Team Members : Kartik (BS19B017) | Vineet (BS19B033)

Assignment No.1(Writeup)

Problem Statement -

➔ Objective - To show the calculations involved and the method used to assign Values with respective cases considered as per the Polymer Movement during protein folding through a 2D Lattice System comprising 6 Polymer, each containing 4 residues.

➔ Coding Language Used – MATLAB Scripting Language

➔ Assumptions –

- The Residues of each polymer can move only in a limited space defined by a mesh grid of variable “SqDim” in code and can move only 1 unit step per move either through end to end or corner move.
- The Residues Can Cross the grid and enter from the opposite Side.
- Each polymer can move translationally either in positive or negative direction along the horizontal or vertical axis of the respective grid.
- Interactions between residues are considered only when the separation between residues is equal to the unit length of the lattice system. Also,kbT=1 while calculating Boltzmann Weight.
- Residues of one polymer do not interact with each other.

Introduction - Proteins play a vital role in everyday bodily function, thus understanding their behaviour is paramount to biological, pharmaceutical and medical research. A protein can only perform correctly if it is in a biologically `active' state, which requires the structure to be precisely folded into the right shape. This is the subject of Levinthal's paradox, which suggests that proteins need to calculate the energies of an enormous number of folded states to determine the native local minimum. In this paper, protein folding is modelled using lattice simulation Monte Carlo techniques, which discretises the folding process. This mechanism presents new insights into the problem of how a protein folds so rapidly, and significantly reduces the computational time compared to that put forward by Levinthal.

In each Monte Carlo time step the programme selects,at random, an amino acid to determine if it can be `moved'. A move constitutes repositioning the selected monomer to one of its neighbouring (unoccupied) lattice positions, whilst not breaking the bonds between the two amino acids adjacent in the primary structure

Step – To -Step Methodology -

Step 1 - Initially we create a 3D array matrix, in a variable named ”Lattice” containing 6 pages with dimensions of each page - (4 Rows X 2 Cols). In “Lattice”, on each page, each row denotes the position values of the specific residue on a 2D Lattice , defined by a mesh grid. Each page contains 2 columns, where the first column and the second column specify X columns and Y columns respectively of residue of the specific polymer.

Step 2 – Now we create(Initialise) variables as defined below for our code –

- “Lattice2” – This variable stores a copy of original lattice(named by variable “Lattice” as per Step 1). After each iteration, it would be reassigned to original lattice, so as to keep it updated with fresh coordinates of residues.
- “SqDim” – Stores the dimension value m so as to create a mesh grid of m x m, which defines our lattice system.
- “EInitial” - It stores the Initial energy of the Lattice System when none of the polymer residues are interacting with each other.
- “Interaction Energy” – It stores the value of Interaction energy between residues as per combination of residues which can interact. It can be changed as per required.Its initial value is set to “Energy of Lattice” one uses to calculate relative energy of systems.
- “StableEnergyIndex” - It is an vector whose initial value is set to 0. After each successful step
- “N” - It stores the number of Iterations one wants to repeatedly run the movements in order to check protein folding.

Step 3 -

- Now, we would be Initialising a loop to iterate over the steps discussed below for a specific number of times given by our “N” variable.
- Then, we will create a loop over all the polymers taken in our lattice system, from 1 to 6, in order to randomly select a particular residue from 1 to 4 for each polymer when we would iterate over each polymer, in which we will check all the movements available for that particular residue.

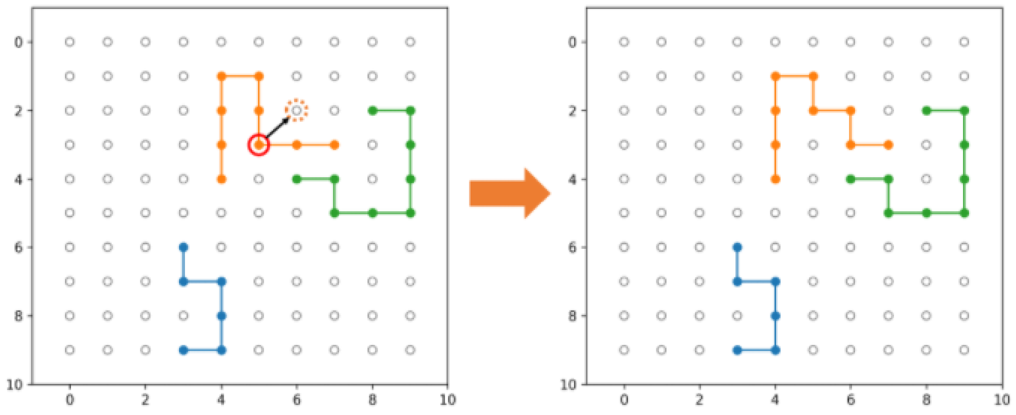
- A polymer can also translate horizontally or vertically by one lattice length in each Monte Carlo Step
- We will move the selected residue for each polymer based on specific conditions and cases as following:

Possible movements for a Residue in polymer:

1)Corner Movement : In this type of movement, a residue will shift to the diagonally opposite location.

Conditions for Corner Movement :

I. Diagonally opposite location in the lattice system should be vacant.
II. The selected Residue before and after the corner movement should remain connected to the preceding and succeeding residue of the same polymer.
III. It should not be in a linear configuration with its preceding and succeeding residues and they should form adjacent sides of a square.
IV. Selected residue should not be located at the ends of a polymer as end residue can't do corner move.



Let x1,y1 be the coordinates of the residue which will be doing corner move

Let x2,y2 be the coordinates of the residue preceding the selected residue

Letx3,y3 be the coordinates of the residue succeeding the selected residue

The adjacent sides formed from the selected residue - preceding residue pair and succeeding residue - selected residue pair form adjacent sides of a square.

So the coordinates of diagonally opposite point of square are given by

(x1 + (x2 + x3 - 2 * x1)),(y1 + (y2 + y3 - 2 * y1))

2)End Movement :

In this type of movement, a residue located at the end of a polymer shifts 90 degrees (clockwise/anticlockwise) while remaining connected to the adjacent residue of the same polymer.

