Иван Тренчев

Приложение на 3D модели в биоинформатични изследвания

Иван Тренчев

ПРИЛОЖЕНИЕ НА 3D МОДЕЛИ В БИОИНФОРМАТИЧНИ ИЗСЛЕДВАНИЯ

2©2©

Ivan Trenchev

APPLICATION OF 3D MODELS IN BIOINFORMATICS RESEARCH

ЗА БУКВИТЕ
О ПИСМЕНЕХ

IVAN TRENCHEV

APPLICATION OF 3D MODELS IN BIOINFORMATICS RESEARCH

Academic Publishing "Za Bukvite – O Pismeneh"
Sofia, 2020

ИВАН ТРЕНЧЕВ

ПРИЛОЖЕНИЕ НА 3D МОДЕЛИ В БИОИНФОРМАТИЧНИ ИЗСЛЕДВАНИЯ

Академично издателство „За буквите – О писменехь“

София, 2020

ISBN 978-619-185-444-8

София, 2020

С Ъ Д Ъ Р Ж А Н И Е

C O N T E N T S

У В О Д

Биоинформатиката, като бързо развиваща се област на науката комбинира компютърни науки, математика, химия и биология, с цел решаване на биоинформатични задачи. Една от тези биоинформатични задачи е задачата за предсказване на пространствена структура на протеините. Решението на тази задача ще има пряко и дълбоко влияние върху нашите разбирания за живота, а именно как тези молекули, които изпълняват почти всяка функция в живите клетки играят своята роля на фундаментално физично ниво.

Структурната биоинформатика е една от основните изследователски области в областта на изчислителната биология *(Zhang et al., 2005; Altman and Dugan, 2005; Clote and Backofen, 2000; Pevzner, 2000; Liljas et al., 2001; Gopakumar, 2012)*. Тя се отнася до анализ и прогнозиране на триизмерната (3D) структура на биологични макромолекули като протеини, РНК и ДНК *(Zhang et al., 2005; Altman and Dugan, 2005)*. Тази структурна информация съответства на 3D макромолекулни структури, получени чрез различни експериментални методи, такива като кристалография (рентгенова дифракция), електронна микроскопия или ядрено-магнитен резонанс ЯМР (nmr). Тази информация позволява да се изследват свойствата на протеините, еволюцията и връзките между структура/функция.

Един от основните изследователски проблеми в структурната биоинформатика е прогнозирането на триизмерните протеинови структури. Протеините са дълги последователности, образувани от 20 различни аминокиселинни остатъци, които във физиологични условия приемат уникална 3D структура *(Anfinsen et al., 1961)*. Познаването на протеиновата

структура позволява изследването на биологичните процеси по-директно, с по-висока разделителна способност и по-фини детайли. Парадигмата за протеиновата последователност (известна още като хипотеза „заключване и ключ“) казва, че протеинът може да постигне биологичната си функция само чрез 3D, структурирано състояние, определено от неговата аминокиселинна последователност *(Anfinsen, 1973)*. Независимо от това, в момента е установено, че не всички протеинови функции са свързани с пространствената структура *(Dunker et al., 2008, 2001; Uversky, 2001; Tompa and Csermely, 2004; Tompa, 2002; Wright and Dyson, 1999)*. В някои случаи те трябва да бъдат „разплетени”, за да изпълняват своите функции *(Gunasekaran et al., 2003)*. Тези протеини се наричат вътрешно неподредени протеини (IDP) и представляват около 30% от последователности. Въпреки наличието на IDP протеини важен аспект на разбиране и тълкуване на функцията на даден протеин включва характеризиране молекулни взаимодействия. Тези връзки могат да се вътрешно молекулни (йонни връзки, ковалентни връзки, метални връзки и т.н.) или между молекулни (водородни връзки и други не – ковалентни връзки като сили на ван дер Ваал). Познаването на структурата на 3D полипептиди дава на изследователите много важна информация да се направи извод за функцията на протеина в клетката *(Branden and Tooze, 1998; Laskowiski сътр, 2005а, б; Lesk, 2002)*.

Структурни функции; катализа при химични реакции; транспорт и съхранение; регулаторни функции; контрол на генната транскрипция; се определят от пространствената структура. Допълнителни подробности за прогнозирането на протеиновите функции могат да бъдат намерени във Whisstock and Lesk (2003), Rentzsch и Orengo (2009) и Lee et al. (1999, 2000, 2004, 2007).

Всяко достатъчно подробно изследване на структурата на живите същества разкрива една изумителна йерархия на организацията. От най-простите аминокиселини, природата

надгражда основополагащата структура по увлекателни начини. Разбирането на тази организация е от решаващо значение за разбирането на това как работи живата материя и как тези операции могат да бъдат представени и визуализирани с помощта на компютри.

Атомите са базовите единици за разглеждане на жизнената ни структура. Тези частици са най-малката единица, която запазва химичните свойства на елемента. Химичните свойства на атомите по същество дефинират дали те привличат или отблъскват други атоми и под какви условия. Молекулите се образуват, когато два или повече атома образуват дължимо разположение.

Живата материя е организирана чрез дълбока структурна йерархия: аминокиселините изграждат протеини; протеините (често) образуват полимери; клетки и клетъчни продукти (например извънклетъчен матрикс) изграждат тъкани и органи; а тъканите и органите създават организми.

Много 3D приложения имат възможности за динамична симулация, които използват вграден двигател по физика. В този случай естествените закони на движение могат да бъдат приложени към всеки модел, който е стимулира определени физически свойства. Прост пример е този на подскачаща топка. Гравитационната сила, приложена към топка върху наклонена повърхност я кара да се търкаля, да падне от ръба и да отскочи, когато удари земята. Атрибути като маса, еластичност и триене се въвеждат от потребителя, а физическият двигател прави останалото. Полезно спестяване на време за аниматорите за бордова динамика на програма като Maya могат да се окажат полезни при типа прогнозно научно моделиране.

Днес 3D моделирането се използва още по-широко от медицината до инженерните приложения. То и визуализацията направиха възможно усъвършенстване на технологията, особено заедно с анимациите, използването на тези модели е по-често, отколкото преди *(Kundrot et al., 1991; Le Grand et al., 1993)*. Ранните

компютърни графики са векторни графики, съставени от тънки линии, докато графиките днес са базирани на пиксели.

Приложението на 3D моделирането в научните изследвания е аспект на тази разработка. Направен е анализ на различните методи за презсказване на пространствената структура на протеините. Представени са авторски разработки за създаване на пресмятане на структурата на съществуващи протеини с предложените в дисертатицония труд математически модели.

Изследвано е влиянието на магнитното поле върху ДНК и теорията на резонанса. Друг аспект на това изследване е използването на клъстърите технологии в биоинфоматичните изследвания, а именно – Функционалната теория на плътността *(Guschin et al. 2018; Arora et al. 1998).* Основната идея на метода е да се използва концепцията за електронна плътност в основно състояние *(Stump et al., 2000; Misra and S. Mukherjee, 2004; Zeyad et al., 2008),* нейното разпределение се описва от едно уравнение на Шрьодингер.

INTRODUCTION

Bioinformatics, as a rapidly developing field of science, combines computer science, mathematics, chemistry and biology in order to solve bioinformatics problems. One of these bioinformatics tasks is the task of predicting the spatial structure of proteins. The solution to this problem will have a direct and profound impact on our understanding of life, namely how these molecules, which perform almost every function in living cells, play their role on a fundamental physical level.

Structural bioinformatics is one of the main research areas in computational biology *(Zhang et al., 2005; Altman and Dugan, 2005; Clote and Backofen, 2000; Pevzner, 2000; Liljas et al., 2001; Gopakumar, 2012).* It refers to the analysis and prediction of the three-dimensional (3-D) structure of biological macromolecules such as proteins, RNA and DNA *(Zhang et al., 2005; Altman and Dugan, 2005).* This structural information corresponds to 3D macromolecular structures obtained by various experimental methods, such as crystallography (X-ray diffraction), electron microscopy or nuclear magnetic resonance NMR (nmr). This information makes it possible to study the properties of proteins, evolution and the relationships between structure/function.

One of the main research problems in structural bioinformatics is the prediction of three-dimensional protein structures. Proteins are long sequences formed by 20 different amino acid residues that, under physiological conditions, assume a unique 3-D structure *(Anfinsen et al., 1961).* Knowledge of the protein structure allows the study of biological processes more directly, with higher resolution and finer details. The protein sequence paradigm (also known as the „lock and key“ hypothesis) says that a protein can only achieve its biological function through a 3D, structured state determined by its amino acid sequence *(Anfinsen, 1973).* However, it has now been found that not all protein functions are related to spatial structure *(Dunker et al., 2008,*

2001; Uversky, 2001; Tompa and Csermely, 2004; Tompa, 2002; Wright and Dyson, 1999). In some cases, they need to be „unraveled“ to perform their functions *(Gunasekaran et al., 2003).* These proteins are called intrinsic proteins (IDPs) and make up about 30% of sequences. Despite the presence of IDP proteins, an important aspect of understanding and interpreting the function of a protein involves characterizing molecular interactions. These bonds can be intramolecular (ionic bonds, covalent bonds, metallic bonds, etc.) or between molecular (hydrogen bonds and other non-covalent bonds such as van der Waal forces). Knowledge of the structure of 3-D polypeptides provides researchers with very important information to infer the function of the protein in the cell *(Branden and Tooze, 1998; Laskowiski et al., 2005a, b; Lesk, 2002).*

Structural functions; catalysis by chemical reactions; transport and storage; regulatory functions; gene transcription control; are determined by the spatial structure. Further details on the prediction of protein functions can be found in Whisstock and Lesk (2003), Rentzsch and Orengo (2009) and Lee et al. (1999, 2000, 2004, 2007).

Any sufficiently detailed study of the structure of living beings reveals an astonishing hierarchy of organization. From the simplest amino acids, nature builds on the basic structure in fascinating ways. Understanding this organization is crucial to understanding how living matter works and how these operations can be represented and visualized using computers.

Atoms are the basic units for considering our life structure. These particles are the smallest unit that retains the chemical properties of the element. The chemical properties of atoms essentially define whether they attract or repel other atoms and under what conditions. Molecules are formed when two or more atoms form a proper arrangement.

Living matter is organized through a deep structural hierarchy: amino acids build proteins; proteins (often) form polymers; cells and cellular products (eg extracellular matrix) build tissues and organs; and tissues and organs create organisms.

Many 3D applications have dynamic simulation capabilities that use a built-in physics engine. In this case, the natural laws of motion can be applied to any model that has stimulated certain physical properties. A simple example is that of a bouncing ball. The gravitational force applied to a ball on a sloping surface causes it to roll, fall off the edge, and bounce when it hits the ground. Attributes such as mass, elasticity and friction are introduced by the user, and the physical engine does the rest. Useful time savings for animators for onboard dynamics of a program like Maya can be useful in the type of predictive scientific modeling.

Today, 3D modeling is used even more widely from medicine to engineering applications. It and visualization have made it possible to improve technology, especially with animations, the use of these models is more common than before *(Kundrot et al., 1991; Le Grand et al., 1993).* Early computer graphics were vector graphics made up of thin lines, while graphics today are based on pixels.

The application of 3D modeling in research is an aspect of this development. An analysis of the various methods for predicting the spatial structure of proteins has been made. Author's works for creating a calculation of the structure of existing proteins with the mathematical models proposed in the dissertation are presented.

The influence of the magnetic field on DNA and the theory of resonance have been studied. Another aspect of this study is the use of cluster technologies in bioinformatics research, namely, Functional Density Theory (Guschin et al. 2018; Arora et al. 1998). The main idea of the method is to use the concept of electron density in the ground state *(Stump et al., 2000; Misra and S. Mukherjee, 2004; Zeyad et al., 2008),* its distribution is described by a Schrödinger equation.

П Ъ Р В А Г Л А В А

ИЗПОЛЗВАНЕ НА 3D МОДЕЛИ В БИОЛОГИЯТА

1.1. Въведение в тематиката

Определяне на протеин структура е, както експериментално скъпа технология (поради разходите, свързани с кристалография, електро микроскопия или NMR), и отнема време *(Guntert, 2004)*. Трудността при определянето и откриването на 3D структурата на протеините породи голямо несъответствие между обема данни (последователности на аминокиселинни остатъци), генерирани от проектите на генома и броя на 3D структури на протеини, които в момента са известни. Само малка част от протеинови последователности експериментално са решили триизмерните структури. Тези факти не само ясно илюстрират необходимостта от разработване на начини на предсказване, но и мотивират по-нататъшни изследвания в методите за прогнозиране на изчислителната структура на протеини. През последните 10 години няколко изчислителни методологии, системи и алгоритми са предложени като решение на триизмерната структура на белтъка.

Изследването на структурата на протеина и предвиждане на техните триизмерни (3D) структури е един от най-важните научни проблеми в структурната Биоинформатика. Прогнозиране на триизмерен протеин, който все още няма шаблон в Protein Data Bank[1] е много трудно, а понякога и почти неразрешима задача. През последните години са разработени много изчислителни методи, системи и алгоритми с цел решаване на този сложен проблем. Проблемът обаче продължава да

[1] **Protein** Data Bank. Available at: < https://www.rcsb.org/> Accesses on: 30.09.2020

предизвиква биолози, биоинформатици, химици, компютърни учени и математици поради сложността и високата размерност на пространството за търсене на конформационни протеини.

Познаването на 3-D структурата на полипептида дава на изследователите много важна информация за функцията на протеина в клетката. Трудността при определянето и откриването на 3-D структурата на протеини генерира голямо несъответствие между обема на данни

Според Dorm класификацията на методите за прогнозиране са четири класа *(Dorn et al., 2014 : 1)* методи на първия принцип без информация от базата данни; (2) първи принципни методи с информация от база данни; (3) евристични модели и (4) сравнителни модели на моделиране и секвениране.

През последните години, най-вероятно най-важните резултати са разработените от хибридни методи. Такива хибридни методи съчетават точността на методите, основани познати структури, сила на полето, физикохимични описание на протеините и квантова механика.

Компютърното моделиране представлява математическо моделиране, при което се използват компютърни системи и технологии. При него се комбинират математически модел и компютърна програма, реализираща алгоритъма за намиране на търсеното решение. Компютърното предсказване на пространствената структура на протеините е от особено практическо значение, тъй като води до създаване на бази от данни, като например Protein Data Bank, които от своя страна водят до компютърно-подпомогне на лекарствения дизайн, може да се използват при откриване на нови лекарства и да се комбинират успешно с експериментални методи, което дава възможност да се опростяват или да се моделират сложни и скъпи експерименти. По този начин се получава ценна информация за случващото се в организмите под действието на определени вещества.

Връзката между първичната структура на ДНК и нейната биологична функция е един от изключителните проблеми в съвременната биология. Има много обективни причини да се смята, че функционалната роля на последователността на ДНК е не само за кодиране протеини, което представлява по-малко от 5% от геномите на бозайници, но също така да допринесе за контролиране на пространствената структура на ДНК в хроматин.

В днешно време е добре разпознато, че динамиката на сгъване и разгъване на ДНК в живи клетки играе основна роля за регулиране на много биологични процеси, като генна експресия, репликация на ДНК, рекомбинация и възстановяване. Както е известно геномната ДНК на еукариотните клетки е плътно опакована в нуклеозоми, които представляват основни единици хроматин. Както е описано експериментално чрез рентгенов анализ с висока разделителна способност, всяка нуклеозома се състои от почти два оборота на ДНК, обвита около октамер от основни хистонови протеини. Допълнителен ДНК фрагмент се отделят последователни нуклеозоми, които са разположени под формата на топчета по ДНК. Този нуклеозомен масив е допълнително организиран в последователни структури от по-висок ред, включително кондензация в 30 nm хроматиново влакно и образуване на хроматинови бримки, до пълна степен на кондензация в метафазни хромозоми. Всъщност структурата и динамиката на хроматина са под контрола на редица механизми, включващи взаимодействия между ДНК и протеин, от които може да зависят локалните зависимости, кодиране на протеина и др.

За да се справи с изчислителната сложност на прогнозиране на 3D протеиновата структура, се използва широк спектър от алгоритми за оптимизация *(Klepeis et al., 2003)*. Мета евристиката се използва за осигуряване на почти оптимални решения. Освен това, като се имат предвид ограниченията на четирите класа методи за прогнозиране на протеиновата структура,

изследователите наскоро разработиха хибридни методи, които комбинират принципите на четирите класа, както може да се наблюдава в последните издания на CASP *(Moult et al., 2014, 2011)*. Например точността, представена от методите за моделиране на хомологията, се комбинира с капацитета на методите Ab initio за предсказване на нови гънки *(Dhingra & Jayaram, 2013; Dorn et al., 2008 a, b; Fan and Mark, 2004)*. За да се намали сложността и голямата размерност на пространството за конформационно търсене, присъщо на методите Ab initio, информацията за структурни мотиви, открити в известни протеинови структури, може да се използва за изграждане на приблизителни сведения. Очаква се тези приблизителни конформации да са достатъчни, за да позволят по-късно преразглеждане с помощта на молекулярна механика (ММ), като симулация на MD *(Van Gunsteren & Berendsen, 1990)*. В етапа на успокояване глобалните взаимодействия между всички атоми в молекулата (включително напр. взаимодействия без връзка) се оценяват и отклоненията в торпесните ъгли на полипептидната главна и страничната верига могат да бъдат коригирани *(Fan and Mark, 2004)*.

Нашата основна цел е да прегледаме методите и изчислителните стратегии, които в момента се използват в прогнозирането на 3D протеиновата структура. За да го направим, представяме най-важните резултати, необходими за разбирането на класа методи на прогнозиране. Освен това, представяме подробности за основните методи за прогнозиране и очертаваме изпълнените изчислителни стратегии.

Ще представя описани в литературата методи за подравняване на нагъвания на протеини чрез само-избягващи се разходки (Self-Avoiding Walks – SAW) в 2D и 3D пространство. Тези методи се фокусират върху правилното дефиниране на контакти между двойки от аминокиселини, а множеството от всички тези контакти се нарича *карта на контакти*. Един от измерителите за сходство, който се използва при подравняване на

нагъвания е Припокриваща Карта на Контактите (Contact-Map Overlap – CMO). Тези измерители представляват база за практически и строги алгоритми за сравняване на структури и анализ *(Caprara & Lancia, 2002; Caprara, et al., 2004; Goldman, et al., 1999; Lancia, et al., 2001)*.

Картата на контакти за дадено нагъване на протеин е граф, който улавя модела на контактите в нагъването. Има различни концепции за контакти. В дадена карта на контактите за нагъване на протеин в HP модела всяка аминокиселина е представена като връх и съществува ребро между двойка аминокиселини i и j, когато образуват контакт в нагъването. Нагъванията на протеините са представени като само-избягващи се криви с аминокиселини, представени като точки на кривите. Обикновено, контакт в дадено нагъване се дефинира като двойка от аминокиселини, които са на разстояние по-малко от определен праг.

SAW = 2 Стека + 1 Опашка **(1)**

Една карта на контакти за само-избягваща се разходка може да се разложи на два „стека“ и една „опашка“. Оказва се, че стековете (наподобяващи α-спирали) съответстват на вериги от аминокиселини в частично подредени структури и представляват обобщения на вторичните структури, разглеждани като графи. От двоична гледна точка, опашките (наподобяващи β-листа) съответстват на анти-вериги в частично подредени структури.

В решетъчния или извън решетъчния двумерен HP модел, структурата на картата на контактите за дадено нагъване на протеин може да се характеризира по следния начин.

Теорема. За всяка HP последователност S, множеството от контакти или картата на контактите за всяко нагъване на S в 2D може да се раздели на два стека и една опашка *(Goldman, 2000;*

Goldman, et al., 1999).

За разлика от двумерния HP модел, в 3D HP модела няма известен приближен алгоритъм на проблема за припокриваща карта на контактите.

Интересно е да се отбележи, че в описаните в литературата изследвания, са открити контактни карти на протеинови структури от PDB, които са разбити на комбиниран брой стекове и опашки *(Agarwal, et al., 2007).*

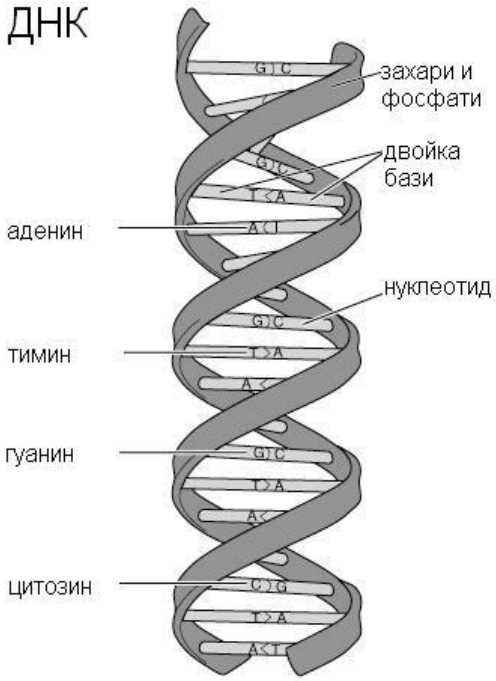
1.2. Основни биоинформатични сведения

ДНК (дезоксирибонуклеинова киселина) е генетичен материал. Генетичния материал на една клетка или на един организъм се отнася на тези материали намерени в ядрото, митохондриите и цитоплазмата, които играят фундаментална роля при определяне на структурата на клетките на веществата, и е способен на самостоятелно размножаване и промяна. Това е информация, която се съхранява в ДНК (Айала и Кигер, 1987), която осъществява организация на неживите молекули във функционални, живи клетки и организми, които са във възможност да регулират техните вътрешни химически състави, ръст и репродукция.

Различните единици, които управляват тези характеристики на генетично ниво, се наричат **гени.** С други думи казано: те са всъщност единици за наследственост в жив организъм.

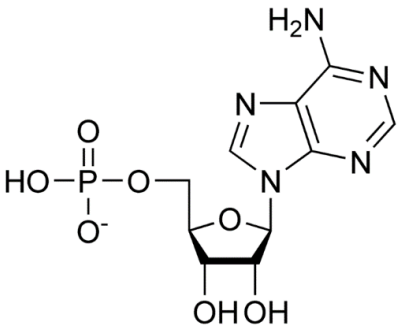
Самите гени съдържат техните информации като специфична редица на нуклеотиди, които се намират в ДНК молекулите. Само четири различни бази се ползват в ДНК молекули: гуанин, аденин, тимин и цитозин (G, A, T и C) **фиг. 1.1.**

Има още един нуклеотид – това е **урацил,** обаче него го няма в ДНК, но се намира в РНК. Урацилът е химичния аналог на тимина в РНК.

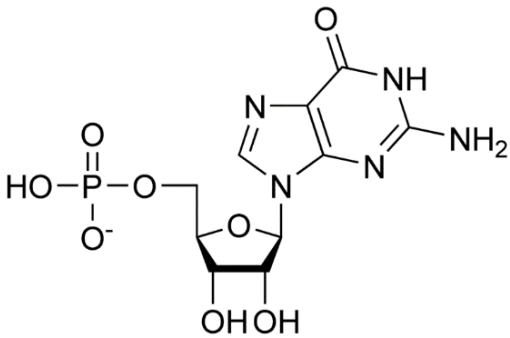


Фиг. 1.1. Представане на ДНК

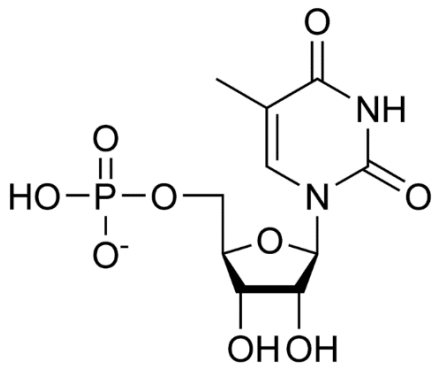
Всяка база е прикрепена на фосфат група и деоксирибоза, за да формира нуклеотиди. Единствената работа, която прави един нуклеотид да се различава от друг е коя нитрогенска база я съдържа Фиг. 1.2 – Фиг. 1.5.



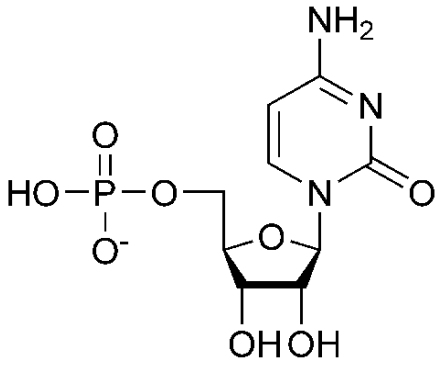
Фиг. 1.2. AMP (Аденозин монофосфат)



Фиг. 1.3. GMP (Гуанозин монофосфат)



Фиг. 1.4. TMP (Тимидин монофосфат)



Фиг. 1.5. CMP (Цитозин монофосфат)

Всяка информация в рамките на всеки ген идва от реда, в който тези четири нуклеотиди се намират по дължината на ДНК молекулите. Сложните гени могат да бъдат дълги хиляди нуклеотиди, и всички генетични инструкции на организъма, неговия геном, може да се поддържа в милиони, даже и милиарди нуклотиди.

Генетичният код е набор от правила, чрез които информацията кодирана в генетичен материал (DNA или mRNA редици) е преведена в протеини (редици от аминокиселинни).

В общият случай предаването на наследствената информация минава следните етапи: разплитане на двойната спирала на ДНК; образуване на две нови двойни спирали, образуване на две нови клетки. Най-общо синтезът на белтъците протича по следния начин: разплитане на участък от ДНК; прочитане и копиране на този участък от разплетената ДНК от ядрото в цитоплазмата; детайлно разчитане на копирания участък и синтез на белтък. Този механизъм на предаване на наследствена информация, създаден от природата, се нарича съвременен генетичен код.

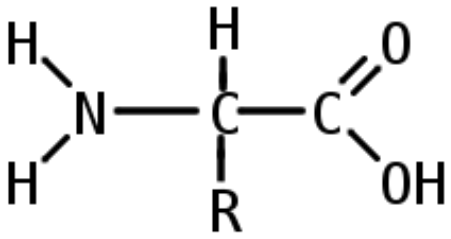
Този код е наречен съвременен, макар че е от милиони години. С какво той е уникален? Защо природата е предпочела

точно него, а не някой друг теоретичен генетичен код. Много изследователи започнаха изследвания в този аспект, а именно доколко съвременния генетичен код е устойчив, т.е. ако в някой етап на предаването на наследствената информация, кодираното съобщение претърпи промяна (мутация), ще бъде ли коректно предадено. Макар че се използват различни модели за изследване на СГК, които се базират на статистически подходи, различни вероятностни разпределения и пр., понастоящем не „се използва" цялото множество на всички теоретични генетични кодове.

Наследствената информация се съдържа в нуклеотидната последователност на ДНК (първичната им структура) под формата на трибуквени думи, наречени кодони. Тази последователност се проявява, чрез детерминиране на специфичната последователност на аминокиселини в белтъците (протеините), които са структурообразуващите органични съединения в организмите (растения и животни). Универсалността на всички живи организми се изразява във факта, че кодът, който трансформира нуклеотидните последователности в ядрото в последователности от аминокиселини в протеините, е един и същ във всички организми: бактерии; растения; животни и в човека. Съществуват много малко на брой изключения – бактерии *(Di Giulio, Medugno, 1998; Knight et all., 1999).*

Аминокиселините са основните единици, от които са съставени протеините. Също както и ДНК, и РНК, протеините са синтезирани като линейни полимери съставени от по-малки молекули. Обаче, за разлика от ДНК и РНК, в които има четири нуклеотиди, от които може да се избира, протеините са изградени от 20 аминокиселини с най-различни големини, форми и химически свойства.

Те имат следващата обща структура:



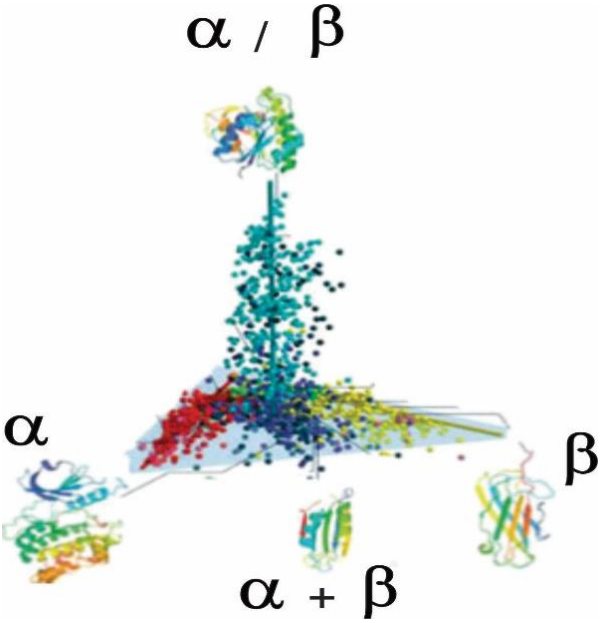
Фиг. 1.6. Обща структура на всяка аминокиселина

Както може да се види от фиг. 1.6, всяка аминокиселина съдържа аминогрупа, карбоксилна група, водороден атом, алфа въглероден атом свързан с амино киселинен остатък (R група; R верига), който всъщност прави всяка аминокиселина уникална.

Протеините (или белтъците) са биологични макромолекули изградени чрез поликондензирането на аминокиселини. Те всъщност са съставени от една последователност от аминокиселини. Функционалните свойства на протеините се основават на тяхната тримерна структура. Протеините са много важни компоненти на всички живи организми. Те изпълняват много жизнено важни функции. Те са структуро образуващите органични съединения в организмите (растения и животни).

