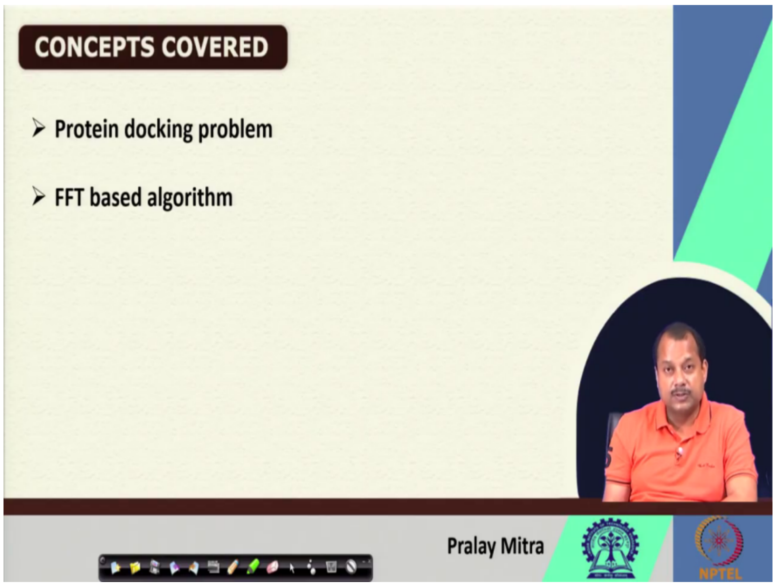
Algorithm for Protein Modelling and Engineering
Professor Pralay Mitra
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture-09
Implementation Details

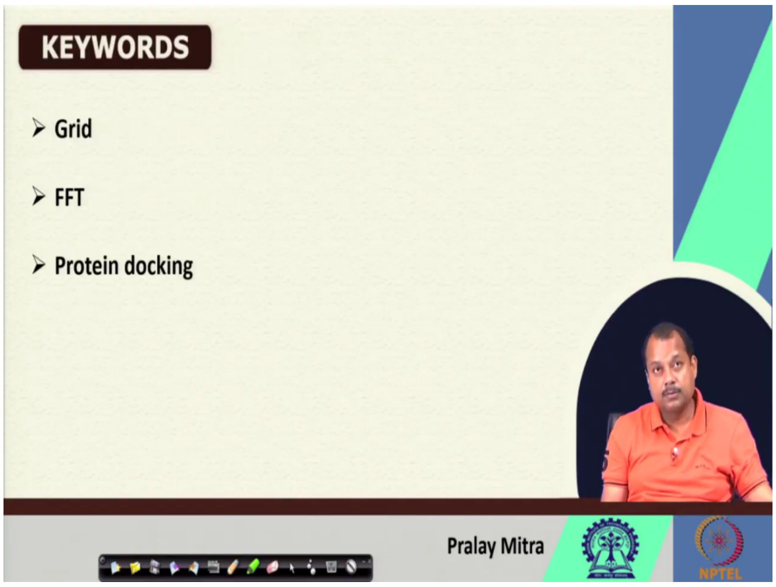
Welcome back. Today, I am planning to give you the details on the implementation of the algorithm that we have discussed in the last two lectures.

(Refer Slide Time: 00:27)


CONCEPTS COVERED

- Protein docking problem
- FFT based algorithm

Pralay Mitra

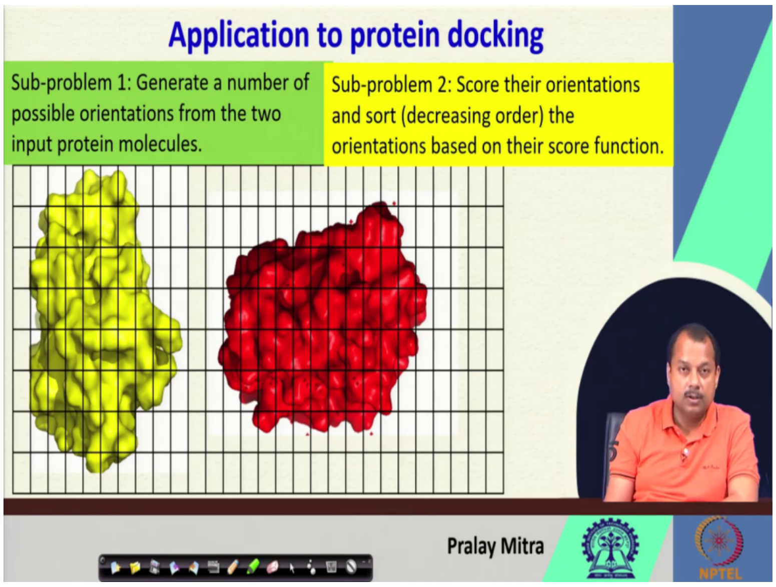

KEYWORDS

- Grid
- FFT
- Protein docking

Pralay Mitra

That's why the coverage I mentioned as Protein docking problem, but it will be the implementation details for the algorithms that we discussed in the last lectures.