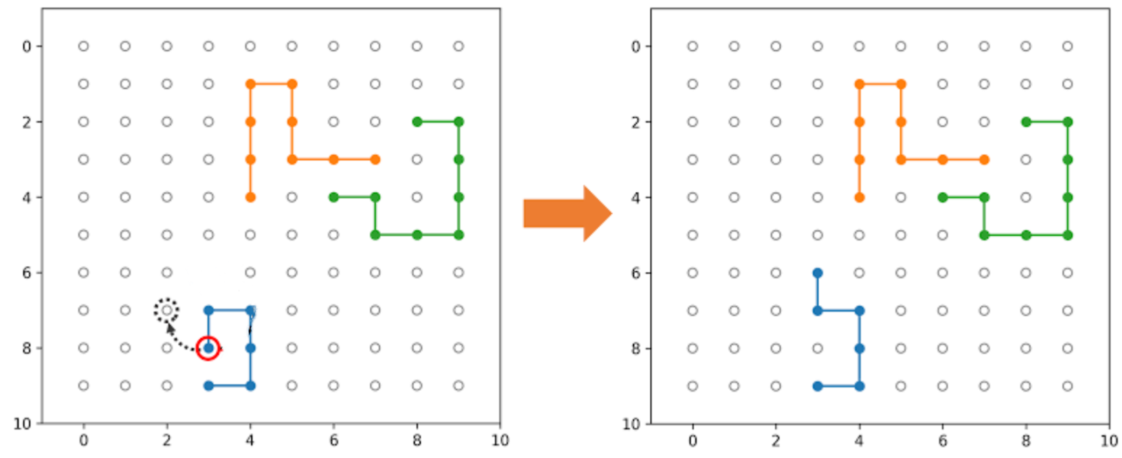
<u>Conditions for End Movement</u>:

I.	It should remain connected to the adjacent residue and should have a distance of 1 unit between them.

II.	The final location should be vacant for our selected residue to make a end to end move

III.	Selected residue should not be located in the middle of the polymer as residues in the middle can't do an end movement.

IV.	It should shift 90 degrees from its original location considering adjacent residue as the center of the circle



<u>To calculate the coordinates of possible end movements, we will be considering two possible cases :</u>

Let x1,y1 be the coordinates of our selected residue

Let x2,y2 be coordinates of the adjacent residue

<u>Case 1</u> - When x coordinate of selected residue and adjacent residue of same polymer, is same :

Possible movement coordinates for end movements for selected residue is given by:

(x1 + 1, y2) and (x1 - 1, y2)

<u>Case 2</u> - When y coordinate of selected residue and adjacent residue of same polymer, is same :

Possible movement coordinates for end movements for selected residue is given by:

(x2 , y1+1) and (x2, y2 - 1)

Note:

The Major Advantage of the methods used to do a corner or end move, is that it can be used even when it crosses the grid to enter again from the opposite side when used with the **"StayInGrid"** function, which will be discussed further.

Step 4 :

- If the energy of our lattice system satisfies Metropolis criteria, we would proceed to make all the above moves we got in this iteration, else we would neglect and will move on to the next Iteration.

  As per Metropolis Criteria,a move can only be made if one of two conditions are met:

  1. The difference in the energy of the protein after the proposed move and its current state, Emove, is negative (i.e. the new energy is less than the current).

  2. $\exp(-\Delta E_{move}/k_B T) > r$ where T is the temperature of the solution and Boltzmann's constant is represented by k_B .Here, (k_B T) is set to unity, and r is a randomly generated number between 0 and 1.

  The second criterion corresponds to calculating the probability that the system, represented by a canonical ensemble of energy microstates, makes an energy transition, and then tests to see if the system makes such a transition, given that probability. This condition accounts for the statistical uncertainty in the system due to entropic considerations. If one of the conditions is met then the residue is moved and the process repeats again for as many steps as specified by the user (Here, it is repeated for 10^6 times).

- After all the Iterations, the final lattice configuration is assumed to be the final lattice conformation reflecting the property of proteins.

Functions Defined :

- isOccupied - This function checks whether the location of a point in the 2D lattice is occupied by any other residue or not and hence, it returns 1 or true if it is occupied by a residue of any other polymer.

- distance - This function calculates the distance between locations of two residues passed to it, in the form of two arrays.

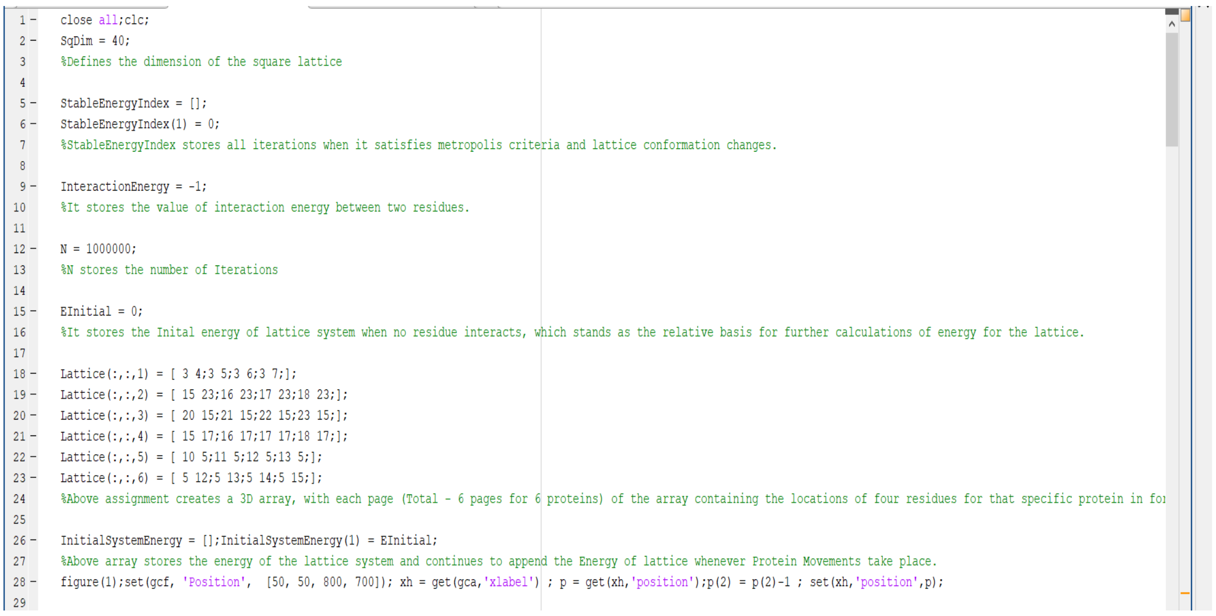
- StayInGrid - This function checks whether the given coordinates of a point lies within our 2D lattice and in case, if it lies outside the 2D lattice, changes its coordinates in such a way, that it enters from the opposite site of the Lattice.