Протеините се сгъват в разнообразие от триизмерни структури с разнообразни топологии добре дефинирани структурни йерархии. Основната структура на протеин, състоящ се от последователността на аминокиселини, които образуват полипептида верига, се само събира във вторични структурни елементи като α-спирала и β-листове, чиито взаимодействия определят протеиновата третична структура. Някои протеини образуват или хомо, или хетеро мултимери и по този начин придобиват кватернерна структура. Парадигматичен пример на протеин с кватернер структурата е тази на хемоглобина. След това гънката на протеина се определя като подредба на вторичните структурни елементи на структурата относително

един към друг в пространството. Броят на известните протеинови структури масово се увеличава в последните десетилетия с усъвършенстване на методите за определяне на структурата – рентген кристалография, ЯМР и по-скоро криоелектронна микроскопия. В края на 2018 г., Протеин Банка данни (PDB) (rcsb.org) изброи впечатляващ брой от 135 000 протеинови структури, от които 90% се определят от рентгеновата кристалография. Въпреки това, има значително структурно съкращение и протеините се сгъват в по-ограничен брой уникални гънки и топологии в протеиновата вселена. Графичното представяне на различни гънки според тяхната структурна топология осигурява впечатляваща илюстрация на разнообразието на протеиновата вселена (Фиг. 1.7).



Фиг. 1.7. Разнообразие от протеинови топологии

С представяне на α-спирала и β-листове Копие от Hou J, Sims GE, Zhang C, Kim S-H (2003) A global representation of the protein fold space. Proc Natl Acad Sci 100:2386–2390

Поддържането на триизмерната структура на протеин е резултат от прибавен принос на различни физически взаимодействия, които като цяло действат на протеина заедно.

Формирането на пространствена структура на даден протеин зависи от специфични и неспецифични ковалентни взаимодействия. Неспецифичните взаимодействия са по същество неполярни (хидрофобни) взаимодействия, van der Waals сили, които са важни за задвижването на процеса на сгъване. Специфичните взаимодействия по същество са електростатични и водородни връзки, които са основни за сгъването на **структурата и динамиката.**

Неполярни (хидрофобни) взаимодействия, установени между неполярна аминокиселинна верига във вътрешността на протеините са сред най-значимите стабилизиращи взаимодействия на протеиновата структура. Те се формират, за да се сведе до минимум експозицията на неполярни региони до водни молекули, които заобикалят протеини и осигуряват значително стабилизиращ принос (* 60–80 kJ mol^{-1}). Както ще обсъдим по-късно, установяването на неполярни хидрофобни взаимодействия е резултат от т.нар хидрофобен ефект, който е основната движеща сила за сгъване на протеини.

Взаимодействията на Ван дер Ваал възникват, когато атом с частичен заряд е наблизо и не заредена, предизвикваща незабавно преразпределение на електронната плътност, която води до слабо атрактивно взаимодействие между съседните атоми. Тези сили се срещат, както в полярни, така и в неполярни групи, включващи преходно индуцирани или постоянни електрически диполи. Тези взаимодействия са индивидуално много слаби (* 1 kJ mol^{-1}), но колективно силни и са изключително къси, което означава оптимизирани в плътно опакована сърцевина от протеини. По този начин взаимодействията на Ван дер Валс плътно свързани с хидрофобната опаковка на протеин.

Електростатичните взаимодействия са взаимодействия между заряд и заряд, установени между постоянно заредени йони и се простират на значителни разстояния. Те най-вече включват полярни групи от странични вериги. Въпреки това протеиновите

гръбначни атоми са частично заредени и могат да участват и в атрактивни и отблъскващи взаимодействия; въпреки че те са по-слаби, тъй като зарядите са по-малки, кумулативният им ефект може да бъде значителен. Мостовете често са близки по протеинова последователност и образуват вътре един и същ вторичен структурен елемент. Това предполага, че тези взаимодействия допринасят за протеинова стабилност чрез ограничаване на движенията на гръбнака. Електростатичните взаимодействия може да се считат от няколко вида. Взаимодействия, включващи полярни аминокиселини, околните водни молекули не влияят на стабилността на протеините, но са доста важни да се моделира разтворимостта на протеини. Взаимодействия, включващи полярни групи разположени на протеинова повърхност имат незначителен принос за протеиновата стабилност като зарядите са защитени една от друга от водни молекули и силата на взаимодействието е отслабено до 17–50 kJ mol^{-1}.

Водородните връзки се установяват между два електро отрицателни атома с водород, свързан към един от тях. Тя може да възникне със заобикалящата вода, докато вътре в протеинови, пептидни групи водородна връзка или към друга пептидна група или към молекула на водата от ядрото на протеина. Най-много от гръбнака CO и NH групите (90%) са свързани с H, благоприятствайки вътрешната организация и ограничаване на протеиновите конформации. Водородните връзки в протеини са насочени, тъй като силата им зависи от диполната ориентация, варираща от 8 до 29kJ mol^{-1}. Това е много важна характеристика, тъй като осигурява специфично взаимодействие. По този начин водородните връзки допринасят за определяне на стабилната третична структура на протеина. Други взаимодействия, специфични за някои протеини, също играят важна стабилизираща роля. Такъв е случаят с **дисулфидните връзки** (S – S), които са ковалентни взаимодействия, установени между окислени странични вериги от цистеинови остатъци. Само

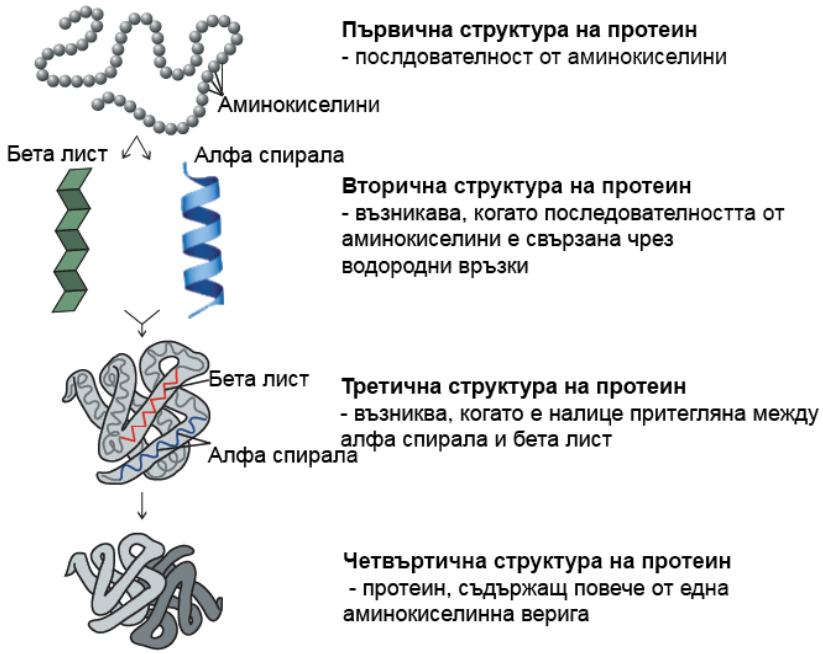
няколко протеина съдържат S – S връзки, някои от които играят ключова структурна роля.

Протеините са гъвкави молекули, които имат цялостна форма, но вътрешно се „клатят" (Ofran et al., 2006; Reva et al., 1998). Движенията в протеините включват бързо локално движение с редки (бавни) промени към различна конформация. Тези промени включват бързи преходи между отделните състояния. Движението на протеина е термично: взаимодействие между вода и разтворители с протеини индуцират вибрации и люлеещи се движения (библиотеки) *(Richardson, 1977; Sanchez & Sali, 2000; Sali et al., 1994a)*. Бързите локални движения са хармонични (симетрични вибрации) и некорелирани.

Водните молекули и протеинното нагъване са изключително важни за разбиране протеинова динамика. Слоят на протеиновата хидратация е слоят от водни молекули, който се образуват около протеина; той е от съществено значение за сгъването и функционирането на протеините, защото водните молекули осигуряват топлинна енергия за протеинови движения, както и Н-свързване към заредените групи и гръбначни атоми, достъпни от повърхността. На динамиката на водата в хидратационния слой около протеин е различна до разстояние 1 nm. Продължителността на контакт с конкретна водна молекула с протеиновата повърхност може да се появи в интервала на суб-наносекунда. В претъпкани разтвори (както в клетките), масата на водата е не повече от 40% от общото количество протеинова маса. Следователно, хидратационният слой (не повече от две водни молекули) е по същество цялата вода, налична във вътреклетъчна среда *(Ishima et al., 2004; Lippincott-Schwartz et al., 2001; Lensink et al., 2019)*. Водните молекули в протеиновите кухини също са важни модулатори на протеинова динамика: вътрешната водна молекула запазва известна подвижност и нейните движения в кухината, позволяват на протеина да реорганизира водородна връзка. Всъщност вътрешните водни молекули се намират в кухини близо до активни слоеве, тъй като това улеснява

протеиновите движения *(Matheson et al., 2018).*

През 50-те години на миналия век американският биохимик Кристиан Анфинсен (1916–1995) прави серия от открития, които показаха, че информацията, необходима за сгъване на полипептидна верига, е добре дефинирана и биологично функционална конформация, която е стриктно кодирана в аминокиселинна последователност *(Lyu et al. 1990; Murzina et al. 1995; Miyazawa et al., 1985, Mendes et al., 1999)*. Тези експерименти разширяват знанията за аминокиселинна структура на протеините при раждането на областта на протеиновата структурна биохимия. Всъщност, структурата на миоглобина, на първия протеин, който има своята триизмерна структура, определена чрез рентгенова кристалография от Джон Кендрю в Кеймбридж през 1958 г.



Фигура 1.8. Йерархична структура на протеин

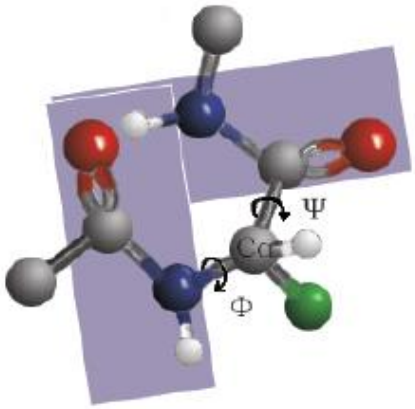
Структурата на протеините е йерархично организирана на четири нива (Фиг. 1.8): първична, вторична, третична и четвъртична структура. Всяко ниво е важно и тяхната комбинация води до един всеобхватен поглед над протеините.

Първичната структура на протеините представлява точната

последователност на свързване на отделните аминокиселини, кодирана от полинуклеотидната последователност на ДНК. В природата протеините не се срещат в своята първична структура. Те могат да я заемат временно.

По-общо казано основната структура на протеините се състои от двумерни, линейно подредени аминокиселини свързани в полипептидна верига *(Jones, 2001).* Аминокиселините са градивните елементи на протеините и всяка една от тях води до превод на 3-кодонен сегмент на иРНК. Кодон (3-кодон) е група от три базови двойки РНК, които кодират една от 20-те възможни аминокиселини. Полипептидната верига е последователност от различни аминокиселини, които са свързани чрез ковалентни пептидни връзки. Пептидната група (определена от три атома C_α-*C*-*N*) на всяка аминокиселина образува основата (гръбнака) на пептидната верига. Атома C_α се отнася за първият въглерод в пептидната група на всяка аминокиселина, която представлява функционална група, отговорна за характеристиките на химичната реакция. Структурата на пептидните групи е едновременно твърда и равнинна, което се демонстрира от Линус Паулинг през 1930 г. и 1940 г. *(Voet & Voet, 1995).* Това откритие води до идентифицирането на торсионните ъгли, които определят една опорна полипептидна структура. Торсионните ъгли (фиг. 1.9) са ъглите на въртене около връзката C_α-*N* (ϕ) и връзката C_α-*C* (ψ).

Тези ъгли са пространствено ограничени и по този начин ограничават обхвата на основната структура. Степента на φ и ψ ъглите характеризира вторичните структури в една тримерна протеинова структура. Първичната структура води до вторична структура *(Voet & Voet, 1995).*



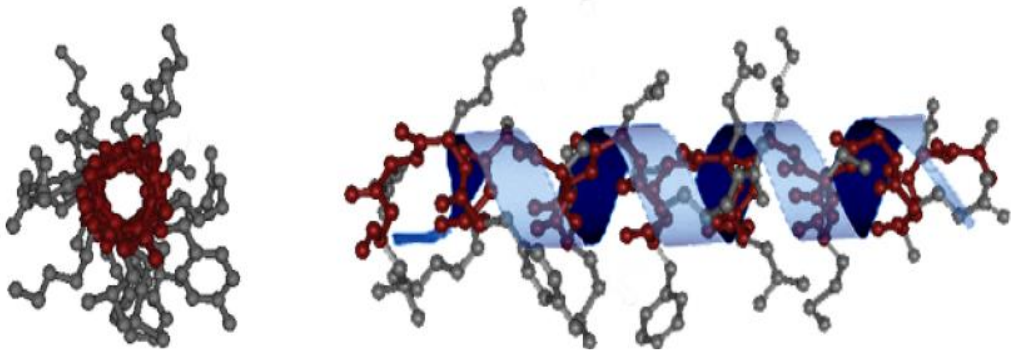
Фиг. 1.9. Полипептиден скелет показващ местоположението и посоката на въртене на торсионните ъгли на пептидните групи: φ и ψ

Вторичната структура представлява локалните нагъвания на основния скелет на полипептидната верига на протеина, които притежават известна периодичност, т.е. пространственото подреждане на аминокиселините *(Jones, 2000)*. Всяка възпроизводима и характерна форма се означава като отделен тип вторична структура: α-спирала, β-лист и β-завои:

- ***α*-Спирала** – *α*-спиралата се получава при възникване на водородни връзки между всяка пептидна група. Тези връзки водят до огъване на полипептидната верига. *α*-спиралата е възможно най-компактната форма на полипептидната верига. Обикновено естествените *α*-спирали са дясно завити.
- ***β*-Лист** – *β*-листа се формира, когато две сравнително опънати полипептидни вериги се намират близо една до друга и техните пептидни групи образуват вътрешно-верижни водородни мостове. Това е сравнително най-опънатата форма на полипептидна верига. Основният скелет на полипептидната верига се извива

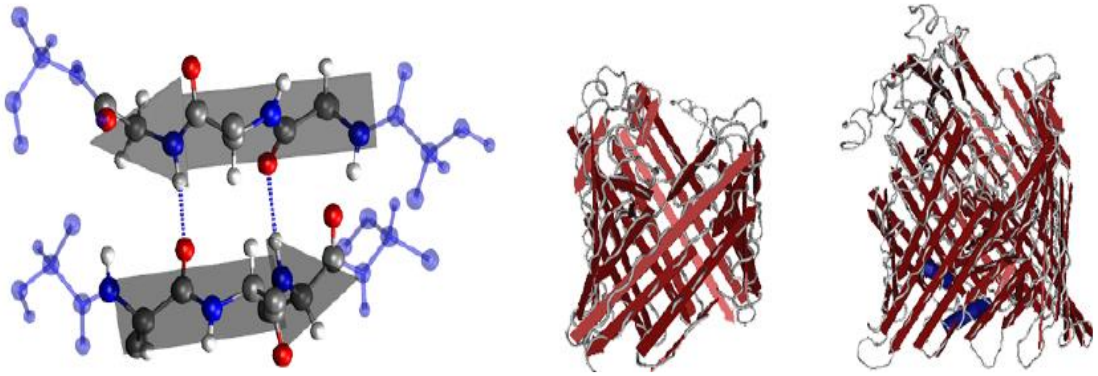
зигзагообразно и наподобява хармоника, в чийто краища се намират *a*-въглеродни атоми.

Всяка от тези единици съдържа водородни връзки. Най-известната и най-разпознаваемата единица във вторичните структури в над 30% от всички аминокиселини в кълбовидните протеини е дясно завитата α-спирала (фиг. 1.10) *(Jones, 2001)*. Стойностите на ϕ и ψ ъглите в α-спиралата, позволяват на атомите на гръбнака да се опаковат много плътно с няколко неблагоприятни контакти, за да образуват водородни връзки.



Фиг. 1.10. Лявата фигура показва кръгъл и пръчковиден модел на α спирала, погледнат отгоре. Дясната фигура е страничен изглед с включена тясна ивица

Подобно на α-спиралата, β-нишката представлява спираловидна структура, чието разположение е удължено с две аминокиселини. β-нишките (фиг. 1.10) са по-трудни за разпознаване и нестабилни, поради липсата на локално стабилизиращи взаимодействия. Стабилността се увеличава, когато последователност от β-нишки се свържат чрез водородни връзки, образувайки β-лист. Листа може да бъде или паралелен или анти-паралелен в зависимост от разположението на β-нишките. В редица протеини β-нишките се свързват в широки извити листа, известни като β-барели (*Jones, 2001*).



Фиг. 1.11. Фигурата вляво показва β-лист, съставен от единични β-нишки, а фигурата в дясно – β-барели съставен от множество листове *(Lenz, 2005)*

Бета-завоите имат универсална роля, която дава възможност на полипептидите да променят своята посока и в някои случаи да се обръщат към себе си (фиг. 1.11). Проучванията показват, че има около 10 различни конформации, базирани както на броя на аминокиселините, които могат да се завъртят, така и на степените на φ и ψ ъглите, свързани с основните аминокиселини. Общите завъртания включват γ-завой, който се характеризира чрез аминокиселина в средата на завоя, която не участва във водородните връзки и β-завой, в който двете средни аминокиселини никога не участват в изграждането на водородни връзки *(Jones, 2001).*

Тези структурни единици, не се срещат заедно във всички протеини. В действителност, протеините се разделят на структурни класове на базата на състава на техните вторични структури. Съществуват четири различни структурни класа на протеините, характеризиращи се с включване или изключване на α-спирали и β-нишки. Тези четири класа са: всички-α, всички-β, (α+β) и α/β *(Ofran & Margalit, 2006; Taylor, 2000).*

Третичната структура представлява пространствено подреждане на полипетидната верига на протеина, образувано чрез специфични стабилизиращи взаимодействия. Тази

структура дава представа за цялостна форма на белтъчната молекула, за връзките и отношенията между отделните вторични структури. Третичната структура не е резултат от действието на случайни сили, а е строго определена от аминокиселинната последователност (първичната структура) на полипептидната верига, т.е. тя е генетично детерминирана. Тримерната форма, която белтъчната молекула приема при третичната структура се нарича *конформация*. Във физиологични условия дадена конформация търпи неголеми колебания, възникнали при взаимодействия с други молекули, свързани с функционалните прояви на белтъците. Стабилната третична структура на протеините се базира на няколко различни взаимодействия, характеризиращи се с относителна сила и честота.

През 1990 г., Дил предполага, че ако се разберат доминиращите сили, участващи в структурата на протеините, ще се разберат функциите на протеините *(Dill, 1990, Dill et al., 1995)*. Базирайки своите открития на хипотезата, че доминиращата сила в третична структура на протеините може да обясни защо естествено нагънатото състояние на протеина е най-изгодно, Дил стига до заключението, че хидрофобните взаимодействия са тази доминираща сила[2]. Взаимодействията като Ван дер Ваалсовите сили и водородните връзки са от решаващо значение за предотвратяване на денатурирането (разгъването) на протеина и при тях липсва обяснение за тяхното предимство в нагънатото състояние. Въпреки, че не е възможно единствено доминиращата сила (хидрофобните взаимодействия) да бъде отговорна за определянето на естествената структура на протеина, тя може да доведе до генериране на относително малък брой конформации, от които да се определи естествената структура на даден протеин, чрез балансиране на всички взаимодействия *(Dill, 1990)*.

Третичната структура на големите протеини може да бъде

[2] **Трайков**, Методи. (2017) Математически модели и алгоритми за предсказване на пространствената структура на протеини. [дисертационен труд]

организирана около повече от една структурна единица. Всяка единица се нарича *домейн*, който се определя като регион, чието нагъване е независимо и притежава хидрофобна сърцевина *(Birney & Ponting, 2000)*.

Четвъртичната структура обяснява наличието на повече от една полипептидна верига в много протеини. Тя използва същите сили на взаимодействие като при третична структура и по това прилича на нея. Различава се по това, че възникват взаимодействия между една или повече полипептидни вериги, които често се наричат субединици *(Jones, 2001)*.

Използвайки експериментални методи даден протеин може да се денатурира и да се изучи неговата първична структура. Чрез рентгенова кристалография или ЯМР може да се проучи тримерната структура на протеините. Това, което не може да се установи е точната причина, която води до нагъването на протеина и защо протеините се нагъват винаги до една и съща конформация *(Whitford, 1995)*.

Проблемът за нагъването на протеините има за цел да определи естествената структурата на даден протеин по неговата линейна последователност от аминокиселини. Обратният проблем за нагъване на протеини представлява проблема за проектиране на линейна последователност от аминокиселини на даден протеин, т.е. дадена е специфична тримерна конформация, целта е да се определи линейна последователност от аминокиселини, която създава тази уникална структура. Решението на обратния проблем за нагъване на протеин има голям потенциал като инструмент за проектиране на лекарства, тъй като може да предостави приближение до основната последователност от аминокиселини за целевия протеин *(Vinson & Valda, 2017)* и в същото време може да се използва при проектиране на вторични структури на протеини, които имат потенциал за създаване на лекарства *(Yu, 2002)*.

1.3. Описание на Математически модели

Под задача на дискретното оптимиране ще разбираме

$$f(x^0) = min\ f(x), \quad x \in G, \tag{2}$$

множество от допустими стойности, което е крайно, $0 \leq |G| = N < \infty$, където $|G|$ е множеството от елементите на G.

Поради това че G е крайно, всички допустими решения могат да бъдат номерирани:

$$x^1, x^2, \ldots, x^N$$

след това изчисляваме $f(x^i)$, $i = 1..N$. Проверката на всички възможни решения е един от подходите за решаване на комбинаторна задача. Поради големият брой решения много често това е практически невъзможно.

Казваме, че даден проблем е проблем с голяма размерност, ако не е изпълнено поне едно от неравенствата

$$T(S) \leq T_0, \quad V(S) \leq V_0,$$

Където $T(S)$ е времето за решаване тази задача на конкретен компютър с определен софтуер, $V(S)$ е паметта необходима за решение на задачата, T_0 V_0, времето и паметта, отделени за решаване на задачата.

Широко приета дефиниция е, че даден проблем е с голяма размерност ако допустимото множество от решения е съразмеримо със задачата на търговския пътник.

Малко са научните термини, които толкова бързо придобиха широка популярност, колкото понятието „NP-пълен проблем“. За краткия период от появата на тази концепция от началото на 70-те години на миналия век се превърна в символ на огромните затруднения, с които разработчиците на алгоритми се сблъскват все по-често, тъй като те решават проблеми с все по-нарастващо

измерение и все по-сложна структура. Многобройни и разнообразни задачи, които често се срещат в математиката, теоретичното оптимиране и изследванията на операциите, се оказват NP-пълни, а списъкът с такива проблеми се актуализира почти ежедневно. Проблемите, завършени с NP, са толкова широко разпространени, че е много важно всеки, който влезе в контакт с изчислителните аспекти на тези области, да разбере смисъла и значението на тази концепция *(Berger & Leighton, 1998; Lathrop, 1994; Hart & Istrail, 1997)*.

Казваме, че алгоритъмът решава проблем, ако е приложим за всеки отделен „подпроблем" и задължително дава решение.

1.4. Нагъване на протеини – HP folding модел

Въпреки че броят на известните 3D структури се увеличил почти експоненциално в последните 20 години, познатите протеинови структури представляват само малка част от известните белтъчни секвенции. Целта на нагъването на протеините е да се предвиди компактна тримерна структура на даден протеин, базирана на негова аминокиселинна последователност.

Методът на *HP* нагъването на протеини в решетка се основава на наблюдението, че в полярна среда, пептидите се нагъват по начин в който има повече хидрофобни аминокиселини в ядро – при контакт между тях – и повече полярни амино киселини в контакт с полярната среда. Такава форма на сгънатия пептид е с минимална енергия и е по-стабилен, така че ние можем да очакваме този случай на 3D структура ще бъде реалния случай.

HP нагъването в решетка се различава от другите подходи за нагъване в две точки:

докато повечето подходи разчитат на цялостната азбука на

аминокиселините (20 букви), *HP* нагъването ползва симплифицирана двусимволна азбука, в която всяка аминокиселина е или Хидрофобна „h" или полярна „p" (таблица 1.1)

3D пространството, в което секвенцията ще се нагъва, е дискретизирано в 3D решетка (в нашия случай, 3D кубична проста решетка) *(Xu et al., 2004; Yanev et al., 2008, 2011, 2017; Yoon, 2006.)*

Таблица 1.1. Хидрофобна/Полярна класификация на 20-те α аминокиселини

Име	***Символ***	***Класификация***	***Име***	***Символ***	***Класификация***
Alanine	A	Хидрофобна	Leucine	L	Хидрофобна
Arginine	R	Полярна	Lysine	K	Полярна
Asparagine	N	Полярна	Methionine	M	Хидрофобна
Aspartic Acid	D	Полярна	Phenylalanine	F	Хидрофобна
Cysteine	C	Полярна	Proline	P	Хидрофобна
Glutamic Acid	E	Полярна	Serine	S	Полярна
Glutamine	Q	Полярна	Threonine	T	Полярна
Glycine	G	Полярна	Tryptophan	W	Хидрофобна
Histidine	H	Полярна	Tyrosine	Y	Полярна
Isoleucine	I	Хидрофобна	Valine	V	Хидрофобна

В *HP* нагъването в решетка, възможно нагъване се нарича **self-avoiding walk** и се състои от поставянето на аминокиселините от секвенция в решетка със следващите ограничения:

- всички аминокиселини от секвенцията могат да бъдат поставени в решетката;
- клетка от решетката може да съдържа най-много една аминокиселина от секвенцията;
- две аминокиселини, които са една след друга в секвенцията могат да бъдат поставени в клетки, които са съседи в решетката.

Хидрофобен контакт (*h-h* контакт) възниква, когато две хидрофобни аминокиселини, които не са съседи в секвенцията, са поставени в съседни клетки в решетката.

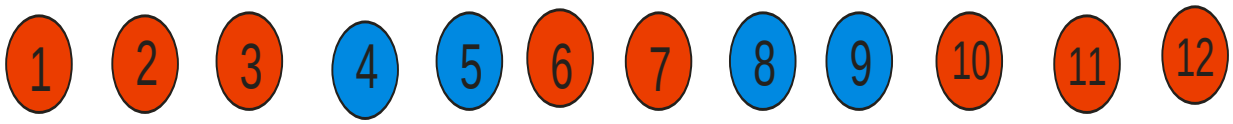
Оптималното нагъване е self-avoiding walk, който притежава максимален брой на хидрофобни контакти.

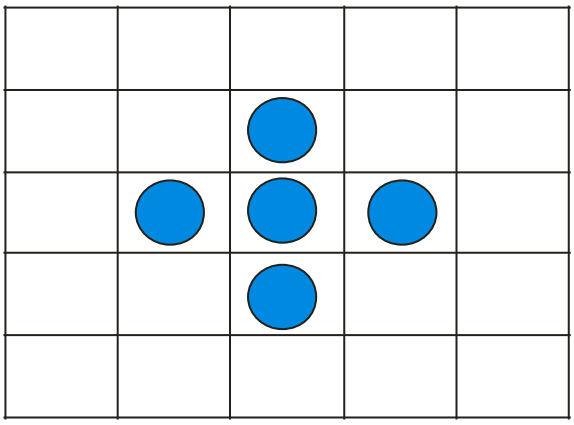
HP нагъването на протеини е *NP пълен проблем*, заради което това изследване се концентрира в създаване на алгоритми за апроксимация. То може да бъде използвано по два различни начини:

- за предвиждане на 3D структура на протеин чрез ползване на неговата първична структура (или последователност);
- за оценка на 3D структура, която е предложена от други модели за нагъване на протеинът.

Нека да имаме последователност от амино киселини, които сме конвертирали в HP последователност:

/12 амино киселини, червените са полярни, сините хидрофобни/

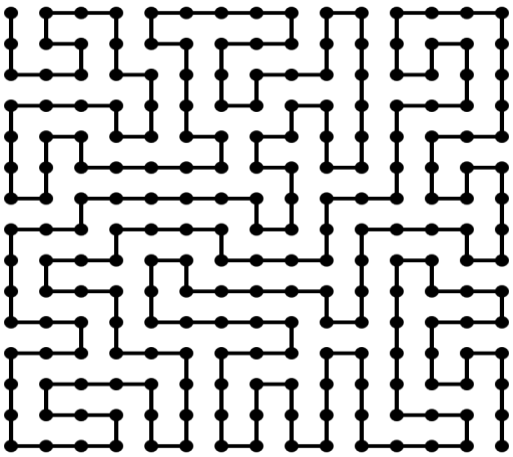




Фиг. 1.12. Графично представяне на протеин от 12 аминокиселини

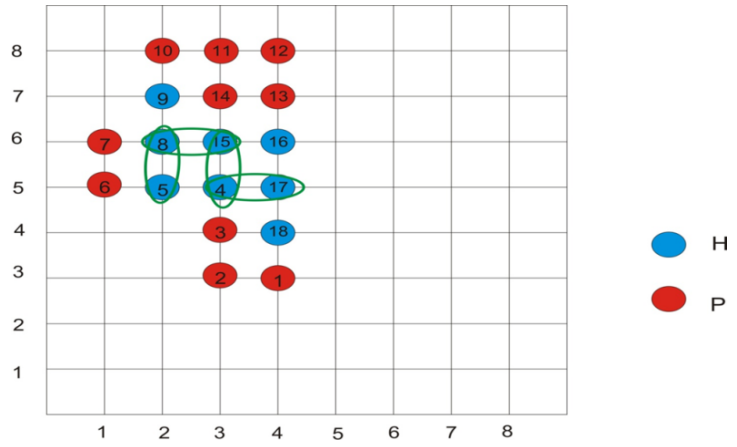
Нашата цел е да поставим червените и сините топки върху двумерна решетка като:

- Всеки две съседни топки от редицата трябва да бъдат и съседни върху решетката;
- В една клетка можем да поставим най-много една топка.



