellvertexId{i})
Walls(counter,1)=CellvertexId{i}(j);
Walls(counter,2)=CellvertexId{i}(j+1);
end
end
end
for i=1:length(Walls)
for j=2:length(Walls)
    if Walls(j,1)==Walls(i,2)&&Walls(j,2)==Walls(i,1)
        Walls(j,:)=0;
    end
```

```
end

end

[~,~,WAll1]=find(Walls(:,1));
[~,~,WAll2]=find(Walls(:,2));
Wall=[WAll1,WAll2];
Walls= unique(Wall,'rows');
for i=1:length(Walls)
    if ((abs((coordinates(Walls(i,2),1)-
coordinates(Walls(i,1),1)))>abs((coordinates(Walls(i,2),2)-
coordinates(Walls(i,1),2))))&&(coordinates(Walls(i,1),1))>coordinates(Walls(i
,2),1))...
            ||((abs((coordinates(Walls(i,2),2)-
coordinates(Walls(i,1),2)))>abs((coordinates(Walls(i,2),1)-
coordinates(Walls(i,1),1))))&&(coordinates(Walls(i,1),2))>coordinates(Walls(i
,2),2))
        wall(i,1)=Walls(i,2);
        wall(i,2)=Walls(i,1);
    else
        wall(i,1)=Walls(i,1);
        wall(i,2)=Walls(i,2);
    end
end

walls=wall;

  %% wall2cell specification final
wall2cell=zeros(length(walls),2);
for i=1:length(walls)
    for j=1:length(CellvertexId)

QQ1=intersect(coordinates(walls(i,1),:),coordinates(CellvertexId{j}(:),:),'ro
ws');

QQ2=intersect(coordinates(walls(i,2),:),coordinates(CellvertexId{j}(:),:),'ro
ws');
    x0(j)=mean(coordinates(CellvertexId{j}(:),1));
    y0(j)=mean(coordinates(CellvertexId{j}(:),2));
    if (~isempty(QQ1)&&~isempty(QQ2))
       Wall_vector= [coordinates(walls(i,2),:)-coordinates(walls(i,1),:),0];
       CellPos_vector=[[x0(j),y0(j)]-coordinates(walls(i,1),:),0];
       angle=cross( Wall_vector,CellPos_vector);
       if angle(3)>0
           wall2cell(i,1)=j;
       elseif angle(3)<0
           wall2cell(i,2)=j;
       end
    end
    end

end


for i=1:length(wall2cell)
```

```
    if wall2cell(i,1)==0
        wall2cell(i,1)=numberofcells;
    end
    if wall2cell(i,2)==0
        wall2cell(i,2)=numberofcells;
    end
end

Coordinates=coordinates;

oldwall=length(walls);
  %% wall2cell specification
  wall2cell=numberofcells*ones(length(walls),2);
for i=1:length(walls)
    ind=[];
    for j=1:length(CellvertexId)
        if ismember(walls(i,:),CellvertexId{j})
            ind=[ind,j];
        end
    end
    wall2cell(i,1:length(ind))=ind;
end

 ba=find( wall2cell(1:(length(walls)),1)==numberofcells|
wall2cell(1:(length(walls)),2)==numberofcells);
   Wall2cell=wall2cell(ba,:);
   Wallbnd=walls(ba,:);
   stoneId=[];
  %% identfying vertices which are not vertices of external walls and walls
to adjacent to pores
  CORDint= find(~ismember(1:length(coordinates),Wallbnd));



  %%
for i=1:length( CellvertexId)

Cellarea(i)=polyarea(coordinates(CellvertexId{i}(:),1),coordinates(Cellvertex
Id{i}(:),2));
end
%% making the corner pores
CellvertexID=CellvertexId;%CellvertexID is used to update the cell vertices
while making the corner pores

for i=1:length(CellvertexId)
    CellvertexId{i}=CellvertexId{i}(1:length(CellvertexId{i})-1);
end

randn('state',2)
coordinate=coordinates;%coordinate is used to update the coordintes of
vertices while making the corner pores
wallp=walls;
cordpore=1:2:length(coordinates);

HID=[];
for i=1:length(hypoId)
```

```
    HID=[HID,CellvertexId{hypoId(i)}];
end
HID=unique(HID);

cordpore=cordpore(~ismember(cordpore,HID));
stonevertex=[];

for i=1:length(stoneId)
    stonevertex=[stonevertex,CellvertexId{stoneId(i)}];
end

CELID=CellvertexId;

for i=1:length(CellvertexId)
    CellvertexId{i}=unique(CellvertexId{i});
end

for k=1:length(CellvertexId)
    CellvertexId{k}=unique(CellvertexId{k});
    centroid{k}=mean(coordinates(CellvertexId{k}(:),:));
    thetarad{k}=atan2((coordinates(CellvertexId{k}(:),2)-
centroid{k}(2)),(coordinates(CellvertexId{k}(:),1)-centroid{k}(1)));

    thetaneg=find(thetarad{k}<0);
    thetarad{k}(thetaneg(:))=thetarad{k}(thetaneg(:))+2*pi;
    [~,I] = sort(thetarad{k});
    CelvertexId{k}=CellvertexId{k};
    CellvertexId{k}=CelvertexId{k}(I);
    CellvertexId{k}=CellvertexId{k}(:)';

    CellvertexId{k}=[CellvertexId{k}(:);CellvertexId{k}(1)];
    xx=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),1);
    yy=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),2);

    x2 = [xx(2:length(xx),1);xx(1,1)];
    y2 = [yy(2:length(yy),1);yy(1,1)];
    %% calculating second moments of inertia
    a = (1/2*sum(x2.*yy-xx.*y2));

    x0(k) = 1/6*sum((x2.*yy-xx.*y2).*(xx+x2))/a;
    y0(k) = 1/6*sum((x2.*yy-xx.*y2).*(yy+y2))/a;
 end

  %% wall2cell specification final
wall2cell=zeros(length(walls),2);
for i=1:length(walls)
    for j=1:length(CellvertexId)

QQ1=intersect(coordinates(walls(i,1),:),coordinates(CellvertexId{j}(:),:),'ro
ws');

QQ2=intersect(coordinates(walls(i,2),:),coordinates(CellvertexId{j}(:),:),'ro
ws');
    if (~isempty(QQ1)&&~isempty(QQ2))
       Wall_vector= [coordinates(walls(i,2),:)-coordinates(walls(i,1),:),0];
```

```
       CellPos_vector=[[x0(j),y0(j)]-coordinates(walls(i,1),:),0];
       angle=cross( Wall_vector,CellPos_vector);
       if angle(3)>0
           wall2cell(i,1)=j;
       elseif angle(3)<0
           wall2cell(i,2)=j;
       end
    end
    end

end



for i=1:length(wall2cell)
    if wall2cell(i,1)==0
    wall2cell(i,1)=numberofcells;
    end
    if wall2cell(i,2)==0
    wall2cell(i,2)=numberofcells;
    end
end
%%
ba=find( wall2cell(:,1)==numberofcells| wall2cell(:,2)==numberofcells);
Wall2cell=wall2cell(ba,:);
Wallbnd=walls(ba,:);
Wallbndf=[];

for i=1:length(Wallbnd)
 [in ,
~]=inpolygon(coordinates(Wallbnd(i,:),1),coordinates(Wallbnd(i,:),2),xk(k1),y
k(k1))  ;
    if ~isempty(find(in==0, 1))
    Wallbndf=[Wallbndf,i];
    end
end

Wallbnd=Wallbnd(Wallbndf,:);
dimension=2;

paliId=[];
epiId=[];
for i=1:length(CellvertexId)
      if mean(coordinates(CellvertexId{i},2))>pali_lim(1)&&
mean(coordinates(CellvertexId{i},2))<pali_lim(2)
          paliId=[paliId;i];
      end
      if (mean(coordinates(CellvertexId{i},2))>epiup_lim(1)&&
mean(coordinates(CellvertexId{i},2))<epiup_lim(2))...
              ||(mean(coordinates(CellvertexId{i},2))>epilow_lim(1)&&
mean(coordinates(CellvertexId{i},2))<epilow_lim(2))
          epiId=[epiId;i];
      end
end

%%
```

```
V0=[];

for i=1:length(CellvertexId)

    if CellvertexId{i}(1)~=CellvertexId{i}(length(CellvertexId{i}))
        CellvertexId{i}=[CellvertexId{i};CellvertexId{i}(1)];
    end

V0{i}=coordinates(CellvertexId{i}(1:length(CellvertexId{i})),:);

end
cordinates=[];

counter=0;
for k=1:length(V0)
    for h=1:length(V0{k})
         counter=counter+1;
         cordinates(counter,1)=(V0{k}(h,1));
         cordinates(counter,2)=(V0{k}(h,2));
    end
end

[coordinates,~,~]=unique(cordinates,'rows');
CellvertexId=[];
for i=1:length(V0)
    for j=1:length(V0{i})
    [~, ~, ib] = intersect(V0{i}(j,:), coordinates(:,:),'rows');
    CellvertexId{i}(j)=ib;
    end
end

%% making wall vertices

Walls=[];
counter=0;
for i=1:length(CellvertexId)
    for j=1:length(CellvertexId{i})
        counter=counter+1;
        if j<length(CellvertexId{i})
            Walls(counter,1)=CellvertexId{i}(j);
            Walls(counter,2)=CellvertexId{i}(j+1);
        end
    end
end

for i=1:length(Walls)
    for j=2:length(Walls)
        if Walls(j,1)==Walls(i,2)&&Walls(j,2)==Walls(i,1)
        Walls(j,:)=0;
        end
    end
end



[~,~,WAll1]=find(Walls(:,1));
```

```
[~,~,WAll2]=find(Walls(:,2));
Wall=[WAll1,WAll2];
Walls= unique(Wall,'rows');
for i=1:length(Walls)
    if ((abs((coordinates(Walls(i,2),1)-
coordinates(Walls(i,1),1)))>abs((coordinates(Walls(i,2),2)-
coordinates(Walls(i,1),2))))&&(coordinates(Walls(i,1),1))>coordinates(Walls(i
,2),1))...
            ||((abs((coordinates(Walls(i,2),2)-
coordinates(Walls(i,1),2)))>abs((coordinates(Walls(i,2),1)-
coordinates(Walls(i,1),1))))&&(coordinates(Walls(i,1),2))>coordinates(Walls(i
,2),2))
        wall(i,1)=Walls(i,2);
        wall(i,2)=Walls(i,1);
    else
        wall(i,1)=Walls(i,1);
        wall(i,2)=Walls(i,2);
    end
end

walls=wall;