- <u>Energy_A</u> - This function is used to calculate the total energy of the lattice in which Residue A of one polymer can interact only with Residue A of another polymer. Similarly, Residue B of one polymer can interact only with Residue B of another polymer and similar case for interactions for Residue C and D.

- <u>Energy_B</u> - _This function is used to calculate the total energy of the lattice system when any residue of one polymer can interact with any residue of another polymer of our lattice system.

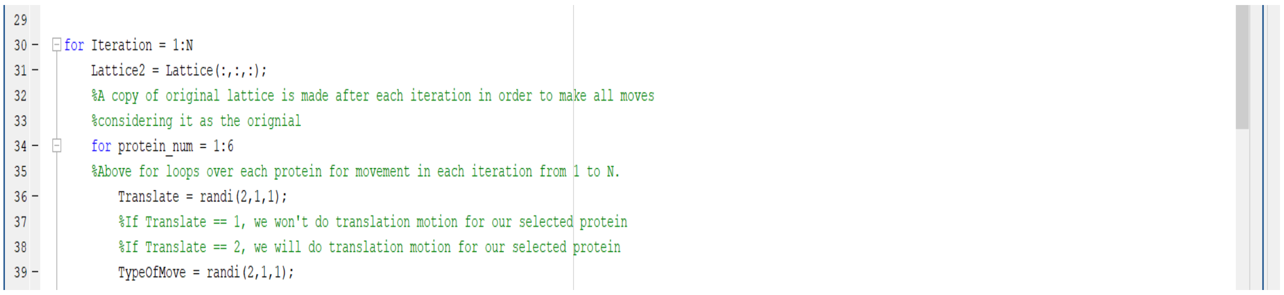
- <u>Energy_C</u> - _This function is used to calculate the total energy of the lattice system when only Residue A and Residue B can interact with Residue A and Residue B of another polymer respectively. Here Residue C and Residue D do not contribute to the intramolecular interaction energy of our lattice.

<u>Code Explanation -</u>



```matlab
close all;clc;
SqDim = 40;
%Defines the dimension of the square lattice

StableEnergyIndex = [];
StableEnergyIndex(1) = 0;
%StableEnergyIndex stores all iterations when it satisfies metropolis criteria and lattice conformation changes.

InteractionEnergy = -1;
%It stores the value of interaction energy between two residues.

N = 1000000;
%N stores the number of Iterations

EInitial = 0;
%It stores the Inital energy of lattice system when no residue interacts, which stands as the relative basis for further calculations of energy for the lattice.

Lattice(:,:,1) = [ 3 4;3 5;3 6;3 7;];
Lattice(:,:,2) = [ 15 23;16 23;17 23;18 23;];
Lattice(:,:,3) = [ 20 15;21 15;22 15;23 15;];
Lattice(:,:,4) = [ 15 17;16 17;17 17;18 17;];
Lattice(:,:,5) = [ 10 5;11 5;12 5;13 5;];
Lattice(:,:,6) = [ 5 12;5 13;5 14;5 15;];
%Above assignment creates a 3D array, with each page (Total - 6 pages for 6 proteins) of the array containing the locations of four residues for that specific protein in fo

InitialSystemEnergy = [];InitialSystemEnergy(1) = EInitial;
%Above array stores the energy of the lattice system and continues to append the Energy of lattice whenever Protein Movements take place.
figure(1);set(gcf, 'Position',  [50, 50, 800, 700]); xh = get(gca,'xlabel') ; p = get(xh,'position');p(2) = p(2)-1 ; set(xh,'position',p);

```

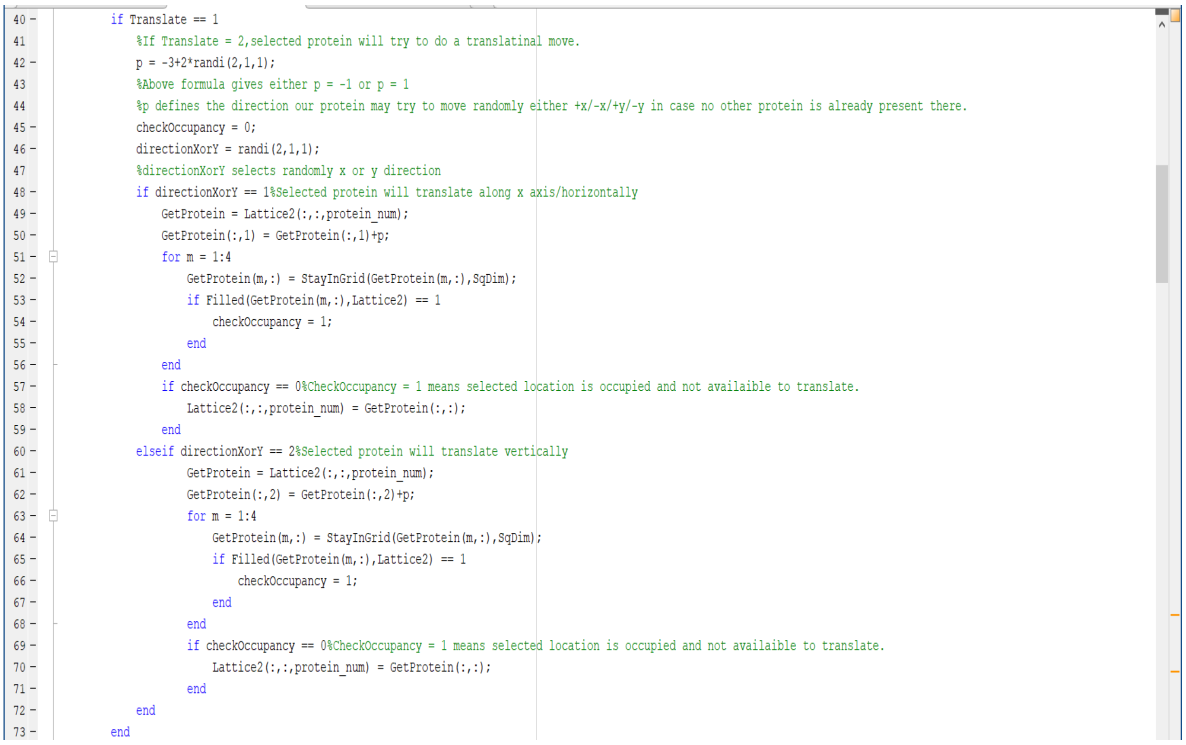
We initialise our variables and set initial location of residues of the given 6 polymers and create a vector named “InitialSystemEnergy” which stores Lattice energy after each Monte Carlo Iteration. Also we create a figure which will be updated after each 2000 moves.



```
29
30 -  for Iteration = 1:N
31 -      Lattice2 = Lattice(:,:,:);
32        %A copy of original lattice is made after each iteration in order to make all moves
33        %considering it as the orignial
34 -      for protein_num = 1:6
35        %Above for loops over each protein for movement in each iteration from 1 to N.
36 -          Translate = randi(2,1,1);
37            %If Translate == 1, we won't do translation motion for our selected protein
38            %If Translate == 2, we will do translation motion for our selected protein
39 -          TypeOfMove = randi(2,1,1);
```