(Refer Slide Time: 00:43)



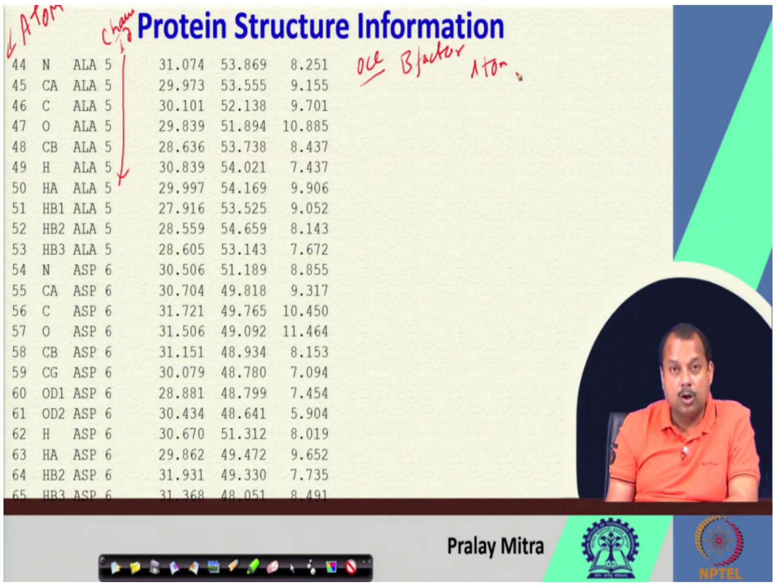
Application to protein docking

Sub-problem 1: Generate a number of possible orientations from the two input protein molecules.

Sub-problem 2: Score their orientations and sort (decreasing order) the orientations based on their score function.

In the last class, I mentioned that there are two subproblems - generating the different orientations in a protein docking problem, and scoring them. I also mentioned that we prefer the integrated scoring, where during the generation phase (generation of all possible orientations), I calculate their score function. Next, based upon that score function and a threshold value, I decide whether to take that particular orientation or reject that particular orientation. Now we go for the implementation details.

(Refer Slide Time: 01:49)



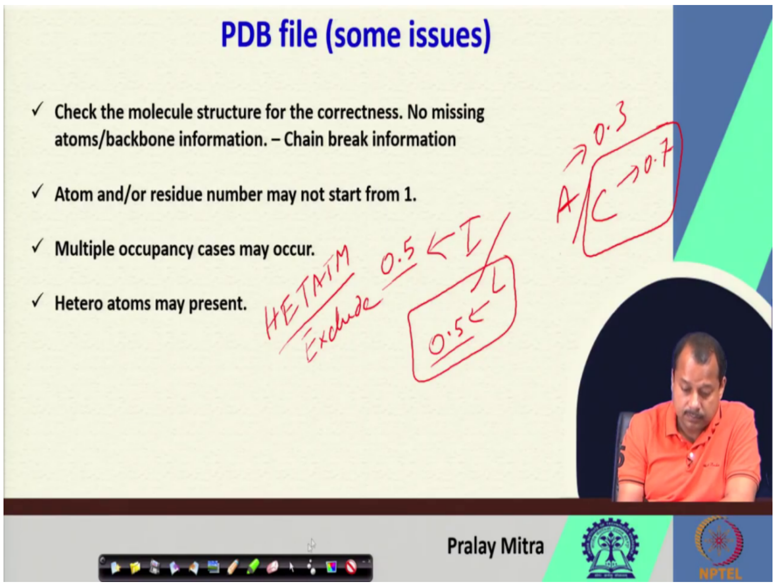
Protein Structure Information

```
↓Atom      Chain
           ID
44  N   ALA 5 |    31.074  53.869   8.251    occ  Bfactor  Atom
45  CA  ALA 5 |    29.973  53.555   9.155
46  C   ALA 5 |    30.101  52.138   9.701
47  O   ALA 5 |    29.839  51.894  10.885
48  CB  ALA 5 |    28.636  53.738   8.437
49  H   ALA 5 ↓    30.839  54.021   7.437
50  HA  ALA 5      29.997  54.169   9.906
51  HB1 ALA 5      27.916  53.525   9.052
52  HB2 ALA 5      28.559  54.659   8.143
53  HB3 ALA 5      28.605  53.143   7.672
54  N   ASP 6      30.506  51.189   8.855
55  CA  ASP 6      30.704  49.818   9.317
56  C   ASP 6      31.721  49.765  10.450
57  O   ASP 6      31.506  49.092  11.464
58  CB  ASP 6      31.151  48.934   8.153
59  CG  ASP 6      30.079  48.780   7.094
60  OD1 ASP 6      28.881  48.799   7.454
61  OD2 ASP 6      30.434  48.641   5.904
62  H   ASP 6      30.670  51.312   8.019
63  HA  ASP 6      29.862  49.472   9.652
64  HB2 ASP 6      31.931  49.330   7.735
65  HB3 ASP 6      31.368  48.051   8.491
```