  %% wall2cell specification final
wall2cell=zeros(length(walls),2);
for i=1:length(walls)
    for j=1:length(CellvertexId)

QQ1=intersect(coordinates(walls(i,1),:),coordinates(CellvertexId{j}(:),:),'ro
ws');

QQ2=intersect(coordinates(walls(i,2),:),coordinates(CellvertexId{j}(:),:),'ro
ws');
    if (~isempty(QQ1)&&~isempty(QQ2))
       Wall_vector= [coordinates(walls(i,2),:)-coordinates(walls(i,1),:),0];
       CellPos_vector=[[x0(j),y0(j)]-coordinates(walls(i,1),:),0];
       angle=cross( Wall_vector,CellPos_vector);
       if angle(3)>0
           wall2cell(i,1)=j;
       elseif angle(3)<0
           wall2cell(i,2)=j;
       end
    end
    end
end



for i=1:length(wall2cell)
   if wall2cell(i,1)==0
     wall2cell(i,1)=numberofcells;
   end
    if wall2cell(i,2)==0
     wall2cell(i,2)=numberofcells;
    end
end

wal=walls;
```

```
w2cell=wall2cell;
cord=coordinates;

walls=wal;
wall2cell=w2cell;
coordinates=cord;
l_n=[];
l=[];

for i=1:length(walls)
  l_n(i)=norm(coordinates(walls(i,1),:)-coordinates(walls(i,2),:),2);
  l(i)=norm(coordinates(walls(i,1),:)-coordinates(walls(i,2),:),2);
end

l_n=l_n';
l=l';



h=figure(34);
hold on
for i=1:length(CellvertexId)
    if ismember(i,[epiId])
plot(coordinates(CellvertexId{i}(:),1),coordinates(CellvertexId{i}(:),2),'r')
;

%text(mean(coordinates(CellvertexId{i}(:),1)),mean(coordinates(CellvertexId{i
}(:),2)),[strcat('c_',num2str(i))],'FontSize',10)
    elseif ismember(i,[paliId])

plot(coordinates(CellvertexId{i}(:),1),coordinates(CellvertexId{i}(:),2),'m')
;


    else

patch(coordinates(CellvertexId{i}(:),1),coordinates(CellvertexId{i}(:),2),'g'
);

    end
hold on
axis('equal')

end
```

Nearestpoint

```
function [IND, D] = nearestpoint(x,y,m) ;
% NEARESTPOINT - find the nearest value in another vector
%
%   IND = NEARESTPOINT(X,Y) finds the value in Y which is the closest to
%   each value in X, so that abs(Xi-Yk) => abs(Xi-Yj) when k is not equal to
j.
%   IND contains the indices of each of these points.
%   Example:
%      NEARESTPOINT([1 4 12],[0 3]) -> [1 2 2]
%
```

```
%   [IND,D] = ... also returns the absolute distances in D,
%   that is D == abs(X - Y(IND))
%
%   NEARESTPOINT(X, Y, M) specifies the operation mode M:
%   'nearest' : default, same as above
%   'previous': find the points in Y that are closest, but preceeds a point
in X
%               NEARESTPOINT([0 4 3 12],[0 3],'previous') -> [NaN 2 1 2]
%   'next'    : find the points in Y that are closets, but follow a point in
X
%               NEARESTPOINT([1 4 3 12],[0 3],'next') -> [2 NaN 2 NaN]
%
%   If there is no previous or next point in Y for a point X(i), IND(i)
%   will be NaN.
%
%   X and Y may be unsorted.
%
%   This function is quite fast, and especially suited for large arrays with
%   time data. For instance, X and Y may be the times of two separate events,
%   like simple and complex spike data of a neurophysiological study.
%
%


%   Nearestpoint('test') will run a test to show it's effective ness for
%   large data sets

% Created       : august 2004
% Author        : Jos van der Geest
% Email         : matlab@jasen.nl
% Modifications :
%  aug 25, 2004 - corrected to work with unsorted input values
%  nov 02, 2005 -
%  apr 28, 2006 - fixed problem with previous points

if nargin==1 & strcmp(x,'test'),

    testnearestpoint ;
    return
end

error(nargchk(2,3,nargin)) ;

if nargin==2,
    m = 'nearest' ;
else
    if ~ischar(m),
        error('Mode argument should be a string (either ''nearest'',
''previous'', or ''next'')') ;
    end
end

if ~isa(x,'double') | ~isa(y,'double'),
    error('X and Y should be double matrices') ;
end
```

```
% sort the input vectors
sz = size(x) ;
[x, xi] = sort(x(:)) ;
[dum, xi] = sort(xi) ; % for rearranging the output back to X
nx = numel(x) ;
cx = zeros(nx,1) ;
qx = isnan(x) ; % for replacing NaNs with NaNs later on

[y,yi] = sort(y(:)) ;
ny = length(y) ;
cy = ones(ny,1) ;

xy = [x ; y] ;

[xy, xyi] = sort(xy) ;
cxy = [cx ; cy] ;
cxy = cxy(xyi) ; % cxy(i) = 0 -> xy(i) belongs to X, = 1 -> xy(i) belongs to
Y
ii = cumsum(cxy) ;
ii = ii(cxy==0).' ; % ii should be a row vector

% reduce overhead
clear cxy xy xyi ;

switch lower(m),
    case {'nearest','near','absolute'}
        % the indices of the nearest point
        ii = [ii ; ii+1] ;
        ii(ii==0) = 1 ;
        ii(ii>ny) = ny ;
        yy = y(ii) ;
        dy = abs(repmat(x.',2,1) - yy) ;
        [dum, ai] = min(dy) ;
        IND = ii(sub2ind(size(ii),ai,1:nx)) ;
    case {'previous','prev','before'}
        % the indices of the previous points
        ii(ii < 1) = NaN ;
        IND = ii ;
    case {'next','after'}
        % the indices of the next points
        ii = ii + 1 ;
        ii(ii>ny) = NaN ;
        IND = ii ;
    otherwise
        error(sprintf('Unknown method "%s"',m)) ;
end

IND(qx) = NaN ; % put NaNs back in
% IND = IND(:) ; % solves a problem for x = 1-by-n and y = 1-by-1

if nargout==2,
    % also return distance if requested;
    D = repmat(NaN,1,nx) ;
    q = ~isnan(IND) ;
    D(q) = abs(x(q) - y(IND(q))) ;
```

```
    D = reshape(D(xi),sz) ;
end

% reshape and sort to match input X
IND = reshape(IND(xi),sz) ;

% because Y was sorted, we have to unsort the indices
q = ~isnan(IND) ;
IND(q) = yi(IND(q)) ;

% END OF FUNCTION

function testnearestpoint
disp('TEST for nearestpoint, please wait ... ') ;
M = 13 ;
tim = repmat(NaN,M,3) ;
tim(8:M,1) = 2.^[8:M].' ;
figure('Name','NearestPointTest','doublebuffer','on') ;
h = plot(tim(:,1),tim(:,2),'bo-',tim(:,1),tim(:,3),'rs-') ;
xlabel('N') ;
ylabel('Time (seconds)') ;
title('Test for Nearestpoint function ... please wait ...') ;
set(gca,'xlim',[0 max(tim(:,1))+10]) ;
for j=8:M,
    N = 2.^j ;
    A = rand(N,1) ; B = rand(N,1) ;
    tic ;
    D1 = zeros(N,1) ;
    I1 = zeros(N,1) ;
    for i=1:N,
        [D1(i), I1(i)] = min(abs(A(i)-B)) ;
    end
    tim(j,2) = toc ;
    pause(0.1) ;
    tic ;
    [I2,D2] = nearestpoint(A,B) ;
    tim(j,3) = toc ;
    % isequal(I1,I2)
    set(h(1),'Ydata',tim(:,2)) ;
    set(h(2),'Ydata',tim(:,3)) ;
    drawnow ;
end
title('Test for Nearestpoint function') ;
legend('Traditional for-loop','Nearestpoint',2) ;
```

Split

```
function
[coordinates,dimension,numberofcells,walls,wall2cell,l_n,Cellarea,CellvertexI
d,t,l_n_0,y,epiId,paliId,KK,geom,adhesive_wall,split_wall,split_turgor_1,spli
t_turgor_2,...

left_boundary,right_boundary,az_top,az_top_2,az_bottom,az_bottom_2,con_point_
pal,con_point_2_pal,con_point_3_pal,con_point_epi,con_point_2_epi,con_point_3
_epi,...
```

```
con_point_spongy,con_point_2_spongy,con_point_3_spongy,con_point_pal_spongy,c
on_point_2_pal_spongy,con_point_3_pal_spongy]=SPLIT(ap,ar,dr,l_n_max_2_l_n_0,
P_h_2_w,geom,string,string11);
cd(string)

load coordinates
load CellvertexId
load walls
load wall2cell
load l_n
load epiId
load paliId
load KK
load adhesive_wall
load split_wall
load split_turgor_1
load split_turgor_2
load left_boundary
load right_boundary
load az_top
load az_top_2
load az_bottom
load az_bottom_2

load con_point_pal
load con_point_2_pal
load con_point_3_pal

load con_point_epi
load con_point_2_epi
load con_point_3_epi

load con_point_spongy
load con_point_2_spongy
load con_point_3_spongy

load con_point_pal_spongy
load con_point_2_pal_spongy
load con_point_3_pal_spongy
cd(string11)

%% remove the comments if only a portion of the walls need to be separated
% SP=split_wall;
% split_wall=[];
% rand('state',5) % the state of rand allows generating various porosities
% split_wall=SP(unique(randi(length(SP),length(SP),1)),:);

ABD=[];
for i=2:2:length(paliId)
    ABD=[ABD,CellvertexId{paliId(i)}];
end
con_left=min(coordinates(:,1));
con_bottom=min(coordinates(:,2));
numberofcells=length(CellvertexId)+1;
```