In the above code, we Initialised a new loop which iterates over 10^6 times and in each iteration, we take a copy of the original lattice 3D array for further use.

For each iteration, we again iterate over all polymers and calculate two random integers each having value as 1 or 2 and they are stored as variables named Translate and TypeOfMove.

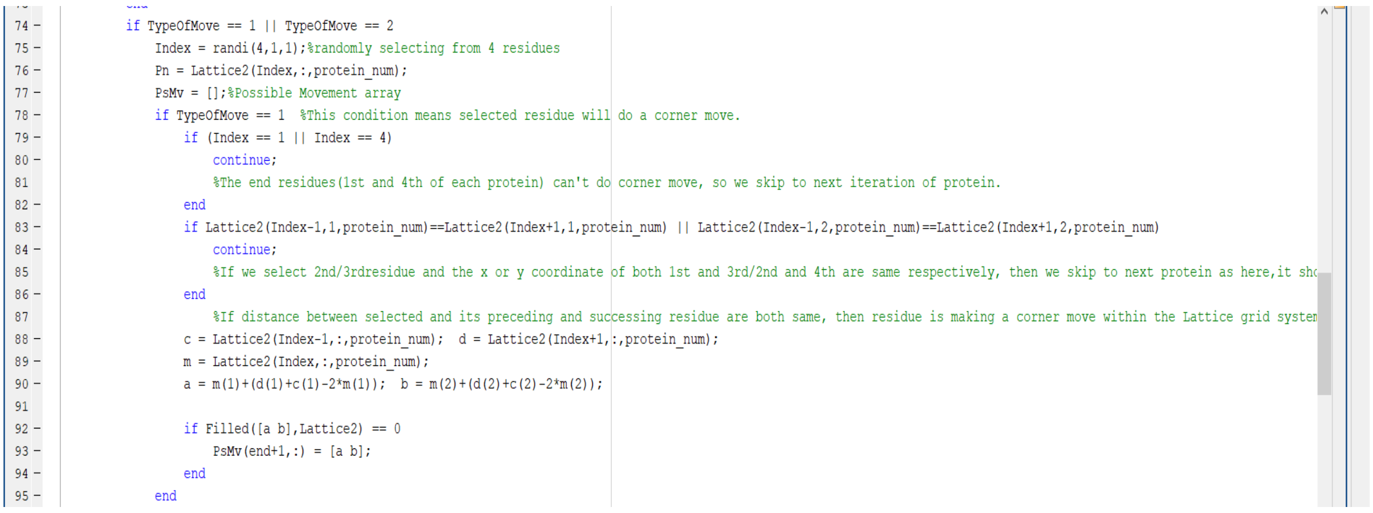


```
40 -          if Translate == 1
41                %If Translate = 2,selected protein will try to do a translatinal move.
42 -              p = -3+2*randi(2,1,1);
43                %Above formula gives either p = -1 or p = 1
44                %p defines the direction our protein may try to move randomly either +x/-x/+y/-y in case no other protein is already present there.
45 -              checkOccupancy = 0;
46 -              directionXorY = randi(2,1,1);
47                %directionXorY selects randomly x or y direction
48 -              if directionXorY == 1%Selected protein will translate along x axis/horizontally
49 -                  GetProtein = Lattice2(:,:,protein_num);
50 -                  GetProtein(:,1) = GetProtein(:,1)+p;
51 -                  for m = 1:4
52 -                      GetProtein(m,:) = StayInGrid(GetProtein(m,:),SqDim);
53 -                      if Filled(GetProtein(m,:),Lattice2) == 1
54 -                          checkOccupancy = 1;
55 -                      end
56 -                  end
57 -                  if checkOccupancy == 0%CheckOccupancy = 1 means selected location is occupied and not availaible to translate.
58 -                      Lattice2(:,:,protein_num) = GetProtein(:,:);
59 -                  end
60 -              elseif directionXorY == 2%Selected protein will translate vertically
61 -                      GetProtein = Lattice2(:,:,protein_num);
62 -                      GetProtein(:,2) = GetProtein(:,2)+p;
63 -                      for m = 1:4
64 -                          GetProtein(m,:) = StayInGrid(GetProtein(m,:),SqDim);
65 -                          if Filled(GetProtein(m,:),Lattice2) == 1
66 -                              checkOccupancy = 1;
67 -                          end
68 -                      end
69 -                      if checkOccupancy == 0%CheckOccupancy = 1 means selected location is occupied and not availaible to translate.
70 -                          Lattice2(:,:,protein_num) = GetProtein(:,:);
71 -                      end
72 -              end
73 -          end
```

Each polymer can translate in any medium in any dir
For each protein, we will do a translation move in horizontal or vertical direction if the value of Translate is 1.