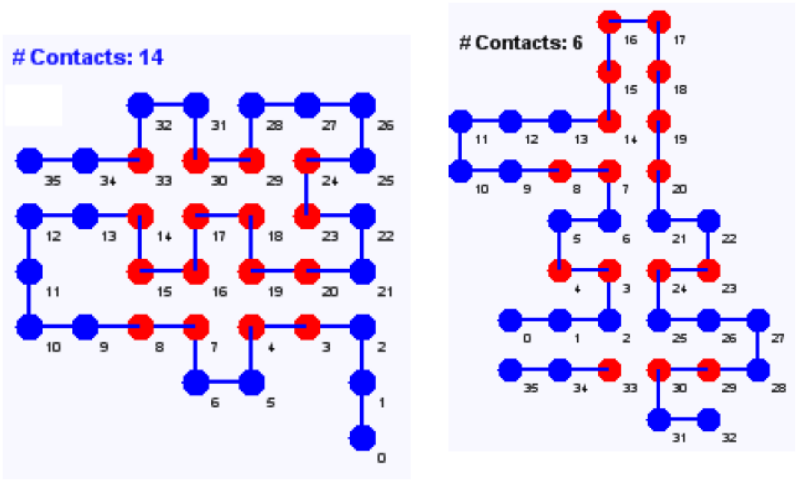
Фиг. 1.13. Self-avoiding walk

Целта ни е да намерим **self-avoiding walk** фиг 1.13**.**



Фиг 1.14. Възможни контакти между сини аминокиселини

Нашата цел е да намерим максимален брой контакти между сините топки (фиг. 1.15)



Фиг 1.15. Оптимално с 14 контакта, и евристично с 6 контакта

Формулираният проблем е NP труден, т.е. не съществува полиноминиален алгоритъм за решаването му. Намирането на оптималното решение е много трудна задача изискваща много изчислително време. На горните две фигури са представени две решения: едното оптимално с 14 контакта, и евристично с 6 контакта (фиг. 1. 15).

HP моделът за проблем d максимизира броят на *HH* двойки

контакти, същевременно задоволявайки следните три ограничения:

- всяка аминокиселина трябва да бъде определена в една от решетковите точки;
- всяка решеткова точка не може да бъде разпределена към повече от една аминокиселина;
- всеки две аминокиселини, които са последователни в протеиновата секвенция трябва да бъдат поставени в една от съседните решеткови точки.

Нека имаме един протеин *q*. Както казахме по-горе, той е представен чрез неговите аминокиселини, т.е. от техните хидрофобни/хидрофилни свойства. Това означава, че всъщност протеина *q* е един бинарен стринг, съдържащ се от 1 и 0, така че:

1 – хидрофобна аминокиселина (H);
0 – полярна аминокиселина (хидрофилна) (P).

Нека да означим със *s* дължината на протеина (дължината на бинарният стринг). Имаме и едно число *n*, така че: $n = \sqrt{s} + 1$.

Както вече споменахме, *HP моделът* на протеините има модели и в двумерното и в тримерното пространство, който обикновено е дефиниран на 2D или 3D целочислени решетки. С цел опростяване на модела, 2D и 3D решетките могат да бъдат трансформирани в 1D решетка чрез следващите правила:

- двумерна координата *(x,y)* в една координата *i*: $i = nx(y-1) + x$;
- тримерна координата *(x,y,z)* в една координата *i*: $i = n^2 x(z-1) + nx(y-1) + x$;

И, разбира се, очевидно е че тъй като решетките са целочислени, което означава че x, $y, z \in N$.

Нека $L(i,k)$ е целочислена решетка. Нека за 2D и 3D решетки

имаме: $N = n^2$ и $N = n^3$ съответно. За всеки връх $i, k \in L$ (в стълб $i \in L$ и ред $k \in L$) са дефинирани бинарни променливи x_{ik} и z_{ikjl} такива че:

$$x_{ik} = \begin{cases} 1, & \textit{ако } k-\textit{тата аминокиселина е в клетката } i \\ 0, & \textit{в останалите случаи} \end{cases}$$

където $i = i...N \; ; \; k = 1...s.$

$$z_{ikjl} = \begin{cases} 1, & \textit{ако ръба } (i,k,j,l) \textit{ е контакт помежду две НН аминокиселини} \\ 0, & \textit{в останалите случаи} \end{cases}$$

където $i, j = 1...N \; ; \; k, l = 1...s.$

Нека множеството $G(i)$ означава всички възможни съседи на *i*. Броя на елементите на $G(i)$ за 2D решетки е 4, а за 3D решетки е 6. За 2D решетките има 4 възможни съседи:

1. $G_1(i) = i + 1$
2. $G_2(i) = i - 1$
3. $G_3(i) = i + n$
4. $G_4(i) = i - n$

За 3D решетките има 6 възможни съседи:

1. $G_1(i) = i + 1$
2. $G_2(i) = i - 1$
3. $G_3(i) = i + n$
4. $G_4(i) = i - n$
5. $G_5(i) = i + n^2$
6. $G_5(i) = i - n^2$

Всички тези характеристики, които споменахме до сега за математическите HP-folding модели са в сила за всички 3 модели за които ще говорим.

Модел № 1[3]

SAW-а е дефиниран от следващите ограничения:

Всяка аминокиселина трябва да заеме връх от решетката, т.е. във всеки стълб k, на точно една променлива $x_{i\,k}$ е зададена стойност 1:

$$\sum_i x_{i,k} = 1 \qquad \forall k\text{, където } i = 1..N,\ k = 1..n. \qquad (1)$$

Една клетка от решетката може да съдържа най-много една аминокиселина, т.е. във всеки ред $i \in L$, най-много една променлива $x_{i\,k}$ може да има стойност 1:

$$\sum_k x_{i,k} \le 1 \qquad \forall i\text{, където } i = 1..N,\ k = 1..n. \qquad (2)$$

Било кои две последователни аминокиселини трябва да бъдат поставени в съседни клетки, т.е. ако в стълбът k върхът $x_{i\,k}$ има стойност 1, тогава в стълба $x_{i\,k+1}$ един от върховете $x_{j\,k+1}$ $(j \in G(i))$, трябва да има стойност 1:

$$\sum_k x_{i,k} \le \sum_{j \in G(i)} x_{j,k+1} \qquad \forall x_{i,k}\text{, където } i = 1..N,\ k = 1..n. \qquad (3)$$

Тези ограничения дефинират SAW-а.

Сега трябва да се моделират връзките помежду върховете и ръбовете. Това се прави чрез следните равенства:

$$x_{i,k} \ge \sum_{j \in G(i)} z_{i,k,j,l} \qquad \forall x_{i,k}\text{, където } i, j = 1..N,\ k, l = 1..n. \qquad (4)$$

[3]Yanev N., Milanov P., **Trenchev I.,** Mirchev I. **Integer programming approaches to HP folding** // [poster] Eight European Workshop in Drug Desing, Certosa di Pontignano, Siena – Italy, May 22nd-29th, 2011

$x_{j,l} \geq \sum_{i \in G(j)} z_{i,k,j,l} \qquad \forall x_{j,l}$ където $i, j = 1 \ldots N,\ k, l = 1 \ldots n.$ (5)

$x_{i,k}$ - binary, where $i = 1..N,\ k = 1..n.$ (6)

На тези ограничения имаме целевата функция:

$max \sum z_{i\,k\,j\,l}$, където $i, j = 1 \ldots N;\ k, l = 1 \ldots s;$

Нека E и O представляват множествата от H аминокиселини в четна и нечетна позиция в *q,* респективно.

Ако променливите $x_{i\,k}$ където $i = 1 \ldots N;\ k = 1 \ldots s;$ не са бинарни, тогава условията (1) – (5) с (7) дефинират линеен проблем (LP задача).

Модел № 2

Моделът № 2 е подобен на моделът № 1. Всъщност, и третият модел е подобен на първият, разликите са само в условията (4) и (.5).

Условията (1) – (.3) и (7) са в сила и в модел №: 2. Условията (1) и (3) се заместват с едно:

$2x_{ik} \geq \sum_{j \in G(i)} z_{i,k,j,l} + \sum_{i \in G(j)} z_{i,k,j,l}$ където $i, j = 1 \ldots N;\ k, l = 1 \ldots s;$ . . . (4)

Тук е добавено и още един клас на ограничения:

$0 \leq z_{i,k,j,l} \leq 1$ където $i, j = 1 \ldots N;\ k, l = 1 \ldots s;$ . . . (5)

Модел № 3

Модел № 3 е подобрен модел № 2 като:

$\sum_i x_{i,k} = y_i \quad \forall k$, където $i = 1.N$, $k = 1..n$. (1) и y_i - binary.

Описание на изчислителния софтуер

Софтуер CPLEX

ILOG CPLEX е инструмент за решаване на линейни оптимизационни задачи (LP Problems), със следващия математически модел:

min (max): $c_1 x_1 + c_2 x_2 + \ldots + c_n x_n$

ограничения:
$$a_{11} x_1 + a_{12} x_2 + \ldots + a_{1n} x_n \approx b_1$$
$$a_{21} x_1 + a_{22} x_2 + \ldots + a_{2n} x_n \approx b_2$$
$$\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{m1} x_1 + a_{m2} x_2 + \ldots + a_{mn} x_n \approx b_m$$

Данните които се предоставят като вход за това линейно програмиране са:

Коефициенти на целевата функция: $c_1, c_2, \ldots, c_n$

Коефициенти на ограниченията:

$$a_{11}, a_{12}, \ldots a_{1n}$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{m1}, a_{m2}, \ldots . a_{mn}$$

Коефициенти на „дясната ръка“:

$b_1, b_2, \ldots, b_n$

Горни и долни граници: $u_1 \ldots u_n \; u \; l_1 \ldots l_n$

Оптималното решение което ILOG CPLEX изчислява и връща е:
Променливи: $x_1, x_2, \ldots, x_n$

ILOG CPLEX също така може да реши и няколко разширения на линейното програмиране:

- **задачи с мрежови потоци**, специален случай на LP, който CPLEX може да го реши много по-бързо чрез използване на структура на проблема;
- **задачи на квадратичното програмиране** (Quadratic Programming (QP) problems), където целевата функция на LP задача е разширена така че тя може да включва квадратични условия;
- **задачи на квадратичното ограничено програмиране** (Quadratically Constrained Programming (QCP) problems). Това са проблеми, които включват квадратични условия в ограниченията. Всъщност, CPLEX може да решава проблеми от типа на „конусово програмиране от втори ред“ (Second Order Cone Programming (SOCP) problems);
- **задачи на смесеното целочислено програмиране** (Mixed Integer Programming (MIP) problems) където било коя или всичките LP, QP, или QCP променливи са допълнително ограничени да приемат целочислени стойности в оптимално решение и където MIP само по себе си е разширено да включва конструкции като „Специално подредени множества“ (Special Ordered Sets (SOS)) и полу-непрекъснати променливи.

CPLEX идва с три форми, за да може да отговаря на широк кръг от нужди на потребителите си:

- **CPLEX Interactive Optimizer –** Симплексовият интерактивен оптимизатор е изпълнима програма, която може да чете проблема интерактивно или от файлове, които са в стандартни формати, да реши проблема, и да достави решението интерактивно или в текстови файлове. Програмата се състои от файла cplex.exe за Windows

платформи или cplex за UNIX платформи.

- Concert Technology е множество от C++, Java и .NET библиотеки от класове, които предлагат помощни средства за моделиране, които позволяват на програмиста да вгради CPLEX оптимизаторите в C++, Java или .NET приложения. Следващата таблица показва файловете, които съдържат библиотеките.

CPLEX предвижда няколко опции за въвеждане на данните за даден проблем. Когато се ползва Interactive Optimizer повечето потребители въвеждат данните си чрез форматирани файлове. CPLEX поддържа стандартния MPS файл формат (Mathematical Programming System), както и собствения CPLEX LP формат – редово ориентиран (row oriented) формат, който много потребители го смятат за много удобен. Интерактивното въвеждане (ползвайки CPLEX LP формата) също така е възможен за малки проблеми.

Имаме следния LP проблем:

Целева функция: max --> $x_1 + 2x_2 + 3x_3$

$$-x_1 + x_2 + x_3 \leq 20$$
$$x_1 - 3x_2 + x_3 \leq 30$$

ограничения: $0 \leq x_1 \leq 40$

$0 \leq x_2 \leq +\infty$

$0 \leq x_3 \leq +\infty$

Следващият пример е изход на екрана от изпълнението на CPLEX Interactive Optimizer, където модела на примера е въведен и решен.

Показания пример е тестван върху сървъра на ЮЗУ.

CPLEX> ***указва на CPLEX prompt, а текстът, който следва, всъщност е вход на потребителя.***

```
Welcome to CPLEX Interactive Optimizer 10.0.0
    with Simplex, Mixed Integer & Barrier Optimizers
    Copyright (c) ILOG 1997-2006
    CPLEX is a registered trademark of ILOG

    Type 'help' for a list of available commands.
    Type 'help' followed by a command name for more
    information on commands.

    CPLEX> enter example
    Enter new problem ['end' on a separate line terminates]:
    maximize x1 + 2 x2 + 3 x3
    subject to -x1 +  x2 + x3 <= 20
                x1 - 3x2 + x3 <= 30
    bounds
    0 <= x1 <= 40
    0 <= x2
    0 <= x3

    End

    CPLEX> optimize
    Tried aggregator 1 time.
    No LP presolve or aggregator reductions.
    Presolve time = 0.00 sec.

    Iteration log . . .
```

```
Iteration:     1 Dual infeasibility =    0.000000
Iteration:     2 Dual objective =        202.500000

Dual simplex - Optimal: Objective = 2.0250000000e+002
Solution time = 0.01 sec. Iterations = 2 (1)

CPLEX> display solution variables x1-x3
Variable Name                Solution Value
x1                                  40.000000
x2                                  17.500000
x3                                  42.500000

CPLEX> quit
```

За търсенето на информация върху биологични, физикохимични, биохимични и др. характеристики на различни АК и нуклеотидни секвенции, както и обектите, върху които са получени, използвахме три вида БД: ExPASy, KEGG и NCBI.

Описание на разработения софтуер за реализиране на математическия модел

За реализиране на входните файлове за софтуерния пакет CPLEX са използвани два програмни езика: Java и C++. Първоначално описаните модели бяха написани програми за реализирането на C++.

Използвахме стандарти библиотеки на клас iostream.h. Генерирахме случайни последователности от HP като целта ни беше да генерираме входните файлове за оптимизационния пакет и след това стартирахме оптимизацията.

За целите на настоящото изследване ще покажем само част от кода.

Конвертиране на реален протеин в HP последователност

данните в 01 последователност.

```
#include <iostream.h>
using namespace std;
      int main(){
       int s=28;
          int n=8, br=0, br1=0;

       int  ac, ac1, sum =0;
       //ac amino acid, ac1 second amono acid;

     char g[] = "FSGPPGLQGRLQRLLQASGNHAAGILTM";
     char hp[s];

  int  i,j,l,t;

 for( ac=0; ac<s; ac++)

        if  (g[ac]=='A'  ||  g[ac]=='I'  ||  g[ac]=='L'  ||  g[ac]=='M'  ||
g[ac]=='F' || g[ac]=='P' || g[ac]=='W' || g[ac]=='V' )

       hp[ac]='1';
       else
        hp[ac]='0';
  for( ac=0; ac<s; ac++)  cout <<hp[ac];
  cout <<'\n';
 cin>>ac;
 }
```

Със следващата програма генерираме входен файл за нагъване на протеин с 11 аминокиселини (11010101011 – 1 1 са хидрофобни, а 0 полярни).

```
include <iostream.h>
#include <fstream.h>
using namespace std;
    int main(){



     int s=11;
       int n=4, br=0;

     int  ac,ac1;
     //ac  amino  acid,  ac1
sZcond amono acid;

    char g[] = "11010101011";
   int  i,j,l,t, k;

ofstream  outf  ("a1.lp",
ios::ate);

outf <<"Maximize"<<'\n';
  outf <<"obj: ";
for( ac=0; ac<s-2; ac++)
   for(  ac1=ac+3;  ac1<s;
ac1++)

     if      (g[ac]=='1'      &&
g[ac1]=='1')
    {
     for (i=0; i<n; i++)

     for (j=0; j<n; j++)

          for (k=0;  k<n; k++)
      {
                    {
              if  (i>0  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<
""<<ac1+1<<""<<i<<"" << j+1 <<
"" << k+1 <<"+";
              if  (i+2<=n  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<
""<<ac1+1<<""<<i+2<<"" << j+1
<< "" << k+1 <<"+";
              if  (j>0  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<  ""
<<ac1+1<<""<<i+1<<"" << j << ""
<< k+1 <<"+";
              if  (j+2<=n  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<  ""
<<ac1+1<<""<<i+1<<"" << j+2 <<
"" << k+1 <<"+";
              if  (k>0  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<
""<<ac1+1<<""<<i+1<<"" << j+1
<< "" << k <<"+";
              if  (k+2<=n  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<  ""
<<ac1+1<<""<<i+1<<"" << j+1 <<
"" << k+2 <<"+";
              }
            //outf <<"+";
```

```
            outf <<'\n';
            }

            }
      outf <<'\n'<<"Subject To"<<'\n';

      for( ac=0; ac<s-2; ac++)
        for( ac1=ac+3; ac1<s; ac1++)

        if (g[ac]=='1' && g[ac1]=='1')
        {
         for (i=0; i<n; i++)

         for (j=0; j<n; j++)

             for (k=0;k<n;k++)
          {


              {
           if (i>0 )
           {
                outf <<"a"<<br<<": "<<"x"<<ac1+1<<""<<i+1<<"" << j+1 << k+1<< "-"<<"Z"<<ac+1<<""<<i+1<<"" << j+1 << "" << k+1 << ""<<ac1+1<<""<<i<<"" << j+1 << "" << k+1 <<">=0 \n"; br = br+1;
                outf <<"a"<<br<<": "<<"x"<<ac1+1<<""<<i<<"" << j+1 << "" << k+1<< "-"<<"Z"<<ac+1<<""<<i+1<<"" << j+1 << "" << k+1 << ""<<ac1+1<<""<<i<<"" << j+1 << "" << k+1 <<">=0 \n"; br = br+1;
                }
           if (i+2<=n )
           {
                outf <<"a"<<br<<": "<<"x"<<ac+1<<""<<i+1<<"" << j+1 << "" << k+1 << "-"<<"Z"<<ac+1<<""<<i+1<<"" << j+1 << "" << k+1 << ""<<ac1+1<<""<<i+2<<"" << j+1 << "" << k+1 <<">=0 \n"; br = br+1;
                outf <<"a"<<br<<": "<<"x"<<ac1+1<<""<<i+2<<"" << j+1 << "" << k+1<< "-"<<"Z"<<ac+1<<""<<i+1<<"" << j+1 << "" << k+1 << ""<<ac1+1<<""<<i+2<<"" << j+1 << "" << k+1 <<">=0 \n"; br = br+1;
                }
           if (j>0 )
           {
                outf <<"a"<<br<<":
```

```
"<<"x"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << "-
"<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j << ""
<< k+1 <<">=0 \n"; br = br+1;
                outf
<<"a"<<br<<":
"<<"x"<<ac1+1<<""<<i+1<<"" <<
j << "" << k+1<< "-
"<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j << ""
<< k+1 <<">=0 \n"; br = br+1;
                }
            if (j+2<=n )
            {
                outf
<<"a"<<br<<":
"<<"x"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << "-"
<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j+2 <<
"" << k+1 <<">=0 \n"; br = br+1;
                outf
<<"a"<<br<<":
"<<"x"<<ac1+1<<""<<i+1<<"" <<
j+2 << "" << k+1 <<"-
"<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j+2 <<
"" << k+1 <<">=0 \n"; br = br+1;
                }
            if (k>0 )
            {
                outf
<<"a"<<br<<":
"<<"x"<<ac+1<<""<<i+1<<"" <<
j+1 << k+1<< "-
"<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 <<
""<<ac1+1<<""<<i+1<<"" << j+1
<< "" << k <<">=0 \n"; br = br+1;
                outf
<<"a"<<br<<":
"<<"x"<<ac1+1<<""<<i+1<<"" <<
j+1 << "" << k <<"-
"<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 <<
""<<ac1+1<<""<<i+1<<"" << j+1
<< "" << k <<">=0 \n"; br = br+1;
                }
            if (k+2<=n )
            {
                outf
<<"a"<<br<<":
"<<"x"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << "-"
<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j+1 <<
"" << k+2 <<">=0 \n"; br = br+1;
                outf
<<"a"<<br<<":
"<<"x"<<ac1+1<<""<<i+1<<"" <<
j+1 << "" << k+2 <<"-
"<<"Z"<<ac+1<<""<<i+1<<"" <<
```

```
j+1 << "" << k+1 << "" <<ac1+1<<""<<i+1<<"" << j+1 << "" << k+2 <<">=0 \n";  br = br+1;
                    }
              }
             //outf <<"+";
             }

             }
       br = 0;

       for (int d=0; d<s; d++)
       {
          br = br+1;
          outf <<"p"<<br<<": ";
              for (i=0; i<n; i++)
                for (j=0; j<n; j++)
           {
              for (k=0;k<n;k++)

              outf <<"x"<<d+1<<""<<i+1<<"" << j+1 << "" << k+1 <<"+";


          }
               outf <<"0 = 1"<<'\n';
       }


       br = 0;
          outf <<"n"<<br<<": "; // brooiach c



          for (i=0; i<n; i++)

            for (j=0; j<n; j++)

                for (k=0;k<n;k++)
             {


                 for (int d=0; d<s; d++)

                    outf <<"x"<<d+1<<""<<i+1<<"" << j+1 << "" << k+1 <<"+";


                    outf <<"0 <= 1"<<'\n';
                    br = br+1;
                    outf <<"n"<<br<<": "; // brooiach c
                }

          br=1;
                for (int d=0; d<s-1; d++)

          for (i=0; i<n; i++)
```

```
                                        outf
<<"c"<<br<<": "; // brooiach c
                                          }

                                    outf
<<'\n'<<"BINARY"<<'\n';

                                    for( ac=0; ac<s-2; ac++)
                                    for(  ac1=ac+3;  ac1<s;
ac1++)

                                     if     (g[ac]=='1'     &&
g[ac1]=='1')
                                    {
                                     for (i=0; i<n; i++)
                                     {
                                        outf <<"\n";

                                     for (j=0; j<n; j++)

                                          for (k=0;k<n;k++)
                                      {

                                            {
                                          if     (i>0     )     outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<
""<<ac1+1<<""<<i<<"" << j+1 <<
"" << k+1 <<" ";
                                              if  (i+2<=n  )  outf
<<"Z"<<ac+1<<""<<i+1<<""  <<
j+1  <<  ""  <<  k+1  <<
""<<ac1+1<<""<<i+2<<""  <<  j+1
<< "" << k+1 <<" ";
      for (j=0; j<n; j++)

           for (k=0;k<n;k++)
       {
                 outf
<<"x"<<d+1<<""<<i+1<<""     <<
j+1 << "" << k+1 <<"-";
                 if   (i>0    )   outf
<<"x"<<d+2<<""<<i<<""  <<  j+1
<< "" << k+1 <<"-";
               if   (i+2<=n    )   outf
<<"x"<<d+2<<""<<i+2<<""     <<
j+1 << "" << k+1 <<"-";
               if     (j>0     )     outf
<<"x"<<d+2<<""<<i+1<<""  <<  j
<< "" << k+1 <<"-";
               if   (j+2<=n    )   outf
<<"x"<<d+2<<""<<i+1<<""     <<
j+2 << "" << k+1 <<"-";
               if     (k>0     )     outf
<<"x"<<d+2<<""<<i+1<<""     <<
j+1 << "" << k <<"-";
               if   (k+2<=n    )   outf
<<"x"<<d+2<<""<<i+1<<""     <<
j+1 << "" << k+2 <<"-";

           }
                       outf  <<"0  <=
0"<<'\n';
                       br = br+1;
```

```
            if (j>0 ) outf
<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j << ""
<< k+1 <<" ";
            if (j+2<=n ) outf
<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j+2 <<
"" << k+1 <<" ";
            if (k>0 ) outf
<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 <<
""<<ac1+1<<""<<i+1<<"" << j+1
<< "" << k <<" ";
            if (k+2<=n ) outf
<<"Z"<<ac+1<<""<<i+1<<"" <<
j+1 << "" << k+1 << ""
<<ac1+1<<""<<i+1<<"" << j+1 <<
"" << k+2 <<" ";
            outf <<'\n';
             }
            //outf <<"+";
            }
            }
            }
             for (int d=0; d<s;
d++)
      { outf <<"\n";

            for (i=0; i<n; i++)
              for (j=0; j<n; j++)

             for (k=0;k<n;k++)
                    outf
<<"x"<<d+1<<""<<i+1<<"" <<
j+1 << "" << k+1 <<"  "<< '\n';
        }
           outf
<<'\n'<<"End"<<'\n';

                  outf.close();
      return                    0;
```

Третият математически модел беше реализиран на Java и пълно описание е дадено в Приложение в края на монографията.

Написан беше и софтуерен за визуализация на резултатите в програмна среда Matlab:

```
c = [31 32 68 104 140 176 212 206 170 134 98 62 61 25 26 27 33 34 28
64 63 99 135 141 105 69 70 106]
for i=1:1:28
   p(i) = c(i) - 1;
   z(i) = idivide(int32(p(i)),6^2)+1;
end
for i=1:1:28
  p(i) = p(i) - (z(i) - 1)*6^2;
  y(i) = idivide(int32(p(i)),6)+1;
end
for i=1:1:28
p(i) = p(i) - (y(i) - 1)*6;
x(i) = p(i) + 1;
end
disp(['z = ',num2str(z)])
disp(['y = ',num2str(y)])
disp(['x = ',num2str(x)])
plot3(x,y,z,'-o','MarkerSize',8),grid on
```

Един от примерите беше визуализиран и в 3DS Max. Предимство на визуализацията там е лесния поглед от различни гледни точки, а намирането на оптимално решение за всеки протеин се използва следния алгоритъм:

- Генериране на входен файл с разширение lp;
- Стартиране на CPLEX. Четене на данните с командата *read* и намиране на оптималния брой контакти между хидрофобни аминокиселини с командата *optimize*;
- Визуализация на данните.

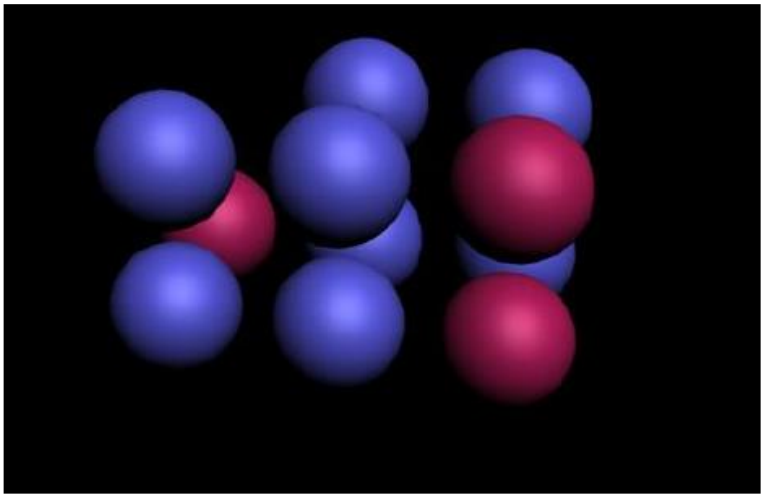
Представяме данните, получени от CPLEX за протеин с 31 аминокиселини: x

37_1	1.000000
x12_2	1.000000
x11_3	1.000000
x10_4	1.000000
x35_5	1.000000
x36_6	1.000000
x31_7	1.000000
x56_8	1.000000
x51_9	1.000000
x2_16	1.000000
x7_17	1.000000
x6_18	1.000000
x1_19	1.000000
x76_10	1.000000
x77_11	1.000000
x52_12	1.000000
x57_13	1.000000
x32_14	1.000000
x27_15	1.000000
x26_20	1.000000
z52_12_27_15	1.000000
z27_15_26_20	1.000000
z37_1_32_14	1.000000
z31_7_32_14	1.000000
z51_9_52_12	1.000000
z51_9_26_20	1.000000
z37_1_36_6	1.000000
z11_3_36_6	1.000000
z11_3_6_18	1.000000
z31_7_6_18	1.000000

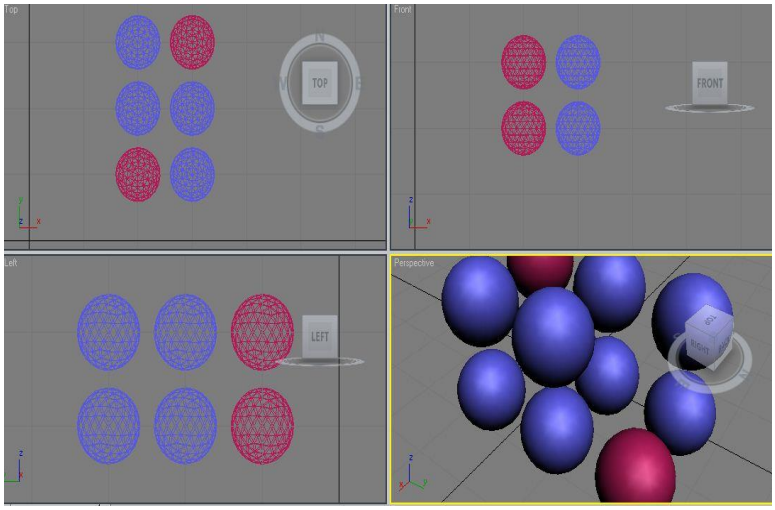
Променливите z представят връзките между хидрофобните аминокиселини. Графично представяне на данните е показано на фиг. 1.16 – 1.20.