```
dimension=2;
  wall2cell(split_turgor_1,2)=numberofcells+1;
  wall2cell(split_turgor_2,1)=numberofcells+1;
  wall2cell(split_turgor_1,1)=numberofcells+2;
  wall2cell(split_turgor_2,2)=numberofcells+2;
  for i=1:length(walls)
      if
ismember(numberofcells,wall2cell(i,:))&ismember(wall2cell(i,1),paliId)
          wall2cell(i,2)=numberofcells+1;
      elseif
ismember(numberofcells,wall2cell(i,:))&ismember(wall2cell(i,2),paliId)
          wall2cell(i,1)=numberofcells+1;
      end
  end

for i=1:length(CellvertexId)

Cellarea(i)=polyarea(coordinates(CellvertexId{i},1),coordinates(CellvertexId{
i},2));
end
Chloro_surf=[];
 for i=1:length(adhesive_wall)



           if ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_3_epi])&ismember(adhesive_wall(i,2),[
con_point_3_epi]))
               g=1
         K_ad(i)=0;
         Chloro_surf=[Chloro_surf;(adhesive_wall(i,:))'];
           elseif ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_2_epi])&ismember(adhesive_wall(i,2),[
con_point_2_epi]))
               K_ad(i)=50;

           elseif ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_3_pal])&ismember(adhesive_wall(i,2),[
con_point_3_pal]))
               K_ad(i)=0;
               g=2
               Chloro_surf=[Chloro_surf;(adhesive_wall(i,:))'];

          elseif ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_2_pal])&ismember(adhesive_wall(i,2),[
con_point_2_pal]))
             K_ad(i)=0;
             g=3
          Chloro_surf=[Chloro_surf;(adhesive_wall(i,:))'];
          elseif ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_pal([1:3:end],:)])&ismember(adhesive_
wall(i,2),[con_point_pal([1:2:end],:)]))
          K_ad(i)=0;
          g=4
          Chloro_surf=[Chloro_surf;(adhesive_wall(i,:))'];
```

```
elseif ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_3_pal_spongy])&ismember(adhesive_wall
(i,2),[con_point_3_pal_spongy]))
          K_ad(i)=0;
          g=5
          Chloro_surf=[Chloro_surf;(adhesive_wall(i,:))'];
           elseif ismember(adhesive_wall(i,:),[split_wall])&
(ismember(adhesive_wall(i,1),[con_point_2_pal_spongy])&ismember(adhesive_wall
(i,2),[con_point_2_pal_spongy]))
         K_ad(i)=100;
           else
          K_ad(i)=100;
           end
           if
ismember(adhesive_wall(i,:),[split_wall])&((ismember(adhesive_wall(i,1),ABD)&
~ismember(adhesive_wall(i,2),ABD))|(ismember(adhesive_wall(i,2),ABD)&~ismembe
r(adhesive_wall(i,1),ABD)))
          K_ad(i)=0;
          Chloro_surf=[Chloro_surf;(adhesive_wall(i,:))'];
           end

        end

%% assigning turgore pressure to the cells, spring constant to the walls  and
damping coefficient for the matrix

turgorpressure(numberofcells,1)=0;
turgorpressure(1:numberofcells-1,1)=1;
turgorpressure(numberofcells+1,1)=0.9;
turgorpressure(numberofcells+2,1)=0.9;

A0=mean(Cellarea); % average cell area of the initial cells
drag=dr; %drag coefficient
l_n_0=l_n;% initial resting length
l_nmax=l_n_max_2_l_n_0*l_n_0; %setting the maximum resting length of the
walls
%% initializing the variables
aml=[];
tau1=200000;
tau=tau1;
%%
velocity=zeros(length(coordinates),dimension);
tstart=0;
tfinal=2000;
at=10^(-ar);% absolute tolerance
rt=10^(-(ar+1));%relative tolerance
pr=1;% initialing turgore pressure
tc=100000;% time constant for torgore pressure to reach maximum
pmax=ap;
y0=[coordinates(:,1);coordinates(:,2);l_n;pr];% intial conditions for the ode
solver
options =
odeset('RelTol',rt,'AbsTol',at,'events',@events,'stats','on','MaxOrder',1,'BD
F','on');
poreId=[];% initialzing pore cells
%% beginning of the solution
```

```
for j=1:1,
BDc=[]; %initializing the boundary cells

ABD=1:length(coordinates);
stopc=ABD(find(~ismember(ABD,[left_boundary,right_boundary,az_top
,az_top_2,az_bottom,az_bottom_2])));
savetime=[[tstart:100:tfinal]];%,[1250:1000:10000],[11000:25000:1000000],[102
5000:250000:tfinal]];
[t,y]=ode15s(@f,[savetime],y0,options); % the ODE solver
end
%% defining the ODE function to be solved

function dydt=f(t,y)

   %% initializing the parameters at every time step
 b=[];
 force=zeros(length(coordinates),dimension);
 pf=zeros(length(coordinates),dimension);
 Tf=zeros(length(coordinates),dimension);

coordinates=[y(1:1*length(coordinates)),y(1*length(coordinates)+1:2*length(co
ordinates))];% new coordinates
 coordinates(left_boundary,1)=con_left;

%Calculate new Length and direction of walls

for i=1:length(walls)
    l(i)=norm(coordinates(walls(i,1),:)-coordinates(walls(i,2),:),2);
    direction(i,:)=(coordinates(walls(i,1),:)-
coordinates(walls(i,2),:))/l(i);
    normal_direction(i,1)=direction(i,2);
    normal_direction(i,2)=-direction(i,1);
    Degree1(i)= atan((coordinates(walls(i,2),2)-
coordinates(walls(i,1),2))/(coordinates(walls(i,2),1)-
coordinates(walls(i,1),1)));

    Fturgor=(turgorpressure(wall2cell(i,2))-
turgorpressure(wall2cell(i,1)))*l(i)*(normal_direction(i,:));

%% tension force
b(i)=0;
Ftension=KK(i)*(l(i)-l_n(i))*direction(i,:);

 %% net force

pf(walls(i,1),:)=pf(walls(i,1),:)+0.5*Fturgor;
pf(walls(i,2),:)=pf(walls(i,2),:)+0.5*Fturgor ;
Tf(walls(i,1),:)=Tf(walls(i,1),:)-Ftension;
Tf(walls(i,2),:)=Tf(walls(i,2),:)+Ftension;

kr(i)=0;

end
% %
for i=1:length(adhesive_wall)
```

```
    ll(i)=1+norm(coordinates(adhesive_wall(i,1),:)-
coordinates(adhesive_wall(i,2),:),2);
    Tf(adhesive_wall(i,1),:)=Tf(adhesive_wall(i,1),:)-
K_ad(i)*(coordinates(adhesive_wall(i,1),:)-
coordinates(adhesive_wall(i,2),:))/ll(i);

Tf(adhesive_wall(i,2),:)=Tf(adhesive_wall(i,2),:)+K_ad(i)*(coordinates(adhesi
ve_wall(i,1),:)-coordinates(adhesive_wall(i,2),:))/ll(i);
end

force=pf+Tf;
force(left_boundary,1)=0;
force([az_bottom],2)=0;
force(right_boundary,1)=0;
force([az_top,az_top_2],2)=0;

velocity(:,1)=force(:,1)/drag;
velocity(:,2)=force(:,2)/drag;
velocity(left_boundary,1)=mean(velocity(left_boundary,1));
velocity(right_boundary,1)=mean(velocity(right_boundary,1));
velocity(az_bottom_2,2)=mean(velocity(az_bottom_2,2));
velocity(az_top_2,2)=mean(velocity(az_top_2,2));
velocity(az_bottom,2)=mean(velocity(az_bottom,2));
velocity(az_top,2)=mean(velocity(az_top,2));

p=0;
dydt=[velocity(:,1);velocity(:,2);b(:);p];

%
end
 %% saving the time history of forces on vertices
function [value,isterminal,direction] = events(t,y)
   for i=1:length( CellvertexId),

Cellarea(i)=polyarea(coordinates(CellvertexId{i}(:),1),coordinates(Cellvertex
Id{i}(:),2));
    end

areacheck=(mean(Cellarea)-5*A0);
NV=norm(velocity(stopc,:));
value=mean(NV)-0.0001;
t1=t;

isterminal = ones(1,1);
direction =-ones(1,1);

end   % End nested function events
end
```

Visualization of outputs

The following set of codes are useful for plotting the generated geometries.

```
% Plotting geometries file; Author: Moges Retta, moges.retta@kuleuven.be
% This program plots the generated geometries and patchs them using the
```

```
% colors specified by r for red, g for green and m for magenta

load cord   %% coordinates of the vertexId for each of the tissues
load CellId %% contains the cellvertexId of each
load PID    %% information about which cell is a palisade in each tissue
load EID    %% information about which cell is a epidermal cell in each
tissue
load WALL   %% information about which vertices defines a wall
load W2C    %%

for i=1:length(CellId)  % For each geometry

 % Initialization
 coordinates=[];
 CellvertexId=[];
 epiId=[];
 paliId=[];

 % Assign values to initialized vectors
 coordinates=cord{i};
 CellvertexId=CellId{i};
 epiId=EID{i};
 paliId=PID{i};

 subplot(length(CellId),1,i)

 for j=1:length(CellvertexId)

        if ismember(j,epiId)

patch(coordinates(CellvertexId{j},1),coordinates(CellvertexId{j},2),'r','l
inewidth',0.01)

        elseif ismember(j,paliId)

patch(coordinates(CellvertexId{j}(:),1),coordinates(CellvertexId{j}(:),2),
'm','linewidth',1)
        else


patch(coordinates(CellvertexId{j},1),coordinates(CellvertexId{j},2),'g')
            hold on
        end
end

end
```

Calcuating anatomical properties

The following set of codes calculated area and aspect ratio of cells.