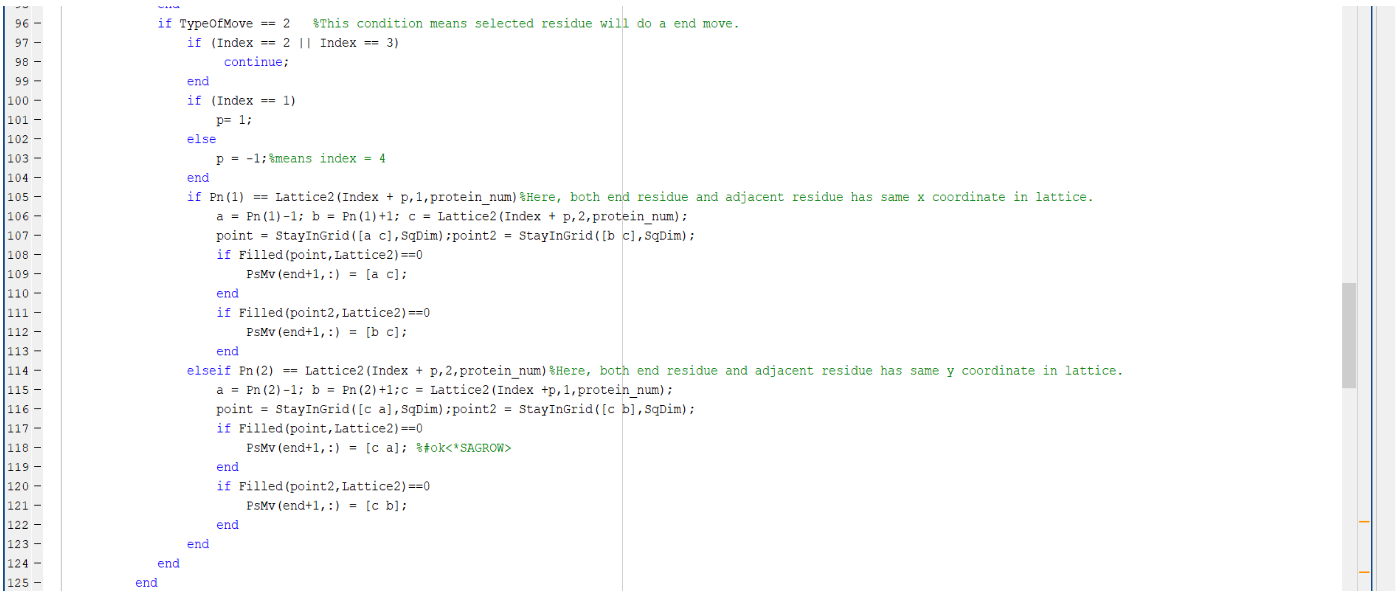
From Line 48 - 60 in the above code, we translate the polymer horizontally where the positive or negative side is decided by a variable p (p takes only +1/-1 value randomly) while checking whether the final location for polymer is vacant or not.

From Line 61 - 73 in the above code, we translate the polymer vertically where the positive or negative side is decided by a variable p (p takes only +1/-1 value randomly) while checking whether the final location for the polymer is vacant or not.



```
74 -        if TypeOfMove == 1 || TypeOfMove == 2
75 -            Index = randi(4,1,1);%randomly selecting from 4 residues
76 -            Pn = Lattice2(Index,:,protein_num);
77 -            PsMv = [];%Possible Movement array
78 -            if TypeOfMove == 1  %This condition means selected residue will do a corner move.
79 -                if (Index == 1 || Index == 4)
80 -                    continue;
81                      %The end residues(1st and 4th of each protein) can't do corner move, so we skip to next iteration of protein.
82 -                end
83 -                if Lattice2(Index-1,1,protein_num)==Lattice2(Index+1,1,protein_num) || Lattice2(Index-1,2,protein_num)==Lattice2(Index+1,2,protein_num)
84 -                    continue;
85                      %If we select 2nd/3rdresidue and the x or y coordinate of both 1st and 3rd/2nd and 4th are same respectively, then we skip to next protein as here,it sho
86 -                end
87                      %If distance between selected and its preceding and successing residue are both same, then residue is making a corner move within the Lattice grid syster
88 -                c = Lattice2(Index-1,:,protein_num);  d = Lattice2(Index+1,:,protein_num);
89 -                m = Lattice2(Index,:,protein_num);
90 -                a = m(1)+(d(1)+c(1)-2*m(1));  b = m(2)+(d(2)+c(2)-2*m(2));
91
92 -                if Filled([a b],Lattice2) == 0
93 -                    PsMv(end+1,:) = [a b];
94 -                end
95 -            end
```

