**Detecting and Visualizing Knots and Slipknots in Protein Chains**

**Master’s Project By: Shashank Mucheli**

**Advisor: Dr. Firas Khatib**

**Committee Members**

**Dr. Haiping Xu**

**Dr. Shelley Zhang**

**Abstract**

Proteins are made up of chains containing a large number of residues and are fundamental entities of all biological creatures. Protein chains very rarely have knots or slipknots and these should generally be avoided when predicting protein structures, as they make it more difficult to reach the correct fold. This project focusses on finding and visualizing knots and slipknots in protein structures. A protein is said to be knotted if the chain loops through itself and not back out again through the same loop. Slipknots occur when the polypeptide chain goes through a loop but the chain eventually doubles back on itself, removing the knot. As slipknots are not mathematically knots, the standard protein knot detection algorithms do not report them.

This project improves one such knot detection algorithm, Knotfind, to find slipknots in proteins. Our new algorithm iteratively searches subsections of a protein chain and checks each of them for knots. If a knotted region is found, we search for the residue that unties the knot resulting in a slipknot. In order to clearly visualize the slipknotted region, our algorithm outputs a modified protein chain that is then loaded into PyMol (a commonly used tool for protein visualization). Using the PyMol API we then highlight the slipknotted region of the protein chain, along with the residues that untie the knot, clearly defining the slipknot.

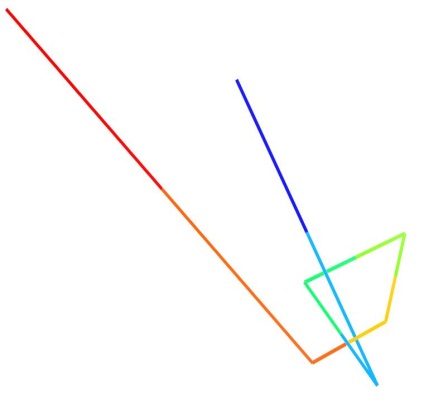
**Introduction**

Proteins are fundamental entities for all living beings. Proteins aid in almost every function that is carried out inside the body, mostly in the cells. They are large and complex molecules. Each protein is associated with a specific function in the body. Few of those functions include:

* DNA replication: Helps in producing two identical chains of DNA.
* Messenger: Transmits signals to co-ordinate biological processes.
* Transportation: Binds small atoms inside them and transport across the body.

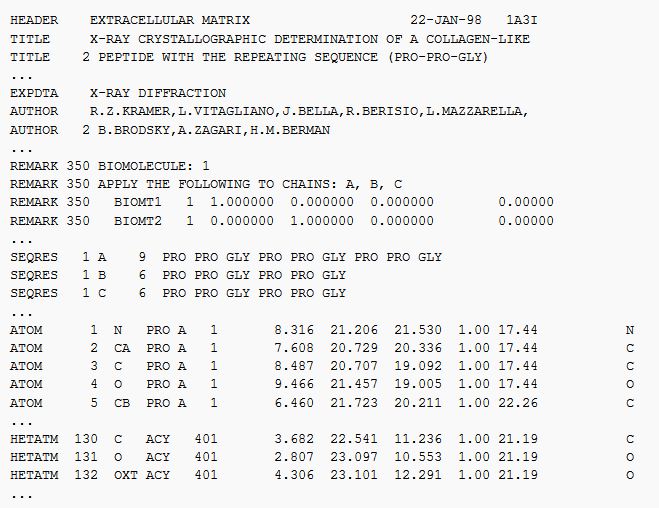
Proteins are made up hundreds or thousands of amino acids which are connected to each other making them long chains. Different amino acids sequences form different proteins. Every protein has a unique structure. The amino acid sequence determines the structure and the function of any given protein.