Представяне на графичните резултати

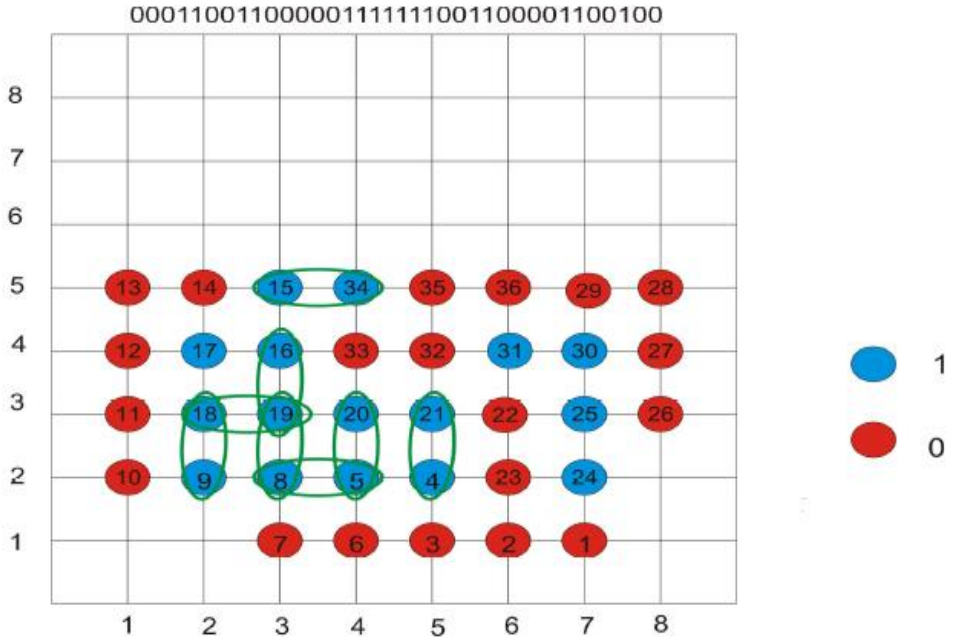
След успешна оптимизация CPLEX намира координатите на аминокиселините и на следващите фигури ще покажем някои решения:



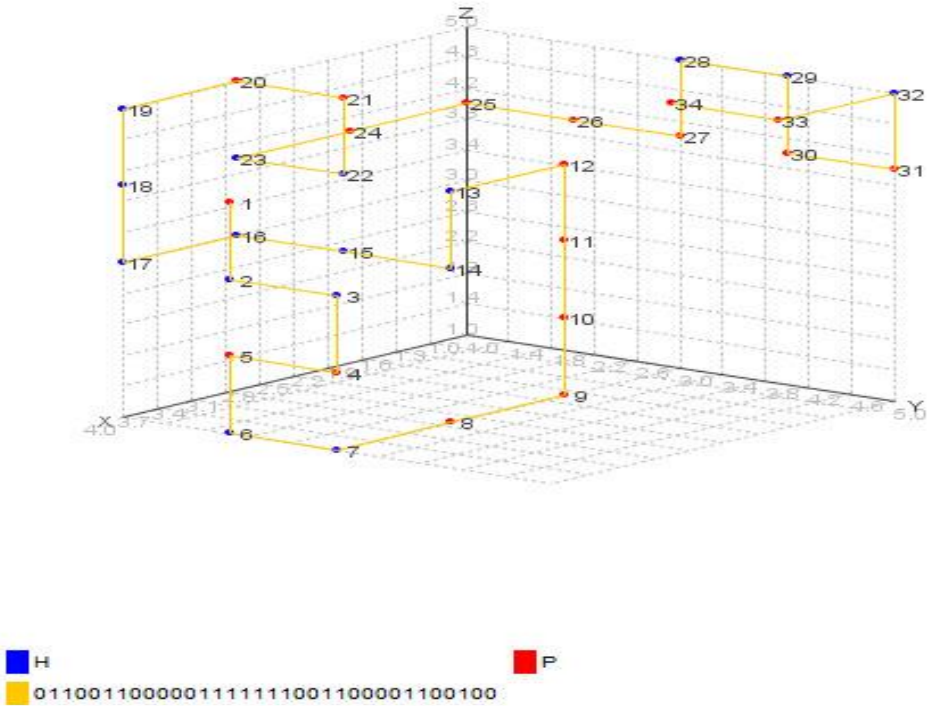
Фиг. 1.16. Решение за протеин с последователност 1101010101.



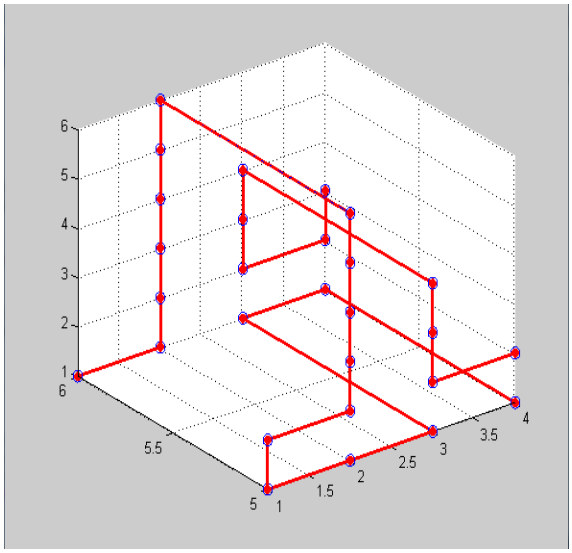
Фиг. 1.17. Решение за протеин с последователност 1101010101 и визуализация с програмата 3Ds Max.



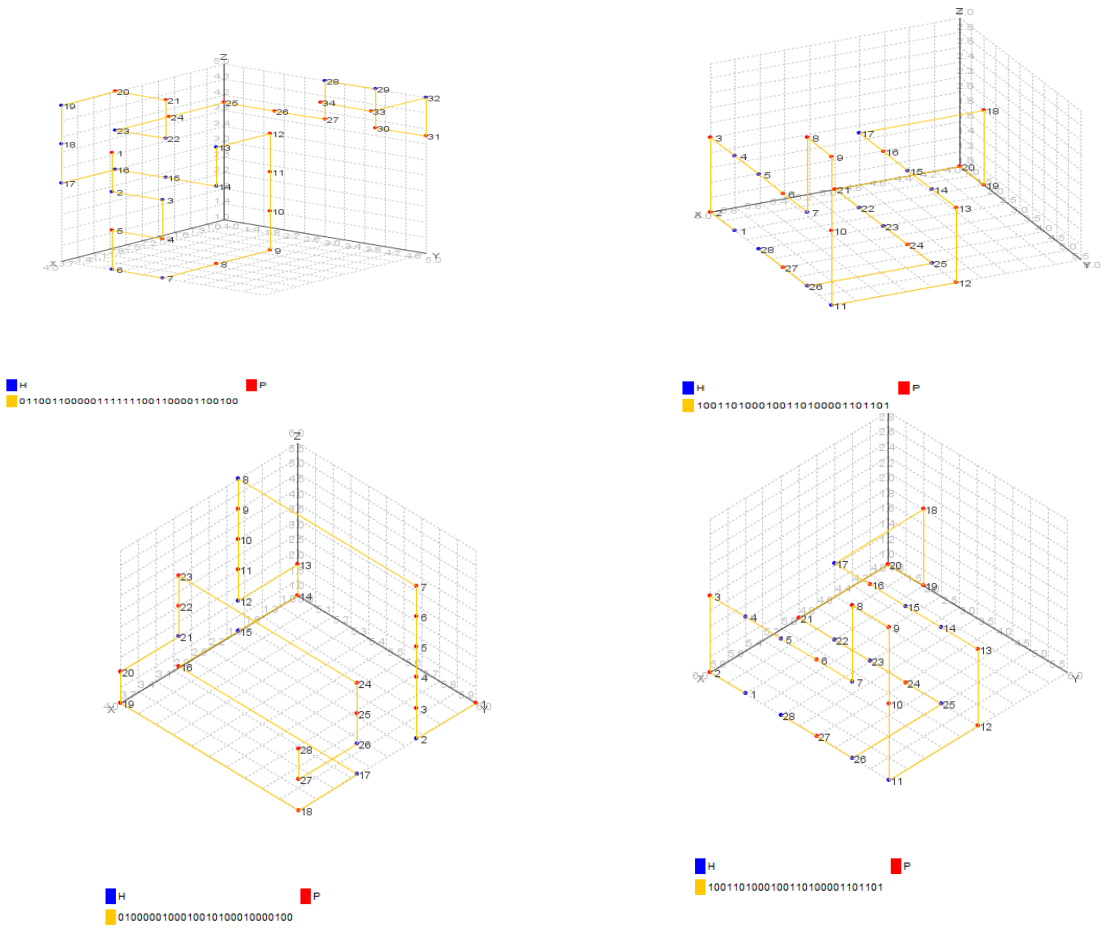
Фиг. 1.18. Решение за протеин с последователност 0001100110000011111111001100001100100.



Фиг. 1.19. Решение за протеин с последователност 10100110100101100101 върху тримерна решетка



Фиг. 1.20. Визуализация на протеини върху 3D решетка с Matlab



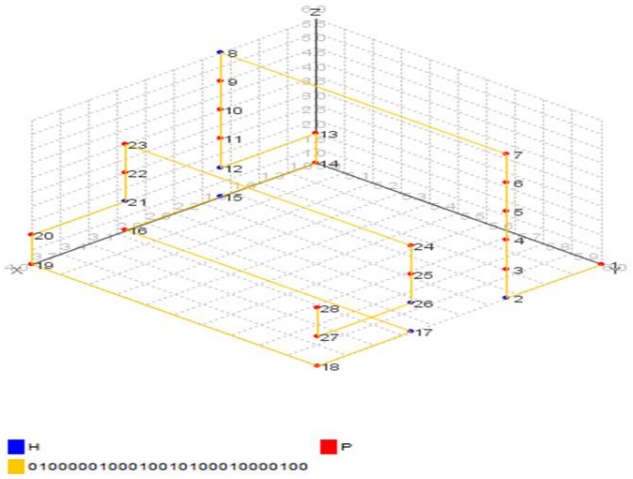
Фиг. 1.20 а. Визуализация на протеини върху 3D решетка с Matlab

За проверка на модела избрахме два реални протеина от биологични бази от данни. Превърнахме първичната им структура в HP последователност и изчислихме оптималната им 3D структура и я сравнихме с реалната.

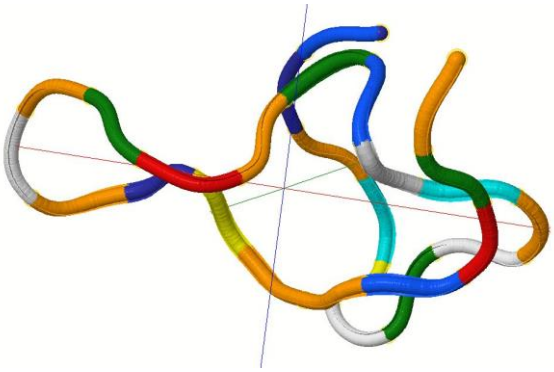
Първичната структура на двата протеина е:

SLDRSSCFTGSLDSIRAQSGLGCNSFRY – orexin peptide;

FSGPPGLQGRLQRLLQASGNHAAGILTM – hypotensive hormone.

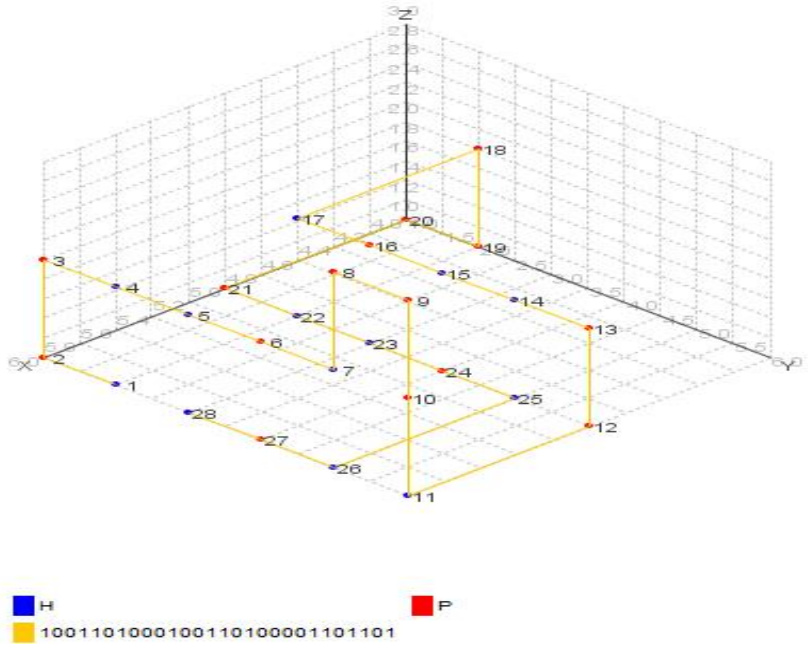


Фиг. 1.21. Графична презентация на протеин orexin peptide (1ANP)

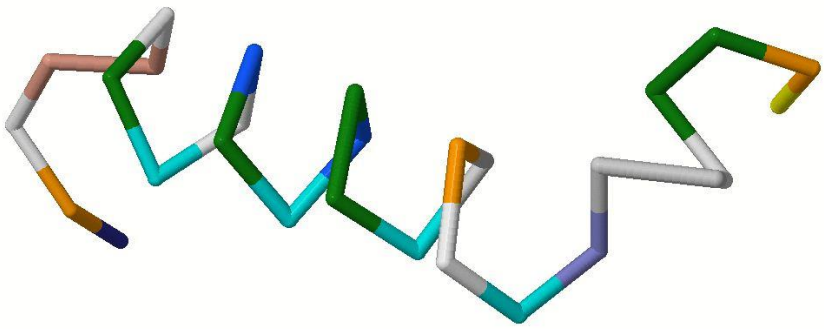


-

Фиг. 1.22. Реалната 3D структура на протеин orexin peptide (1ANP)



Фиг. 1.23. Графична презентация на протеин hypotensive peptide (1CQ0)



Фиг. 1.24. Реалната 3D структура на протеин hypotensive peptide (1CQ0)

От направеното сравнение от фиг. 1.21 до фиг. 1.24 за hypotensive peptide може да се направят следните изводи:

Предсказване на 3D структурата на протеин с HP folding е реална и може да се използва в практиката.

Този начин на моделиране трябва да се комбинира с модели базиращи се върху молекулната механика за по голяма точност.

Направени бяха експерименти да изчислим реалната структура на протеин с 199 аминокиселини, но сървърът спря поради големия входен файл – 400 MBt.

1.5. Използване на Евристичен модел[4]

В този раздел се разглежда разработения модел за предсказване на триизмерна форма на белтъците, разновидност на HP модела, при който:

1. Аминокиселините се разполагат с триизмерни целочислени координати, започвайки от (0,0,0) и (0,0,1), следвайки номерата в аминокиселинната последователност, като на случаен принцип се избира позицията на всяка следваща от възможните 5 посоки – по x,y,z в отрицателна и в положителна посока, без обратно в предишна позиция. Използва се генератор на случайни числа на средата за програмиране на C++, като се взима остатъкът от делене на 6 и нека това бъде числото r и тогава:

Ако r=0, то $x_{i+1}=x_i+1$, $y_{i+1}=y_i$, $z_{i+1}=z_i$

Ако r=1, то $x_{i+1}=x_i-1$, $y_{i+1}=y_i$, $z_{i+1}=z_i$

Ако r=2, то $x_{i+1}=x_i$, $y_{i+1}=y_i+1$, $z_{i+1}=z_i$

Ако r=3, то $x_{i+1}=x_i$, $y_{i+1}=y_i-1$, $z_{i+1}=z_i$

Ако r=4, то $x_{i+1}=x_i$, $y_{i+1}=y_i$, $z_{i+1}=z_i+1$

Ако r=5, то $x_{i+1}=x_i$, $y_{i+1}=y_i$, $z_{i+1}=z_i-1$

където $(x_{i+1}, y_{i+1}, z_{i+1})$ са координатите на новодоставена

[4] **Данните** в тази глава се разработени с моя докторант Иван Тодорин, който защити успешно дисертацията "IN SILICO ПРЕДСКАЗВАНЕ НА ТРИИЗМЕРНАТА СТРУКТУРА НА ПРОТЕИНИТЕ ЧРЕЗ ЕВРИСТИЧНИ АЛГОРИТМИ", представеният модел е публикуван в I.Todorin, I. Trenchev. An Off-Lattice HP Model for Protein folding with three componential evaluation function Proceedings of the 15th International Conference for Informatics and Information Technology, 2018, 18-21, ISBNT978-608-4699-08-8

аминокиселина, а (x_i, y_i, z_i) са координатите на предишни поставени аминокиселини.

Разстоянието между съседни аминокиселини е единица нормирано разстояние, а формата на протеина се представя опростено единствено чрез тези координати и под разполагане на аминокиселините ще се разбира разполагане на гръбначна структура на пептидна верига *(Michalewicz & Fogel, 2004; Thilagavathi & Amudha, 2015; Traykov et al., 2016).*

2. При разполагане на аминокиселините се спазват следните ограничения:

- не може да се разполагат на координати, които вече са заети от друга аминокиселина:

$$(x_i=x_j)\cap(y_i=y_j)\cap(z_i=z_j)$$

където (x_i, y_i, z_i) са координатите на ново поставената аминокиселина, а (x_j, y_j, z_j) са координатите на всяка от поставените до момента аминокиселини;

- не може да се разполагат по-далеч от центъра на молекулата от разстояние, зададено от променливия параметър за допустимо разпростиране, по всяка от трите ортогонални посоки:

$$(x_i-x_c)^2+(y_i-y_c)^2+(z_i-z_c)^2<s.l,$$

където (x_i, y_i, z_i) са координати на поставяната аминокиселина, (x_c, y_c, z_c) са координати на центъра на молекулата построена до момента, l е дължината на пептидната верига, s е променливият параметър за допустимо разпростиране;

- при получаване на недопустима позиция, се задава нова посока и така докато се изчерпят всичките 5 възможности. При невъзможност за разполагане на следващата аминокиселина, се прекратява построяване на текущата конформация, актуализира се броячът на провалените конформации и се започва следваща. Това се осъществява в цикъл с определен брой изпълнения, вложен в друг цикъл с определен брой изпълнения, като процента на

проваленитe конформации (възможни триизмерни форми) при изпълнение на вътрешния цикъл определя нова стойност на променливия параметър за допустимо разпростиране за следващата итерация на външния цикъл.

3. За да се формира стойността на оценъчната функция, следва да се определи кои аминокиселини са в контакт – ако имат съседни координати. Проверката за близост се дефинира с формулата:

$$(x_i-x_j)^2+(y_i-y_j)^2+(z_i-z_j)^2=1,$$

където (x_i, y_i, z_i) и (x_j, y_j, z_j) са координатите на двете аминокиселини.

Оценъчната функция е сума на стойности на хидрофобност на аминокиселини във всеки контакт плюс коефициент за отчитане на взаимодействие между водни молекули:

$$SF = \sum_{i,j\in C}^{n} Hi + Hj + w$$

където SF е стойността на оценъчна функция, w е коефициент за отчитане на взаимодействие между водни молекули Hi и Hj са стойности на хидрофобност на аминокиселините с номера i и j, а C е множество на двойките в контакт.

В сравнение с класическия HP модел, този има възможност за гъвкаво изменяне на ограничението за разпростиране в пространството, вместо фиксиран размер на решетката, което позволява да се намери по-добър баланс между процента провалени конформации, поради недостатъчно място, и построяване на огромен брой конформации с ниска оценъчна функция, поради прекалено много място *(Shmygelska & Hoos, 2005; Skolnick & Kolinski, 1990; Song, et al., 2006; Sternberg & Thornton, 1977; Strickland, et al., 2005)*. Също така, оценъчната функция позволява да се отчита ролята на всички контакти, като водещата роля остава на хидрофобните.

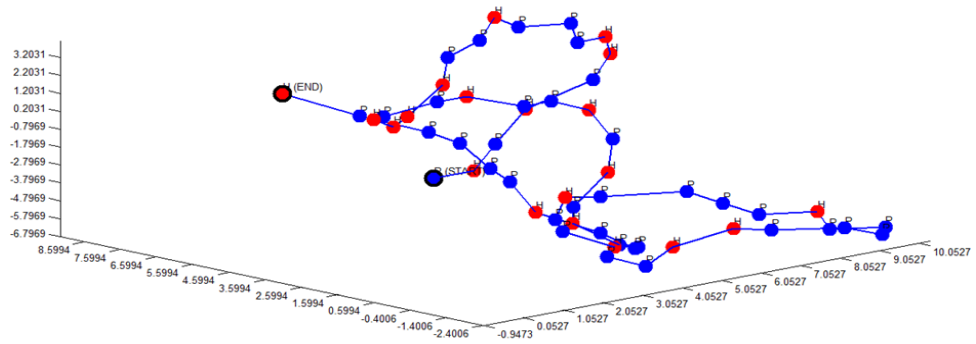
Използва се генератора на случайни числа на средата за програмиране на C++, като се взима остатъкът от делене на 72 и нека това бъде числото r, а r1=r/12 и r2=rmod12, тогава:

$$x_i = x_{i-1} + \sin(90 - 30r1),$$
$$y_i = y_{i-1} + \sin(30r2)\cos(90 - 30r1),$$
$$z_i = z_{i-1} + \cos(30r2)\cos(90 - 30r1),$$

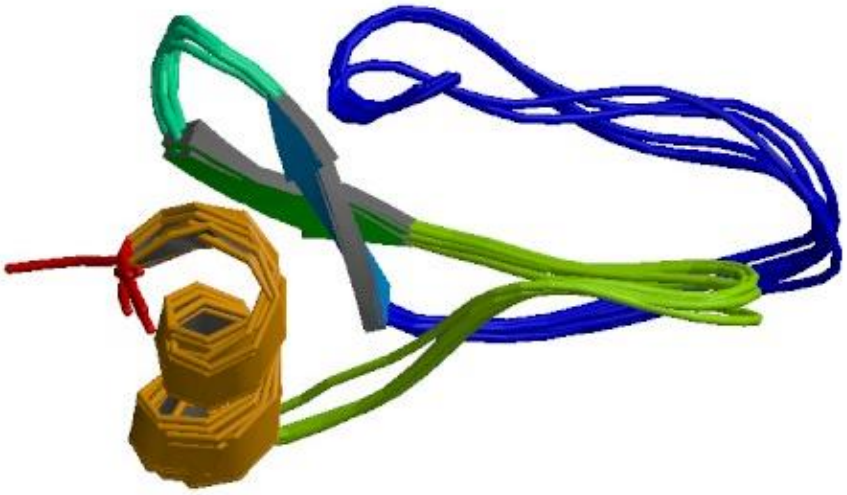
където (x_i, y_i, z_i) са координати на новопоставената аминокиселина, а (x_{i-1}, y_{i-1}, z_{i-1}) и (x_{i-2}, y_{i-2}, z_{i-2}) са координати на предходните аминокиселини.

Разлика при анализа с предишната модификация е, че по-голямата свобода на завиване дава по-голям брой възможни структури, така че вече параметър брой намиране на структурата с най-висока оценъчна функция няма да бъде следен. Поради наблюдаваното при досегашни опити преобладаващо значение на параметъра процент провалени пред стъпката и началната стойност на коефициента на разпростиране **s**, то той ще се изменя. Също така ще има два варианта на оценъчна подфункция за хидрофобните контакти – с параметър за влиянието на водата **wtw = 8** и **wtw = 0** – в първия случай контактите между полярни аминокиселини носят положителна стойност, макар и значително по-малка от хидрофобните, докато във втория вариант носят отрицателна стойност.

При 90% провалени и wtw = 8 се получи следната най-добра структура със 179 контакти и 26 съвпадащи със сниманата структура със 125 контакти.



Фиг. 1.25. Протеин 1UUB – добре изразена третична структура с хидрофобни контакти



Фиг. 1.26. Реалиня протеин 1UUB

Извършените експериментални симулации имат за цел, от една страна, да се намерят най-подходящите параметри за намиране на конформация с най-висока оценъчна функция *(Wind, et al., 1968; Wiers, et al., 2018, Rodriguez-Tello et al., 2018)* и брой повторно намиране на същата, а от друга страна да се провери доколко получените „най-добри" конформации съответстват на действителната триизмерна форма (верификация на модела) – за целта са намерени съседните номерирани двойки аминокиселини, които са в близост (контакт) по същия критерий за същия изследван протеин, чиито координати са взети от PDB файла в базата данни, получени чрез ядреномагнитен резонанс.

Данните се отнасят до изследване на протеина 1UUB с 56 аминокиселини – (PDB файла има 19 модела), чиито координати и контакти, както и обяснение за избора му, са дадени в предходната глава, където изследването е върху същия протеин, за да са равноправно съпоставими данните.

Първоначално, съобразно първата цел – намиране на конформация с най-висока оценъчна функция при различни параметри и едно и също изчислително време – в случая изпробването на 1,000,000,000=10^9 случайни конформации (при

възможни в този модел 36^{56}), изискването за максимално разстояние между аминокиселини за формиране на контакт е заложено да бъде 1,4 пъти нормирано разстояние, а не 2 пъти, както е дадено в описание на модела, тъй като следващите изследвания по втората цел (верификация на модела) показват, че за контакт трябва да се счита 2 пъти нормираното разстояние – достатъчно близо за формиране на всички връзки, понижаващи общата енергия. Независимо от това, тези резултати служат за постигане на първа цел.

Експерименталните данни започват с построяване на конформации при начална стойност на променливия параметър за допустимо разпростиране **s=0,5** и стъпка на увеличение **st=0,01** – увеличението настъпва при по-висок процент на провалени конформации в рамките на вътрешния цикъл от 1000000 конформации, като тази проверка във външния цикъл се извършва 1000 пъти – т.е. стойността на параметъра бързо ще достигне и ще се задържи на тази, при която процентът на провалени конформации е колкото зададения.

При **s=0,5, st=0,01** и **91%** провалени конформации се получава следната структура:

- списък на координати на аминокиселините:

0,0,0,0
1,1,0,0
2,2,-0.984406,-0.175913
3,3,-0.537128,0.0787322
4,4,0.0283648,1.14581
5,5.36616,0.810881,2.17935
6,6.54924,0.560844,1.92882
7,7.64078,0.263833,0.81846
8,7.202,0.290457,-0.236721
9,6.48353,1.17033,-1.26431
10,5.13975,1.78539,-2.27811
11,3.96786,2.09292,-1.785
12,2.88192,1.30612,-1.19884
13,1.83895,-0.0278493,-0.566157
14,0.976355,-1.19483,0.690212
15,-0.096847,-1.01154,1.8184
16,-0.135748,-1.78725,2.88249
17,-0.499297,-3.11403,3.91453
18,-0.180612,-4.27742,5.29632
19,0.153076,-3.87444,6.48721
20,0.81762,-4.54029,7.58265
21,0.505546,-5.63795,8.63038
22,0.990193,-6.68679,8.38643
23,0.891411,-7.7112,9.20449
24,1.70818,-8.72341,10.1133
25,2.88078,-9.72951,9.92273
26,3.96709,-9.59005,10.5937
27,5.01024,-8.53535,10.7564
28,6.51608,-7.50801,10.6611
29,7.769,-7.63563,9.84614
30,8.89546,-8.19783,8.57171
31,8.81678,-7.71215,7.43449
32,8.27561,-8.33427,6.36588
33,8.50548,-7.77957,5.33158
34,8.27632,-8.44115,4.31443
35,9.0263,-9.27447,3.30585
36,8.41619,-9.86312,2.30156
37,7.46923,-9.39067,1.29942
38,7.33835,-8.21496,0.298347
39,6.77383,-7.1271,0.664366
40,6.66278,-6.08317,-0.137859
41,6.94986,-5.06121,0.40051
42,6.10885,-4.05023,0.844823
43,6.33147,-3.04474,1.83274
44,6.09869,-2.04199,1.38776
45,4.99775,-1.04062,1.34041
46,3.94728,-0.198079,2.25648
47,2.92204,0.224784,1.71452
48,1.90943,-0.20508,0.676243
49,1.05902,-1.35895,-0.342895
50,-0.233137,-2.43588,-1.35085

51,-1.37921,-2.10779,-
2.35391
52,-2.45225,-2.12044,-
1.87118
53,-3.48877,-2.29876,-
2.61491
54,-4.50703,-2.38633,-
3.98677
55,-4.0319,-2.60681,-5.1727

- стойност на оценъчната функция – 1704
- s=1.12
- брой пъти намерена същата - 76
- брой хидрофобни контакти, брой дисулфидни мостове и списък на двойки хидрофобни аминокиселини в контакт:
- h-h 12
- s-s 1

1,13	2,13	3,48
1,14	2,14	13,48
1,48	2,48	14,48
1,49	2,49	14,49

- общ брой контакти и списък на двойки аминокиселини в контакт:

14

1,13	2,14	4,47
1,14	2,48	13,48
1,48	2,49	14,48
1,49	3,48	14,49
2,13	4,46	

При тази структура прави впечатление големия брой повторения на намерената структура – това дава по-голяма вероятност това да е най-добрата възможна.