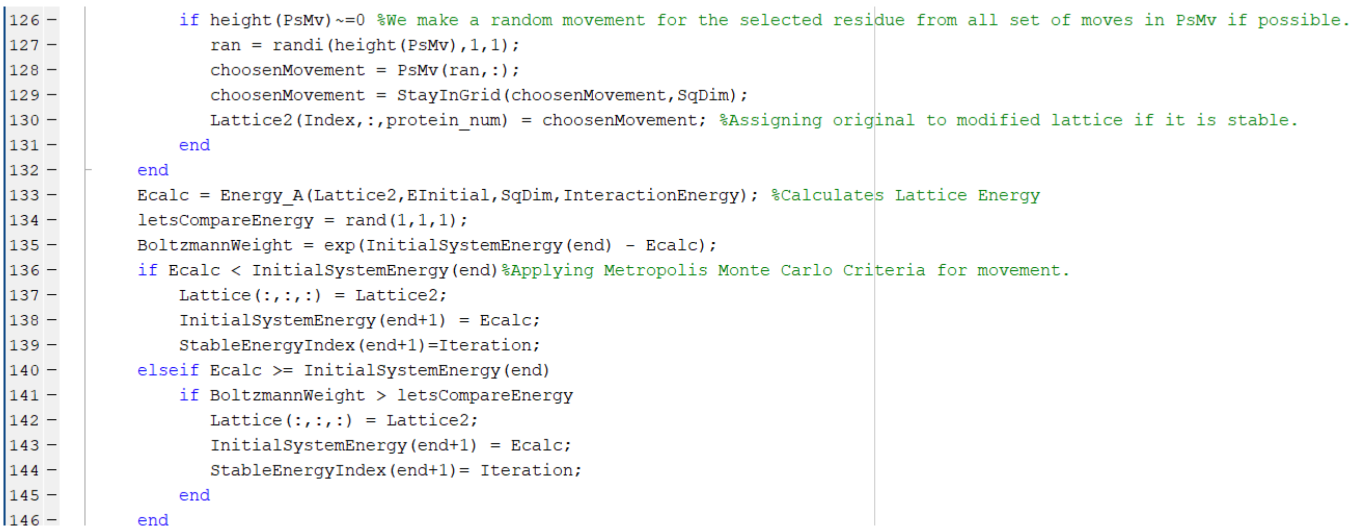
In Line 74, we proceed to do a corner or end move, after which we randomly select a residue randomly of the polymer over which we are iterating.
From Line 78 - 95, We seek the possibility of a corner move for the specific residue as per conditions and formulae discussed above in step 3, only if the value of variable “TypeOfMove” is 1. We store possible movements in the array named “PsMv” if it can move to those positions.



```
96 -            if TypeOfMove == 2   %This condition means selected residue will do a end move.
97 -                if (Index == 2 || Index == 3)
98 -                     continue;
99 -                end
100 -               if (Index == 1)
101 -                   p= 1;
102 -               else
103 -                   p = -1;%means index = 4
104 -               end
105 -               if Pn(1) == Lattice2(Index + p,1,protein_num)%Here, both end residue and adjacent residue has same x coordinate in lattice.
106 -                   a = Pn(1)-1; b = Pn(1)+1; c = Lattice2(Index + p,2,protein_num);
107 -                   point = StayInGrid([a c],SqDim);point2 = StayInGrid([b c],SqDim);
108 -                   if Filled(point,Lattice2)==0
109 -                       PsMv(end+1,:) = [a c];
110 -                   end
111 -                   if Filled(point2,Lattice2)==0
112 -                       PsMv(end+1,:) = [b c];
113 -                   end
114 -               elseif Pn(2) == Lattice2(Index + p,2,protein_num)%Here, both end residue and adjacent residue has same y coordinate in lattice.
115 -                   a = Pn(2)-1; b = Pn(2)+1;c = Lattice2(Index +p,1,protein_num);
116 -                   point = StayInGrid([c a],SqDim);point2 = StayInGrid([c b],SqDim);
117 -                   if Filled(point,Lattice2)==0
118 -                       PsMv(end+1,:) = [c a]; %#ok<*SAGROW>
119 -                   end
120 -                   if Filled(point2,Lattice2)==0
121 -                       PsMv(end+1,:) = [c b];
122 -                   end
123 -               end
124 -           end
125 -       end
```

From Line 96 - 125, We seek the possibility of an end move for the selected specific residue as per conditions and formulae discussed above in step 3, only if the value of variable “TypeOfMove” is 2. We store possible movements in the array named “PsMv” if it can move to those positions.

Here, We use the Filled function to check whether the location for a possible move is filled or not and add those movements to array “PsMv” accordingly.
The p variable manages to change the cases for index selected as 1 or 4 accordingly.
Different cases used in Line 105 and Line 114 were discussed in End Movement case of Step 3.

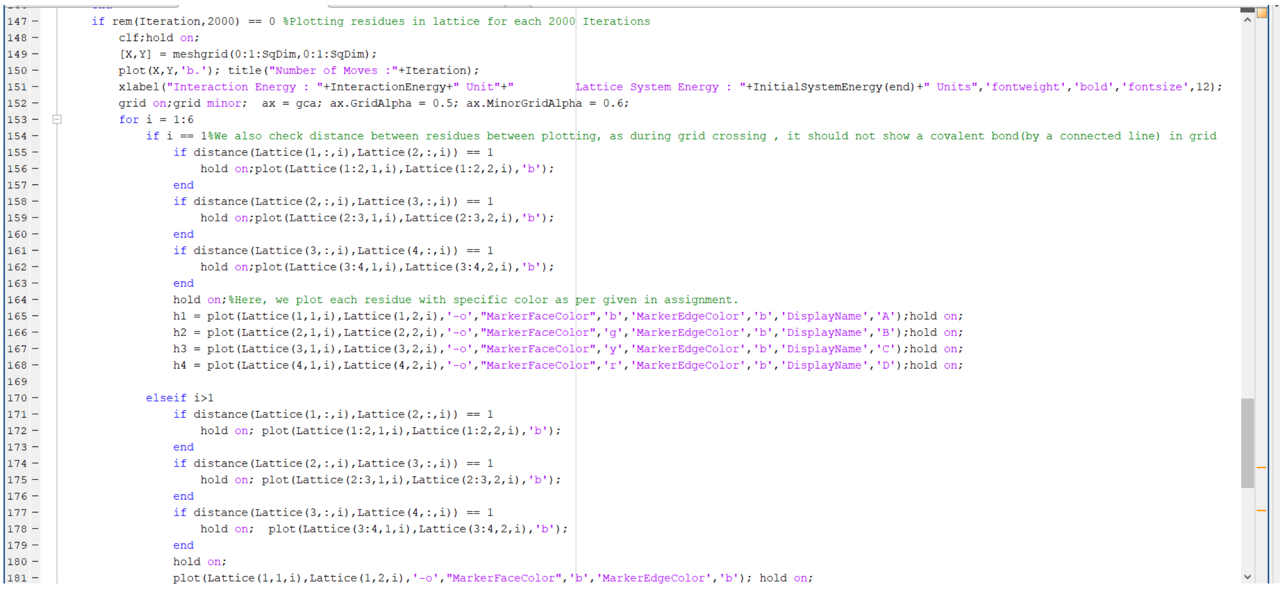


```
126 -            if height(PsMv)~=0 %We make a random movement for the selected residue from all set of moves in PsMv if possible.
127 -                ran = randi(height(PsMv),1,1);
128 -                choosenMovement = PsMv(ran,:);
129 -                choosenMovement = StayInGrid(choosenMovement,SqDim);
130 -                Lattice2(Index,:,protein_num) = choosenMovement; %Assigning original to modified lattice if it is stable.
131 -            end
132 -        end
133 -        Ecalc = Energy_A(Lattice2,EInitial,SqDim,InteractionEnergy); %Calculates Lattice Energy
134 -        letsCompareEnergy = rand(1,1,1);
135 -        BoltzmannWeight = exp(InitialSystemEnergy(end) - Ecalc);
136 -        if Ecalc < InitialSystemEnergy(end)%Applying Metropolis Monte Carlo Criteria for movement.
137 -            Lattice(:,:,:) = Lattice2;
138 -            InitialSystemEnergy(end+1) = Ecalc;
139 -            StableEnergyIndex(end+1)=Iteration;
140 -        elseif Ecalc >= InitialSystemEnergy(end)
141 -            if BoltzmannWeight > letsCompareEnergy
142 -                Lattice(:,:,:) = Lattice2;
143 -                InitialSystemEnergy(end+1) = Ecalc;
144 -                StableEnergyIndex(end+1)= Iteration;
145 -            end
146 -        end
```