```
% Program for calculating the exposed length of palisade and spongy
% mesophyll, area and aspect ratio of the cells

clear all
```

```
currentFolder=pwd;
result_tot=[];
result_tot2=[];

for z=1:length(p_h_2_w)


load cord   %% coordinates of the vertexId for each of the tissues
load CellId %% contains the cellvertexId of each vertex
load PID    %% information about which cell is a palisade in each tissue
load EID    %% information about which cell is a epidermal cell in each
tissue
load WALL   %% information about which vertices defines a wall
load W2C    %% information about which wall belongs to a cell

palisade_wall_exposed_length=[];
spongy_wall_exposed_length=[];

result=[];  %% Matrix containing the results

for zz=1:length(CellId)  % For each geometry
 % Initialization
 coordinates=[];
 CellvertexId=[];
 epiId=[];
 paliId=[];
 walls=[];
 wall2cell=[];
 spongyvertex=[];
 spongyId=[];
 V_walls=[];        % Will contain all vertices that contains internal
boundaries of pallisade parenchyma

 % Assign values to initialized vectors
 coordinates=cord{zz};
 CellvertexId=CellId{zz};
 epiId=EID{zz};
 paliId=PID{zz};
 walls=WALL{zz};
 wall2cell=W2C{zz};

%% to calculate exposed palisade to to total length

 spongyId=find(~ismember([1:length(CellvertexId)],[paliId;epiId]));

for i=1:length(walls) % For each cell wall
    if ((ismember(wall2cell(i,1),paliId)&
~ismember(wall2cell(i,2),paliId))...
        |(ismember(wall2cell(i,2),paliId)&
~ismember(wall2cell(i,1),paliId)))...
        &
(~ismember(wall2cell(i,1),[spongyId';epiId])&~ismember(wall2cell(i,2),[spo
ngyId';epiId]))
```

```
palisade_wall_exposed_length=[palisade_wall_exposed_length;norm(coordinate
s(walls(i,1),:)-coordinates(walls(i,2),:),2)];

    end
end

EPL2LW=sum(palisade_wall_exposed_length)/(max(coordinates(:,1))-
min(coordinates(:,1)));

for i=1:length(walls) % For each cell wall
    if ((ismember(wall2cell(i,1),spongyId)&
~ismember(wall2cell(i,2),spongyId))...
        |(ismember(wall2cell(i,2),spongyId)&
~ismember(wall2cell(i,1),spongyId)))...
        &
(~ismember(wall2cell(i,1),[paliId;epiId])&~ismember(wall2cell(i,2),[paliId
;epiId]))

spongy_wall_exposed_length=[spongy_wall_exposed_length;norm(coordinates(wa
lls(i,1),:)-coordinates(walls(i,2),:),2)];

    end
end

ESL2LW=sum(spongy_wall_exposed_length)/(max(coordinates(:,1))-
min(coordinates(:,1)));
total=(sum(spongy_wall_exposed_length)+sum(palisade_wall_exposed_length))/
(max(coordinates(:,1))-min(coordinates(:,1)));

Leaf_width=max(coordinates(:,1))-min(coordinates(:,1));
Thickness=max(coordinates(:,2))-min(coordinates(:,2));

counter1=0;
counter2=0;
counter3=0;

for k=1:length(CellvertexId),
if ismember (k,epiId)
    counter1=counter1+1;
  patch(coordinates(CellvertexId{k},1),coordinates(CellvertexId{k},2),'r')
    hold on

text(mean(coordinates(CellvertexId{k},1)),mean(coordinates(CellvertexId{k}
,2)),[[strcat(num2str(k))]],'FontSize',10);

A_1_epi(counter1)=polyarea(coordinates(CellvertexId{k},1),coordinates(Cell
vertexId{k},2));
  xx=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),1);
    yy=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),2);

    x2 = [xx(2:length(xx),1);xx(1,1)];
    y2 = [yy(2:length(yy),1);yy(1,1)];
    %% calculating second moments of inertia
```

```
a = (1/2*sum(x2.*yy-xx.*y2));
p=sum(sqrt((x2-xx).*(x2-xx)+(y2-yy).*(y2-yy)));

x0(k) = 1/6*sum((x2.*yy-xx.*y2).*(xx+x2))/a;
y0(k) = 1/6*sum((x2.*yy-xx.*y2).*(yy+y2))/a;
Ixx= sum((yy.*yy +yy.*y2+y2.*y2).*(x2.*yy-xx.*y2))/(12);
Ixy= sum((2*xx.*yy +xx.*y2+x2.*yy+2*x2.*y2).*(x2.*yy-xx.*y2))/(24);
Iyy= sum((xx.*xx +xx.*x2+x2.*x2).*(x2.*yy-xx.*y2))/(12);

%% calculating  centroidal moments

u20=Ixx-a*(y0(k)*y0(k));

u02=Iyy-a*(x0(k)*x0(k));

u11=Ixy-a*(x0(k)*y0(k));

%% calculating major and minor diameter of a hypothetical ellipse that is
% the best fit to the set of points

L1=sqrt((2*(u20+u02+sqrt(4*u11*u11+(u02-u20)*(u02-u20)))/u11));
L2=sqrt((2*(u20+u02-sqrt(4*u11*u11+(u02-u20)*(u02-u20)))/u11));

%% calculating the dominant orientation

beta(k)=1/2*atan(2*u11/(u02-u20))*180/pi;
orientation(k)=beta(k);

%% calculating the aspect ratio

AR(k)=1-min(L2/L1,L1/L2);
Aspectratio_1_epi(counter1)=AR(k);

elseif ismember(k,paliId)
    counter2=counter2+1
  patch(coordinates(CellvertexId{k},1),coordinates(CellvertexId{k},2),'m')
    hold on

text(mean(coordinates(CellvertexId{k},1)),mean(coordinates(CellvertexId{k}
,2)),[[strcat(num2str(k))]],'FontSize',10);

A_1_pali(counter2)=polyarea(coordinates(CellvertexId{k},1),coordinates(Cel
lvertexId{k},2));

  xx=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),1);
    yy=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),2);

    x2 = [xx(2:length(xx),1);xx(1,1)];
    y2 = [yy(2:length(yy),1);yy(1,1)];
    %% calculating second moments of inertia
a = (1/2*sum(x2.*yy-xx.*y2));
p=sum(sqrt((x2-xx).*(x2-xx)+(y2-yy).*(y2-yy)));

x0(k) = 1/6*sum((x2.*yy-xx.*y2).*(xx+x2))/a;
```

```
y0(k) = 1/6*sum((x2.*yy-xx.*y2).*(yy+y2))/a;
Ixx= sum((yy.*yy +yy.*y2+y2.*y2).*(x2.*yy-xx.*y2))/(12);
Ixy= sum((2*xx.*yy +xx.*y2+x2.*yy+2*x2.*y2).*(x2.*yy-xx.*y2))/(24);
Iyy= sum((xx.*xx +xx.*x2+x2.*x2).*(x2.*yy-xx.*y2))/(12);

%% calculating  centroidal moments

u20=Ixx-a*(y0(k)*y0(k));

u02=Iyy-a*(x0(k)*x0(k));

u11=Ixy-a*(x0(k)*y0(k));

%% calculating major and minor diameter of a hypothetical ellipse that is
% the best fit to the set of points

L1=sqrt((2*(u20+u02+sqrt(4*u11*u11+(u02-u20)*(u02-u20)))/u11));
L2=sqrt((2*(u20+u02-sqrt(4*u11*u11+(u02-u20)*(u02-u20)))/u11));

%% calculating the dominant orientation

beta(k)=1/2*atan(2*u11/(u02-u20))*180/pi;
orientation(k)=beta(k);

%% calculating the aspect ratio

AR(k)=1-min(L2/L1,L1/L2);

Aspectratio_1_pali(counter2)=AR(k);
else
    counter3=counter3+1;

patch(coordinates(CellvertexId{k},1),coordinates(CellvertexId{k},2),'g')
    hold on

text(mean(coordinates(CellvertexId{k},1)),mean(coordinates(CellvertexId{k}
,2)),[[strcat(num2str(k))]],'FontSize',10);

A_1_spongy(counter3)=polyarea(coordinates(CellvertexId{k},1),coordinates(C
ellvertexId{k},2));

   xx=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),1);
    yy=coordinates(CellvertexId{k}(1:length(CellvertexId{k}(:))-1),2);

    x2 = [xx(2:length(xx),1);xx(1,1)];
    y2 = [yy(2:length(yy),1);yy(1,1)];
    %% calculating second moments of inertia
a = (1/2*sum(x2.*yy-xx.*y2));
p=sum(sqrt((x2-xx).*(x2-xx)+(y2-yy).*(y2-yy)));

x0(k) = 1/6*sum((x2.*yy-xx.*y2).*(xx+x2))/a;
y0(k) = 1/6*sum((x2.*yy-xx.*y2).*(yy+y2))/a;
Ixx= sum((yy.*yy +yy.*y2+y2.*y2).*(x2.*yy-xx.*y2))/(12);
Ixy= sum((2*xx.*yy +xx.*y2+x2.*yy+2*x2.*y2).*(x2.*yy-xx.*y2))/(24);
```

```
Iyy= sum((xx.*xx +xx.*x2+x2.*x2).*(x2.*yy-xx.*y2))/(12);

%% calculating  centroidal moments

u20=Ixx-a*(y0(k)*y0(k));

u02=Iyy-a*(x0(k)*x0(k));

u11=Ixy-a*(x0(k)*y0(k));

%% calculating major and minor diameter of a hypothetical ellipse that is
% the best fit to the set of points

L1=sqrt((2*(u20+u02+sqrt(4*u11*u11+(u02-u20)*(u02-u20)))/u11));
L2=sqrt((2*(u20+u02-sqrt(4*u11*u11+(u02-u20)*(u02-u20)))/u11));


%% calculating the dominant orientation

beta(k)=1/2*atan(2*u11/(u02-u20))*180/pi;
orientation(k)=beta(k);

%% calculating the aspect ratio

AR(k)=1-min(L2/L1,L1/L2);

Aspectratio_1_spongy(counter3)=AR(k);

end
end


A_epi_1=([A_1_epi]);
A_pali_6=([A_1_pali]);
A_spongy_6=([A_1_spongy]);
Mean_A_e= mean(A_epi_1);
std_A_e=std(A_epi_1);
Mean_A_p= mean(A_pali_6);
std_A_p=std(A_pali_6);
Mean_A_s= mean(A_spongy_6);
std_A_s=std(A_spongy_6);

Mean_A_p= mean(A_pali_6);
Mean_A_s= mean(A_spongy_6);

aspect_epi=[Aspectratio_1_epi];
aspect_pali_6=[Aspectratio_1_pali];
aspect_spongy_6=[Aspectratio_1_spongy];
Mean_As_p=mean(aspect_pali_6);
Std_As_p=std(aspect_pali_6);
Mean_As_S=mean(aspect_spongy_6);
Std_As_s=std(aspect_spongy_6);
Mean_As_e=mean(aspect_epi);
Std_As_e=std(aspect_epi);
```

```
% mean values
result=[result;EPL2LW ESL2LW total Mean_A_e std_A_e Mean_A_p std_A_p
Mean_A_s std_A_s Mean_As_e Std_As_e Mean_As_p Std_As_p Mean_As_S Std_As_s
Thickness Leaf_width];


end
result_tot=[result_tot;result];

end
```

Protocol S3. Matlab code for solving the gas exchange model

Matlab code for solving microscale model of gas exchange during C_3 photosynthesis using virtual leaf tissue geometries.