**Motivation**:

Knots and slipknots in protein chains make the protein folding process complicated and should generally be avoided. Reaching the native fold or the state with the most minimum energy in these types of proteins is very difficult. The motivation behind developing this algorithm was to simplify the protein folding process in general by making sure that any given amino acid sequence does not have a knot or slipknot in it and also simplifying them rapidly and effectively. Biochemists around the world can use this algorithm to check if their protein models have knots and slipknots in them and can avoid them in their models without spending too much time to check for knots and slipknots manually. In most cases visualizing knots and slipknots is a tedious task, this project also focusses on providing supporting python scripts to clearly visualize the deep knots and slipknots in any given protein chains so that the simplification process becomes easy.

**Protein Data Bank:**

Protein data bank (PDB) is a text file which contains the metadata and three dimensional structural data of a protein structure. Every known protein has a unique ID through which it can be referenced. Every atom of every amino acid will have a row in this file which contains data like atom number, residue type, chain and most importantly its structural data(x, y, z co-ordinates). For our algorithm we only consider the structural data of the cα atoms which form the backbone of the protein structure. We filter out only the cα atoms and store them in appropriate data structures. Once the algorithm terminates all the remaining atoms are logged back onto PDB files which later aid in visualizing the knots. Three dimensional protein structures can be visualized using tools like PyMol.



An example of how a protein data bank would look like is presented in the above image. Image courtesy: https://en.wikipedia.org/wiki/Protein\_Data\_Bank\_(file\_format)

**Knots in Proteins**

Knots in proteins are very rare and they should be avoided in the protein folding process. Proteins are considered knotted if their backbone is entangled to form a knot. Imagine pulling a protein by holding both its termini (N and the C terminals), if the sequence ends up having a loop which goes through itself, then we consider it as a knotted protein. In general a protein with no knots should get disentangled and it should not have any loops in it. Knots in proteins are tough to understand as the complexity to reach its native state becomes tougher and complicated to determine its function. Structure Prediction methods do not model the protein folding process itself but rather try to seek only the native state. Consequently, the knot prediction mechanisms are not relevant to the protein modeling process, which means the possibility of finding knots are high in protein models.

**Knotfind:**

We have extended the knotfind algorithm in this project to detect slipknots and visualize both knots and slipknots. Knotfind is an efficient knot prediction algorithm which checks all the residues in a protein chain and searches for knots else it keeps on simplifying the residue chain until only the termini remain. If the protein is knotted the algorithm returns a PDB file with the atoms which constitute the knot.

Procedure:

Knotfind uses only the alpha carbon atoms (cα) in a protein chain to detect the knots. The PDB is parsed into a Java method to extract only the data of the alpha carbon atoms and is stored onto a list. We only need the atom number and co-ordinates of the each cα atom. The atom data is later passed onto different methods for further processing of the chain.

Knotfind uses an iterative approach to simplify and eliminate atoms from the residue chains. It initially starts with ‘n’ atoms (total number of cα’s in the given protein chain) and goes on until the size of the chain is two (for unknotted proteins) and proteins with knots will have two plus the unsimplified atoms.

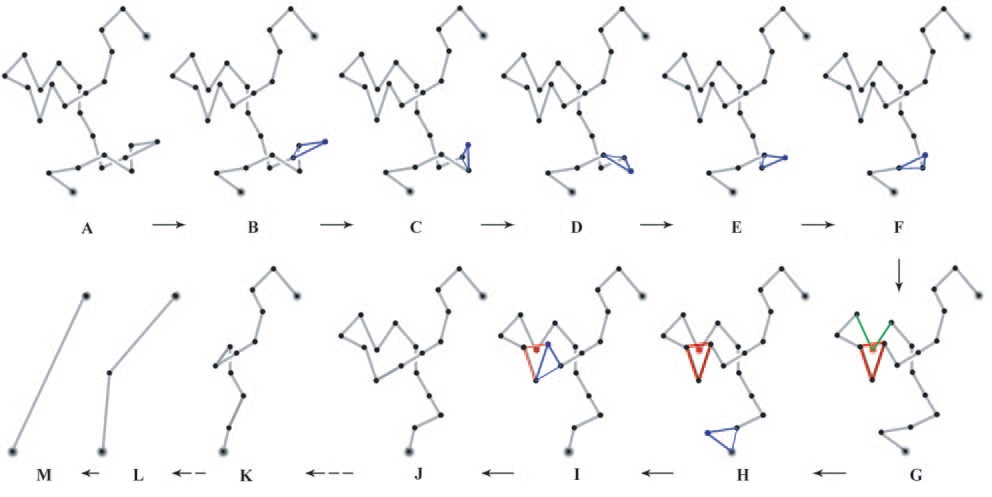
Cα atoms arranged in the increasing order of i-1 to i+1 cartesian distance. Sets of three consecutive cα atoms i-1, i, i+1 are considered in each iteration. If there is no line segment, j, j+1 cutting through the triangle formed by connecting i-1, i, i+1, then cα i, is removed from the residue chain. If the line segment defined by j, j+1 is cutting through the triangle defined by i-1, i, i+1 then the cα i, is not simplified and the next set of i-1, i, i+1 is considered for simplification. This procedure is repeated until the last set of atoms are selected and simplified.