При **s=0,5, st=0,01** и **90%** и при **92%** провалени конформации се получава следната структура:

- списък на координатите на аминокиселини:

0,0,0,0
1,1,0,0
2,2,0.49977,0.866158
3,3,0.749655,2.29924
4,4.36456,0.372071,3.51578
5,5.54684,1.18328,4.12484
6,6.63798,2.52891,4.08827
7,7.00764,3.70172,3.08558
8,7.8356,4.78813,3.34999
9,7.48228,5.83133,2.8409
10,7.80332,6.85293,1.71901
11,8.60697,8.1295,0.658064
12,8.83288,9.26778,-0.856815
13,8.44399,8.97195,-2.11425
14,7.26445,8.65205,-3.24297
15,6.17468,7.84715,-3.04311
16,5.1298,7.268,-1.95892
17,4.26326,6.03949,-0.916819

18,4.59626,4.92523,0.246747
19,4.11842,3.60337,1.32853
20,4.52018,2.44244,1.10162
21,4.8954,1.36197,1.97284
22,4.21606,0.321742,1.91007
23,3.87401,-0.698374,0.878683
24,2.83602,-1.70682,-0.13701
25,1.81703,-1.8677,-1.58406
26,2.17369,-1.44836,-2.80759
27,2.69163,-2.17926,-3.91935
28,2.9514,-1.54471,-4.97524
29,4.06555,-1.40414,-6.00318
30,5.12262,-0.991994,-5.57739
31,6.15116,-1.55373,-4.72382
32,7.16543,-1.49124,-5.23624
33,8.17256,-2.40057,-5.15284
34,9.17613,-2.35408,-5.9765
35,10.1779,-1.33084,-6.38754
36,9.69371,-0.319219,-6.76505
37,10.3178,0.686592,-6.45403
38,9.76283,1.6895,-6.79691
39,10.4251,2.69095,-6.6265
40,9.77172,3.69168,-6.36617
41,8.50473,4.69204,-6.57635
42,7.37123,4.42442,-6.04076
43,6.30448,3.52587,-6.41732
44,5.27111,2.20925,-6.10789
45,5.51864,1.05094,-6.59813
46,5.14333,-0.02822,-5.9767
47,4.09031,-1.0678,-5.16483
48,3.0638,-1.24573,-3.81915
49,2.20944,-0.394675,-2.6463
50,2.54853,0.67337,-1.55988
51,1.95078,0.566097,-0.51667
52,1.99451,1.45194,0.504935
53,1.14942,1.39647,1.51574
54,0.0849734,2.13552,2.52114
55,-0.947252,3.14756,3.79011

- стойност на оценъчна функция – 1201.2

-s=1.08

- брой пъти намерена същата – 1

- брой хидрофобни контакти, брой дисулфидни мостове и списък на двойки хидрофобни аминокиселини в контакт:

h-h 5
s-s 2
2,52 26,48 **27,48**
3,22 26,49

- общ брой контакти и списък на двойки аминокиселини в контакт:

13
1,51 26,48 29,47
2,51 26,49 30,46
2,52 **27,48** 30,47
2,53 28,47
3,22 28,48

При тази структура прави впечатление, че е намерена само един път и е с по-ниска стойност – следователно ефективността е много чувствителна към параметъра за процент провалени конформации и точно при 91% се получава най-висока стойност. Дадените с удебелен шрифт двойки аминокиселини са съседни и в структура, получена чрез ядреномагнитен резонанс.

Следващият експеримент е свързан с търсенето на формата при започване от ниска стойност на s и st и по-нисък процент провалени конформации, за да може изчислението да премине при различно ограничение и плавно увеличение на s:

При **s=0,4, st=0,001** и **90%** провалени конформации се получава следната структура:

- списък на координати на аминокиселините:
0,0,0,0

1,1,0,0
2,2,0.34335,-0.939207
3,3,1.0148,-0.542653
4,4,2.1173,-0.986282
5,5.14068,3.1685,-1.9759
6,6.21103,3.1916,-1.60615
7,7.2462,2.33824,-1.92311
8,8.26378,1.0441,-1.58389
9,7.83228,-0.102841,-1.75464
10,8.11423,-1.17635,-2.70736
11,7.75612,-2.21311,-2.31717
12,7.40429,-3.23149,-1.13711
13,6.28808,-4.08104,-0.0470815
14,5.22997,-4.6778,-0.487165
15,4.20092,-3.99122,-0.879984
16,3.18639,-3.64793,-0.0763932
17,2.17913,-3.13294,-0.613805
18,1.1755,-3.37797,-0.0179497
19,0.173683,-4.48572,0.451186
20,-0.66833,-5.53959,1.62578
21,-0.104514,-5.89297,2.71308
22,0.519997,-5.13018,3.75672
23,1.0066,-3.7641,4.77855
24,1.58951,-2.58106,4.34889
25,1.11366,-1.48954,3.49277
26,1.05009,-0.443775,4.04939
27,0.516465,0.579105,3.46274
28,0.42086,1.59055,2.18418
29,0.712665,1.1557,1.0449
30,1.85857,0.938274,-0.024737
31,2.93152,-0.109371,-0.903655
32,3.96799,-1.61843,-1.17191
33,3.54703,-2.87296,-0.962681
34,2.39624,-4.00022,-1.19843
35,1.32085,-5.06224,-2.18325
36,0.283158,-5.59326,-1.67566
37,-0.735689,-5.85876,-0.421873
8,-0.30536,-5.64966,0.705027
39,0.775963,-5.04534,1.76848
40,1.81662,-4.24341,3.16636
41,3.27671,-3.50059,4.3653
42,4.50675,-3.12759,3.96477
43,5.62177,-3.58238,2.99721
44,6.67928,-4.45107,1.74614
45,7.38238,-3.90074,0.620602
46,7.90514,-4.6108,-0.442166

47,7.30116,-4.46469,-1.47355
48,7.49963,-3.52587,-2.48924
49,8.24199,-2.2907,-3.49709
50,7.62807,-1.17311,-4.173
51,8.30593,-0.114322,-4.3374
52,7.70566,0.915074,-4.07625
53,7.90598,1.92977,-3.07991
54,7.5043,2.93712,-3.44671
55,8.24322,3.9408,-3.28825
- стойност на оценъчната функция – 1663.
-s=0.4999
- брой пъти намерена същата – 5
- брой хидрофобни контакти, брой дисулфидни мостове и списък на двойките хидрофобни аминокиселини в контакт:

h-h 8
s-s 1
1,30
2,30
2,31
3,30
3,31
11,48
11,49
13,45

- общ брой контакти и списък на двойки аминокиселини в контакт:

9
8,49
15,36
17,36
21,38
22,38
23,39
26,52
27,51
27,52

При тази структура се оказва, че най-висок резултат е постигнат при ниска стойност на s, а стойността на оценъчна функция и броя повторения заемат средно положение между предишните два опита. Интерес представлява и фактът, че двете структури с най-висока до момента оценъчна функция – първата и третата, имат 1 дисулфиден мост, голям брой хидрофобни контакти и съвсем малък брой нехидрофобни. За сметка на това, втората структура има два дисулфидни моста (както и получената

от ЯМР), малко хидрофобни контакти и много нехидрофобни – въпреки по-ниската стойност на оценъчна функция, тя има повече съвпадащи двойки аминокиселини с тази, получена чрез ЯМР. Това поставя въпроса за бъдещо усъвършенстване на оценъчната функция – засега тя просто отчита създаване на всички връзки, но теглата им в общата стойност ще трябва да се прецизират. При това изглежда, че ще трябва да се намали теглото на хидрофобното взаимодействие – а тази под функция е цялата оценъчна функция при стандартния HP модел.

1.6. Използване на High-performance computing HPC

Функционалната теория на плътността (Ren, DFT) е метод за изчисляване на електронна структура на много системи от частици в квантовата физика и квантовата химия[5]. По-специално, той се използва за изчисляване на електронна структура на молекули и кондензирана материя. Това е един от най-широко използваните и универсални методи в изчислителна физика и изчислителна химия. Твърдото тяло се разглежда като система, състояща се от голям брой електрони, еднакво взаимодействащи помежду си, държани заедно от решетка от атомни ядра. Основната идея на метода е да се използва концепция за електронна плътност в основно състояние, нейното разпределение се описва от едно частно уравнение на Шрьодингер *(Bernevig, 2019; Shi, Weiqun 2020; Ren et al., 2019).*

Традиционните методи за определяне на електронна структура, по-специално методът Хартри-Фок и неговите производни, описват система, използвайки мулти-електронна вълнова функция. Основна цел на теорията за плътността на функциите е да замени многоелектронна вълнова функция с електронната плътност, когато описва електронна подсистема. Това води до значително опростяване на проблема, тъй като зависи от много електронна вълнова функция 3N променливи – 3

[5] **Donald** J. Kouri, in Encyclopedia of Physical Science and Technology (Third Edition), 2003: Available at: https://www.sciencedirect.com/topics/chemistry/quantum-chemistry. Last Accessed: 20.09.2020

пространствени координати за всеки от N електрони, докато плътността е функция само от три пространствени координати *(Milin et al., 2002; Yamaguchi et al., 2017; Liang et al., 2015).*

Като правило методът на теория на плътност на функционалност се използва заедно с формализма на Кон - Шам, при който неразрешимата задача за описване на няколко взаимодействащи електрона в статично външно поле (атомни ядра) се свежда до по-опростен проблем на независими електрони, които се движат в някакъв ефективен потенциал. Този ефективен потенциал включва статичния потенциал на атомни ядра и също така взема предвид кулоновски ефекти, по-специално обменно взаимодействие и електронна корелация.

Описание на последните две взаимодействия е основна сложност на метода на теория за плътността във функционалната форма на Кон - Шам. Най-простото приближение тук е приближение на локална плътност, основаващо се на точно изчисляване на обменната енергия за пространствено равномерен електронен газ, което може да се извърши в рамките на модела на Томас - Ферми и от което може да се получи и корелационна енергия на електронния газ.

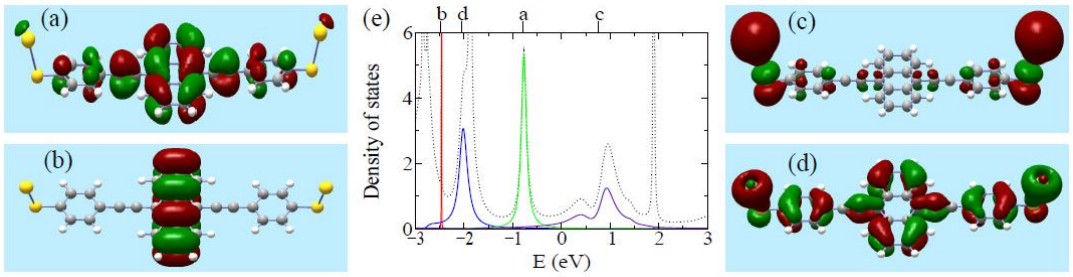
Методът на функционална теория за плътност се използва широко за изчисления във физика на твърдото тяло от 70-те години. В някои случаи дори използваме просто приближаване на локална плътност дава задоволителни резултати, които съответстват на експериментални данни, а изчислителната сложност на метода е ниска в сравнение с други подходи към проблема с много частици в квантова механика. Независимо от това, дълго време методът не е бил достатъчно точен за изчисления в областта на квантова химия, докато през 90-те години на миналия век не се забелязва значително изменение в описание на обменните и корелационните взаимодействия. Понастоящем методът на теория на плътността е основния подход и в двете области. Въпреки това, в теорията, все още има проблеми при прилагане на метода към описание на

между молекулни сили, по-специално на сили на Ван дер Ваалс и дисперсионно взаимодействие, както и при изчисляване на пролуката на лентата в полупроводници.

Трудностите при изчисляване на дисперсионно взаимодействие в рамки на теория за функционална плътност (които възникват, поне в случая, когато този метод не се допълва от други) правят метода на теория на плътността функционален, неподходящ за системи, в които преобладават дисперсионни сили (например, когато се обмисля взаимодействие между атоми на благородни газове) или системи, в които силите на дисперсия са в същия ред като другите взаимодействия (например в органични молекули). Решението на този проблем е предмет на съвременни изследвания.

Метод: **ab initio** квантова химия (DFT) и техники на оптимизационна функция.

$G = \frac{2e^2}{h}\sum_i T_i$ - Формула на Ландауер,



Фиг. 1. 27. Схематично представяне на енергията

Теорията на Хартри-Фок е добре установен подход за получаване на приблизително решение на многоелектронно уравнение на Шрьодингер, като се напише обща (фиг. 1.27) (електронна) вълнова функция на система като единствен детерминант на Слейтер, съдържащ едноелектронни орбитали *(Kostova 2005; Kolodziej et al., 1997; Foresman et al., 1996) Becke, 1988)*. В самостоятелния метод на Hartree-Fock за изчисляване на

електронни състояния на многоелектронни системи, ние обяснихме също вариационна процедура на Hartree-Fock Roothaan[6] и теорията за смущения на Møller-Plesset [7](MP2). Взимайки вълновата функция на многоелектронна система като определящ фактор, липсващият термин в SCF модела на Хартри (т.е. обменно взаимодействие) може да се вземе предвид в приближението на Хартри-Фок *(Poshusta 1975).* Прилагането на вариационния принцип към детерминанта на Слейтер води до набор от N уравнения, известни като уравнения на Хартри-Фок *(Bailly et al., 2004; Thijssen 1999; Szabo & Ostlund, 1996).* Тогава се въвежда базисен набор и той преобразува пространствените интегро-диференциални уравнения на Хартри-Фок в затворен корпус в набор от алгебрични уравнения, известни като уравнения на Ротоан *(Barzegar et al., 2011; Marković et al., 2016).*

За да се изпълни антисиметричния принцип, вълновата функция за N-електронна система се взема във форма на детерминанта на Слейтър[8] и използвайки приближението на Борн-Опенхаймер[9], Хамилтонианът на многоелектронната система е даден като:

$$\mathrm{H} = -\sum_{i=1}^{\mathrm{N}} \frac{1}{2}\nabla_i^2 - \sum_{i=1}^{N} \sum_{A=1}^{M} \frac{Z_A}{r_{iA}} + \sum_{i=1}^{N} \sum_{j>i}^{N} \frac{1}{r_{ij}}$$

Методът на Хартри-Фок търси онези орбитали χ_i, които свеждат до минимум интеграцията на променлива енергия Eq. (1) и

---

[6] **Joy,** Hubert & Schaad, L. & Handler, George. (1967). Hartree—Fock—Roothaan Wavefunctions for Lithium. The Journal of Chemical Physics. 46. 10.1063/1.1840505.

[7] **Krzysztof** Wolinski, Harrell L. Sellers, Peter Pulay, Consistent generalization of the Møller-Plesset partitioning to open-shell and multiconfigurational SCF reference states in many-body perturbation theory, Chemical Physics Letters, Volume 140, Issue 3, 1987, Pages 225-231, ISSN 0009-2614. Accessable on:
<https://indico.in2p3.fr/event/2481/contributions/24469/contribution.pdf>

[8] **Определение** за Слейтър детерминантата. Достъпен на:
<https://bg.termwiki.com/BG/Slater_determinant> Последно посетен на: 09.09.2020 г.

всяка орбитала се приема за нормализирана $\langle\phi|\phi\rangle = 1$, т.е. след това използвайки Eq. (1) и уравнение. (2), получаваме:

$$E = \langle\Psi|H|\Psi\rangle = \sum_{a=1}^{N} h_{aa} + \frac{1}{2}\sum_{a=1}^{N}\sum_{b>a}^{N}(J_{ab} - K_{ab}) \qquad (3)$$

където, $\mathbf{h}_{aa}$ е средната кинетична и ядрена енергия на привличане на електрона, описана от вълновата функция $\Psi_a(r_1)$, J_{ab} е класическо отблъскване между облаци $|\Psi_a(r_1)|^2$ на заряда и $|\Psi_b(r_2)|^2$ се нарича кулонов интеграл или потенциал на електростатично отблъскване K_{ab}, възникващ поради електрона b при неговото положение се осреднява по пространство и се нарича обменния интеграл, който възниква поради асиметрия на общата вълнова функция и изисква спиновете на електрони a и b да бъдат успоредни *(Scheicher 2004; Pilar et al., 1968).*

За атоми и молекули със конфигурация на затворена обвивка, енергийния израз Eq. (3) може да се запише като:

$$E = 2\sum_{a} h_{aa} + \sum_{a=1}\sum_{b=1}(2J_{ab} - K_{ab})$$

Минимизиране $E[\{\chi_a\}]$ по отношение на спиновите орбитали, при условие че ограниченията на спинови орбитали остават ортогонални.

$$F(1)|\chi_a(1)\rangle = \sum_{b=1}^{N}\varepsilon_{ba}|\chi_a(1)\rangle$$

Където $F(1) = h(1) + \sum_{b=1}^{N}\{J_b(1) - K_b(1)\}$, е известен като оператора Fock. Екв. (4) са известни като уравнения на Хартри-Фок. Използвайки орбиталите на Хартри-Фок, където $\psi_i = \sum_{\mu} C_{\mu i}\varphi_{\mu}$ и

$i = 1,2,...k$ пространствено интегро-диференциално уравнение, получаваме:

$$\sum_{\mu} F_{\nu\mu} C_{\mu i} = \varepsilon_i \sum_{\mu} S_{\nu\mu} C_{\mu i}: \qquad i=1, 2, \ldots k \qquad (5)$$

където,

$$S_{\nu\mu} = \int dr_1 \varphi_\nu(1) \varphi_\mu(1), \quad \& \quad F_{\nu\mu} = \int dr_1 \varphi_\nu(1) F(1) \varphi_\mu(1)$$

Екв. (5) са известни като уравнения на Роутан.

Теорията на смущения на Møller-Plesset (MP) добавя по-високи възбуждания към теорията на Hartree-Fock[10] като неитеративна корекция. Той се занимава с разделяне на хамилтоновата (H) на системата на две различни части: необезпокоявана и смутена хамилтонова, което предполага, че вълновата функция и енергия могат да бъдат изразени като мощностни серии по отношение на параметъра, поради добавяне на малките срокове за корекция (McQuarrie, 1983), т.е.

$$\Psi = \Psi^{(0)} + \lambda^1 \Psi^{(1)} + \lambda^2 \Psi^{(2)} + \lambda^3 \Psi^{(3)} + \ldots \qquad (6)$$

$$E = E^{(0)} + \lambda^1 E^{(1)} + \lambda^2 E^{(2)} + \lambda^3 E^{(3)} + \ldots \qquad (7)$$

Подмяна на функцията и енергията на смутената вълна в независимото от времето вълново уравнение на Шрьодингер.

[10] **Sherrill,** C. David. An Introduction to Hartree-Fock Molecular Orbital Theory. Accessible at: <http://vergil.chemistry.gatech.edu/notes/hf-intro/hf-intro.pdf> Accessed on: 01.09.2020

$$\hat{H}\Psi = E\Psi, \text{ получаваме, } E^{(0)} = \left\langle \Psi^{(0)} \middle| \hat{H}_0 \middle| \Psi^{(0)} \right\rangle,$$

$$E^{(1)} = \left\langle \Psi^{(0)} \middle| \hat{H}^{(1)} \middle| \Psi^{(0)} \right\rangle \text{ и } E^{(2)} = -\sum_t \frac{\left| \left\langle \Psi^{(0)} \middle| \hat{H}^{(1)} \middle| \Psi_t \right\rangle \right|^2}{\left(E_t - E^{(0)} \right)} \qquad (8)$$

Екв. (8) показва, че смущения от втори ред към енергията на Хартри-Фок са отрицателни. Намаляването на енергията е това, което трябва да направи точната корекция. Например, трябва да се отбележи, че теорията на смущения на Мьоллер-Плест не е променлива и е възможно енергията на основното състояние на многоелектронната система да бъде прекомерно коригирана с помощта на теорията за смущения на Мьолер-Плест.

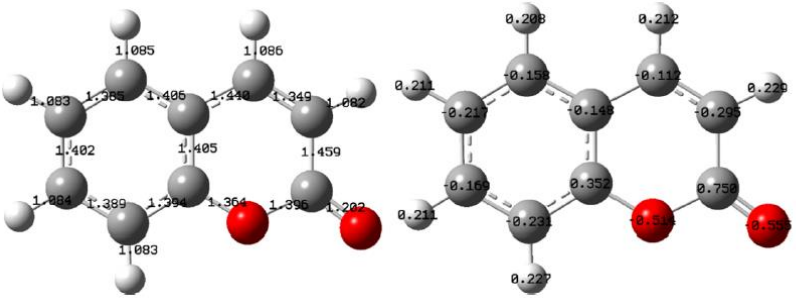
След компютърни симулации на HPC клъстер са изследвани всички възможни монохидроксикумаринови производни и са моделирани с помощта на изчисления на функционалната теория на плътността с цел да изследва ролята на позицията на хидроксилната група за промяната на радикалите и антиоксидантна активност на тези съединения. Геометричната оптимизация се извършва с помощта на функционалността B3LYP с 6-311 + + G (d, p) основен набор.

Промените на енталпията се оценяват в газова фаза и в неявна вода, използвайки модела на поляризирания континуум. Структурно-реактивните модели са очертани. Най-реактивните изомери като както и най-вероятният механизъм на взаимодействие между монохидроксикумарините и свободните радикали са изследвани.

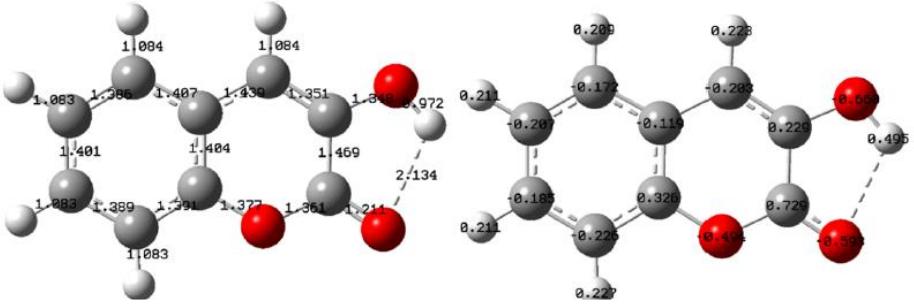
Функционална теория за плътността (DFT) вече е използван за прогнозиране на промяната на радикали механизми на кумариновите производни. Дериватите на 3-фенил-4-хидроксикумарин (флавонов структурен аналог) са теоретично изучени в B3LYP / 6-31G (d, p) ниво. Тяхната радикално поглъщаща активност като феноли е била също се оценява. Ролята на

фениловия пръстен в позиция 3 за резонансната стабилизация на синтетичните кумарини е била оценена. Установено е, че въвеждането на електрондониране функционални групи като метил, метокси и хидроксилните групи в този пръстен водят до намаляване на O-H връзка енталпия на дисоциацията (BDE). Наличието на хлорен атом на позиция 6 в кумариновия скелет има подобен ефект. Най- антиоксидантна активност на синтетичните 3-арил-4-хидроксикумарини има е оценен експериментално спрямо наличен DPPH и 2,2'-азино-бис (3-етилбензотиазолин-6-сулфонова киселина) радикали.

Квантово-химично проучване на тези съединения показва - това присъствието на освобождаващи електронни заместители във фенила пръстенът е особено важен за радикално почистващата активност. Отлична корелация между O-H BDE енергиите и експериментални стойности за активност за намаляване на радона спрямо DPPH е намерен (фиг. 1. 28 и фиг. 1. 29).



Фиг. 1.28. Дължини на връзките в Å (вляво) и естествени орбитални връзки (NBO) заряди (вдясно) на кумарин във вакуум



Фиг. 1.29. Hydroxycoumarin: дължина на връзка в Å (вляво) и NBO промените (вдясно) във вакуум

Таблица 1.2. Получени резултати (в kJ / mol) за BDE, Йонизационни потенциали (IP) и протонен афинитет (PA) на неутралните молекули, както и за протона Енталпия на дисоциацията (PDE) на катион-радикалите и на Енталпия на електронен трансфер (ETE) на анионите на изучаваните топоизомери

	BDE	IP	PDE	PA	ETE
		In a Vacuum			
3-OH-coumarin	351.1	793.4	879.6	1398.2	274.7
4-OH-coumarin	358.3	820.0	860.1	1346.1	334.0
5-OH-coumarin	339.1	816.6	844.2	1363.3	297.5
6-OH-coumarin	340.5	784.1	878.2	1397.3	265.1
7-OH-coumarin	348.8	795.1	875.5	1365.1	305.5
8-OH-coumarin	360.9	804.0	878.7	1413.8	268.9
		In Water			
3-OH-coumarin	330.7	372.6	−42.1	108.5	222.0
4-OH-coumarin	364.2	406.2	−42.2	75.4	288.6
5-OH-coumarin	337.1	386.5	−49.5	102.2	234.7
6-OH-coumarin	337.6	364.9	−27.5	126.0	211.4
7-OH-coumarin	344.5	374.7	−30.3	102.8	241.6
8-OH-coumarin	339.8	380.2	−40.7	119.7	219.9

Изследването (таблица 1.2) обхваща всички възможни монохидроксил кумарин производни, целящи хвърляне на светлина върху ролята на хидроксила група позиция за отстраняване на радикали и цялостна реактивност от тези съединения. Промените на енталпията бяха оценени първо в газова фаза, за да се анализират вътремолекулните фактори засягащи реактивността и за оценка на реактивността на изомерите в неполярна среда и след това в неявна вода, така че моделите могат да имитират по-надеждно реалните условия на реакция за антиоксидантна активност в живите организми *(Evans et al., 1992; Kancheva et al., 2017).*

Най-вероятният механизъм на взаимодействието между монохидроксикумарини и радикали в неполярна среда е (HAT-Hydrogen Atom Transfer) и във вода е (SPLET-Sequential Proton Loss Electron Transfer). Във вода, отделната стъпка е отделянето на електрона от лесно образуван анион.

Най-реактивният изомер в газовата фаза е 5-хидроксикумарин, последван от 6-, 7- и 3-хидроксикумарини, докато 4- и 8-хидроксикумарините се отличават с по-ниска реактивност. Във вода, най-реактивният изомер е 6-хидроксикумарин, последвани от 8- и 3-хидроксикумарини, докато 5-, 7- и накрая, 4-хидроксикумарините са по-малко реактивни, въпреки че последните най-лесно се депротонира. Причините за Реактивността на серията е взаимодействието между електрон и спин разпределение на плътността.

1.7. Математическо описание на ДНК

Много от механичните количества в ДНК може да се опишат емпирично, което не е така за последните модели и моделът е изчислително ефективен в това параметрите на супер-навиването могат да бъдат изчислени от едновременно решаване на алгебрични уравнения.

ДНК молекулата може да се разглежда като еластична отсечка с кръгло сечение и радиус r. Това описание може да се третира като хомогенно, в разширение и линейно еластичен *(Rushdi, 2016, Dummer 2016; Gurwitsch, 1922)*.

Усукването на дъгата S с n-леви завои на усукване е описано в Stump et al., 2000. Този модел показва един балансиран слой с дължина D и ъгъл на супер (Zhour et al., 2008) спирала ß, където нишките са в контакт помежду си и две спирали с половин дължина L (Swigon, 1998),

$$L = 3.1489d\beta^{\frac{31}{2}} - 0.1365d\beta^{-1} - 1.041d\beta^{\frac{1}{2}} - 1.375d\beta \ ,$$

При условие, запазващата (променлива) дължина на дъгата *s* се дава от:

$$s = \frac{2D}{\cos\beta} + 2ndL \ ,$$

където n е броя на спиралите.

Описанието на вектора на нишката - спирала е дадено от (*Stump et al., 2000):*

$$R(t) = (r\cos t)\hat{i} + (r\sin t)\hat{j} + (\cos\beta)t\hat{k} = (r\cos t, r\sin t, \cos\beta)$$

.

Тук $x = r\cos t$ и $y = r\sin t$ описват кръг с радиус *r,* но $z = \cos t$ увеличава (или намалява) косвено от t.