```
% Running file; Author: Moges Retta, moges.retta@kuleuven.be
% Comsol linked with Matlab is required to run this file.
% Comsol version 5.0 and Matlab 2017 were used to write this code
% The code calls for the function comsol_model. Inputs for comsol_model are
% geometry information. These are matrices containing indices of cells,
% vertices and  coordinates for each vertex of a cell. In addition, PID
% and EID matrices specific which of the cells belong to palisade and
% epidermis. Cell ids that are not in EID and PID are then ids for spongy
% cells. These inputs are outputs from the geometry generator.
% cord: Coordinates of the corresponding indices of the vertices
% CellID: Indices of each cell, {cells ID, no. of geometries}
% PID: indices of palisade cells
% EID: indices of epidermal cells
% model_name: name of the comsol model
% result_Iinc: the rate of photosynthesis in response to irradiance(Iinc)
% result_Ci: the response of photosynthesis to intercellular CO2 (Ci)

load cord
load CellId
load PID
load EID

result_Iinc=[];
result_Ci=[];

for j= 1:length(CellId)

    model_name=['model',num2str(j)];

    % Initializing matrices to contain information for each
    % geometry in CellID
    coordinates=[];
    CellvertexId=[];
    epiId=[];
    paliId=[];

    coordinates=cord{j};
    CellvertexId=CellId{j};
    epiId=EID{j};
    paliId=PID{j};


model=Comsol_model(coordinates,CellvertexId,paliId,epiId,model_name,result_Ci
,result_Iinc);

end
```

```
% comsol_model. Author: Moges Retta, moges.retta@kuleuven.be
% Comsol linked with Matlab is required to run this file.
% Comsol version 5 and Matlab 2017 were used to write this code
% The code produces the solved model of gas diffusion and photosynthesis
% kinetics. In addition, the response of photosynthesis to irradiance (Iinc)
% and intercellular CO2 (Ci) are produced. Inputs for comsol_model are
% geometry information. These are matrices containing indices of cells,
% vertices and  coordinates for each vertex of a cell. In addition, PID
% and EID matrices specificity which of the cells belong to palisade and
% epidermis. Cell ids that are not in EID and PID are then ids for spongy
% cells.

function out =
Comsol_model(coordinate,CellvertexIds,paliIds,epiIds,model_name,result_A_Ci,r
esult_A_Iinc)


model = ModelUtil.create(model_name);
import com.comsol.model.*
import com.comsol.model.util.*
model.geom.create('geom1', 2);  % model dimension

% creating label for each cell type
model.geom('geom1').selection.create('csel1', 'CumulativeSelection');
model.geom('geom1').selection('csel1').label('Epidermis');

model.geom('geom1').selection.create('csel2', 'CumulativeSelection');
model.geom('geom1').selection('csel2').label('Palisade');

model.geom('geom1').selection.create('csel3', 'CumulativeSelection');
model.geom('geom1').selection('csel3').label('Spongy');

% Making polygons from x,y coordinates of the vertices of the cells and
% converting them to solid objects. These objects are then assigned to each
% cell label shown above.

for i=1:length(CellvertexIds)

    x=coordinate(CellvertexIds{i},1);
    y=coordinate(CellvertexIds{i},2);
    x=x';
    y=y';
    XX = regexprep(num2str(x,17), '\s*', ',');
    YY= regexprep(num2str(y,17), '\s*', ',');

string=['csol',num2str(i)];
string1=['csol',num2str(length(CellvertexIds)+i)];

model.geom('geom1').feature.create(string, 'Polygon');
model.geom('geom1').feature(string).set('type', 'closed');
model.geom('geom1').feature(string).set('y', XX);
model.geom('geom1').feature(string).set('x', YY);
model.geom('geom1').run(string);
```

```
model.geom('geom1').create(string1, 'ConvertToSolid');
model.geom('geom1').feature(string1').selection('input').set({string});
model.geom('geom1').feature(string1').set('repairtol', '1.0E-6');
model.geom('geom1').run(string1');

model.geom('geom1').feature(string1).set('createselection', 'on');
model.geom('geom1').feature(string).set('createselection', 'on');

        if ismember(i,epiIds)
        model.geom('geom1').feature(string1).set('contributeto', 'csel1');
        elseif ismember(i,paliIds)
        model.geom('geom1').feature(string1).set('contributeto', 'csel2');
        else
        model.geom('geom1').feature(string1).set('contributeto', 'csel3');
        end

end

% correctly aligning the geometry in comsol geometry window
model.geom('geom1').create('rot1', 'Rotate');
model.geom('geom1').feature('rot1').selection('input').named('csel1');
model.geom('geom1').feature('rot1').set('rot', '90');
model.geom('geom1').run('rot1');

model.geom('geom1').create('rot2', 'Rotate');
model.geom('geom1').feature('rot2').selection('input').named('csel2');
model.geom('geom1').feature('rot2').set('rot', '90');
model.geom('geom1').run('rot2');

model.geom('geom1').create('rot3', 'Rotate');
model.geom('geom1').feature('rot3').selection('input').named('csel3');
model.geom('geom1').feature('rot3').set('rot', '90');
model.geom('geom1').run('rot3');

model.geom('geom1').create('mir1', 'Mirror');
model.geom('geom1').feature('mir1').selection('input').named('csel1');
model.geom('geom1').run('mir1');
model.geom('geom1').feature.duplicate('mir2', 'mir1');
model.geom('geom1').runPre('mir2');
model.geom('geom1').feature('mir2').selection('input').named('csel2');
model.geom('geom1').run('mir2');
model.geom('geom1').create('mir3', 'Mirror');
model.geom('geom1').feature('mir3').selection('input').named('csel3');
model.geom('geom1').run('mir3');

model.geom('geom1').runPre('fin');
model.geom('geom1').feature('fin').set('repairtol', '1.E-4');


% Adding a rectangular box encompassing the geometry
Xmin_= min(coordinate(:,1));
Ymin_= min(coordinate(:,2))-5;
```

```
Xmax_= max(coordinate(:,1));
Ymax_= max(coordinate(:,2))+10+abs(Ymin_);
model.geom('geom1').create('r1', 'Rectangle');
model.geom('geom1').feature('r1').set('size', {num2str(Xmax_)
num2str(Ymax_)});
model.geom('geom1').feature('r1').set('pos', {num2str(Xmin_)
num2str(Ymin_)});
model.geom('geom1').run('r1');
model.geom('geom1').create('spl1', 'Split');
model.geom('geom1').feature('spl1').selection('input').set({'mir1' 'mir2'
'mir3' 'r1'});
model.geom('geom1').feature('spl1').set('keep', 'on');
model.geom('geom1').run('spl1');

model.geom('geom1').run('fin');
model.geom('geom1').lengthUnit([native2unicode(hex2dec({'00' 'b5'}),
'unicode') 'm']);

% Model parameters
model.param.set('P', '101325[Pa]', 'Total pressure');
model.param.set('T', '298[K]', 'Temperature');
model.param.set('R', '8.314[(m^3*Pa)/(mol*K)]', 'Universal gas constant');
model.param.set('H', '0.83[mol/m^3/(mol/m^3)]', 'Dimensionless Henry
constant');
model.param.set('ci_sweep', '3.5E-4', 'Ambient [CO2]');
model.param.set('ci', 'ci_sweep*P/R/T', 'Intercellular CO2 for A-Ci');
model.param.set('ci_d', '244[umol/mol]*P/R/T');
model.param.set('O', '210[mmol/mol]*P', 'M&M constant of Rubisco for O2');
model.param.set('Dl', '1.89e-9[m^2/s]', 'Diffusion coefficient of CO2');
model.param.set('D_HCO3', '1.17e-9[m^2/s]', 'Diffusion coefficient of
bicarbonate');
model.param.set('Dw', '0.2*Dl', 'Diffusivity of CO2 in cell wall');
model.param.set('h_coeff', '3.5e-3[m/s]', 'Permeability to CO2 of biological
membranes');
model.param.set('h_chloro', 'h_coeff/2', 'CO2 permeability of chl envelope');
model.param.set('h_HCO3', '5.6e-7[m/s]', 'Permeability to bicarbonate');
model.param.set('Vc_max', '274[umol/m^2/s]', 'Maximum capacity of RuBP
carboxylation');
model.param.set('jm', '223.2e-6', 'Rate of electron transport');
model.param.set('Tp', '13.6[umol/m^2/s]', 'Trios phosphate utilization');
model.param.set('kmC', '267[umol/mol]*P/R/T*H', 'M&M constant of Rubisco for
CO2');
model.param.set('kmO', '164[mmol/mol]*P', 'M&M constant of Rubisco for O2');
model.param.set('ratio_O_Km', 'O/kmO', 'The ratio of [O2] to kmO');
model.param.set('Rd', '2.65[umol/m^2/s]', 'Rate of day-respiration');
model.param.set('gama', '(0.5/3260)*O/R/T*H', 'gama star');
model.param.set('k1', '0.039[1/s]', 'Rate constant of hydration of CO2');
model.param.set('k2', '23[1/s]', 'Rate constant of dehydration of
bicarbonate');
model.param.set('ka1', '2.5e-4[mol/l]', 'Equilibrium constant of CO2
(de)hydration');
model.param.set('protonHc', '10^(-7.8)[mol/l]', 'pH, chloroplasr');
model.param.set('tcw', '0.2[um]', 'Cell wall thickness');
model.param.set('tcw_s', '0.2[um]', 'Cell wall thickness of spongy');
model.param.set('tmem', '0.02[um]', 'Plasma membrane thickness');
model.param.set('kcc', '1.5[mol/m^3]','M&M constant of CA');
```