When the algorithm terminates it should only have N and C terminal cα atoms such that the chain has been simplified into a straight line. If the chain is not fully simplified, as in, there are other cα atoms remaining with the N and C terminals, and then those atoms define the knotted region of the protein.

When a knot is detected, to double check and verify the knot an alternate method is also used, Where the area of the triangle formed by connecting i-1, i, i+1 is considered, If area covered by the triangle is being interested by any line j, j+1 which or also in the same plane then the i’th is not simplified else, the i’th atom is simplified and the process is repeated until the all the residues are simplified or checked. A tolerance of 0.0003 Å is used to round off errors. This is considered as a possible line width of the line connecting j, j+1.

The algorithm keeps a log of all the simplified residues and the unsimplified residues. When the algorithm terminates the remaining residues are stored back onto a new PDB file. This is then visualized.

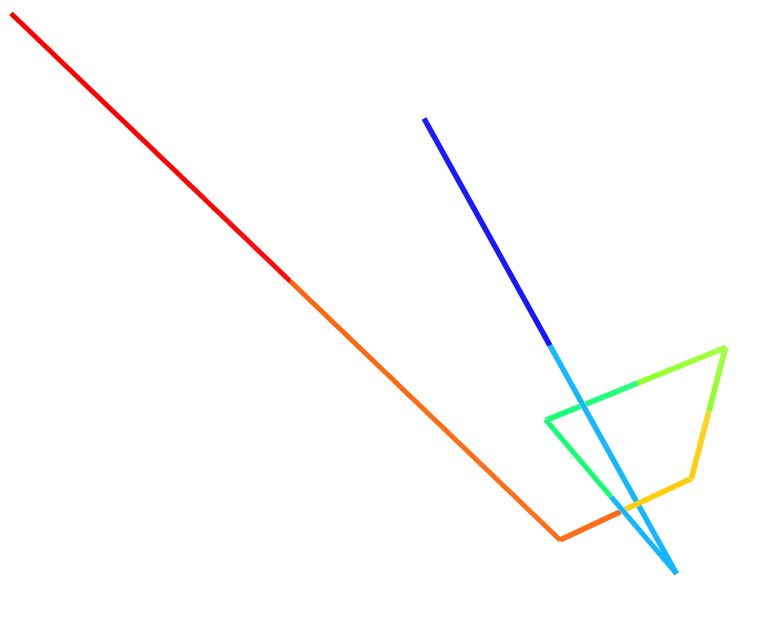
Knotfind simplification explained:



Firas Khatib, M. T. (2006). Rapid knot detection and application to protein structure.