When we are working with a protein structure undoubtedly our database will be Protein Data Bank that we discussed. I also mentioned that it is not the complete entry by in each row it is truncated column-wise. On the left-hand side, it starts with the ATOM keyword. The chain information intentionally I omitted. Here there will be chain information chain ID and after this one there will be occupancy information, B factor information, optionally atom name. Currently, that is not of interest to us, that is why I removed it.

While we are working with the implementation of the algorithm, incomplete structure information becomes a problem for us. Therefore, you have to be very careful about parsing or reading the information from this file. I am going to discuss a few known issues that might occur. Apart from those you may be encountering some other issues that you can report to me.

(Refer Slide Time: 03:36)



PDB file (some issues)

- ✓ Check the molecule structure for the correctness. No missing atoms/backbone information. – Chain break information
- ✓ Atom and/or residue number may not start from 1.
- ✓ Multiple occupancy cases may occur.
- ✓ Hetero atoms may present.

The first issue is to check the molecule structure for correctness. No missing atoms or backbone information chain break information etc. It may possible that in your structure side-chain atoms are missing or some backbone like C Carboxyl C or say Amino N information is not there - it is missing. So you have to check during your parsing stage whether all the atoms are present or not, if not, then you have to fix that if it is a small or minor mistake or minor omission.

But, if it is a major one like one Amino acid has missed completely, then perhaps we do not have any other option rather we have to discard that entry. We cannot work with that. Why can we not work with that one? Let us illustrate with an example, remember what we have done in the surface matching - we digitize the protein molecule put in the grid cell and look for the overlapping information.

Now, if one surface residue is missing because of some mistake maybe at the experimental level or refinement level, then the surface match will be inconsistent. That leads to an incorrect calculation of the surface matching score.

Therefore, you have to discard those entries unless otherwise, it is not fixed. Some techniques are there which can fix it. Mostly, they are time-consuming. We are not discussing that fixing problem since we have plenty of good structures. We shall focus on good or correct structures. But if the missing information is known then better to avoid or ignore that structure.

The first atom or residue number may not start with 1. I mentioned this in my previous lecture. Sometimes it may start with some negative number sometimes it may start with a positive number like 567. My previous example also starts the atom number at 44, the residue number starts with 5.

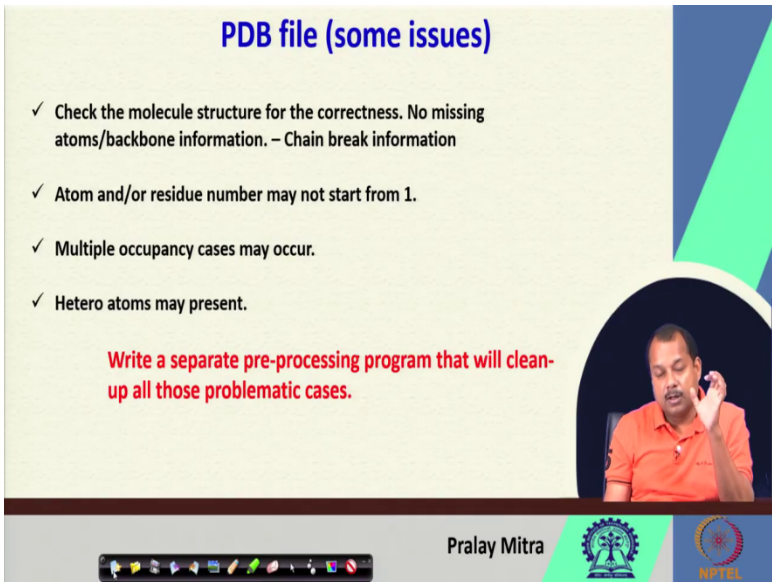
If you are planning to store the atom, and amino acid information in an array then irrespective of the language, either the array index will start with 0 in the case of C or 1 in the case of languages like Fortran or Pascal. So, if the atom or amino acid number is starting with anything other than 0 or 1, then you cannot use that residue number or the atom number as your index for the array. Therefore, have your indexes and store the atom number and the residue number as an integer type in a data structure.