Zeyad (2008) дефинира „ъгъла на наклона" Ξ чрез уравнение зависещо от *z* и *t*.

$$\Xi = 2\pi\left|\cos\beta\right|$$

Бинормалният вектор на ДНК кривата (триедър на Френе) се дефинира чрез следната детерминанта:

$$B = \begin{vmatrix} \hat{i} & \hat{j} & \hat{k} \\ -\dfrac{r\sin t}{\sqrt{r^2+\cos^2\beta}} & \dfrac{r\cos t}{\sqrt{r^2+\cos^2\beta}} & -\dfrac{r\sin t}{\sqrt{r^2+\cos^2\beta}} \\ -\cos t & -\sin t & 0 \end{vmatrix} .$$

Усукването на кривата T_W може да бъде описано чрез:

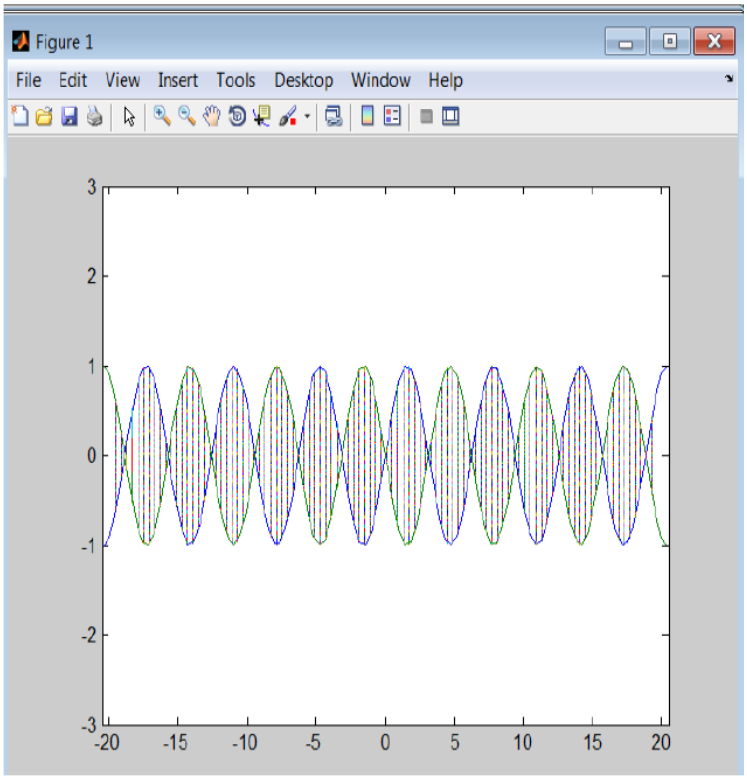
$$T_W = \pm \frac{\cos \beta}{\sqrt{r^2 + \cos^2 \beta}}.$$

Силата на електрическото отблъскване (E_f) се описва по следния начин:

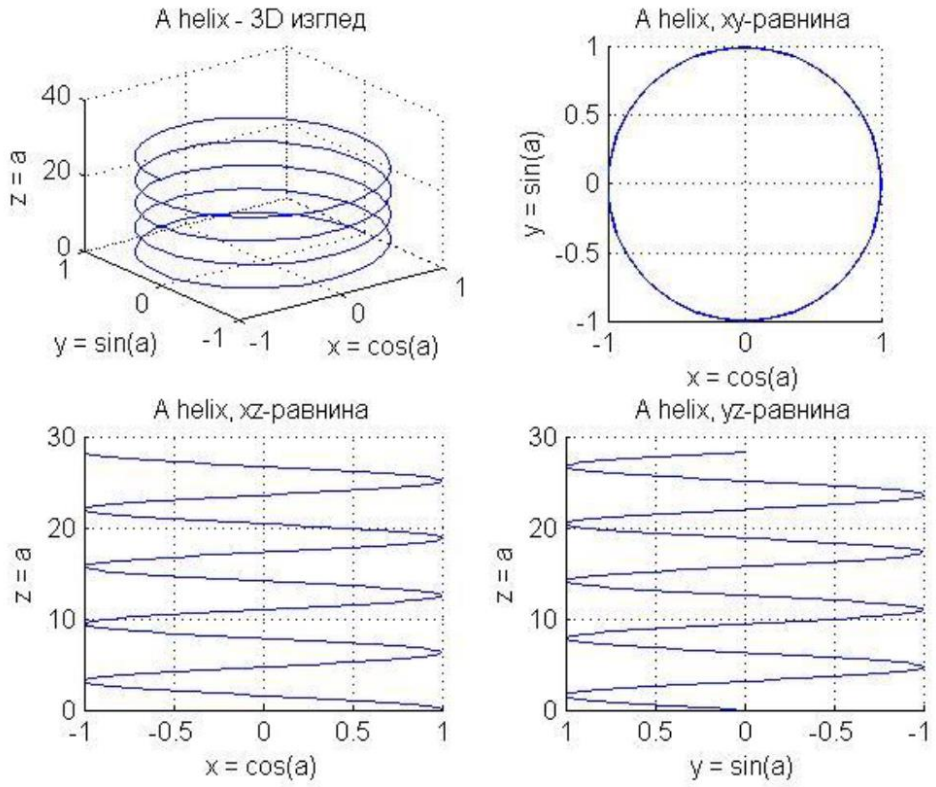
$$E_f = \frac{\sin^4 \beta}{d^3 \cos 2\beta},$$

където d е разстоянието между двете криви.

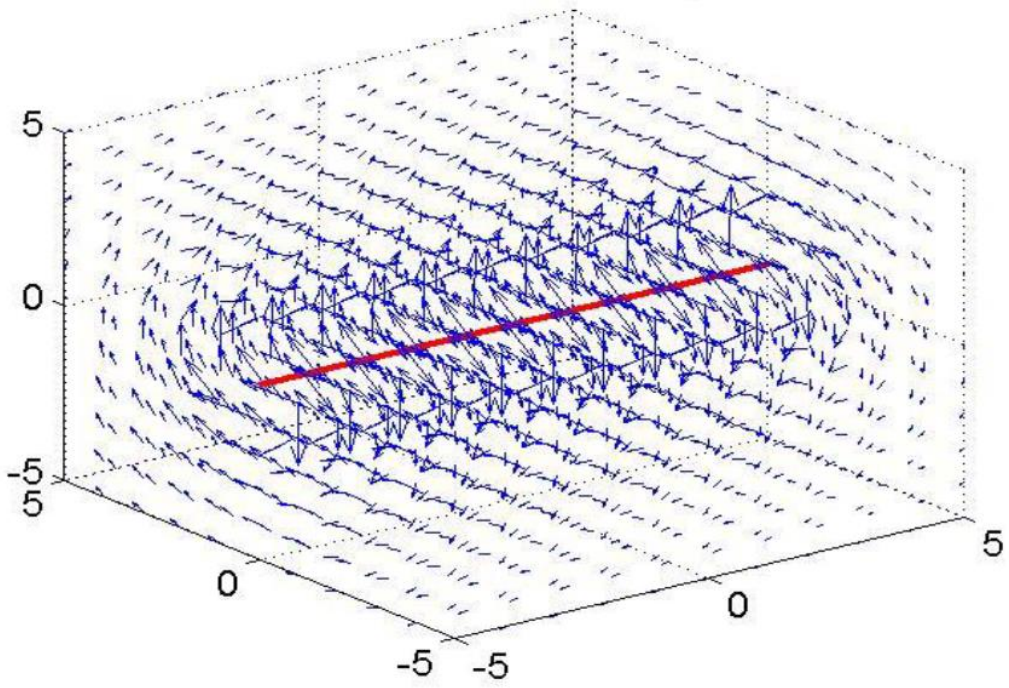
Използвайки горните уравнения на следващите фигури ще покажем компютърни симулации на ДНК в магнитно поле.



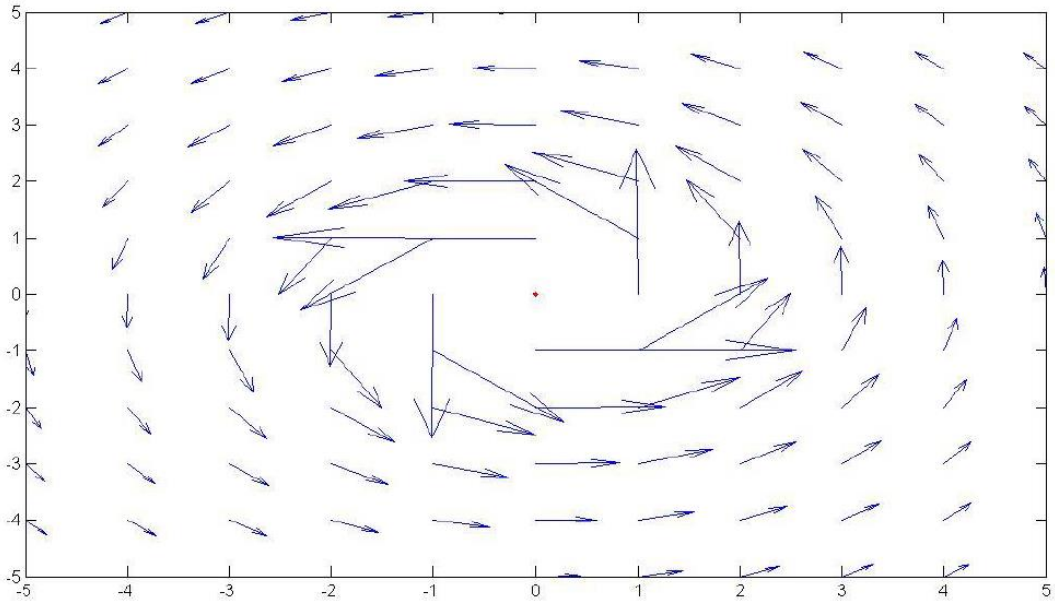
Фиг. 1.30 а. Симулации на магнитно поле в система на Матлаб



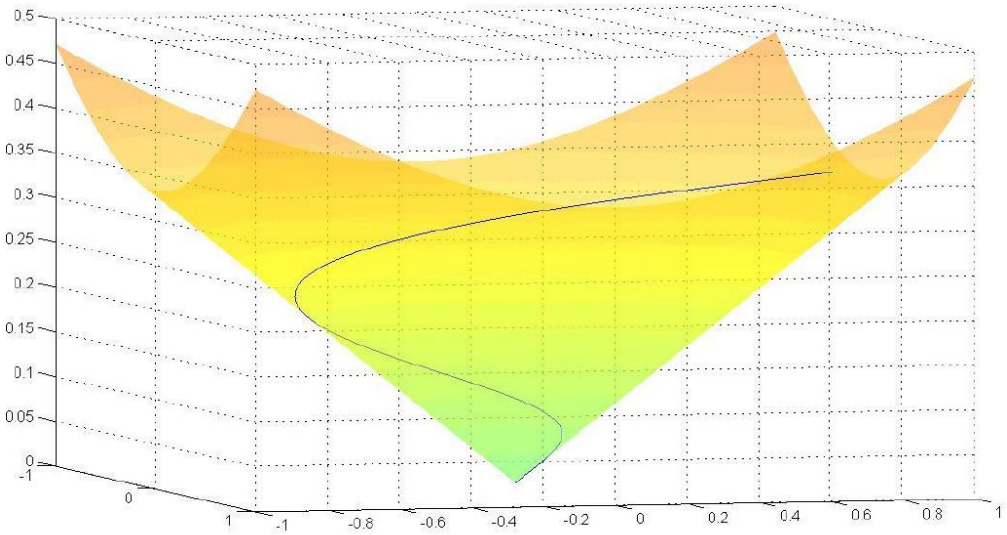
Фиг. 1.30 b. Симулации на магнитно поле в система на Матлаб



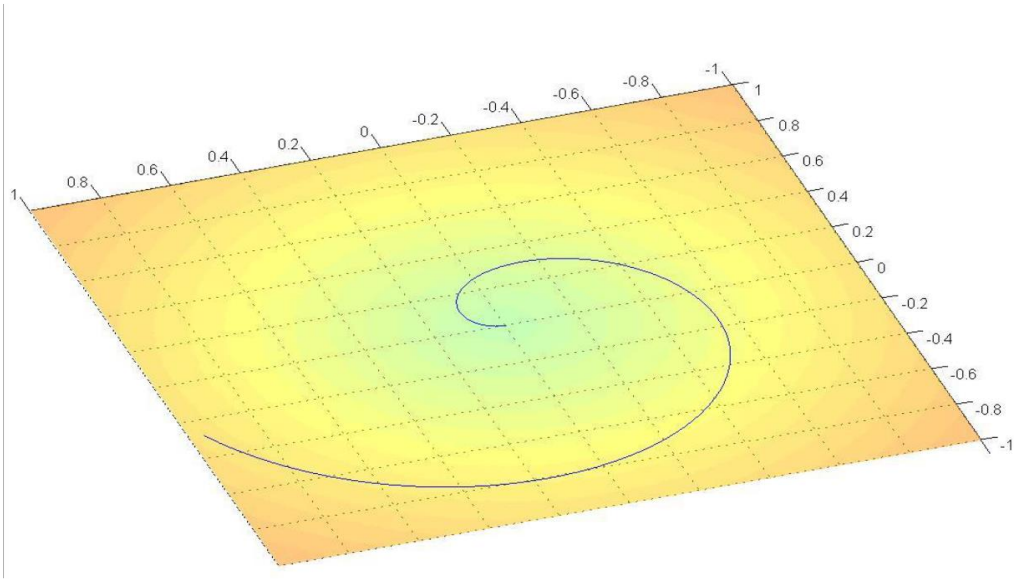
Фиг. 1.30 c. Симулации на магнитно поле в система на Матлаб



Фиг. 1.30 d. Симулации на магнитно поле в система на Матлаб



Фиг. 1.30 e. Симулации на магнитно поле в система на Матлаб



Фиг. 1.30 f. Симулации на магнитно поле в система на Матлаб

Нашата симулация беше извършена с нестатично поле (Фиг. от 1.30 а до 1.30 е). Причината за това е, че то може да се колебае и има спирална природа. Кинетичната енергия на пръстеновидните токове може да се трансформира в потенциална енергия на магнитното поле и обратно, създавайки класическа електромагнитна намотка, както в LC вериги *(Guschin, 2018; Craddock 2014; Dummer 2016).*

Нашите симулации потвърждават хипотезата, че магнитното поле на ДНК зависи от нуклеотидната последователност на ДНК. Може да се направи извода, че промените на магнитното поле пораждат резонанси между подобни последователности в генома. Друг аспект на тези колебания е да възникнат резонанси между подобни геномни последователности *(Craddock 2014, Hameroff 2008; Sheldrake 2009).* Бихме могли да формулираме и противоположната хипотеза, че резониращи последователности могат да имат различна първична последователност, ако формират модели, които могат да породят резонанс. Например, вероятно само пурино-пиримидиновият характер на базите е от значение за резонанса, а не точната идентичност на A, G, C, T бази.

И накрая, резонансите могат да възникнат между отделните клетки и между тъканите. Тъй като всяка клетка има почти

идентична последователност на ДНК и тъй като броят на клетките е голям брой в тялото (около 30 трилиона клетки), Q коефициентът на ЕМ трептенията в тялото също би се подобрил от броя и концентрацията на резонаторите. Тази идея за ЕМ резонансите в тялото отговаря на прогнозите за съществуването на морфогенно поле, направено от Gurwitch *(Gurwitsch, 1922),* преди почти сто години. Сега теорията му е в процес разработена от Sheldrake *(Sheldrake, 2009).* Първоначалната идея и наблюдения на Gurwitch бяха, че клетките и тъканите произвеждат електромагнитно поле, което координира развитието (морфогенезата) и основно определяне (морфостаза) на формата на тялото и структурата на органите. Че всяка клетка е предавател и приемник – общо взето клетките създават морфогенно поле и индивидуално получават и следват инструкциите от него. Съвременната идея на морфогенното поле е, че ДНК на клетките създава морфогенно поле въз основа на геномната последователност и то получава и интерпретира инструкциите от морфогенно поле според геномната програма. По някакъв начин тя е подобна на всяка клетка, която има смартфон, така че може да каже координатите си в тялото и нуждите си и въз основа на това да променя поведението си.

Моделирането на магнетизма на ДНК може да доведе до разбиране на нови електромагнитни механизми от която геномната последователност контролира морфогенезата. Това може да доведе до нови терапевтични приложения, свързани за органно инженерство и терапия на много нарушения, свързани със структурата и формата на тялото и органите. Значението на ароматните хетероцикли за проводимостта е подчертано от Hameroff *(Stuart Hameroff, 2008, 2002).* Това вероятно обяснява сходството в проводимостта в ДНК и микротръбите. Подобно на ДНК, която съдържа подредени ароматни основи, микротръбите съдържат триптофан и други ароматни аминокиселини периодично подредения тубулин *(Stuart Hameroff, 2002),* който се предлага, осигурява проводимост и способност за електромагнитен резонанс *(Craddock, 2014; Stuart R Hameroff, 1994).*

ВТОРА ГЛАВА

ВИЗУАЛИЗАРАНЕ НА 3D ОБЕКТИ

2.1. Общи сведения за компютърната графика

Перспективата в изображенията отразява събирането на успоредните линии при поглед в далечината. От съществено значение е формата на обектите, които се виждат на компютърния екран. Знае се, че полезрението на всяко око има конична форма. Ако се гледа право напред, се създава усещането, че полезрението ни е конусообразно. Например ако прекараме прави линии от всяка съществена точка на куб до окото ни, или центъра на проекцията, точките, в които те пресичат проекционната равнина образуват образ, който се нарича образ при централна проекция. Перспективата се характеризира с това по колко оси се конвертират линиите. Едноточковата перспектива (централната проекция) събира линиите по една ос (предимно по Z). При двутактовата имаме конвертиране по две оси. Тази перспектива е по-реалистична от едноточковата. В реалния свят конвертирането е по трите оси. Триточковата перспектива обаче е много трудна за изобразяване и обикновено не се използва. В рекламите, архитектурата и при инженерите най-използвания метод е двутактовата перспектива. Що се отнася до компютърната графика, по-специално програмите, които генерират автоматично перспективни изображения, триточковата перспектива не е проблем[11].

Програмите, които моделират три измерения работят с множества от данни. Тези множества обикновено са просто списък с координатите по X, Y, и Z в пространството. Програмата обхожда

[11] **Роджерс Д.,** Адамс Дж. Математические основы машинной графики. М.: Мир, 2001. 604 с.

един по един обектите и съставя проекцията върху равнината, която гледаме, т.е. компютърния екран, след което изчертава образите. Има много методи за улесняване на визуализирането на образа. Един такъв трик е премахването на задните стени. Това означава, че всички части на обектите, които не са обърнати към нас, не се показват. Най-лесен метод за скриване на задните части е метода на Z-буфера. При него в паметта се поддържа отделен екранен буфер съдържащ дълбочината на всеки пиксел от екрана[12]. Когато програмата обхожда списъка с координатите, проектирайки ги върху екрана, тя съблюдава Z-координатите за всяка двойка координати (X,Y). Ако същата двойка се срещне отново при обхождането на обектите и съответната Z-координата е по-близо до нас от координатата, записана в буфера, то тя се записва на мястото на предишната точка. По този начин се изобразяват само точките, които са най-близо до нас и закриват останалите. Този метод е прост и лесен за реализиране, но има два недостатъка. Първият е, че изисква много допълнителна памет за буфера. Вторият недостатък е, че понеже не се поддържа връзка между съседните пиксели, се получава стълбищен ефект (aliasing). Има специални техники за преодоляването му (antialiasing), но те работят трудно с метода на Z-буфера.

Към тримерните преобразувания, най-общо спадат преобразуванията от хомогенни координати в 3D транслация и ротация *(Luce, 1966; Mann 2014).* Матрицата за преобразуване, на хомогенните координати в триизмерни, е: P=[x,y,z,1]. Стрелките показват положителната посока на въртене по осите. За да може да се правят някакви преобразувания с даден обект, трябва да се умножат точките които го описват с някоя от следните матрици, в зависимост от действието което се извършва върху него. За лекота, точките на обектите също се разполагат в матрици, а четвъртият ред и колона, се слагат за да не се променят самите матрици при умножението им. Има случай когато трябва да се извършват две

[12] **Роджерс Д**., Адамс Дж. Математические основы машинной графикир Мир 2001, ISBN: 5-03-002143-4

трансформации върху една и съща точка *(Milgram 1994, Sutherland, 1968).* Тогава могат да умножат двете матрици на трансформация, и вече с получената матрица да се умножат и точките.

Матрицата за транслация при транслиране на позиция dx,dy,dz е 4x4:

$$T(dx, dy, dz)=\begin{pmatrix} 1 & 0 & 0 & dx \\ 0 & 1 & 0 & dy \\ 0 & 0 & 1 & dz \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матрицата за мащабиране при мащабиране с коефициенти Sx,Sy,Sz

$$\text{има вида } S(Sx,Sy,Sz)=\begin{pmatrix} Sx & 0 & 0 & 0 \\ 0 & Sy & 0 & 0 \\ 0 & 0 & Sz & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Матриците на ротация, са различни спрямо това около коя от осите се извършва въртенето (Ryan, 2018). Те биват три типа:

На ротация около OX – при въртене на ъгъл около оста OX, координатите на точка X остават непроменени, а по Y и по Z се променят съответно:

$X'=X$

$Y'=Y\cos - Z\sin$

$Z'=Y\sin + Z\cos$

А матрицата има следният вид:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos & -\sin & 0 \\ 0 & \sin & \cos & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Съответно за OY се извършват следните промени:
X′= Zsin +Xcos
Y′=Y
Z′= Zcos – Xsin

Матрицата на ротация, при въртене около оста OY има вида:

$$\begin{pmatrix} \cos & 0 & \sin & 0 \\ 0 & 1 & 0 & 0 \\ -\sin & 0 & \cos & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Аналогично за OZ имаме:
X′= Xcos-Ysin
Y′= Xsin+Ycos
Z′= Z

Една програмна система за интерактивна графика има три компонента *(Balreira et al., 2018):*

- моделът, който се създава, обработва и визуализира;
- приложната програма, която се грижи за създаването му, извършването на операции върху него, както и за организирането му във вид удобен за визуализация;
- графичната система – предоставя набор от средства за визуализация, които приложната програма използва, за да изобрази графично данни от този приложен модел. Графичната система е тази част от програмното осигуряване, която е най-тясно свързана с техническите устройства и която фактически извършва визуализацията, след като приложната програма точно е задала какво трябва да се изобрази.

Естествено е приложни програми да се разработват много по-често от базови графични системи *(Yoshida et al., 2020; Bouvier 2002).* Всяка приложна програма отразява спецификата на

съответната приложна област, а дори и отделните програми в една и съща област могат да бъдат много различни. Въпреки огромните различия, всяка приложна графична програма извършва три основни дейности, които имат определено значение:

- моделиране;
- описание на модела за графичната система;
- интерактивна работа.

Моделът е обект от различно естество, способен да замени друг обект, благодарение на определено съответствие между свойствата на обектите.

Моделирането е процес на построяване на модела и изследване на съответствието между модела и изходния обект, с цел получаване на нова информация за него *(Patricia, 2019)*. Структурата на моделирането включва:

- постановка на задачата;
- създаване или избор на най-икономичен модел;
- изследване на съответствието между модел и изходен обект.

Всяка приложна програма решава конкретна задача:

- изобразяване на молекулната структура на различни химични вещества;
- проектирането на вътрешната архитектура на една сграда;
- анализ на движението на различни механизми;
- преместване и визуализация на различни векторни полета и др.

За решаването на тази задача приложната програма построява и използва геометричен модел на обектите, с които работи. Огромна част от програмистки труд се влага именно в проектирането, създаването, обработването и съхраняването на този модел. Моделите биват най-различни (*Schindler, 1997).*

Описанието на геометричната форма в модела е пряко свързано с поставения приложен проблем и е напълно независим от графическата система. За да бъде визуализиран приложния модел или някаква част от него, е необходимо той да бъде представен в

термините на графичната система, която извършва самото изобразяване. Ако например графичната система може да изобразява само вектори, а моделът е съвкупност от пространствени фигури, то приложната програма трябва да може да опише всяка пространствена фигура чрез вектори. Моделът е това, което е съхранено за един обект, а дейността описание за графичната система е един вид интерпретация на съхранените данни за получаване на тези, които са необходими за визуализация. Тук винаги стои дилемата – каква информация да се съхранява в модела и каква да се изчислява непосредствено преди визуализация. Например: правоъгълник със заоблени върхове може да се съхранява в модела като затворен контур от отсечки и допирателни дъги и тогава описанието му за графична система, която визуализира отсечки и дъги ще е тривиално. Същият правоъгълник може също да бъде представен чрез краищата на диагонала си и радиуса на заобляне на върховете му. Тогава модулът за описанието на модела за същата графична система ще трябва да изчисли краищата на отсечките и дъгите на този заоблен правоъгълник по данните в модела (Hanisch et al., 2005). Приложната програма трябва също така да може да представи за визуализация само определена част от модела или някакъв определен негов изглед. При визуализирането на модел от пространствени фигури с двумерна графична система например трябва да се зададе проекцията и да се генерира плоският образ на пространствената сцена съобразно тази проекция.

Кривите могат да се задават по няколко начина:

- задаване на криви чрез уравнения за променливите x, y, z. При това задаване x е независима променлива т.е.:

$$x = x$$
$$y = f(x)$$
$$z = g(x)$$

- параметрично задаване на криви – параметрични кубически криви. Уравненията $x = x(t)$, $y = y(t)$ и $z = z(t)$ се задават с полиноми от трета степен на параметъра *t*.

$$x(t) = a_x t^3 + b_x t^2 + c_x t + d_x$$
$$y(t) = a_y t^3 + b_y t^2 + c_y t + d_y$$
$$z(t) = a_z t^3 + b_z t^2 + c_z t + d_z$$

при $0 \le t \le 1$ за един сегмент от крива. Ако един вектор-ред от степените на *t* е $T = \left[t^3 t^2 t 1 \right]$, а

$$C = \begin{bmatrix} a_x & a_y & a_z \\ b_x & b_y & b_z \\ c_x & c_y & c_z \\ d_x & d_y & d_z \end{bmatrix},$$

то произволна точка от кривата $P(t) = T.C$.

Производните $\left[\frac{dx}{dt} \frac{dy}{dt} \frac{dz}{dt} \right]$ на $x = x(t)$, $y = y(t)$ и $z = z(t)$ по отношение на *t* определят допирателния вектор в произволна точка от кривата:

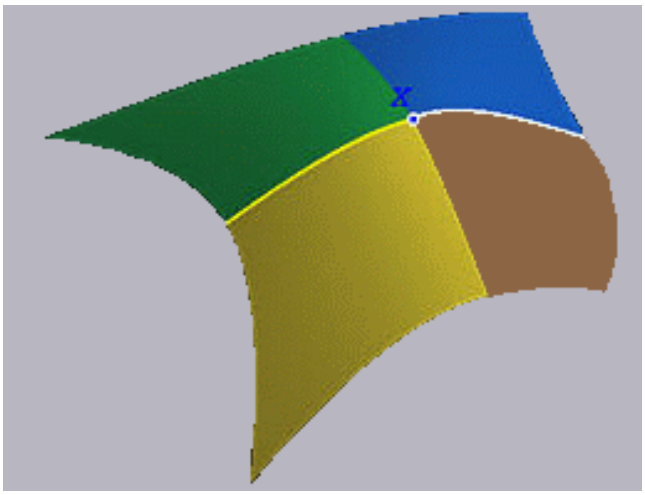
$$\frac{dx}{dt} = 3a_x t^2 + 2b_x t + c_x$$

Аналогично за *y* и *z*

$$\frac{d}{dt} P(t) = \frac{d}{dt} T.C = \left[3t^2 \quad 2t \quad 1 \quad 0 \right].C$$

Ръбовете и стените на едно тяло могат да бъдат съответно криви, сегменти (дъги) или повърхнинни сегменти (кръпки), отколкото линейни сегменти (отсечки) и многостенни. Обаче това може да предизвика някои проблеми *(Parilov & Zorin, 2008; Shoemake, 1995)*. На фигурата по-долу са показани криволинейни кръпки, съединени заедно. Двете гранични дъги, се срещат на върха **X**. Тези

две криви се описват чрез **f**(u) и **g**(v), където u и v имат съответно стойности в интервалите [a,b] и [m,n]. Проблемът е как можем да бъдем сигурни, че тези криви се съединяват гладко (Фиг. 2.1).



Фиг. 2.1. Повърхнина от втора степен

Разглеждаме „десния край“ на кривата **f**(b) и „левия край“ на кривата **g**(m). Ако **f**(b) и **g**(m) са равни, както е показано на фиг. 2.1 се казва, че кривите **f**(u) и **g**(v) са C^0-непрекъснати в точка **f**(b)=**g**(m). Ако за всички i <= k, i-тите производни в **f**(b) и **g**(m) са равни, се казва, че кривите са C^k-непрекъснати в точка **f**(b)=**g**(m).

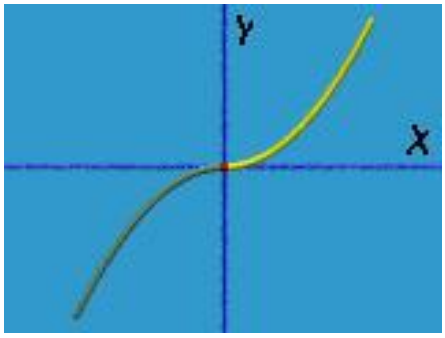
Ясно е от дефиницията, че ако две дъги са *C^k-непрекъснати* в **f**(b)=**g**(m), те са също *C^i-непрекъснати* за всички i по-малки или равни на k. От друга страна, ако k-тите производни на две дъги в точката им на съединяване не са равни, те не могат да бъдат *C^i-непрекъснати* за някое i по-голямо или равно на k.

Разглеждаме един прост пример. Следната крива се състои от две параболи:

$$\mathbf{f}(u) = (u, -u^2, 0),$$
$$\mathbf{g}(v) = (v, v^2, 0),$$

където лявата крива **f**() и дясната крива **g**() имат съответно дефиниционни интервали [-1;0] и [0;1]. Лесно се проверява, че **f**(0) = **g**(0) е координатното начало (*Gortler 2015*). Тогава проверяваме дали

тези две криви са C^2-*непрекъснати* (фиг. 2.2).