In the above image, we can see the changes in the state of the protein chain when the algorithm is running. Image A is the original structure of the backbone of the protein formed by connecting the cα atoms. The algorithm considers i-1, i, i+1such that the distances between them is the shortest. In image B, since no line segment is intersecting the lines connected by i-1, i, i+1 (The triangle marked in blue) the cα atom i is eliminated from the chain. The same process is repeated in the steps C through F. In the step G, an atom is intersecting the lines connected (The triangle marked in red), so i’th atom here is ignored and the next set i-1, i, i+1is considered for simplification. After iteration in steps H, one of the atoms with intersect in step G gets simplified and all the following atoms will get simplified in the later steps. When the algorithm terminates, since no knot is detected, it will have only the two terminal atoms left in the protein chain.

Visualizing proteins with knots:



PDB: 1js1

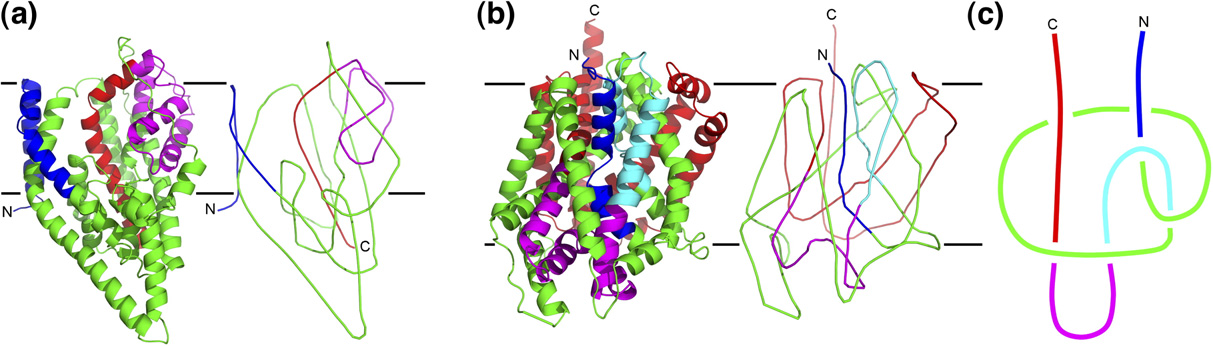
If the algorithm returns a knot for any given PDB file, all its unsimplified atoms are stored in a new PDB file along with the terminals. We use PyMol (A tool to visualize proteins) and it’s sophisticated Python API to visualize the knot and make the knot evident. We have written a supporting Python script which takes in all the residues which remain in the PDB and connect them from N to C terminals sequentially and color them using spectrum coloring, in which the chain starts with blue color and continues to red color which indicates the line from N through C terminal.

**Slipknots:**

Slipknots are a different class of knots which rarely occur in the protein models. They are different when compared to the traditional knots and detecting them is a tedious task. It is also different from detecting the commonly found knots.

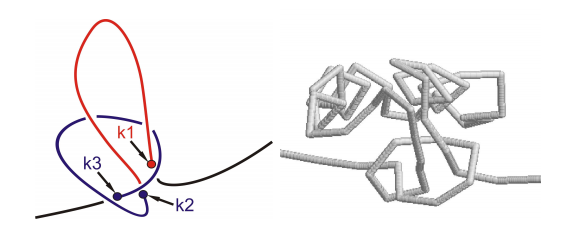
Slipknots are knots in which the knotted protein chain doubles back on itself and mathematically gets untied. If we assume to hold both N and C terminals and pull from both the ends, the knotted chain would get unknotted and would simplify into a straight line. But, a knot is still present in the chain and is affecting the energy of the protein model, even these should be processed and removed to have a protein with most minimum energy or a protein with the correct fold. Slipknots are not mathematically knots and hence the normal knot detection algorithms cannot report them, even Knotfind cannot report them when the entire backbone is passed to it at once. The Knotfind algorithm can simplify all the atoms in the chain other than N and C termini and would report no knots when the entire chain is passed. However, slipknots have a knot in them which when processed in subsections can be reported using knotfind.

We leverage this property of slipknots to find them and report them. We extend the already existing knotfind algorithm to detect slipknots. We divide the entire protein chain into small subsections and iteratively run the knotfind algorithm on these subsections and increase the size of the chain after every pass until all the residues are checked. While processing, slipknots are mathematically knots until it finds that one atom (which we mark as the k1 atom) which unties the whole chain and makes the chain unknotted.