Multiple occupancy cases may occur which says that during the experiment or the refinement stage sometimes there is confusion between multiple amino acids, for example, confusion between Isoleucine and Leucine, between Alanine or Cysteine. It may occur during the experimental technique, then the authors, I mean the experimentalists deposit both the information in the PDB with a probability value. For example, probability 0.5 for Isoleucine and probability 0.5 for Leucine; or probability 0.3 for Alanine and 0.7 for Cysteine, definitely all of them will sum up to probability 1. This means one atom (or residue) is there, but I am not sure which atom (or residue) it is.

Under this circumstance both the residues can not exist at the same position, so you have to pick one. In this case, perhaps it is easy - you can pick with the higher probability. For equal probability like 0.5 and 0.5, you may break the tie arbitrarily and pick the first one say.

Another issue lies with the Hetero atoms including water molecules that you need to exclude. There may be a separate section for the heteroatom or it may be a part of a chain, all those entries that start with HET can be excluded for the time being. Similarly, you may exclude non-standard amino acids.

(Refer Slide Time: 10:56)



PDB file (some issues)

- ✓ Check the molecule structure for the correctness. No missing atoms/backbone information. – Chain break information
- ✓ Atom and/or residue number may not start from 1.
- ✓ Multiple occupancy cases may occur.
- ✓ Hetero atoms may present.

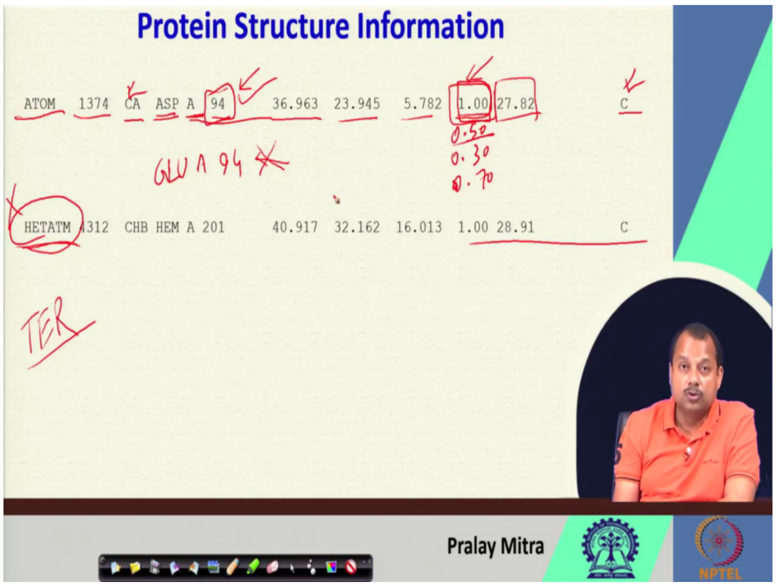
Write a separate pre-processing program that will clean-up all those problematic cases.

Pralay Mitra

To handle these situations, my suggestion is that write a separate pre-processing program that will clean up all those messes or all those problematic cases or all those issues from the PDB and give you the refined structure file on which you can work. Otherwise, whether the program fails because of your coding mistake or because of your wrong calculation, or because of your wrong protein structure file you will not be able to judge. It is always a good idea to have some pre-processing algorithm and almost all the software refine the protein structures using their algorithm. Now in your library file, apart from 3 letter amino acid to 1 letter amino acid conversion and vice versa, you have to write one more program to clean up all those issues from your PDB.

Another problem that may arise is that you will see say residue number 94 is not changing. Assuming 94th residue is aspartic acid (ASP) Glutamic acid may also assume 94 residue number when corresponding to 94th residue multiple residues occurs. Therefore, two different residue names, but the same residue number. Is it a mistake? For that case, look for their occupancy you will see their occupancy adds up to 1. It indicates that at the 94th position authors are not confident that which residue it is, it may be aspartic acid, or it may be glutamic acid.