```
model.param.set('xa', '0.27[mol/m^3]','Concetrionatio of CA');
model.param.set('ka', '2e5[1/s]','Turn-over rate of CA');
model.param.set('keq', '5.6e-7[mol/l]','Equilib. const. of CA');
model.param.set('kh', '34[mol/m^3]','M&M constant of CA');
model.param.set('tcyt', '0.212[um]','Thickness of cytosol');
model.param.set('fII', '0.50','Thcikness of cytosol');
model.param.set('k2LL', '0.357','Conversion efficiency of light to electron
transport');
model.param.set('teta', '0.797','convexity factor');
model.param.set('I_inc', '1500[umol/m^2/s]','Thcikness of cytosol');
model.param.set('J', '1.1479451129999999E-4[mol/(m^2*s)]', 'Parametric sweep
value');


% Photosynthesis kinetics and diffusion equations
model.variable.create('var1');
model.variable('var1').model('mod1');
model.variable('var1').selection.geom('geom1', 2);
model.variable('var1').set('Ac', 'Vc_max*(c1)/(c1+kmC*(1+ratio_O_Km))');
model.variable('var1').set('Aj', 'J_m*(c1)/(4*c1+8*gama)');
model.variable('var1').set('Ap', '3*Tp/(max(eps,1-(gama/(c1))))');
model.variable('var1').set('AG', 'min(Ac,Aj)');
model.variable('var1').set('A', 'min(AG,Ap)');
model.variable('var1').set('hh', 'ka*xa*(c1-
c2*protonHc/keq)/(kcc+kcc*c2/kh+c1)');
model.variable('var1').set('rpc', 'Vc_max*gama/(c1+kmC*(1+ratio_O_Km))');
model.variable('var1').set('rpj', 'J_m*(gama)/(4*c1+8*gama)');
model.variable('var1').set('rp', 'min(rpc,rpj)');
model.variable('var1').set('conc', 'c1');
model.variable('var1').set('concH', 'c2');
model.variable('var1').set('Anet', 'A-Rd-rp');
model.variable('var1').set('J_m', '(k2LL*I_inc+jm-sqrt((k2LL*I_inc+jm)^2-
4*teta*k2LL*I_inc*jm))/(2*teta)');
model.variable('var1').set('f_chloro', 'Vmeso/Vtissue');

model.view.create('view2', 2);

model.variable.create('var2');
model.variable('var2').model('mod1');
model.variable('var2').selection.geom('geom1', 2);

model.cpl.create('intop1', 'Integration', 'geom1');
model.cpl.create('intop2', 'Integration', 'geom1');

model.physics.create('tds', 'DilutedSpecies', 'geom1');
model.physics('tds').prop('TransportMechanism').set('Convection', '0');
model.physics('tds').field('concentration').field('c1');
model.physics('tds').field('concentration').component({'c1' 'c2'});
model.physics('tds').feature('cdm1').set('D_c2', {'D_HCO3' '0' '0' '0'
'D_HCO3' '0' '0' '0' 'D_HCO3'});
model.physics('tds').feature('cdm1').set('D_c1', {'Dl' '0' '0' '0' 'Dl' '0'
'0' '0' 'Dl'});

model.physics('tds').create('reac1', 'Reactions', 2);
model.physics('tds').create('reac2', 'Reactions', 2);
```

```
M1=model.physics('tds').feature('reac2').selection.named('geom1_csel2_dom').i
nputEntities(); % palisade
M2=model.physics('tds').feature('reac2').selection.named('geom1_csel3_dom').i
nputEntities(); % spongy

% Identifying the boundaries of each cell, cell types using box selection
N=model.geom('geom1').getNDomains();
if length(M1)==N %
    core_pali=[];
for i=1:length(CellvertexIds)
    if ismember(i,paliIds)
    x=coordinate(CellvertexIds{i},1);
    y=coordinate(CellvertexIds{i},2);
    core_pali=[core_pali;x y];
    end
end

xminp=min(core_pali(:,1));
yminp=min(core_pali(:,2))+8;

xmaxp=max(core_pali(:,1));
ymaxp=max(core_pali(:,2))-8;

model.geom('geom1').create('boxsel5', 'BoxSelection');
model.geom('geom1').feature('boxsel5').set('xmax', num2str(xmaxp));
model.geom('geom1').feature('boxsel5').set('xmin', num2str(xminp));
model.geom('geom1').feature('boxsel5').set('ymin', num2str(yminp));
model.geom('geom1').feature('boxsel5').set('ymax', num2str(ymaxp));
model.geom('geom1').selection.create('csel10', 'CumulativeSelection');
model.geom('geom1').selection('csel10').label('pali');
model.geom('geom1').feature('boxsel5').set('contributeto', 'csel10');
model.geom('geom1').run('boxsel5');
M1=model.physics('tds').feature('reac2').selection.named('geom1_csel10_dom').
inputEntities(); %
end

Meso=union(M2,M1); %

 model.physics('tds').selection.set(Meso);
 model.variable('var1').selection.set(Meso);
 model.variable('var2').selection.set(Meso);
 model.cpl('intop1').selection.set(Meso);
 model.variable('var2').set('Vmeso', 'intop1(1)');

 yy=min(coordinate(:))-50;
 yy2=min(coordinate(:))-2;
 % lower part of the rectangle
model.geom('geom1').create('boxsel2', 'BoxSelection');
model.geom('geom1').feature('boxsel2').set('xmax', num2str(Xmax_));
model.geom('geom1').feature('boxsel2').set('xmin', num2str(Xmin_));
model.geom('geom1').feature('boxsel2').set('ymin', num2str(yy));
model.geom('geom1').feature('boxsel2').set('ymax', num2str(yy2));
model.geom('geom1').selection.create('csel5', 'CumulativeSelection');
model.geom('geom1').selection('csel5').label('lower rect');
```

```
model.geom('geom1').feature('boxsel2').set('contributeto', 'csel5');
model.geom('geom1').run('boxsel2');

Ymax2=max(coordinate(:,2));
Xmin=min(coordinate(:,1));
% Upper part of the rectangle
model.geom('geom1').create('boxsel3', 'BoxSelection');
model.geom('geom1').feature('boxsel3').set('xmax', num2str(Xmax_));
model.geom('geom1').feature('boxsel3').set('xmin', num2str(Xmin));
model.geom('geom1').feature('boxsel3').set('ymin', num2str(Ymax2+2));
model.geom('geom1').feature('boxsel3').set('ymax', num2str(Ymax2+50));
model.geom('geom1').selection.create('csel6', 'CumulativeSelection');
model.geom('geom1').selection('csel6').label('Upper rect');
model.geom('geom1').feature('boxsel3').set('contributeto', 'csel6');
model.geom('geom1').run('boxsel3');

%
id22=mphselectbox(model,'geom1',[-inf -inf;inf inf]','domain');
M3=model.physics('tds').feature('reac2').selection.named('geom1_csel5_dom').i
nputEntities(); %
M4=model.physics('tds').feature('reac2').selection.named('geom1_csel6_dom').i
nputEntities(); %
Extra_geoms=union(M3,M4);
Tissue=setdiff(id22,Extra_geoms);

model.variable.create('var3');
model.variable('var3').model('mod1');
model.variable('var3').selection.geom('geom1', 2);
model.variable('var3').selection.set(Tissue);
model.variable('var3').set('Vtissue', 'intop2(1)');
model.cpl('intop2').selection.set(Tissue);

L=Xmax_-Xmin_;
L=L*1e-6; % µm
string=['Vtissue/',num2str(L)];

model.variable('var3').set('d', string); % average thickness of the tissue

model.physics('tds').feature('reac1').selection.named('geom1_csel2_dom');
model.physics('tds').feature('reac1').set('R_c2', 'hh');
model.physics('tds').feature('reac1').set('R_c1', '-Anet/(d*f_chloro)-hh');
model.physics('tds').feature('reac2').selection.named('geom1_csel3_dom');
model.physics('tds').feature('reac2').set('R_c2', 'hh');
model.physics('tds').feature('reac2').set('R_c1', '-Anet/(d*f_chloro)-hh');
model.physics('tds').create('fl1', 'Fluxes', 1);
model.physics('tds').feature('fl1').set('FluxType',
'ExternalForcedConvection');
model.physics('tds').feature('fl1').set('species', {'1'; '0'});
model.physics('tds').feature('fl1').set('kc',
{'1/(1/h_coeff+tcw/Dw+1/h_chloro+tcyt/Dl)'; '0'});
model.physics('tds').feature('fl1').set('cb', {'ci*H'; '0'});
model.physics('tds').create('tdb1', 'ThinDiffusionBarrier', 1);
model.physics('tds').feature('tdb1').selection.all; % all
model.physics('tds').feature('tdb1').set('ds', '2*(tcw+tmem)');
model.physics('tds').feature('tdb1').set('Ds', {'2*Dw+2*h_coeff*tmem';
'2*h_HCO3*tmem+2*D_HCO3'});
```

```
model.physics('tds').feature('init1').setIndex('initc', 'eps', 0);
model.physics('tds').create('tdb2', 'ThinDiffusionBarrier', 1);
model.physics('tds').feature('tdb2').set('ds', '2*(tcw_s+tmem)');
model.physics('tds').feature('tdb2').set('Ds', {'2*Dw+2*h_coeff*tmem';
'2*h_HCO3*tmem+2*D_HCO3'});
model.physics('tds').create('tdb3', 'ThinDiffusionBarrier', 1);
model.physics('tds').feature('tdb3').set('ds', 'tcw+tcw_s+2*tmem');
model.physics('tds').feature('tdb3').set('Ds', {'2*Dw+2*h_coeff*tmem';
'2*h_HCO3*tmem+2*D_HCO3'});