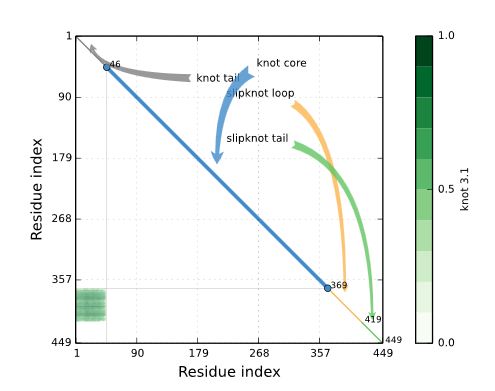
Neil P. King1, E. O. (2007). Identification of Rare Slipknots in Proteins and Their Implications for Stability and Folding.

The transition from image A to C represents the simplification of the protein chain to have just the alpha carbon atoms after smoothening the backbone of the initial structure.In the above image, in section C, we can notice that pulling both N and C terminals would untie the chain and make the whole protein unknotted. These types of knots go undetected when we run traditional knot find algorithms on them.



The above image defines the points which form a slipknot. Atom k3 to k2 form the knot and continues to be knotted until k1 is found which unties the whole chain and makes the protein chain unknotted.

Slipknot Classification:



KnotProt for 1ALK chain A

Image: http://knotprot.cent.uw.edu.pl/view/1alk/A/

A slipknotted protein chain is classified into four different regions and each of them represent a part of the knot. Following are its details:

* Knot Core: The shortest sub chain with a knot.
* Knot Tail: A segment between one terminal and the Knot core.
* Slipknot Tail: The longest segment starting at one terminal, for which no change in topology is detected.
* Slipknot Loop: The segment between knot core and the slipknot tail.

Extending Knotfind to detect Slipknots:

We extend the knotfind algorithm to find slipknots, we run knotfind algorithm on subsections of the protein chains. We start the processing with only three atoms in the chain, Knotfind would return false with a chain size of three as no line segment could possibly intersect a line drawn to connect these three atoms. After every attempt of knotfind we increase the size of the chain by one, as in we append the next available cα atom in the pool to the sub chain which we are considering to process slipknots, and run the knotfind algorithm on it.

For any knotted or slipknotted sample, we first focus on getting the k3 and k2 atoms which define the knot core for any knotted sample. If no knot core is found during the entire process it is evident that the sample is not knotted or slipknotted. If the knotfind algorithm returns a knot for any given subsection of the protein we report the k3 and k2 atoms and continue increasing the size of the chain till ‘n’.

Once a knot is found, the algorithm checks until what length the algorithm would return true. If at any point the algorithm returns false, then we mark that atom which removes the knot as k1 atom and report it along with k3 and k2. And we mark the entire chain as slipknotted. The regions from k3 to k2 would form the knot core and k2 to k1 would be the slipknot loop. Atoms from one to k3-1 would be marked as knot tail and atoms from k1+1 to ‘n’ would be marked as slipknot tail.

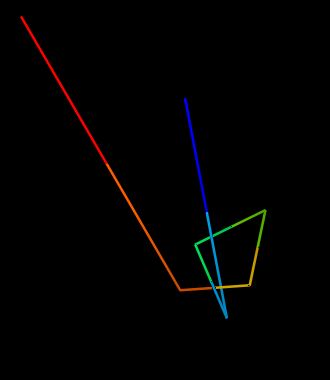
The algorithm logs all the residues which are unsimplified cα atoms from k3 to k1 onto a PDB file along with all the atoms which constitute the knot tail and slipknot tail. With the knot tail and slipknot tail we can visualize the slipknots better.

**Visualizing Knots**

At the end of the algorithm, based on what type of knot the protein has, we create PDB files with the corresponding atom data in it. We later load them using PyMol a commonly used protein visualization tool. The new PDB files will have only the cα data in it for knotted structures and knot tail and slipknot tail along with the unsimplified cα atoms for the slipknotted protein structures.