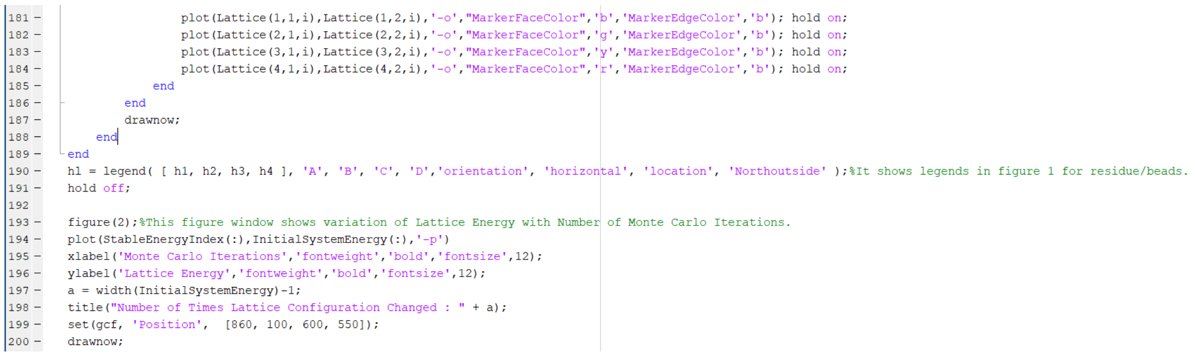
From Line 126- 131 , we are choosing a random movement for our selected residue from the set of movements stored in “PsMv”Array that satisfy conditions required to make the move.
After that the change in location of residue is made to the copy of the original lattice we copied before and then the same is repeated for residue of the next polymer in the same iteration.

Once we have repeated the above procedure for all six polymers and made changes to the copy of original lattice, we check whether the copy of the lattice satisfies conditions of Metropolis criteria for monte carlo simulation as given from Line 133 - 146.

The function “Energy_A” can be replaced with other functions “Energy_B”, “Energy_C” as per the combination of interactions given in the problem statement.