(Refer Slide Time: 12:33)

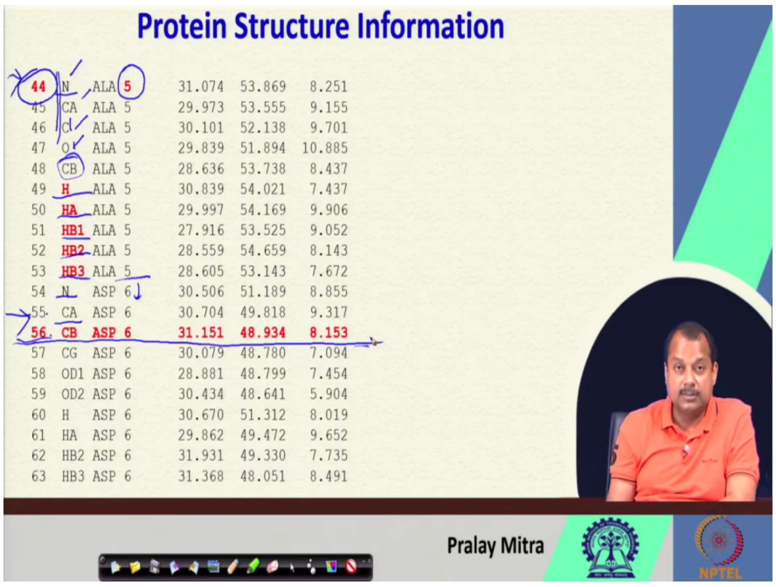


Protein Structure Information

```
ATOM  1374  CA  ASP A  94    36.963  23.945   5.782  1.00 27.82           C
                GLU A 94                               0.50
                                                       0.30
                                                       0.70
HETATM 1312  CHB HEM A 201    40.917  32.162  16.013  1.00 28.91           C
TER
```

That’s why I said that you have to pick one maybe the first one or maybe with the higher occupancy. Usually, the highest occupancy occurs fast, so my suggestion will be to pick the first one and discard or ignore the rest. There may be some Hetero atom entry mostly the hetero atom entries occurs at the end after the entry of the atom is over and they are terminated by TER. But it may not be the case, hetero atom entry may occur very much inside the atomic entry. I mean the atom then hetero atom after that one TER will occur and if it is the case for the time being our suggestion is that you exclude or you ignore that hetero atom entry completely.

(Refer Slide Time: 15:31)


Protein Structure Information

```
44  N   ALA 5      31.074  53.869   8.251
45  CA  ALA 5      29.973  53.555   9.155
46  C   ALA 5      30.101  52.138   9.701
47  O   ALA 5      29.839  51.894  10.885
48  CB  ALA 5      28.636  53.738   8.437
49  H   ALA 5      30.839  54.021   7.437
50  HA  ALA 5      29.997  54.169   9.906
51  HB1 ALA 5      27.916  53.525   9.052
52  HB2 ALA 5      28.559  54.659   8.143
53  HB3 ALA 5      28.605  53.143   7.672
54  N   ASP 6      30.506  51.189   8.855
55  CA  ASP 6      30.704  49.818   9.317
56  CB  ASP 6      31.151  48.934   8.153
57  CG  ASP 6      30.079  48.780   7.094
58  OD1 ASP 6      28.881  48.799   7.454
59  OD2 ASP 6      30.434  48.641   5.904
60  H   ASP 6      30.670  51.312   8.019
61  HA  ASP 6      29.862  49.472   9.652
62  HB2 ASP 6      31.931  49.330   7.735
63  HB3 ASP 6      31.368  48.051   8.491
```

Look at the structure, I am picking the blue ink so that I can discriminate. I mention the atom number may not start with 1 or 0 which may be required for your program implementation in C, Python, C++, Fortran, Pascal, or any other language. Better you define array index and store this information for processing purposes.