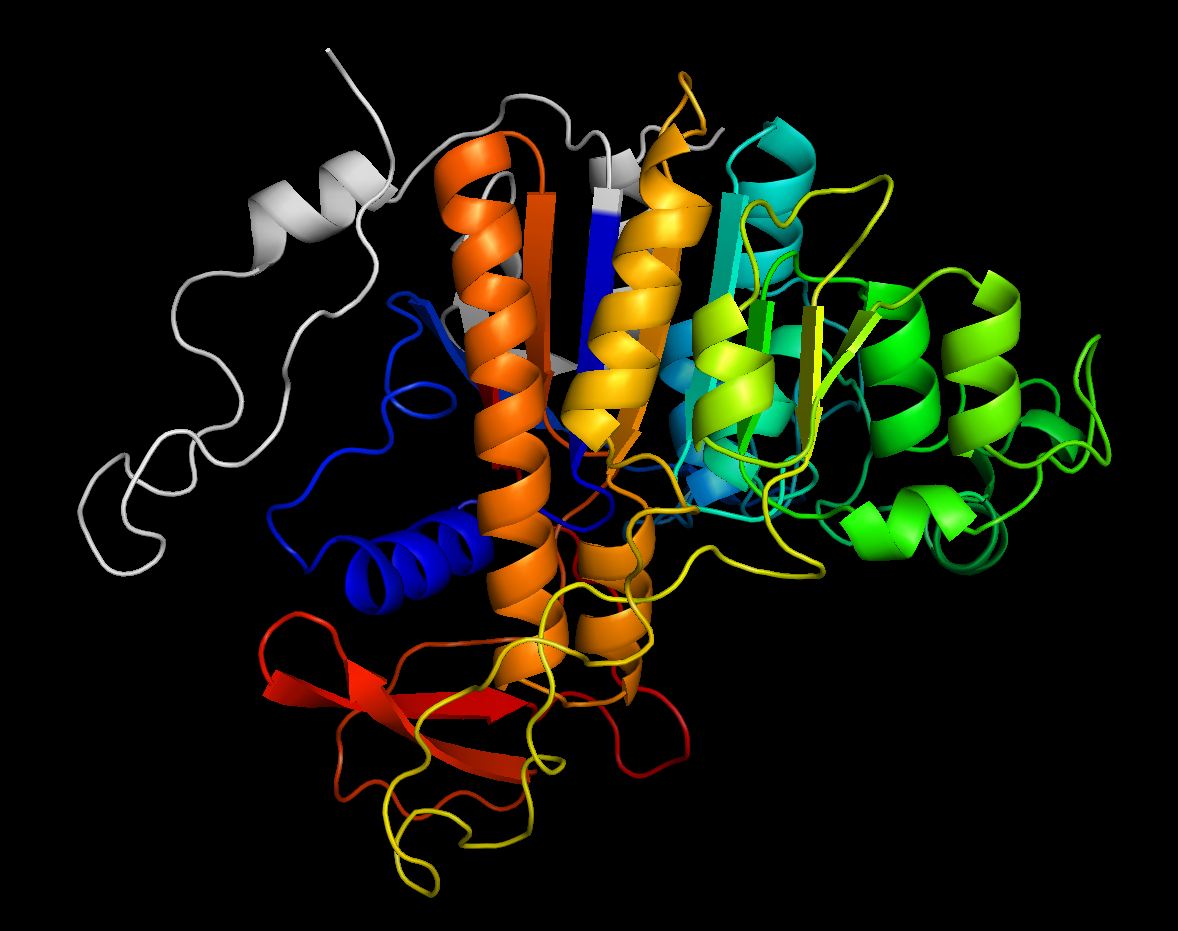
PyMol has an API of its own written in Python which we have used here to make the knots evident. We wrote a python script to connect all the atoms and visualize them serially.