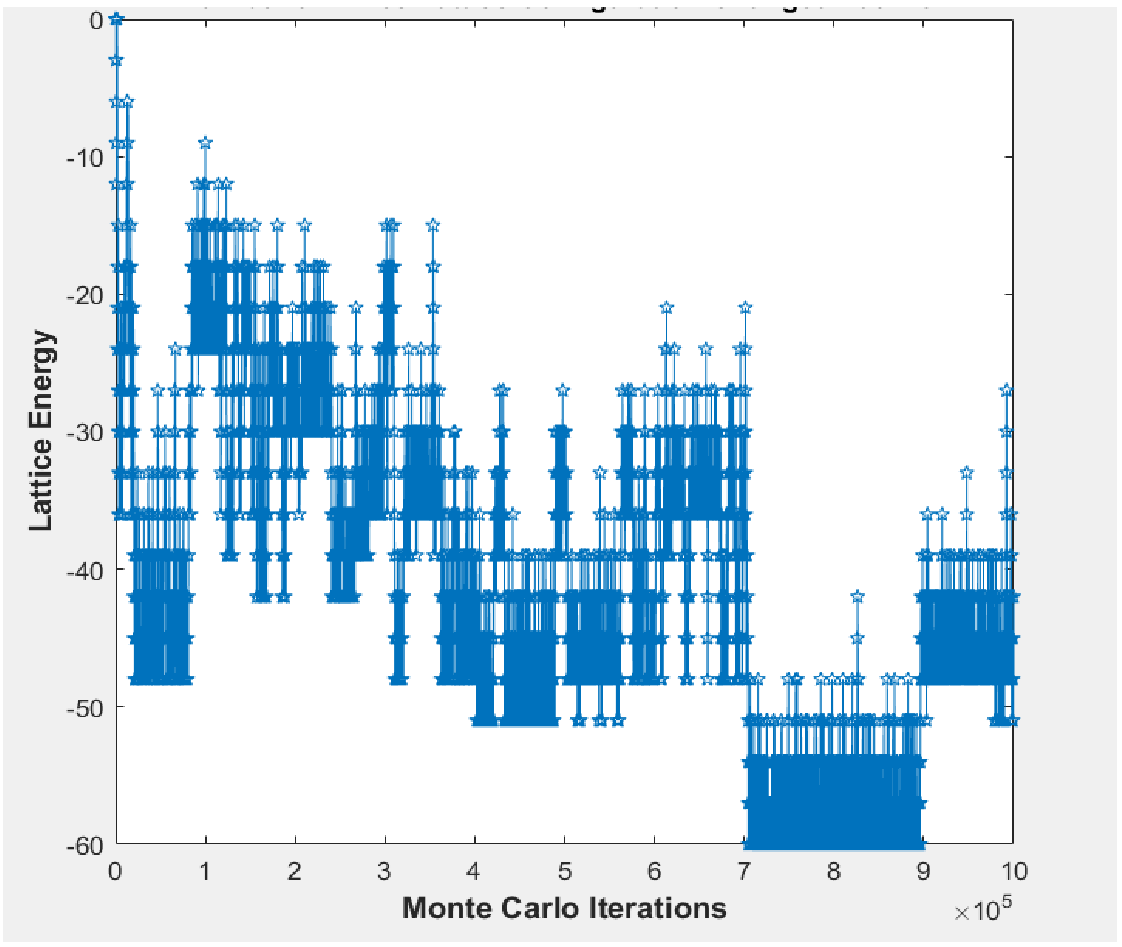
```
147 -        if rem(Iteration,2000) == 0 %Plotting residues in lattice for each 2000 Iterations
148 -            clf;hold on;
149 -            [X,Y] = meshgrid(0:1:SqDim,0:1:SqDim);
150 -            plot(X,Y,'b.'); title("Number of Moves :"+Iteration);
151 -            xlabel("Interaction Energy : "+InteractionEnergy+" Unit"+"          Lattice System Energy : "+InitialSystemEnergy(end)+" Units",'fontweight','bold','fontsize',12);
152 -            grid on;grid minor;  ax = gca; ax.GridAlpha = 0.5; ax.MinorGridAlpha = 0.6;
153 -            for i = 1:6
154 -                if i == 1%We also check distance between residues between plotting, as during grid crossing , it should not show a covalent bond(by a connected line) in grid
155 -                    if distance(Lattice(1,:,i),Lattice(2,:,i)) == 1
156 -                        hold on;plot(Lattice(1:2,1,i),Lattice(1:2,2,i),'b');
157 -                    end
158 -                    if distance(Lattice(2,:,i),Lattice(3,:,i)) == 1
159 -                        hold on;plot(Lattice(2:3,1,i),Lattice(2:3,2,i),'b');
160 -                    end
161 -                    if distance(Lattice(3,:,i),Lattice(4,:,i)) == 1
162 -                        hold on;plot(Lattice(3:4,1,i),Lattice(3:4,2,i),'b');
163 -                    end
164 -                    hold on;%Here, we plot each residue with specific color as per given in assignment.
165 -                    h1 = plot(Lattice(1,1,i),Lattice(1,2,i),'-o',"MarkerFaceColor",'b','MarkerEdgeColor','b','DisplayName','A');hold on;
166 -                    h2 = plot(Lattice(2,1,i),Lattice(2,2,i),'-o',"MarkerFaceColor",'g','MarkerEdgeColor','b','DisplayName','B');hold on;
167 -                    h3 = plot(Lattice(3,1,i),Lattice(3,2,i),'-o',"MarkerFaceColor",'y','MarkerEdgeColor','b','DisplayName','C');hold on;
168 -                    h4 = plot(Lattice(4,1,i),Lattice(4,2,i),'-o',"MarkerFaceColor",'r','MarkerEdgeColor','b','DisplayName','D');hold on;
169
170 -                elseif i>1
171 -                    if distance(Lattice(1,:,i),Lattice(2,:,i)) == 1
172 -                        hold on; plot(Lattice(1:2,1,i),Lattice(1:2,2,i),'b');
173 -                    end
174 -                    if distance(Lattice(2,:,i),Lattice(3,:,i)) == 1
175 -                        hold on; plot(Lattice(2:3,1,i),Lattice(2:3,2,i),'b');
176 -                    end
177 -                    if distance(Lattice(3,:,i),Lattice(4,:,i)) == 1
178 -                        hold on;  plot(Lattice(3:4,1,i),Lattice(3:4,2,i),'b');
179 -                    end
180 -                    hold on;
181 -                    plot(Lattice(1,1,i),Lattice(1,2,i),'-o',"MarkerFaceColor",'b','MarkerEdgeColor','b'); hold on;
```



```
                plot(Lattice(1,1,i),Lattice(1,2,i),'-o',"MarkerFaceColor",'b','MarkerEdgeColor','b'); hold on;
                plot(Lattice(2,1,i),Lattice(2,2,i),'-o',"MarkerFaceColor",'g','MarkerEdgeColor','b'); hold on;
                plot(Lattice(3,1,i),Lattice(3,2,i),'-o',"MarkerFaceColor",'y','MarkerEdgeColor','b'); hold on;
                plot(Lattice(4,1,i),Lattice(4,2,i),'-o',"MarkerFaceColor",'r','MarkerEdgeColor','b'); hold on;
            end
        end
        drawnow;
    end
end
hl = legend( [ h1, h2, h3, h4 ], 'A', 'B', 'C', 'D','orientation', 'horizontal', 'location', 'Northoutside' );%It shows legends in figure 1 for residue/beads.
hold off;

figure(2);%This figure window shows variation of Lattice Energy with Number of Monte Carlo Iterations.
plot(StableEnergyIndex(:),InitialSystemEnergy(:),'-p')
xlabel('Monte Carlo Iterations','fontweight','bold','fontsize',12);
ylabel('Lattice Energy','fontweight','bold','fontsize',12);
a = width(InitialSystemEnergy)-1;
title("Number of Times Lattice Configuration Changed : " + a);
set(gcf, 'Position',  [860, 100, 600, 550]);
drawnow;
```

As the code takes long time to execute dynamic updates after every step, we proceed to make dynamic update each time after 2000 iterations,
After completing 2000 iterations, we make updates to our figure 1 which shows the lattice system in the form of a symmetric grid as discussed before.
Also, we create a figure representing the Energy of Lattice system vs Number of Monte Carlo Steps.

Results:



After every Monte Carlo Iteration , the new energy of the protein was
calculated and recorded.
Above figure shows that starting with 6 unfolded polymers and allowing it to undergo the natural folding process of protein folding, its energy swiftly decreases and then gradually levels off, as expected. Sometimes in the middle range of Monte Carlo iteration steps, the energy can be seen to rise to a local maximum value suggesting that the protein had been in a false local energy minimum, shallow enough to be able to escape.

Conclusion :

The energy is finally relatively stable around a specific value at the end of all iterations , indicating a significantly deeper energy well, and thus suggesting that the protein may be in a `native' molecular state for the given conditions.
The above Lattice Energy vs Monte Carlo Iteration graph also shows characteristics of Interaction Energy between Residues, as the volatility in the graph is a direct measure of how easily residual interactions break off, which signifies low Interaction energy at a constant temperature and volume.
Whereas low volatility or less jumps in the graph gives a good estimate of Interaction energy between residues in which high interactions between residues does not allow the covalent bonds between residues to break off easily.