Фиг. 2.2. Графика на непрекъсната крива

За тази цел е необходимо да се направят следните изчисления:

$$\mathbf{f}'(u) = (\,1,\, -2u,\, 0\,)$$
$$\mathbf{f}''(u) = (\,0,\, -2,\, 0\,)$$
$$\mathbf{g}'(u) = (\,1,\, 2v,\, 0\,)$$
$$\mathbf{g}''(v) = (\,0,\, 2,\, 0\,)$$

Тъй като $\mathbf{f}'(0) = \mathbf{g}'(0) = (1, 0, 0)$, тези две криви са C^1-*непрекъснати* в координатното начало. Обаче, понеже $\mathbf{f}''(0) = (0, -2, 0)$ не е равен на $\mathbf{g}''(v) = (0, 2, 0)$, кривите не са C^2-*непрекъснати* в координатното начало. Всъщност, $\mathbf{f}''(u)$ и $\mathbf{g}''(v)$ са противоположни постоянни вектори (т.е. не зависят от u и v). Следователно, като пресича координатното начало една движеща се точка от зелената дъга към жълтата дъга, си променя внезапно направлението на втората си производна. Следователно дъгите не са C^2-*непрекъснати* в координатното начало (Bus, 2001). Изчисляваме техните кривини κ:

$$\kappa(\mathbf{f}(u)) = 2/(1 + 4u^2)^{3/2}$$
$$\kappa(\mathbf{g}(v)) = 2/(1 + 4v^2)^{3/2}$$

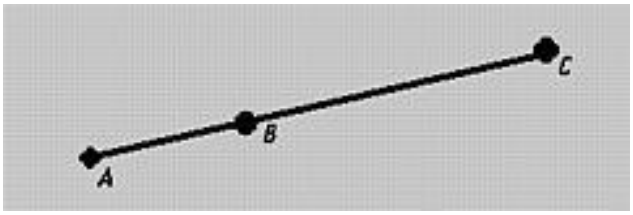
Те имат една и съща форма, въпреки че u е в [-1;0], а v е в [0;1]. Всъщност, кривината на тези две дъги е една и съща върху цялата крива и следователно двете дъги са *непрекъснати* в точката на

съединяването им (т.е. $\mathbf{f}(0)= \mathbf{g}(0)$). Оттук се вижда, че две дъги могат да бъдат C^1-*непрекъснати* и дори *кривинно непрекъснати*, обаче да не бъдат C^2-*непрекъснати*.

C^k-*непрекъснатостта* изглежда добър начин за установяване дали две криви се съединяват гладко. Обаче се появява проблем. Разглеждаме следните две отсечки[13]:

$$\mathbf{f}(u) = \mathbf{A} + u(\mathbf{B} - \mathbf{A})$$
$$\mathbf{g}(v) = \mathbf{B} + v(\mathbf{C} - \mathbf{B})$$

където **A**, **B** и **C** са три колинеарни точки, както са показани на фиг. 2.3.



Фиг. 2.3. Три колинеарни точки

Както u (съотв. v) се променя от 0 до 1, $\mathbf{f}(u)$ (съотв. $\mathbf{g}(v)$) пробягва от **A** до **B** (съотв. от **B** до **C**). Отсечките $\mathbf{f}(u)$ и $\mathbf{g}(v)$ са очевидно C^0-*непрекъснати* в точката на съединяване **B**. Проверяваме дали има C^1-*непрекъснатост*?

$$\mathbf{f}'(u) = \mathbf{B} - \mathbf{A}$$
$$\mathbf{g}'(v) = \mathbf{C} - \mathbf{B}$$

Следователно, $\mathbf{f}'(u) = \mathbf{B} - \mathbf{A}$ е изобщо различно от $\mathbf{g}'(v) = \mathbf{C} - \mathbf{B}$ и затова тези две отсечки не са C^1-*непрекъснати* в точката на съединяване **B**! Това е проблем на параметризацията. Ако заместим направляващите вектори $\mathbf{B} - \mathbf{A}$ и $\mathbf{C} - \mathbf{B}$ с единични вектори и променим интервалите на параметрите u и v, този проблем ще изчезне. Това означава, че горните уравнения се променят както

[13] **Примерът** и теорията се базират на книгата DEROSE, T. Three-Dimensional Computer Graphics: A Coordinate-Free Approach. Unpublished, 1992.

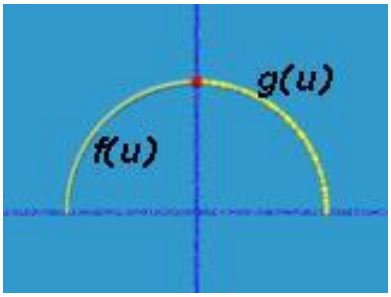
следва:

$$\mathbf{F}(u) = \mathbf{A} + u(\mathbf{B} - \mathbf{A})/ \mid \mathbf{B} - \mathbf{A} \mid$$
$$\mathbf{G}(v) = \mathbf{B} + v(\mathbf{C} - \mathbf{B})/ \mid \mathbf{C} - \mathbf{B}$$

където u е в интервала от 0 до $\mid \mathbf{B} - \mathbf{A} \mid$, а v е в интервала от 0 до $\mid \mathbf{C} - \mathbf{B} \mid$. Сега, понеже има равенство на $\mathbf{F}'(u)$, $\mathbf{G}'(v)$ и единичния вектор по направлението **AC**, отсечките са C^1-*непрекъснати*. Следователно смяната на параметризацията на дъгите може да преодолее проблема.

Разглеждаме друг пример, където π е лудолфовото число 3,1415926..., а u и v са от [0;1] (фиг. 2.4).

$$\mathbf{f}(u) = (-\cos(u^2 \pi/2), \sin(u^2 \pi/2), 0)$$
$$\mathbf{g}(v) = (\sin(v^2 \pi/2), \cos(v^2 \pi/2), 0)$$



Фиг. 2.4. Графика на две криви

Когато u се мени от 0 до 1, $\mathbf{f}(u)$ описва лявата част на полуокръжността. Аналогично, когато v приема стойности от 0 до 1, $\mathbf{g}(u)$ описва дясната част на полуокръжността. Двете части имат точка на съединяване, показана в червено, при (0,1,0) = $\mathbf{f}(1)$ = $\mathbf{g}(0)$. Получава се следното:

$$\mathbf{f}'(u) = (\pi u \sin(u^2 \pi/2), \pi u \cos(u^2 \pi/2), 0)$$
$$\mathbf{f}''(u) = (\pi^2 u^2 \cos(u^2 \pi/2), -\pi^2 u^2 \sin(u^2 \pi/2), 0)$$
$$\mathbf{f}'(u) \times \mathbf{f}''(u) = (0, 0, -\pi^3 u^3)$$
$$\mid \mathbf{f}'(u) \mid = \pi u$$
$$\mid \mathbf{f}'(u) \times \mathbf{f}''(u) \mid = \pi^3 u^3$$
$$\boldsymbol{\kappa}(u) = 1$$

$$\mathbf{g}'(v) = (\pi v \cos(v^2 \pi/2), -\pi v \sin(v^2 \pi/2), 0)$$
$$\mathbf{g}''(v) = (-\pi^2 v^2 \cos(v^2 \pi/2), -\pi^2 v^2 \cos(v^2 \pi/2), 0)$$
$$\mathbf{g}'(v) \times \mathbf{g}''(v) = (0, 0, -\pi^3 u^3)$$
$$| \mathbf{g}'(v) | = \pi v$$
$$| \mathbf{g}'(v) \times \mathbf{g}''(v) | = \pi^3 v^3$$
$$\boldsymbol{\kappa}(v) = 1$$

Двата вектора $\mathbf{g}'(0)$ и $\mathbf{g}''(0)$ са нулеви и не са добре дефинирани. Като резултат не може да се говори за непрекъснатост в точката на съединяване изобщо. Обаче от фигурата изглежда сякаш тези две криви имат някаква непрекъснатост, понеже най-малкото те имат обща допирателна.

Репараметризираме тези криви (т.е. сменяме техните параметрични уравнения без да променяме тяхната форма). Нека $u^2 = p$ в $\mathbf{f}(u)$ и нека $v^2 = q$ в $\mathbf{g}(v)$. Новите уравнения са:

$$\mathbf{f}(p) = (-\cos(p \pi/2), \sin(p \pi/2), 0)$$
$$\mathbf{g}(q) = (\sin(q \pi/2), \cos(q \pi/2), 0)$$

Производните им са:

$$\mathbf{f}'(p) = ((\pi/2) \sin(p \pi/2), (\pi/2) \cos(p \pi/2), 0)$$
$$\mathbf{f}''(p) = ((\pi/2)^2 \cos(p \pi/2), -(\pi/2)^2 \sin(p \pi/2), 0)$$

$$\mathbf{g}'(q) = ((\pi/2) \cos(q \pi/2), -(\pi/2) \sin(q \pi/2), 0)$$
$$\mathbf{g}''(q) = (-(\pi/2)^2 \sin(q \pi/2), -(\pi/2)^2 \cos(q \pi/2), 0)$$

$$\mathbf{f}'(p) \times \mathbf{f}''(p) = \mathbf{g}'(q) \times \mathbf{g}''(q) = (0, 0, -(\pi/2)^3)$$
$$| \mathbf{f}'(p) \times \mathbf{f}''(p) | = | \mathbf{g}'(q) \times \mathbf{g}''(q) | = (\pi/2)^3$$
$$| \mathbf{f}'(p) | = | \mathbf{g}'(q) | = \pi/2$$
$$\kappa(p) = \kappa(q) = 1$$

Следователно, след смяната на неизвестните, и $\mathbf{f}'(1)$ и $\mathbf{g}'(0)$ са равни на (п/2, 0, 0) и оттук съставната крива е C^1-*непрекъсната*. Освен това, и $\mathbf{f}''(1)$ и $\mathbf{g}''(0)$ са равни на (0, -(п/2)^2, 0) и следователно кривата е C^2-*непрекъсната*. Дъгите са също и *кривинно непрекъснати*, защото имат кривина 1 навсякъде. По такъв начин, ре параметризацията или смяната на неизвестните има драматично влияние върху непрекъснатостта.

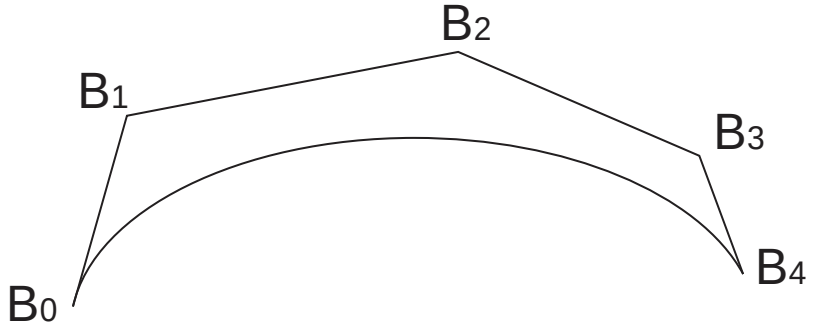
Триизмерните или пространствени криви се използват широко при проектирането и разработването на различни продукти: автомобили, кораби, самолети, обувки, бутилки, сгради и др. Те също са от голямо значение за описанието и тълкуването на физическите явления в геологията, физиката и медицината.

Преди прилагането на математически и компютърни модели в производствения процес, проектирането и производството е използвана описателна геометрия. Много от неговите методи са пренесени в компютърната графика.

Повърхностите често са изобразявани като мрежа от криви, разположени в ортогонални секантни равнини с триизмерни контури на части.

Криви на Безие

Пиер Безие предложи метод за създаване на извивки и повърхности с всякаква форма. Той извежда математическата основа на своя метод от геометрични изображения. Резултатът му е еквивалентен на базата на Бернщайн или на функцията на полиномното приближение.



Фиг. 2.5. Крива на Безие през 5 точки

Кривата на Безие се определя от многоъгълник, както е показано на фигурата. Тъй като основата на Безие е тази на Бернщайн, някои свойства на кривите на Безие веднага са известни (фиг. 2.5).

Степента на полинома, която определя частта на кривата, е една по-малка от броя на точките в съответния многоъгълник.

Първата и последната точки на кривата съвпадат със съответните точки на определящия многоъгълник.

Допирателните вектори в краищата на кривата в посока съвпадат с първата и последната страна на многоъгълника.

Кривата се намира вътре в изпъкналия корпус на многоъгълника, т.е. вътре в най-големия многоъгълник, изграден от дадени точки.

Математическото параметрично представяне на кривата на Безие има формата:

$$B_{n,i}(u) = \frac{n!}{i!\,(n-i)!} u^i (1-u)^{n-i}$$

$$i = 0, 1, \ldots, n$$

където функцията на апроксимация използва полинома на Бернщайн.

Общата формула е:

$C(u) = \sum_{i=0}^{n} B_{n,i}(u) P_i$, където P$i$ са контролните точки $i = 0, 1, \ldots, n$.

Б-*сплайн*

Основните Б-сплайн функции, подобно на основните функции на Безие (полиномите на Бернщайн), са също полиномни функции на една реална променлива, която обикновено се изменя в интервала [0,1].

Основните различия с функциите на Безие са:

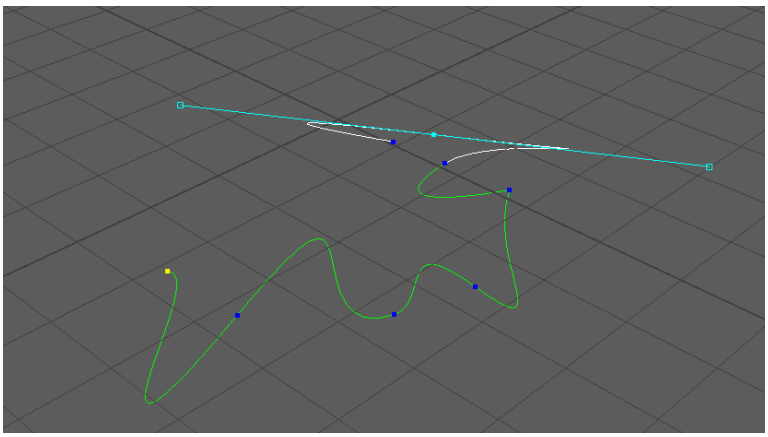
- Б-сплайн функциите представляват съставни полиномни дъги, съединени с някаква степен на гладкост;
- Б-сплайн функциите не са ненулеви в целия интервал на изменение на аргумента;
- Б-сплайн функциите се дефинират над възлов вектор – последователност от реални числа $u_0 \le u_1 \le \ldots \le u_m$, принадлежащи на интервала [0,1].

Основните Б-сплайн функции се дефинират чрез рекурентната формула на Кокс-де Бор.

$$N_{i,0}(u) = \begin{pmatrix} 1 & u \in [u_i, u_{i+1}) \\ 0 & u \notin [u_i, u_{i+1}) \end{pmatrix},$$

а функциите от по-висока степен (p≥1) се пресмятат (фиг. 2.6) , чрез:

$$N_{i,p}(u) = \frac{u - u_i}{u_{i+p} - u_i} N_{i,p-1}(u) + \frac{u_{i+p+1} - u}{u_{i+p+1} - u_{i+1}} N_{i+1,p-1}(u)$$



Фиг. 2.6. Крива на Безие в Maya

Повърхности

Повърхностите и тяхното описание играят важна роля при проектирането и производството. Очевидни примери за това са развитието и производството на автомобилни каросерии, корабни корпуси, фюзелажи и крила на самолети; витла, турбини, компресори и лопатки на вентилатора; прибори, мебели и обувки. В този случай същността на конструкцията, или по функционални или естетически причини, е формата или геометрията на повърхността. Описанието на повърхността също играе важна роля при представянето на данните, получени в медицината, геологията, физиката и други природни науки.

Традиционният начин за представяне на повърхността е използването на няколко ортогонални проекции. По същество повърхността се дефинира от мрежа от ортогонални равнинни криви, разположени на секантни равнини и няколко ортогонални проекции на определени „характерни" пространствени линии. Тези криви първоначално могат да бъдат създадени на хартия или взети (дигитализирани) от триизмерен модел, например в автомобилната индустрия, дизайнерите традиционно използват модел от глина.

В компютърната графика и компютърния дизайн е изгодно да се разработи „истински" триизмерен математически повърхностен модел. Такъв модел позволява да се анализира характеристиките на повърхността, например, кривина или физически количествени характеристики в зависимост от повърхността, например обем, повърхностна площ, инерционен момент и т.н., в ранен етап е сравнително лесно. Визуализацията на повърхността е опростена, използвана за разработване или наблюдение на напредъка на развитието. Освен това, в сравнение с традиционния метод, използващ решетка от линии, генерирането на информация, необходима за производството на повърхността, например програми за управление на машина с цифрово управление, също е значително опростена.

Ранните творби на Безие, Сабина, Питърс (Farin, 1983) и други

автори демонстрират осъществимостта на този подход. Наскоро разработените методи за описание на повърхности достигнаха такъв етап на развитие, че позволяват „почти“ да изключат традиционното описание на повърхността, използвайки решетка от линии.

Има две основни идеи, които са в основата на методите за описание на повърхността В първия, свързан главно с името на Кунс (Hering, 1983), те се опитват да създадат математическа повърхност според известни по-рано данни. Във втория, свързан главно с името Безие, те се опитват да създадат математическа повърхност ab initio (от самото начало). Отначало индустриите, свързани с числови параметри, например дизайнът, гравитираха към първия подход, докато отраслите, които отчитаха визуални или естетически фактори, като дизайнери и художници, гравитираха към втория. Работа на Роджърс върху интерактивни системи за разработване на корабни корпуси и Коен за развитие на повърхността показа, че тези два подхода са съвместими.

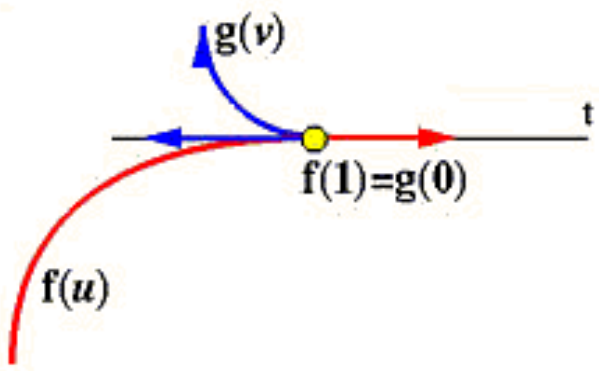
Много C^1-*непрекъснати* криви са *кривинно непрекъснати*, но не C^2-*непрекъснати* в точката на съединяване и някои от тях могат дори да не са двукратно диференцируеми. Тези криви изглеждат гладки в точката на съединяване и също изглеждат гладки когато се придвижва точка от един към друг сегмент. Нещо повече, както беше отбелязано по-рано, след смяна на параметъра някои от тях могат да станат C^2-*непрекъснати* в точката на съединяване. Обаче необходимата репараметризация може да е трудно да се намери. Тогава, трябва да се намали изискването за C^2-*непрекъснатост* до следното: две дъги се наричат *геометрично непрекъснати от степен k* (G^k-*непрекъснати*) в точката на съединяване, тогава и само тогава, когато всички *i*-ти производни, $i <= k$, изчислени спрямо естествен параметър, съвпадат в точката на съединяване.

Използването на естествения параметър не е задължително, както е показано в следната еквивалентна дефиниция: две дъги се наричат *геометрично непрекъснати от степен k* (G^k-*непрекъснати*) в

точката на съединяване, тогава и само тогава, когато съществуват две параметризации – по една за всяка дъга, такива че всички *i*-ти производни, *i* <= *k*, изчислени спрямо тези нови параметризации, съвпадат в точката на съединяване.

Две C^0-*непрекъснати* дъги се наричат G^1-*непрекъснати* в точката на съединяване, тогава и само тогава, когато векторите f'(*u*) и g'(*v*) са с еднакво направление в точката на съединяване. Освен това f'(*u*) и g'(*v*) са изчислени в точката на съединяване.

Тъй като допирателните вектори в точката на съединяване имат еднакво направление, и двете криви имат еднакви допирателни в точката на съединяване. Две дъги, имащи една и съща допирателна, не винаги са G^1-*непрекъснати* в точката на съединяване. На фиг. 2.7 кривите f(u) и g(v) имат обща точка, в която допирателните съвпадат, но допирателните вектори са противоположни и в резултат те не са G^1-*непрекъснати* в точката на съединяване (фиг. 2. 7).



Фиг. 2. 7. Две криви с обща допирателна точка

Критерий на Нийлсон за G^2-*непрекъснатост*:

Две C^1-*непрекъснати* дъги се наричат G^2-*непрекъснати* в точката на съединяване, тогава и само тогава, когато векторът f''(*u*) - g''(*v*) е колинеарен на допирателния вектор в точката на съединяване, т.е.

$$f''(u) - g''(v) = a.\ f'(u) = a.g'(v),\ a=\text{const}.$$

Функциите f'(u), f''(u) и g'(v), g''(v) са изчислени в точката на съединяване.

Хубавото на това характеризиране на G^2-*непрекъснатост* е, че то работи за всяка параметризация, но преди да се използва това характеризиране, трябва да се провери необходимото условие за C^1-*непрекъснатост*. Ако няма C^1-*непрекъснатост*, тогава се проверява чрез по-горните дефиниции, зависещи от параметризацията.

Ако a=0 в критерия на Нийлсон, тогава вторите производни са равни в точката на съединяване и следователно дъгите са C^2-*непрекъснати* в тази точка. Очевидно от C^2-*непрекъснатост* следва G^2-*непрекъснатост*, но не и обратното.

Нека да използваме пример, за да илюстрираме последния резултат.

Криви на Хермит

а сегмент от крива се задават двете крайни точки P_1 и P_2 и допирателните вектори в тях R_1 и R_4.

$$x'(t) = \begin{bmatrix} 3t^2 & 2t & 1 & 0 \end{bmatrix} . C_x \qquad x(t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} a_x \\ b_x \\ c_x \\ d_x \end{bmatrix} = T . C_x$$

$$\begin{aligned} x(0) = P_{1x} &= \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} . C_x \\ x(1) = P_{4x} &= \begin{bmatrix} 1 & 1 & 1 & 1 \end{bmatrix} . C_x \\ x'(0) = R_{1x} &= \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} . C_x \\ x'(1) = R_{4x} &= \begin{bmatrix} 3 & 2 & 1 & 0 \end{bmatrix} . C_x \end{aligned}$$

или обединени в матричен вид

$$\begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{bmatrix} . C_x$$

$$C_x = \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} . \begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix} = M_H . G_{Hx}$$

където M_H е матрицата на Hermite, а

$$G_{Hx} = \begin{bmatrix} P_{1x} \\ P_{4x} \\ R_{1x} \\ R_{4x} \end{bmatrix}$$

е геометричен вектор на Hermite.

Тогава

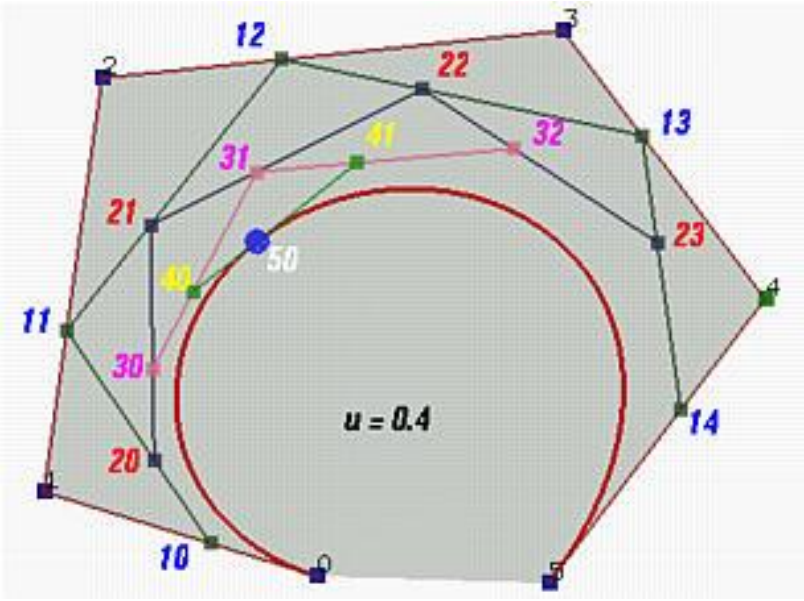
$$x(t) = T . M_H . G_{Hx}$$
$$y(t) = T . M_H . G_{Hy}$$
$$z(t) = T . M_H . G_{Hz}$$

Като се има предвид построяването на крива на Безие, следващата важна задача е да се намери точката P(u) върху кривата, определена за дадено u. Това става чрез *алгоритъма на дьо Кастелжо (De Casteljau)*. Един прост начин е да се развие дефиницията на кривата на Безие в традиционната форма f(u) = ($f(u)$, $g(u)$, $h(u)$) и като се замести с дадено u в това уравнение, да се получи f(u). Контролните точки са: 00 за P_0, 01 за P_1, ..., 0i за P_i, ..., 0n за P_n. Нулите в тези номера показват, че това е началната или *0*-вата итерация. После могат да бъдат заменени с 1, 2, 3 и т.н.

Основната идея на алгоритъма на дьо Кастелжо е избирането на точка C от отсечката AB, така че разстоянието между A и C а разстоянието между A и B да имат дадено отношение, да кажем u. Намираме начин да определим точка C.

Векторът от A до B е B - A. Тъй като u е отношение в интервала от 0 до 1, точка C се задава чрез u(B - A). Като се има предвид положението на A, точката C е A + u(B - A) = (1 - u)A + uB. Следователно задавайки u, (1 - u)A + uB е точката C между A и B, такава че отношението на разстоянието между C и A и разстоянието между A и B е u.

Идеята на алгоритъма на дьо Кастелжо идва по следния начин. Предполага се, че искаме да намерим **P**(u), където u е от [0,1]. Започвайки с първата начупена линия (полигон), 00-01-02-03...-0n, на рамото (т.е. отсечката) от 0i до 0(i+1), използва се горната формула да се намери точка 1i, така че отношението на разстоянието между 0i и 1i и разстоянието между 0i и 0(i+1) да е u. По този начин се получават n точки 10, 11, 12, ..., 1(n-1). Те определят нова начупена линия от n - 1 рамена (фиг. 2.8).

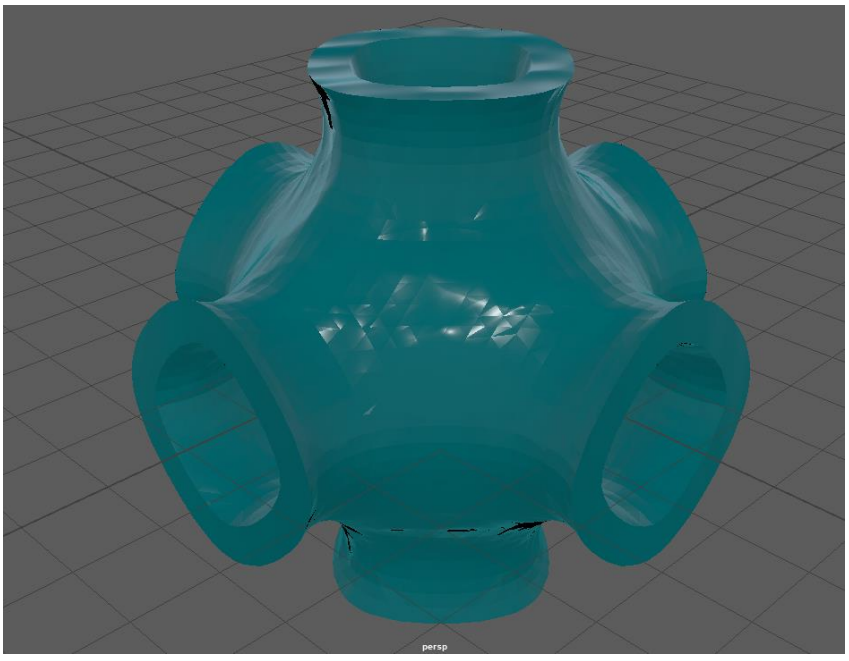


Фиг 2.8. Представяне на криви и допирателни

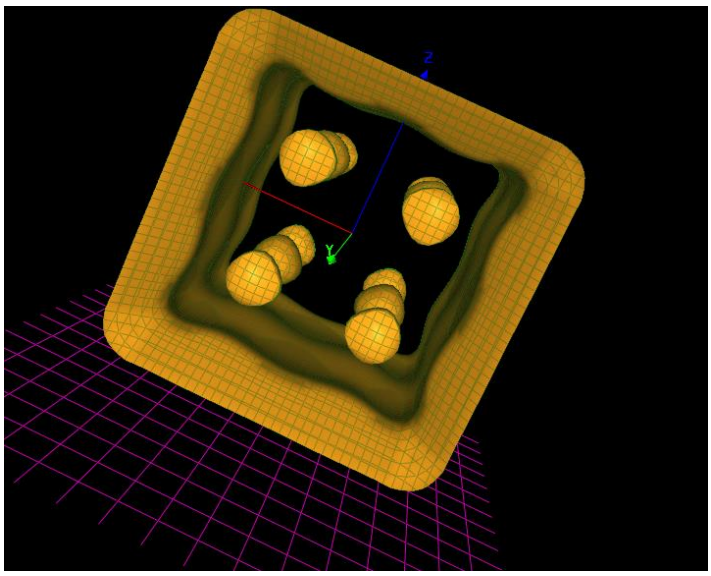
В горната линия u е 0.4. Точката 10 лежи на рамото 00-01, точката 11 – на рамото 01-02, ..., а точката 14 – на рамото 04-05.

Новите точки са номерирани с 1i. Прилага се процедура към този нов полигон и се получава втори полигон от n - 1 точки 20, 21, ..., 2(n-2) и n - 2 рамена. Започвайки с този полигон, може да се конструира трети полигон от n - 2 точки 30, 31, ..., 3(n-3) и n - 3 рамена. Повтаряйки този процес n пъти се стига до единствена точка n_0. Дьо Кастелжо е доказал, че това е точката P(u) върху кривата, която съответства на u.