Here is an example of how a knotted protein would look like before and after the script:

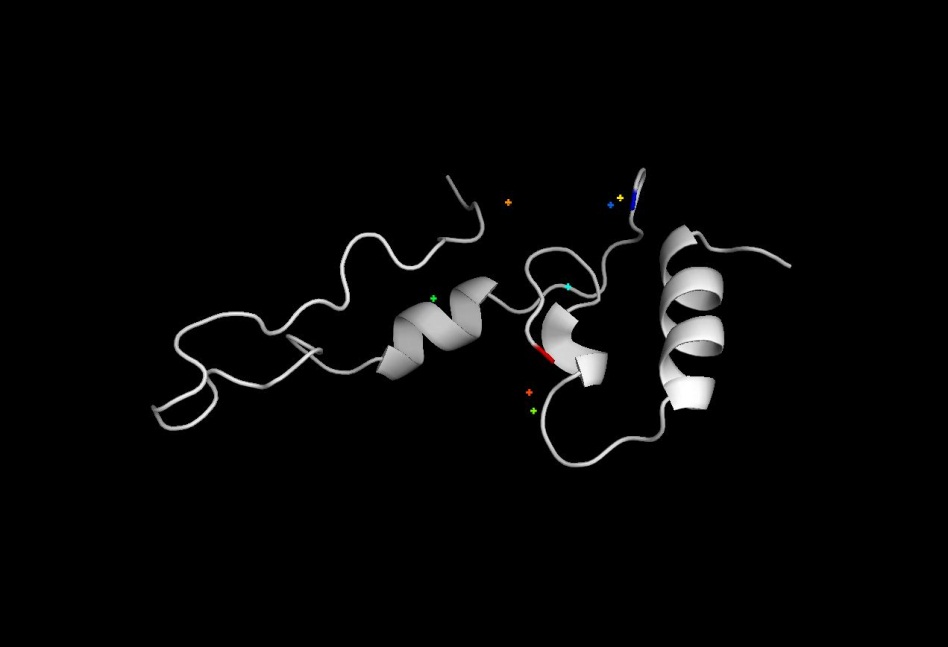


The above images are the unsimplified atoms of the protein 1js1. The knot can be visualized clearly in the second image after connecting the atoms and color them with spectrum coloring. Once the PDB is loaded into PyMol we just run a supporting script in it to connect the atoms.