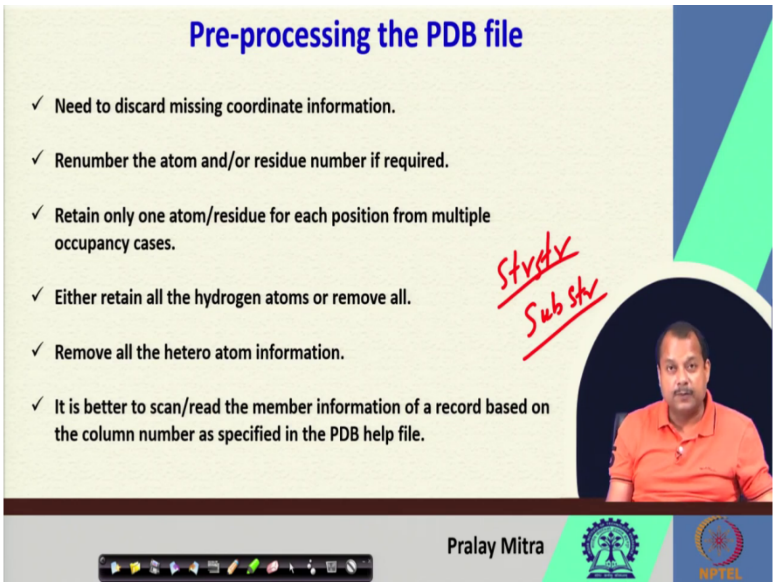
Regarding hydrogen atoms, mostly they are absent in the crystal structures but is present in NMR structures. Thus, either you include all the hydrogens or exclude them altogether. If hydrogen atoms are partly present then use some software to incorporate the hydrogen atoms. We shall discuss this part later. However, if you are not dealing with the hydrogen atoms for the score calculation, better remove all the hydrogen atoms.

By this time you are familiar with the way the PDB stores the residue and atom information - amino N, C alpha, then Carboxyl C, then Carboxyl O - this is part of a backbone or main chain. After that one side starts. Since it is Alanine, only one CB, C, and then H are there. If it is not Alanine other side chain information will be appended. After that residue number will change from, atom number will keep on incrementing. Again it will start from the backbone amino N, after that CA, now the C and O are missing here that you will note and when it is missing there may be the change in atom number after 55 two atoms are missing, so it may start from 58 or it may start at 56. I do not know. But there is no information that it is missing. Thus, again I suggest you should have one pre-processing algorithm which will pre-process all these issues and fixes those before we do the actual algorithmic implementation.

These are some of the issues you may encounter some errors like segmentation fault in C/C++ programming language. But whatever may be the error you need to understand and fix it during your pre-processing stage. Make sure that your input is correct and for that if required you have one pre-processing program which will pre-process your protein and clean up the input protein file. PDB houses a lot of protein structures, if those corner cases are removed or ignored then also you will have plenty of structures to work with.

The list of errors that you need to handle is not complete. You may come up with some other idea you can keep on adding that one. First, you need to discard missing coordinate information straight away, that is my suggestion. You can fix it, but otherwise, you can discard that one also. Renumber, the atom and or residue number if required for your indexing purpose this index is dealing with your array index where you are storing the information of atoms and residue. Retain only one atom or residue for each position from multiple occupancy cases. As I mentioned it is better to keep the first one. Well if in your PDB file you are not able to identify the occupancy column then you may not think about it. Mostly it is present if it is an experimental structure. For computational structure, it may not be present.

(Refer Slide Time: 20:07)



Pre-processing the PDB file

- ✓ Need to discard missing coordinate information.
- ✓ Renumber the atom and/or residue number if required.
- ✓ Retain only one atom/residue for each position from multiple occupancy cases.
- ✓ Either retain all the hydrogen atoms or remove all.
- ✓ Remove all the hetero atom information.
- ✓ It is better to scan/read the member information of a record based on the column number as specified in the PDB help file.

Study
Sub Str

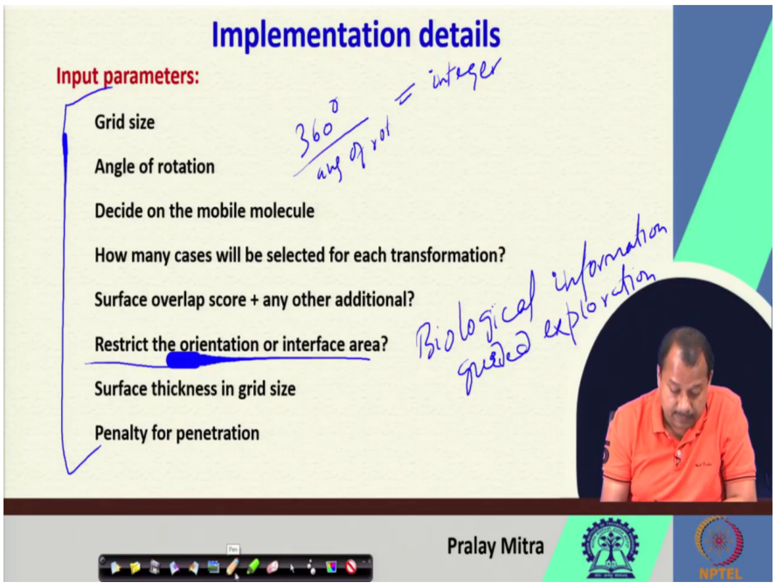
Pralay Mitra

Either retain all the hydrogen atoms or remove all, sometimes you will see that hydrogen atoms are required, if required then you add missing hydrogen atoms using some software like VMD. If not required you remove all the hetero atom information mostly we do not need that information.