% Finding exposed mesophyll boundary
Xmin1= min(coordinate(:,1))+0.1;
Xmax1= max(coordinate(:,1))-0.2;
model.geom('geom1').create('boxsel1', 'BoxSelection');
model.geom('geom1').feature('boxsel1').set('entitydim', '1');
model.geom('geom1').feature('boxsel1').set('xmin', num2str(Xmin1));
model.geom('geom1').feature('boxsel1').set('xmax', num2str(Xmax1));
model.geom('geom1').selection.create('csel4', 'CumulativeSelection');
model.geom('geom1').selection('csel4').label('testRbnd');
model.geom('geom1').feature('boxsel1').set('contributeto', 'csel4');
model.geom('geom1').run;
model.geom('geom1').run('boxsel1');

h1=model.physics('tds').feature('fl1').selection.named('geom1_csel1_bnd').inp
utEntities(); % epi+RL
h2=model.physics('tds').feature('fl1').selection.named('geom1_csel2_bnd').inp
utEntities(); % pali+RL
h3=model.physics('tds').feature('fl1').selection.named('geom1_csel3_bnd').inp
utEntities(); % spongy+RL
h4=model.physics('tds').feature('fl1').selection.named('geom1_csel4_bnd').inp
utEntities(); % all except RL

total=union(h1,h2);
total2=union(total,h3);
total3=setdiff(total2,h4); % Right (R) and left (L) boundaries of the
rectangle

index=setdiff(h2,total3); %  pali except the right boundary of the rectangle

total4=setdiff(index,total3); % exposed pali_epi_upper+Palisade

index_=setdiff(h3,h2); % exposed spongy  + spongy_epi_lower + R&L boundaries

total5=setdiff(index_,total3); % exposed pali_spongy+spongy+epi_lower_spongy

index1_=intersect(h2,h1); % Pali_epi_upper interface
index2_=intersect(h3,h1); % Spongy_epi_lower interface

total6=setdiff(total5,index2_); % exposed pali_spongy+ exposed spongy_epi +
spongy

expo1=union(total6,total4);
expo=setdiff(expo1,index1_);

model.physics('tds').feature('fl1').selection.set(expo);% exposed
```

```
% mphviewselection(model,'geom1',expo_pali_2,'boundary')
hh=intersect(h1,h2); % pali epi interface
hh2=intersect(h2,h3); % spongy pali interface

expo_pali_int=setdiff(h2,hh); % pali bnd except exposed_pali_epi

expo_spongy=setdiff(h3,hh2); % spongy bnd except the pali_spongy interface

expo_pali_2=setdiff(expo_pali_int,hh2); % pali internal+ exposed +RL -
pali_epi and  spongy_epi interface

model.physics('tds').feature('tdb1').selection.set(expo_pali_2); % all
model.physics('tds').feature('tdb2').selection.set(expo_spongy); % all
model.physics('tds').feature('tdb3').selection.set(hh2); % all

model.study.create('std2');
model.study('std2').create('stat', 'Stationary');

model.study.create('std1');
model.study('std1').create('stat', 'Stationary');
model.study('std1').feature('stat').activate('tds', true);

model.sol.create('sol1');
model.sol('sol1').study('std1');

model.study('std1').feature('stat').set('notlistsolnum', 1);
model.study('std1').feature('stat').set('notsolnum', '1');
model.study('std1').feature('stat').set('listsolnum', 1);
model.study('std1').feature('stat').set('solnum', '1');

model.sol('sol1').create('st1', 'StudyStep');
model.sol('sol1').feature('st1').set('study', 'std1');
model.sol('sol1').feature('st1').set('studystep', 'stat');
model.sol('sol1').create('v1', 'Variables');
model.sol('sol1').feature('v1').set('control', 'stat');
model.sol('sol1').create('s1', 'Stationary');
model.sol('sol1').feature('s1').create('fc1', 'FullyCoupled');
model.sol('sol1').feature('s1').feature('fc1').set('dtech', 'auto');
model.sol('sol1').feature('s1').feature('fc1').set('maxiter', 50);
model.sol('sol1').feature('s1').feature('fc1').set('initstep', 0.01);
model.sol('sol1').feature('s1').feature('fc1').set('minstep', 1.0E-6);
model.sol('sol1').feature('s1').create('d1', 'Direct');
model.sol('sol1').feature('s1').feature('fc1').set('linsolver', 'd1');
model.sol('sol1').feature('s1').feature('fc1').set('dtech', 'auto');
model.sol('sol1').feature('s1').feature('fc1').set('maxiter', 50);
model.sol('sol1').feature('s1').feature('fc1').set('initstep', 0.01);
model.sol('sol1').feature('s1').feature('fc1').set('minstep', 1.0E-6);
model.sol('sol1').feature('s1').create('p1', 'Parametric');
model.sol('sol1').feature('s1').feature('p1').set('pname', {'ci_sweep'});
model.sol('sol1').feature('s1').feature('p1').set('plistarr', {'54.03649337,
77.2741493, 130.1631251, 184.1316069, 239.4225968, 429.9634441, 608.3342773,
788.646187, 974.8097509, 1338.08416, 1724.306284'});
model.sol('sol1').feature('s1').feature('p1').set('punit', {'umol/mol'});
model.sol('sol1').feature('s1').feature('p1').set('sweeptype', 'sparse');
```

```
model.sol('sol1').feature('s1').feature('p1').set('preusesol', 'no');
model.sol('sol1').feature('s1').feature('p1').set('pcontinuationmode', 'no');
model.sol('sol1').feature('s1').feature('p1').set('plot', 'off');
model.sol('sol1').attach('std1');

model.study('std1').create('param', 'Parametric');
model.study('std1').feature('param').setIndex('pname', 'P', 0);
model.study('std1').feature('param').setIndex('plistarr', '', 0);
model.study('std1').feature('param').setIndex('punit', '', 0);
model.study('std1').feature('param').setIndex('pname', 'P', 0);
model.study('std1').feature('param').setIndex('plistarr', '', 0);
model.study('std1').feature('param').setIndex('punit', '', 0);
model.study('std1').feature('param').setIndex('pname', 'ci_sweep', 0);
model.study('std1').feature('param').setIndex('plistarr', '54.03649337,
77.2741493, 130.1631251, 184.1316069, 239.4225968, 429.9634441, 608.3342773,
788.646187, 974.8097509, 1338.08416, 1724.306284', 0);
model.study('std1').feature('param').setIndex('punit', 'umol/mol', 0);
%

model.mesh.create('mesh1', 'geom1');
model.mesh('mesh1').autoMeshSize(3);
model.mesh('mesh1').run;

model.sol('sol1').runAll;

model.study('std2').create('param', 'Parametric');

model.study('std1').feature('param').setIndex('pname', 'P', 0);
model.study('std1').feature('param').setIndex('plistarr', '', 0);
model.study('std1').feature('param').setIndex('punit', '', 0);
model.study('std1').feature('param').setIndex('pname', 'P', 0);
model.study('std1').feature('param').setIndex('plistarr', '', 0);
model.study('std1').feature('param').setIndex('punit', '', 0);
model.study('std1').feature('param').setIndex('pname', 'ci_sweep', 0);
model.study('std2').feature('param').setIndex('pname', 'P', 0);
model.study('std2').feature('param').setIndex('plistarr', '', 0);
model.study('std2').feature('param').setIndex('punit', '', 0);
model.study('std2').feature('param').setIndex('pname', 'P', 0);
model.study('std2').feature('param').setIndex('plistarr', '', 0);
model.study('std2').feature('param').setIndex('punit', '', 0);
% parametric sweep to solve the model for various Iinc and Ci values
model.study('std1').feature('param').setIndex('plistarr', '54.03649337,
77.2741493, 130.1631251, 184.1316069, 239.4225968, 429.9634441, 608.3342773,
788.646187, 974.8097509, 1338.08416, 1724.306284', 0);
model.study('std1').feature('param').setIndex('punit', 'umol/mol', 0);
model.study('std2').feature('param').setIndex('pname', 'ci_sweep', 0);
model.study('std2').feature('param').setIndex('plistarr',
'404.8941252,392.9087017,366.6411861,348.2094222,307.4157621,286.4205483,271.
4775008,264.5839562,247.6041048', 0);
model.study('std2').feature('param').setIndex('punit', 'umol/mol', 0);
model.study('std2').feature('param').setIndex('pname', 'P', 1);
model.study('std2').feature('param').setIndex('plistarr', '', 1);
model.study('std2').feature('param').setIndex('punit', '', 1);
model.study('std2').feature('param').setIndex('pname', 'P', 1);
model.study('std2').feature('param').setIndex('plistarr', '', 1);
model.study('std2').feature('param').setIndex('punit', '', 1);
```

```
model.study('std2').feature('param').setIndex('pname', 'I_inc', 1);
model.study('std2').feature('param').setIndex('plistarr',
'24.43005753,50.69877625,100.0078634,150.1953735,301.3746033,501.2943624,750.
8420817,999.7507731,1499.285156', 1);
model.study('std2').feature('param').setIndex('punit', 'umol/m^2/s', 1);

model.sol.create('sol2');
model.sol('sol2').study('std2');

model.study('std2').feature('stat').set('notlistsolnum', 1);
model.study('std2').feature('stat').set('notsolnum', '1');
model.study('std2').feature('stat').set('listsolnum', 1);
model.study('std2').feature('stat').set('solnum', '1');

model.study('std1').label('[CO2] response');
model.study('std2').label('Light response');