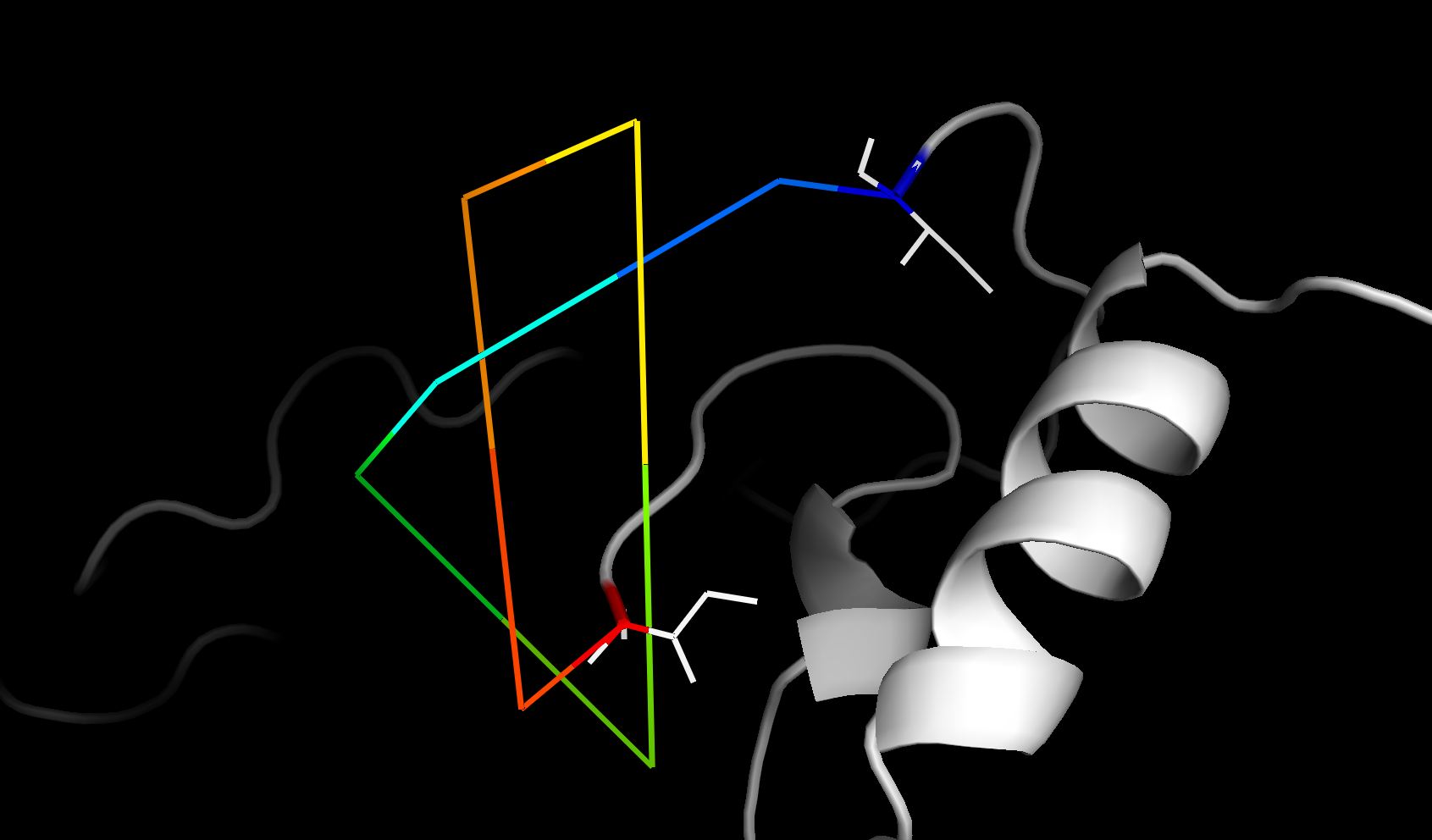
**Visualizing Slipknots:**



The above image is an example of how a slipknotted region would look like. The colored region forms the knot core and the slipknot loop and the tails are marked in gray. It is very hard to figure out how the colored region would form a slipknot and is a tedious task. After we run the slipknot algorithm on the above protein, 1ALK, we get the following result:



The above image has a few atoms which are not connected. These are the atoms which define the slipknot here is what we get upon connecting these cα atoms.



By connecting all the atoms, the slipknot in the above PDB becomes evident. And atoms from blue to through red define the knot core and the slipknot loop.

**Future Work:**

With the presented algorithm we can make the protein modeling process a little easier as the biochemists would not have to worry about processing knots manually.

* Reduce time complexity: The algorithm now compares every atom with one another to check for knots, which is a time consuming process. We are considering different approaches to reduce the time taken by this algorithm.
* Create a web server: We are planning to implement a web server which can enable the biochemists around the world just to login to it and test their proteins online.
* Run on plethora of protein structures: There are hundreds or thousands of protein models already available, we intend to run this algorithm on them to report previously unreported knots and slipknots.

**References:**

* Yuhan Zhang, Created the base code for the slipknot detector algorithm.
* Firas Khatib, M. T. (2006). Rapid knot detection and application to protein structure. *Rapid knot detection and application to protein structure*, 8.
* Joanna I. Sułkowskaa, 1. E. (2012). Conservation of complex knotting and slipknotting patterns in proteins. *Conservation of complex knotting and slipknotting patterns in proteins*, 9.
* Neil P. King1, E. O. (2007). Identification of Rare Slipknots in Proteins and Their Implications for Stability and Folding. *Identification of Rare Slipknots in Proteins and Their Implications for Stability and Folding*, 14.