It is better to scan or read the member information of a record based on the column number as specified in the PDB help file. Be careful if you are using C/C++ programming language or any language that uses a space as a deliminator. At times, between two fields you may not get a space. If it is the case, then it will create some problems. Better you look at the PDB help file and pick column-based information. if you are implementing in C programming language then use substring or *strstr()* function to read the particular field.

Assuming that the PDB file is preprocessed, we get that clean protein structure file. After that, we go for the implementation of that FFT-based generation and the scoring algorithm that we discussed in the last lecture.

(Refer Slide Time: 23:57)



Implementation details

Input parameters:

- Grid size
- Angle of rotation $\frac{360^0}{\text{ang of rot}} = \text{integer}$
- Decide on the mobile molecule
- How many cases will be selected for each transformation?
- Surface overlap score + any other additional?
- Restrict the orientation or interface area? Biological information guided exploration
- Surface thickness in grid size
- Penalty for penetration

Pralay Mitra

Now we need to decide about a few input parameters. The first one is the grid size that is very easy to calculate using the X_{min}, Y_{min}, Z_{min}, X_{max}, Y_{max}, and Z_{max}. Also, it is a good idea if along all the axes you add some extra space like some percent plus delta or something. Next, you decide on the angle of rotation. I also mentioned that this angle of rotation if I divide 360 by angle of rotation then it will be interior value so that the number of steps for the rotation will be an integer in nature.

Next, decide on the mobile molecule for computational efficiency. Truly, the translation and rotation of the small molecule are computationally less expensive. Thus my suggestion is that at the beginning you look at the two protein molecules consider the smaller molecule (in terms of the number of atoms) as a mobile molecule and which one is the larger you consider that as a static molecule.

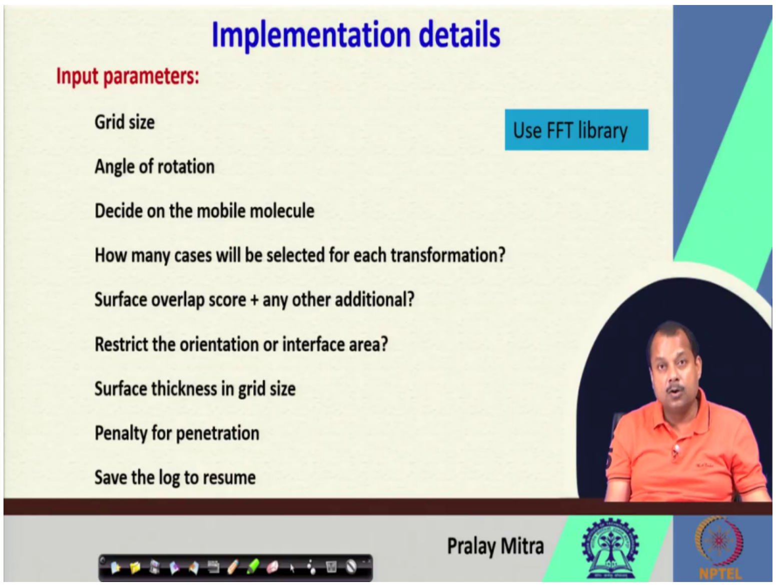
Question - how many cases will be selected for each transformation? I got one orientation, using the Fast Fourier transform I am getting one peak. From that peak, whether the peak only will be considered or apart from that peak three other cases will be considered how many such cases you have to keep that you have to decide.

Next decision on the score function – will it be only surface overlap score or any additional score. We discussed the surface matching wherein one grid cell if there is overlapping from molecule A and molecule B then we will compute that one or we will consider that one. Apart from that one whether any other physicochemical features, we wish to incorporate? I mean, it is not only the overlapping, but whether the overlapping is between one base amino acid and other acid amino acid or they are hydrophilic and hydrophilic? Will those cases also be considered as part of the score function? You need to decide that. If you consider that one then definitely you will get an edge, I mean the advantage compared to if you just consider the surface overlap. These things you have to carefully think and accordingly you have to incorporate.

