```
function atomplot(pos)
%The function atomsplot plots spheres at the 3dimensional coordinates
 specified by the vectors
%x,y and z in pos.
%R - radii of the atoms
%T - atom type an integer from 1 to the number of types of atoms

R=0.3;
[Xs,Ys,Zs]=sphere(25);     %Creates spheres of precision 25
Xs=Xs*R;                   %creates the radii size of the spheres
Ys=Ys*R;
Zs=Zs*R;

color{1} = [1 1 1];
color{2} = [1 0 0];

clf reset
hold on
for t=1:size(pos,1)
    ball(t)=surf(Xs+pos(t,1),Ys+pos(t,2),Zs+pos(t,3));   %plots the
 balls

    set(ball(t),'facecolor','r',...
        'edgecolor','none','facelighting','phong' );
    %sets colors and effects for the ball plotting

    for t2=t+1:size(pos,1)
        r=sum((pos(t,:)-pos(t2,:)).^2);
        if r>0.95 && r<1.05
            line([pos(t,1) pos(t2,1)],[pos(t,2) pos(t2,2)],...
                [pos(t,3) pos(t2,3)],'Color',[0 0 0],'LineWidth',4);
        elseif r<0.05
            %error('Atoms overlapping!');
        end
    end

end

axis equal          %keeps from distorting the relations of the axis
camlight right

hold off
end
```

Published with MATLAB® R2018b

```
% ALEX KLEIN GATEWAY COMPUTING PROJECT C FUNCTION GRAPHENE

% FUNCTION OVERVIEW
    % inputs n and m determine the orientation of the graphene sheet
    % input len is the length of the sheet (future tube) in bond
 lengths
    % output is a 3xn matrix 'pos' with the x,y,z coordinates of #n
 atoms

function pos = Graphene (n, m, len)

%  *** CREATE ATOM GRID ***

% create initial pos matrix
pos = [0;0;0] ;

% loop for x coordinates
for i = 0:(4*n + len)

    % loop for y coordinates
    for j = -m:(4*m + len)

        % set coordinates
        z = 0 ;
        x = (3^(0.5))/2*i ;
        y = 1.5*j ;

        % pull down every other point by 0.375 (1.5/4)
        if mod(i, 2) ~= 0 % row is even
            if mod(j, 2) == 0 % column is even
                y = y - 0.5 ;
            end
        elseif mod(i, 2) == 0 % row is even
            if mod(j, 2) ~= 0 % column is odd
                y = y - 0.5 ;
            end
        end

        % add to matrices
        atomPos = [x;y;z] ;
        pos = [pos, atomPos] ;

    end

end


% *** DEFINE VECTORS ***

% define a1/2 matrices
a1 = [(3^(0.5));0];
a2 = [(3^(0.5))/2;-3/2];
```

```
% define vector C
C = [ n*a1(1,1) + m*a2(1,1) ; n*a1(2,1) + m*a2(2,1) ] ;

% define vector T
T = ( [0,-1;1,0] * (C./norm(C)) ) * len ;
% (vector T is an orthonormal vector to C times variable len) process:
% 'C./norm(C)' finds the unit vector C
% '[0,-1;1,0]' is the rotation matrix to rotate T 90 degrees CCW of C
% '* len' scales the perpecdicular unit vector T to the inputed length


% *** CUT AWAY ATOMS OUTSIDE OF BOUNDS ***

% define polygon
xv = [ 0 , C(1,1) , T(1,1) + C(1,1) , T(1,1) , 0 ] ;
yv = [ 0 , C(2,1) , T(2,1) + C(2,1) , T(2,1) , 0 ] ;

% for loop to test each point
for k = 1:size(pos,2)

    % define point
    xq = pos(1,k) ;
    yq = pos(2,k) ;

    % test if point is in polygon
    in = inpolygon(xq,yq,xv,yv) ;

    % set point to zero if in polygon
    if in == 1
    else
        pos(:,k) = [0;0;0] ;
    end

end


% *** REDUCE POS MATRIX ***

% (removing all columns of [0;0;0])

% create initial array for logical indexing
posZero = ones(1, size(pos,2)) ;

% create boolean vector, for loop to go through pos matrix
for k = 1:size(pos,2)

    % test if the column is all zeros (ignoring z coordinate)
    if pos(1,k) == 0 && pos(2,k) == 0

        % set boolean vector to zero if column of pos is all zeros
        posZero(1,k) = 0 ;

    end
```

```
end

% convert to logical vector
posLogical = logical(posZero) ;

% apply logical indexing to columns
pos = pos(:,posLogical) ;

% add origin back into pos matrix
pos = [[0;0;0],pos];


% *** PLOT GRAPHENE SHEET ***

% transpose pos to fit atomplot function input requirments
pos = pos';

% call atomplot for 3d viewing
figure(2);
atomplot(pos);

end
```