model.sol('sol2').create('st1', 'StudyStep');
model.sol('sol2').feature('st1').set('study', 'std2');
model.sol('sol2').feature('st1').set('studystep', 'stat');
model.sol('sol2').create('v1', 'Variables');
model.sol('sol2').feature('v1').set('control', 'stat');
model.sol('sol2').create('s1', 'Stationary');
model.sol('sol2').feature('s1').create('p1', 'Parametric');
model.sol('sol2').feature('s1').feature('p1').set('pname', {'ci_sweep'
'I_inc'});
model.sol('sol2').feature('s1').feature('p1').set('plistarr',
{'404.8941252,392.9087017,366.6411861,348.2094222,307.4157621,286.4205483,271
.4775008,264.5839562,247.6041048'
'24.43005753,50.69877625,100.0078634,150.1953735,301.3746033,501.2943624,750.
8420817,999.7507731,1499.285156'});
model.sol('sol2').feature('s1').feature('p1').set('punit', {'umol/mol'
'umol/m^2/s'});
model.sol('sol2').feature('s1').feature('p1').set('sweeptype', 'sparse');
model.sol('sol2').feature('s1').feature('p1').set('preusesol', 'no');
model.sol('sol2').feature('s1').feature('p1').set('pcontinuationmode', 'no');
model.sol('sol2').feature('s1').feature('p1').set('plot', 'off');
model.sol('sol2').feature('s1').feature('p1').set('plotgroup', 'Default');
model.sol('sol2').feature('s1').feature('p1').set('probesel', 'all');
model.sol('sol2').feature('s1').feature('p1').set('probes', {});
model.sol('sol2').feature('s1').set('control', 'stat');
model.sol('sol2').feature('s1').create('fc1', 'FullyCoupled');
model.sol('sol2').feature('s1').feature('fc1').set('dtech', 'auto');
model.sol('sol2').feature('s1').feature('fc1').set('maxiter', 50);
model.sol('sol2').feature('s1').feature('fc1').set('initstep', 0.01);
model.sol('sol2').feature('s1').feature('fc1').set('minstep', 1.0E-6);
model.sol('sol2').feature('s1').create('d1', 'Direct');
model.sol('sol2').feature('s1').feature('fc1').set('linsolver', 'd1');
model.sol('sol2').feature('s1').feature('fc1').set('dtech', 'auto');
model.sol('sol2').feature('s1').feature('fc1').set('maxiter', 50);
model.sol('sol2').feature('s1').feature('fc1').set('initstep', 0.01);
model.sol('sol2').feature('s1').feature('fc1').set('minstep', 1.0E-6);

model.sol('sol2').runAll;
```

```
model.study('std1').feature('stat').set('notlistsolnum', 1);
model.study('std1').feature('stat').set('notsolnum', '1');
model.study('std1').feature('stat').set('listsolnum', 1);
model.study('std1').feature('stat').set('solnum', '1');

model.result.create('pg1', 'PlotGroup2D');
model.result('pg1').label('Concentration (tds)');
model.result('pg1').set('data', 'dset1');
model.result('pg1').set('oldanalysistype', 'noneavailable');
model.result('pg1').feature.create('surf1', 'Surface');
model.result('pg1').feature('surf1').label('Surface 1.1');
model.result('pg1').feature('surf1').set('oldanalysistype', 'noneavailable');
model.result('pg1').feature('surf1').set('descractive', true);
model.result('pg1').feature('surf1').set('data', 'parent');

model.result.numerical.create('int1', 'IntSurface');
model.result.numerical('int1').set('expr', 'Anet/Vmeso[m^2]*1e6');
% model.result.numerical('int1').selection.named('geom1_csel2_dom');
model.result.numerical('int1').selection.set(Meso);

model.result.table.create('tbl1', 'Table');
model.result.table('tbl1').comments('Surface Integration 1
(Anet/Vmeso[m^2]*1e6)');
model.result.numerical('int1').set('table', 'tbl1');
model.result.numerical('int1').setResult;

model.result('pg1').run;
model.result('pg1').feature('surf1').set('expr', 'c1*R*T/H/P*1e6');
model.result('pg1').run;
model.result.numerical('int1').set('table', 'tbl1');
model.result.numerical('int1').setResult;
model.result.numerical('int1').set('data', 'dset1');
model.result.numerical('int1').set('table', 'tbl1');
model.result.numerical('int1').setResult;
model.result.numerical.create('int2', 'IntSurface');
model.result.create('pg3', 'PlotGroup2D');
model.result('pg3').label('Concentration (tds) 1');
model.result('pg3').set('data', 'dset2');
model.result('pg3').set('oldanalysistype', 'noneavailable');
model.result('pg3').set('solvertype', 'none');
model.result('pg3').set('solnum', 1);
model.result('pg3').set('showlooplevel', {'off' 'off' 'off'});
model.result('pg3').set('oldanalysistype', 'noneavailable');
model.result('pg3').set('data', 'dset2');
model.result('pg3').feature.create('surf1', 'Surface');
model.result('pg3').feature('surf1').label('Surface 1.1');
model.result('pg3').feature('surf1').set('oldanalysistype', 'noneavailable');
model.result('pg3').feature('surf1').set('solvertype', 'none');
model.result('pg3').feature('surf1').set('descractive', true);
model.result('pg3').feature('surf1').set('data', 'parent');


model.sol('sol2').runAll;
```

```
model.result('pg3').run;
model.result('pg1').run;
model.result('pg3').run;
model.result('pg3').feature('surf1').set('expr', 'c1*R*T/H/P*1e6');
model.result('pg3').run;
model.result('pg3').feature('surf1').set('data', 'dset2');
model.result('pg3').run;
model.result.numerical('int2').set('data', 'dset2');
model.result.numerical('int2').selection.set(Meso);

% calculating the response of photosynthesis to Iinc and Ci
model.result.numerical('int2').set('expr', 'Anet/Vmeso[m^2]*1e6');
model.result.table.create('tbl2', 'Table');
model.result.table('tbl2').comments('Surface Integration 2
(Anet/Vmeso[m^2]*1e6)');
model.result.numerical('int2').set('table', 'tbl2');
model.result.numerical('int2').setResult;
model.result.numerical.create('int3', 'IntSurface');
model.result.numerical('int3').set('data', 'dset1');
model.result.numerical('int3').selection.set(Meso);

model.result.numerical('int3').set('expr', 'c1/Vmeso[m^2]*R*T/H/P*1e6');
model.result.table.create('tbl3', 'Table');
model.result.table('tbl3').comments('Surface Integration 3
(c1/Vmeso[m^2]*R*T/H/P)');
model.result.numerical('int3').set('table', 'tbl3');
model.result.numerical('int3').setResult;
model.result.numerical.create('int4', 'IntSurface');
model.result.numerical('int4').set('data', 'dset2');
model.result.numerical('int4').selection.set(Meso);

model.result.numerical('int4').set('expr', 'c1/Vmeso[m^2]*R*T/H/P*1e6');
model.result.table.create('tbl4', 'Table');
model.result.table('tbl4').comments('Surface Integration 4
(c1/Vmeso[m^2]*1e6*R*T/H/P)');
model.result.numerical('int4').set('table', 'tbl4');
model.result.numerical('int4').setResult;

model.result.create('pg8', 'PlotGroup1D');
model.result('pg8').create('tblp1', 'Table');
model.result('pg8').feature('tblp1').set('table', 'tbl1');
model.result('pg8').feature('tblp1').set('xaxisdata', '1');

model.result.numerical('int4').set('data', 'dset2');
model.result.table('tbl4').clearTableData;
model.result.numerical('int4').set('table', 'tbl4');
model.result.numerical('int4').setResult;
model.result('pg8').feature('tblp1').set('plotcolumninput', 'manual');
model.result('pg8').feature('tblp1').set('plotcolumns', {'2'});
model.result('pg8').feature('tblp1').set('xaxisdata', '1');
model.result('pg8').run;

model.result.create('pg10', 'PlotGroup1D');
model.result('pg10').run;
model.result('pg10').set('data', 'dset2');
model.result('pg10').create('tblp1', 'Table');
```

```
model.result('pg10').feature('tblp1').set('table', 'tbl2');
model.result('pg10').feature('tblp1').set('xaxisdata', '2');
model.result('pg10').feature('tblp1').set('plotcolumninput', 'manual');
model.result('pg10').feature('tblp1').set('plotcolumns', {'3'});
model.result('pg10').run;

model.result.create('pg11', 'PlotGroup1D');
model.result('pg11').run;
model.result('pg11').create('tblp1', 'Table');
model.result('pg11').feature('tblp1').set('table', 'tbl3');
model.result('pg11').feature('tblp1').set('xaxisdata', '1');
model.result('pg11').feature('tblp1').set('plotcolumninput', 'manual');
model.result('pg11').feature('tblp1').set('plotcolumns', {'2'});
model.result('pg11').run;



model.result.create('pg12', 'PlotGroup1D');
model.result('pg12').run;
model.result('pg12').set('data', 'dset2');
model.result('pg12').create('tblp1', 'Table');
model.result('pg12').feature('tblp1').set('table', 'tbl4');
model.result('pg12').feature('tblp1').set('xaxisdata', '2');
model.result('pg12').feature('tblp1').set('plotcolumninput', 'manual');
model.result('pg12').feature('tblp1').set('plotcolumns', {'3'});
model.result('pg12').run;

model.result('pg8').label('A_Ci');
model.result('pg10').run;
model.result('pg10').label('A_Iinc');
model.result('pg11').run;
model.result('pg11').label('Cc_Ci');
model.result('pg12').run;
model.result('pg12').label('Cc_Iinc');

    % A-Ci
    A_mod=model.result.table('tbl1').getTableData(false);
    A_mod=str2num(A_mod);
    A_mod=reshape(A_mod, 11, []);
    A_Ci_mod=A_mod(:,2);

    % A-Iinc
    A_mod2=model.result.table('tbl2').getTableData(false);
    A_mod2=str2num(A_mod2);
    A_mod2=reshape(A_mod2, 9, []);
    A_I_mod=A_mod2(:,3);

    % A-Ci
    Cc_mod=model.result.table('tbl3').getTableData(false);
    Cc_mod=str2num(Cc_mod);
    Cc_mod=reshape(Cc_mod, 11, []);
    Cc_Ci_mod=Cc_mod(:,2);

    % A-Iinc
    Cc_mod2=model.result.table('tbl4').getTableData(false);
```

```
    Cc_mod2=str2num(Cc_mod2);
    Cc_mod2=reshape(Cc_mod2, 9, []);
    Cc_I_mod=Cc_mod2(:,3);

    result_A_Ci=[result_A_Ci;A_Ci_mod Cc_Ci_mod ];
    result_A_Iinc=[result_A_Iinc;A_I_mod Cc_I_mod];

    save result_A_Ci_repair result_A_Ci
    save result_A_Iinc_repair result_A_Iinc
    mphsave(model,model_name);

    ModelUtil.remove(model_name);

out = model;
```