Next is the restriction on the orientation of the interface area. It is crucial in the context of biological information. A biologist may give you some additional information in terms of the binding site or not binding site. For binding site information when you can accept those decoys with more confidence for not binding site information you may add some penalty to reject or block those cases. The blocking or putting some restrictions may be suggested by the biologists. Then you can restrict those or allow only others. This particular approach is sometimes called guided - you are guiding the exploration.

Surface thickness in the grid size (α,β,γ) and penalty for penetration is also required to be decided. Whether will it be a high negative number like -100 or -200 or a small positive number? In your program development, you allow the user to decide on this, but at the same time, you should have some default value if the user is not that competent and does not provide any such information.

(Refer Slide Time: 30:28)



Implementation details

Input parameters:

Grid size

Angle of rotation

Decide on the mobile molecule

How many cases will be selected for each transformation?

Surface overlap score + any other additional?

Restrict the orientation or interface area?

Surface thickness in grid size

Penalty for penetration

Save the log to resume

Use FFT library

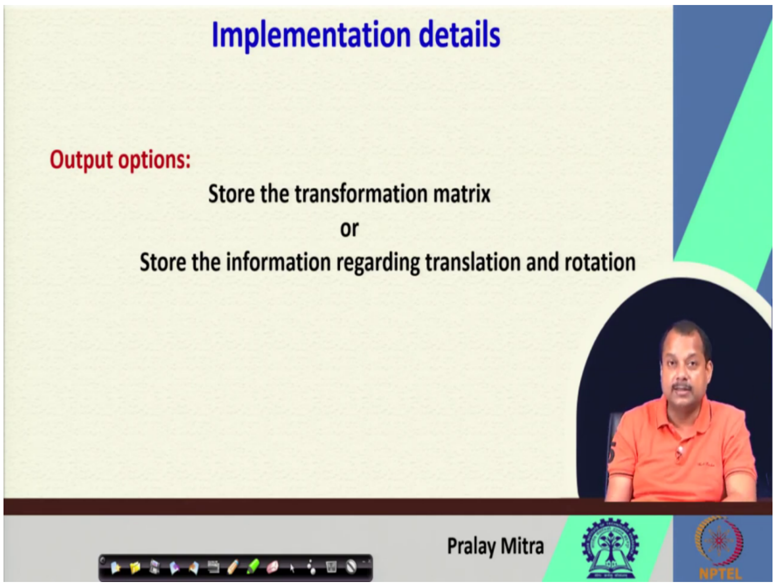
Pralay Mitra

Thus, a set of default parameters are required in your program which will be called if a user does not know anything or set any parameters. Also as a user, if you set you can not set some random values. It must be logical and based upon your understanding and the justification must be there for this one.

Finally, there should be a provision not only here for all the long-running programs to save the intermediate state so that it can be resumed if it stops for any failure like power cutoff. Saving the intermediate state will allow you to resume right after where it has ended (the last saved state) instead of restarting it from the beginning. Regarding saving the state, it is also required to be decided the frequency of the saving the state. Since this is a complete overhead in terms of time and space. It is better if you have built a habit of doing this specifically when you are writing a program that will take a huge amount of time for the simulation.

Apart from that you existing library functions like FFT library whenever those are available. No need to write that explicitly. A lot of FFT library functions are. You need to install that one in your current path and you use it.

(Refer Slide Time: 31:59)



Implementation details

Output options:

Store the transformation matrix

or

Store the information regarding translation and rotation

In the output, you need to store the orientation. Either you store the complete coordinate information of the decoy complex or store the initial coordinate and the transformers matrix so that you can generate the decoys whenever you need this. The former option requires a huge amount of storage space and is not a good idea. For example, if you store the complete orientation, and there are 1000 atoms, then that will take a huge amount of storage space.The latter option is followed by the existing software.

If you store only the initial coordinates and the transformation matrix or the translation and rotation information, then whenever you need the orientation, you use that one, there will be one program that will take these two protein molecules as input and apply that transformation or the translation rotation to generate the protein complex.

Your approach should be like this for generating the output, not the complete orientation, so input or some initial orientation of two protein molecules or input protein molecules along with that transformation matrix corresponding to each orientation or 3 translations, 3 rotations information corresponding to each orientation. So, that is it. Thank you very much.