Published with MATLAB® R2018b

```
% ALEX KLEIN GATEWAY COMPUTING PROJECT C FUNCTION NANOTUBE

% FUNCTION OVERVIEW
    % inputs n and m determine the orientation of the graphene sheet
    % input len is the length of the sheet (future tube) in bond
 lengths
    % output is a 3xn matrix 'pos' with the x,y,z coordinates of #n
 atoms

function pos = Nanotube (n, m, len)

% *** CALL GRAPHENE FUNCTION ***

% creates graphenePos matrix of graphene sheet atom coordinates
graphenePos = (Graphene (n, m, len))';


% *** DEFINE VECTORS ***

% set initial pos matrix for new coordinates
pos = zeros(3,size(graphenePos,2));

% define a1/2 matrices
a1 = [(3^(0.5));0];
a2 = [(3^(0.5))/2;-3/2];

% define vector C
C = [ n*a1(1,1) + m*a2(1,1) ; n*a1(2,1) + m*a2(2,1) ] ;

% define vector T
T = ( [0,-1;1,0] * (C./norm(C)) ) * len ;
% (vector T is an orthonormal vector to C times variable len) process:
% 'C./norm(C)' finds the unit vector C
% '[0,-1;1,0]' is the rotation matrix to rotate T 90 degrees CCW of C
% '* len' scales the perpecdicular unit vector T to the inputed length

% define radius of tube
r = (1/(2*pi)) * (norm(C)) ;


% *** FIND CYLINDRICAL COORDINATES ***

% loop through each point and change coordinate to cylindrical
 positioning
for k = 1:size(graphenePos,2)

    % define c for specific point (current atom corrdinate)
    c = [ graphenePos(1,k) ; graphenePos(2,k) ] ;

    % define t for specific point (distance along vector T)
    t = dot(c,T) / dot(T,T) * T ; % this is the projection of c onto T
```

```
        % define s for specific point (distance along vector C)
        s = dot(c,C) / dot(C,C) * C ; % this is the projection of c onto C

        % set current corrdinate of atom in cylindrical position
        x = r * cos( norm(s)/r ) ;
        y = r * sin( norm(s)/r ) ;
        z = norm(t) ;

        % add current coordinate to pos matrix
        pos(:,k) = [x;y;z];

    end


    % *** REMOVE OVERLAPPING ATOMS ***

    % adjust pos matrix for armchair tube
    if n == m
        pos(:,1) = [];
    end

    % run through each atom
    for t=1:size(pos,2)

        % check against every other atom
        for t2=t+1:size(pos,2)-1

            % determine distance between two atoms
            r=sum((pos(:,t)-pos(:,t2)).^2);

            % if statement to determine if both atoms are touching
            if r<0.05

                % delete from matrix
                pos(:,t) = [];

            end

        end

    end


    % *** PLOT ***

    % remove origin point
    pos(:,1) = [];

    % transpose matrix
    pos = pos';

    % atomplot for 3d viewing
    figure(3);
    atomplot(pos);
```

```
end
```

Published with MATLAB® R2018b