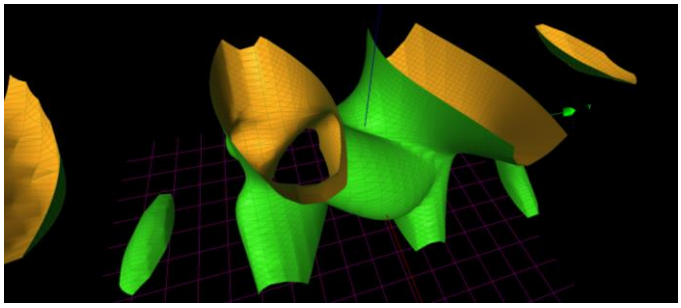
Нека 20 е точката на рамото 10-11, такава че отношението на разстоянието между 10 и 20 и разстоянието между 10 и 11 е u. Аналогично се избират точка 21 върху рамото 11-12, 22 върху рамото 12-13 и 23 върху 13-14. Получава се трети полигон, дефиниран чрез 20, 21, 22 и 23. Този трети полигон има n - 1 точки и n - 2 рамена. Продължавайки с този процес, се получава нов полигон от три точки 30, 31 и 32. Тогава, от този четвърти полигон се получава петия, съдържащ две точки 40 и 41. Като се направи това още веднъж, се получава 50 - точката P(0.4) върху кривата (фиг. 2.8). На следващите фигури представяме примери на примитиви, създадени чрез описаните по горе криви (фиг. 2.9 – 2.11)



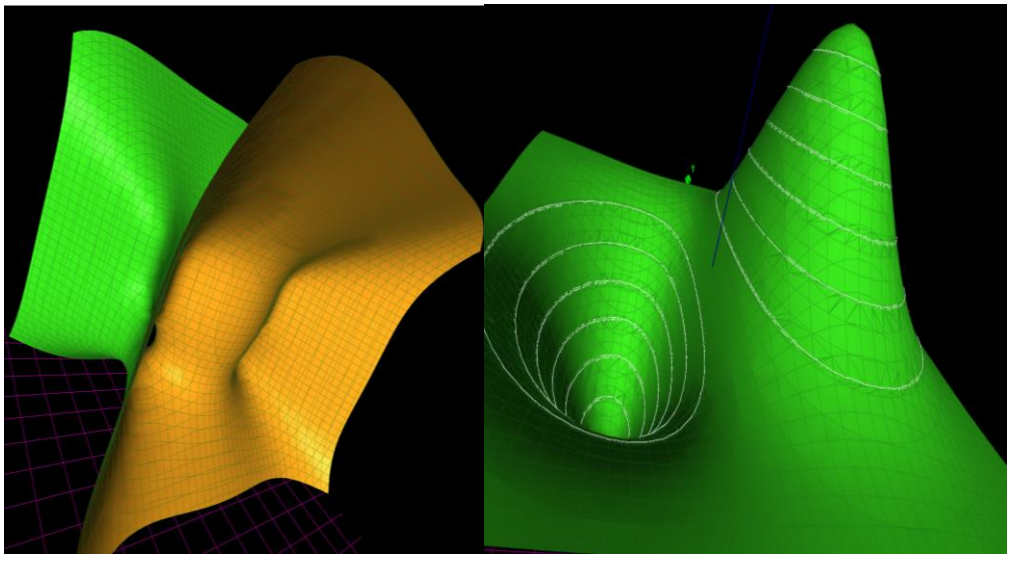
Фиг. 2.9. Повърхност от Maya



Фиг. 2.10. Есплицитно описание на повърхнина



Фиг. 2.11 a. Есплицитно описание на повърхнина



Фиг. 2.11 b. Есплицитно описание на повърхнина

2.2. 3D визуализация на биологичните структури

Когато изискванията за моделиране надхвърлят възможностите на бордовите двигатели, инструментите за програмиране ни позволяват да разработим модела, за да отговори на нашите нужди.

Подобно на истинска сцена, цифровата сцена е тъмна, докато не се добавят светлини. Повечето 3D приложения изключват пакет от налични светлини, имитиращи тези, открити на филмов комплект или във фотографски снимки. Тук се включват прожектори, областни светлини, точкови светлини и инфititни светлини сред другите, но варират в действителното име от приложение до приложение. Обикновено това е по подразбиране светлина за осигуряване на общо осветление, преди да започнете да добавяте светлини към сцената. Светлините могат да бъдат цветни и да им се зададе номер на атрибути, които произвеждат специални ефекти като оцветяване или видим лъч светлина. Сенките, хвърлени от осветени предмети в една сцена, могат да бъдат много полезно устройство за предаване реализъм и за подчертаване на пространствените отношения в една сцена. CGI сенките може да имат твърди или меки ръбове (както правят реалните сенки) и обикновено са зададени и регулирани в рамките на контролите за дадена светлина *(Gallier, 2001; Ge & Ravan, 1994; Heckbert, 1989)*.

В ежедневието се използват камери за преглед и запис на модели и действия в нашата 3D среда. Мая, подобно на много други инструменти за 3D анимация, реализира същата идея да помогне на планирането и как ще се изобразят събитията от вашия 3D виртуален свят. Когато се създава нова сцена в Мая, камера по подразбиране осигурява изгледа Perspective, който се вижда. Докато „маневрират" в пространството, за да получим желания изглед на сцената, ние всъщност правим въртене и може би увеличаване на виртуалната камера на Maya. 3D камери осигуряват ортогонален и

перспективен изглед и имат много от тях атрибути на реални камери, като настройки на експозицията, ъгъл на обектива и фокусно разстояние. Тези атрибути, заедно с превода и въртене на камерата, могат да бъдат анимирани и ключови за нашите цели.

В 3D CGI засенчването се отнася до комбинираните ефекти на осветление, цвят на повърхността, повърхност текстура и геометрия, определящи финалния вид на вашите модели.

Когато първият обект на Maya е създаден, му се назначава шейдър по подразбиране (наричан също материал) с атрибути на външния вид, включително цвят, непрозрачност и характеристики на повърхността като текстура или геометричен модел. Шейдърите могат да бъдат създадени и приложени към обекти по много начини за подражание на реални повърхности и в някои случаи обеми като стъкло или мъгла[14].

Производството на изображения от 3D сцена се нарича рендъринг, сложен обект който комбинира ефектите на светлини, камери и засенчване. Изображенията се запазват като индивидуална картина или като група в един филм и след това може да бъде показан. Колективно изобразените изображения често се наричат кадри. Форматът на изображението и разделителната способност на пиксела на кадрите са зададени в рендериране на настройките на 3D приложението, определени от крайната цел на анимацията. Например, обикновено ще ни е необходим различен формат и разделителна способност за малък филм, предназначен за гледане в интернет браузър, в сравнение с характеристика на голям екран. Важно е да знаем изискванията преди това настройка на камери и изобразяване, особено ако създавате анимация за установен формат като NTSC или съществуваща уеб страница. 3D приложения предоставят гама от стандартни формати и резолюции, които може да избираме, както и персонализирани

[14] **Autodesk Maya**. Knowledge Network. Accessible to: <https://help.autodesk.com/view/MAYAUL/2020/ENU/?guid=GUID-6B531DDB-3440-4216-A322-FB6CD1EA83A1> Last aceessed on: 10.10.2020

настройки.

Render двигателите поддържат редица ефекти от фотореализъм, които могат да бъдат полезни при разработването *(Debevec, 1996; Dorsey, 2008).*

Разсейване на повърхността, при което светлината прониква в повърхност, разпръсква се и се появява отново (както в реално полупрозрачните материали като кожа и восък). Това може да създаде впечатление от полупрозрачни, подобни на гел вещества.

Оклузия на околната среда, която моделира намаляването на околната светлина там, където идват повърхности – огледално езеро.

Глобалното осветление, което е изчислително скъп начин за моделиране на осветлението в реалния свят, при което светлината отскача по различен начин около една сцена и цветът на един обект може да „изтича“ върху друг в близост.

Какъвто и подход да изберете от гледна точка на външния вид на нашата сцена, ще е необходима ефикасна стратегия за производство на финални рендери. Стратегията трябва да има два компонента *(Feynman , 1963).*

1. Композиционен план. Компютърна анимационна сцена, като тези, които виждате във филми и по телевизията рядко се представя като едно цяло. Обикновено се съставя индивидуален кадър на слоеве, номериращи някъде от две до десетки (или дори стотици) поотделно изобразени изображения. Понякога проходите са съставени от различни равнини на изображението (напр. преден план, среден и фонов), понякога те са с различни „знаци“ (напр. Взаимодействащи протеини), а понякога те са от отделни компоненти на изображението (напр. на цвят на текстурата, сенки и подчертаване). Много са практични причини за това: някои ефекти са твърде трудни или се изчисляват интензивно, за да бъдат направени директно (напр. дълбочината на фиксирания ефект) и сравнително лесно да се добави на етапа на композиране; и

промените са по-лесни за извършване, когато е нужен само един компонент на една сцена, а не цялата сцена. Също така, чрез изобразяване на елементи като светлини и сенки отделни проходи, те могат лесно да се настройват за максимален ефект. Изборът, следователно, ще зависи от наличното време и крайната употреба.

2. План за управление на данните. Предвид огромния брой рендери, генерирани от типичен анимационен проект и общата практичност на изобразяването в множество проходи (което може да умножи броя на рендеринга по-малко пъти), важно е да се поддържа разумна стратегия за управление на данни. Тя има един важен компонент – Йерархиите в директорията на проекти се използва за организиране на файлове по тип. Например, Мая по подразбиране ще запази изобразените файлове в директорията с изображения в рамките на текущата Папка[15].

От ранните модели на Джон Далтън, който през 1808 г. предлага цялата материя да се състои от атоми, учените са създали 3D изображения на молекули, за да разберат тяхната структура. Структурата на свой ред изяснява функцията; как една молекула работи, когато изпълнява задълженията си вътре в клетката. Познаването на функцията на биомолекулите, особено нуклеиновите киселини (ДНК, РНК и др.) И протеините, в крайна сметка води до разработването на терапии, лечения и стратегии за превенция на болести. 3D моделите са от съществено значение за Джеймс Уотсън и Франсис Крик, тъй като те решават геометричната структура на ДНК. От друга страна, подреждането, което разкри нуклеотидните бази в противоположни двойки и механизма за копиране на генетичен материал. Това е само един от многото примери за жизненоважната роля на визуализацията в процеса на откриване в биологията. Всяко подробно изследване на структурата на живите същества разкрива удивителна йерархия на организацията. От най-простите аминокиселини нагоре, природата

[15] **In Jason Sharpe** et al., Silico: 3D Animation and Simulation of Cell Biology with Maya and MEL. Morgan Kaufmann Publishers is an imprint of Elsevier. 2012

се гради върху основната структура. Разбирането на тази организация е от решаващо значение за разбирането как функционира живата материя и как тези операции могат да бъдат визуализирани с помощта на компютри. Дори и най-удивителната идея за нови начини за гледане на света е безсилна, без инструментите и техническите средства за привеждане на новата визия в практическо приложение.

При правилните условия актиновите полипептиди образуват редовни верижни полимери, наречени актинови нишки. Индивидуалната актинова верига се нарича субединица в нишката и а мономер, когато е в неполимеризирано състояние (фиг. 2.12). Актиновите нишки са основен компонент от стареенето на протеиновия скелет или цитоскелет, които придават форма на клетката.

Element	C	H	N	O	P	S
vdW radius (Å)	1.70	1.20	1.55	1.40	1.90	1.85
CPK color	gray	white	blue	red	orange	yellow
RGB values (normalized from 0 to 1)	R: 0.7 G: 0.7 B: 0.7	R: 1.0 G: 1.0 B: 1.0	R: 0.0 G: 0.5 B: 1.0	R: 1.0 G: 0.0 B: 0.0	R: 1.0 G: 0.5 B: 0.0	R: 1.0 G: 1.0 B: 0.0

Фиг. 2.12 Размер на молекулярните и атомните[16]

Молекулите, включително големите молекули, съдържащи живи клетки, са комбинация на електрони и атоми в движение през пространството. Тъй като електроните са не само твърде незначителни за око, за да видим дали те могат да бъдат

[16] Copy from **Pauling** L. The nature of the chemical bond and the structure of molecules and crystals, Cornell University Press, Ithaca, NY, 1960

„приковани", но всъщност не могат да бъдат прикачени към атоми и молекулите нямат визуален вид в смисъла, в който обикновено прилагаме този термин – вид на лицето на приятел или великолепен залез. Снимки на атоми и молекули, които ние виждаме в днешните медии, следователно не са реално представяне на визуална визия т.е. всички щяхме да видим дали само молекулата може да се разшири до размера на бейсболно игрище. Тези снимки са визуални интерпретации, които имат художниците, работещи с учени направени за определени жизнени свойства на тези молекули. Две от тях са много важни, за да имате предвид при работата с PDB и други данни за молекулна структура. Първо, атомните ядра са много по-тежки отколкото техните по-ярки електроните и те много често не се изобразяват при компютърни симулации.

Координатата в PDB файла, която виждаме за местоположението на атома, е измерване това казва къде се намира центърът на молекулата.

Файловият формат на Protein Data Bank (pdb) е текстов файлов формат, описващ триизмерните структури на молекули, съхранявани в банката данни за протеини. Съответно pdb форматът осигурява описание и анотация на протеинови и нуклеинови киселинни структури, включително атомни координати, вторични структурни задачи, както и атомна свързаност. Освен това се съхраняват експериментални метаданни. PDB форматът е наследеният файлов формат за Protein Data Bank, който сега съхранява данни за биологични макромолекули в по-новия mmCIF файлов формат[17].

[17] https://en.wikipedia.org/wiki/Protein_Data_Bank_(file_format)

A typical PDB file describing a protein consists of hundreds to thousands of lines like the following (taken peptide):

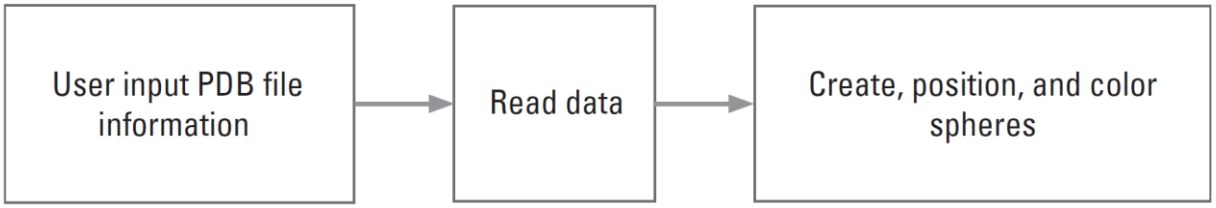
```
HEADER    EXTRACELLULAR MATRIX                    22-JAN-98   1A3I
TITLE     X-RAY CRYSTALLOGRAPHIC DETERMINATION OF A COLLAGEN-LIKE
TITLE    2 PEPTIDE WITH THE REPEATING SEQUENCE (PRO-PRO-GLY)
...
EXPDTA    X-RAY DIFFRACTION
AUTHOR    R.Z.KRAMER,L.VITAGLIANO,J.BELLA,R.BERISIO,L.MAZZARELLA,
AUTHOR   2 B.BRODSKY,A.ZAGARI,H.M.BERMAN
...
REMARK 350 BIOMOLECULE: 1
REMARK 350 APPLY THE FOLLOWING TO CHAINS: A, B, C
REMARK 350   BIOMT1   1  1.000000  0.000000  0.000000        0.00000
REMARK 350   BIOMT2   1  0.000000  1.000000  0.000000        0.00000
...
SEQRES   1 A    9  PRO PRO GLY PRO PRO GLY PRO PRO GLY
SEQRES   1 B    6  PRO PRO GLY PRO PRO GLY
SEQRES   1 C    6  PRO PRO GLY PRO PRO GLY
...
ATOM      1  N   PRO A   1       8.316  21.206  21.530  1.00 17.44           N
ATOM      2  CA  PRO A   1       7.608  20.729  20.336  1.00 17.44           C
ATOM      3  C   PRO A   1       8.487  20.707  19.092  1.00 17.44           C
ATOM      4  O   PRO A   1       9.466  21.457  19.005  1.00 17.44           O
ATOM      5  CB  PRO A   1       6.460  21.723  20.211  1.00 22.26           C
...
HETATM  130  C   ACY   401       3.682  22.541  11.236  1.00 21.19           C
HETATM  131  O   ACY   401       2.807  23.097  10.553  1.00 21.19           O
HETATM  132  OXT ACY   401       4.306  23.101  12.291  1.00 21.19           O
...
```

Фиг. 2.13. Представяне на PDB файл

Форматът на PDB файловете е изобретен през 1976 г. (фиг. 2.13), който позволи на изследователите да обменят протеинови координати чрез система от бази данни. Форматът на ширината му с фиксирана колона е ограничен до 80 колони, който се основава на ширината на компютърните перфокарти, използвани преди за обмен на координати. През годините форматът на файла претърпя много промени и ревизии. Към 13 юли 2011 г. най-новата редакция е 3.30.

Колоната във формат PDB е широка с един знак. Преди това един тип данни често заемат няколко колони. Например колони от 1 до 6 са запазени за записа име (например ATOM или HETATM). Колони 7 до 11 са запазени за данни, наречени „сериен номер на Atom“ и т.н. Във всички запазени са 16 типа данни за ATOM и HETAM записи. Реалните PDB обаче рядко съдържат всички видове. Това е резултат в празни колони, което не е проблем, стига програмата, която чете и интерпретира филатомата, да преброява символите (включително празни интервали) правилно. Ако, от

друга страна, програмата чете с данни дума по дума вместо знак по знак, той може погрешно да приписва данни на грешна колона (фиг. 2.14).



фиг. 2.14. Алгоритъм за визуализиране на макромолекули

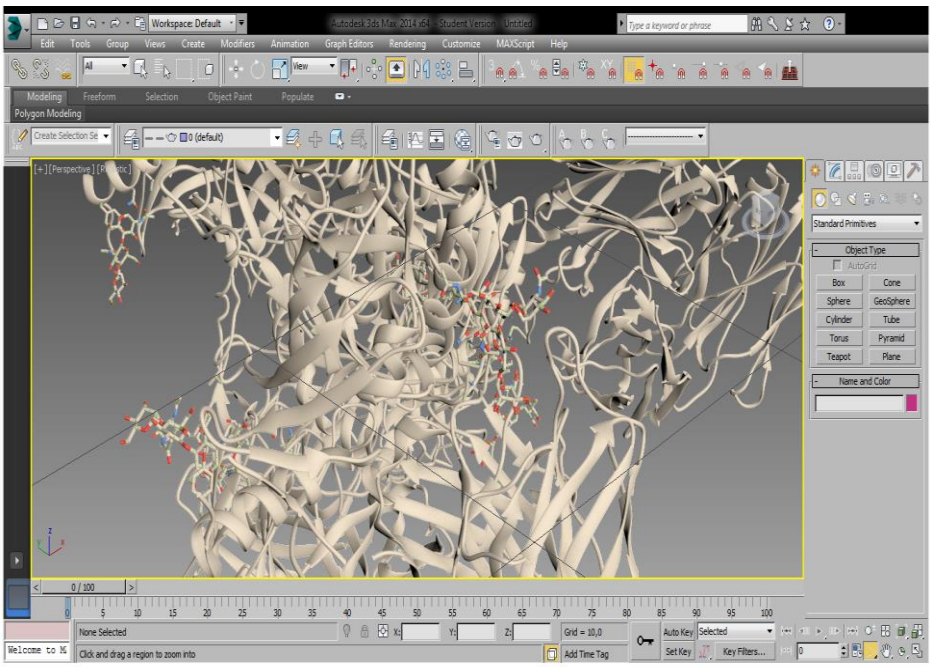
Maya е приложение, което се използва за генериране на 3D активи за използване във филми, телевизия, разработване на игри и архитектура. Първоначално софтуерът беше издаден за операционната система IRIX. Тази подкрепа обаче беше преустановена през август 2006 г. след издаването на версия 6.5. Maya беше достъпна и в двете издания „Пълно“ и „Неограничено“ до август 2008 г., когато беше превърната в един пакет. Освен това за Maya може да бъде закупен постоянен със студентски лиценз. Този лиценз не изтича и версията на студента може да бъде надстроена до търговската версия със значителна отстъпка. Тя може да се използва дори и след завършването на студентите, като единственото ограничение е некомерсиалната употреба. Не се създават водни знаци по време на изхода, което прави студентската версия на Maya [1, 3, 4] подходяща за създаване на портфолио. Файловете, запазени с тази версия, обаче се разпознават от всички версии на Maya като файлове, създадени от студентска версия. Постоянният студентски лиценз също така позволява създаването на некомерсиални активи за некомерсиална употреба в игрови двигатели като Unreal Development Kit.

За визуализиране използваме готов програмен код, реализиран през 2007 г. от Jason Sharpe, Charles Lumsden, Nick Woolridge

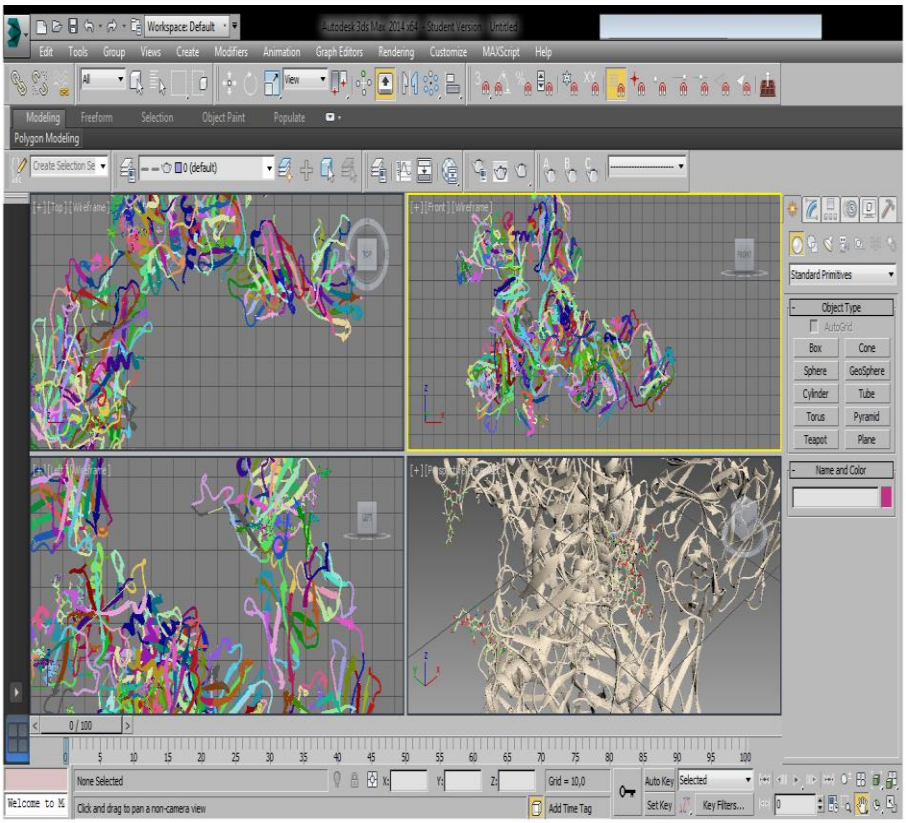
Представяме част от кода

```
/***** cpk.mel *****/
/*
Date: created February 0_1 20_0_6; modified August 15 20_0_7.
Authors: Jason Sharpe, Charles Lumsden, Nick Woolridge.
Description:
This procedure reads ATOM entries from a PDB file. It calls a second
procedure to make and shade a sphere to represent each atom according
to its element (oxygen, carbon, etc). The end result is a CPK-style model
built from sphere representing the van der Waals radii of the constituent
atoms.
The procedure arguments are as follows:
$chaincol The pdb file column in which the chain letters reside.
$xcol The column in which the x-coordinates reside
(starting with 1).
$elemcol The column in which the element (atom) names reside.
To use this script:
Save the entire script in a text file, using the .mel extension, in
your Maya Scripts directory, then source it through Maya's Script
Editor. Alternately, you can copy and paste the entire script into
Maya's Script Editor.
*/
global proc cpk(int $chainCol, int $xCol, int $elemCol) {
// Start of cpk()..
```

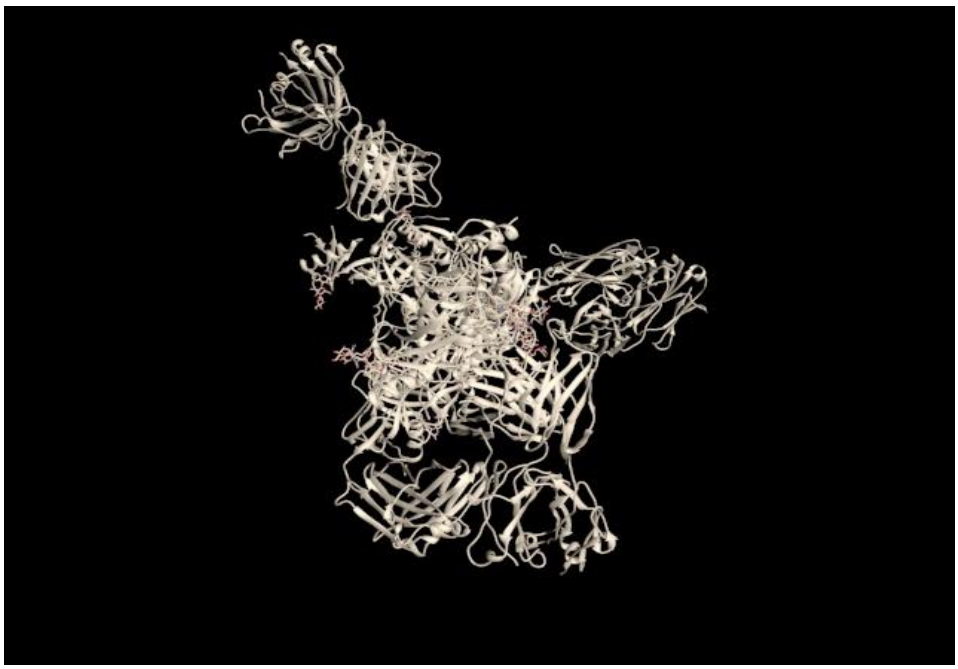
Той е достъпен в Jason Sharpe et al. In Silico: 3D Animation and Simulation of Cell Biology with Maya and MEL, ISBN-13: 978-0-12-373655-0, Morgan Kaufmann. На следващите фигури ще покажем няколко примера, които дават примери за четиримерната структура на протеин (фиг. 2-15 – 2.18).



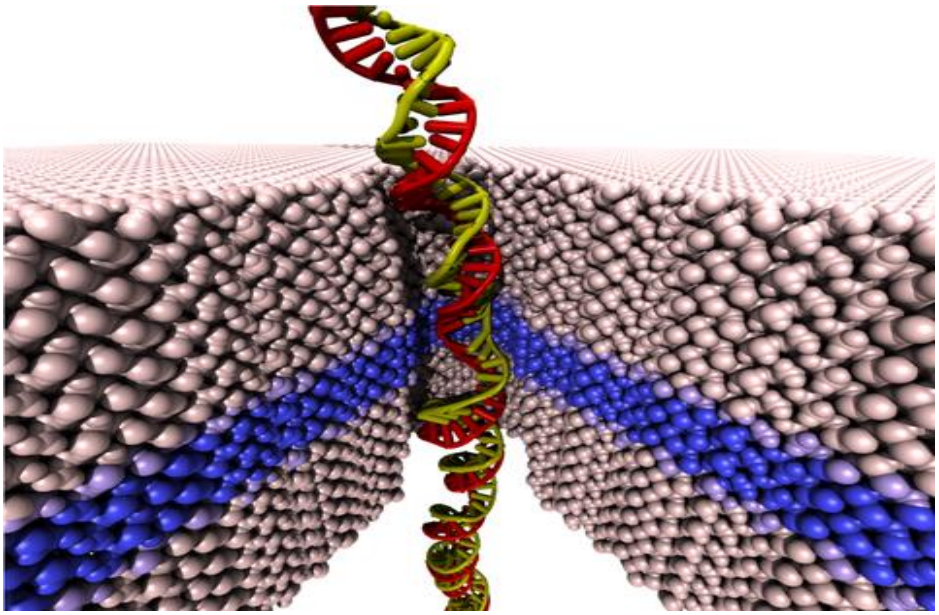
Фиг. 2.15. Представяне на макромолекулата на 3CSY.pdb



Фиг. 2.16. Представяне различни изгледи на макромолекулата на 3CSY.pdb



Фиг. 2.17. Фигура след рендъринг на 3CSY.pdb в Maya



Фиг. 2.18. Представяне на макромолекулата на ig. 4 Modeling Nanopores for Sequencing DNA, copy of image is from] http://bionano.physics.illinois.edu/sites/default/files/nanopore-protocols.pdf

2.3. Системи за геометрично моделиране

Графичните системи могат само да предлагат средства за моделиране (то само някои) *(Gallier, 2001*), а приложната програма е тази, която създава модела. Приложната програма извършва следните основни дейности, в центъра на които стои геометричният модел:

- **поддържане –** създаване, изтриване и модифициране на елементи и връзки в модела;
- **обхождане** за визуализация: представяне на геометричната информация модела за графичната система. Тази дейност може да бъде елементарна за модели, чиито обекти са примитиви на графичната система: отсечки, дъги и начупени. Тя може да бъде и сравнително сложна, ако обектите не се съхраняват във вид на графични примитиви, а всеки обект е сам по себе си алгоритъм за получаването на тези примитиви;
- **обхождане** при търсене и анализ: тази дейност е свързана с изпълнението на обработващите алгоритми, както и на някои специфични анализи върху него. Поради важността на модела и при наличието на мощни средства за неговото поддържане по-сложните интерактивни приложни програми се наричат системи за геометрично моделиране *(Dorsey et al., 2008).*

Освен изискванията за ефективно извършване на горните три дейности, в системите за геометрично моделиране се спазват и следните общи принципи:

- **правилност:** моделът да отразява вярно свойствата на реалните обекти, да не съдържа противоречиви данни;
- **мощност:** системата да позволява моделирането на достатъчно сложни реални обекти;
- **пълнота:** наличната информация да позволява

изчисляването на различни геометрични характеристики: площ, въртящ момент и др.

- **компактност:** възможност за ефективна програмна реализация;
- **отвореност:** възможност за добавяне на нови типове обекти, връзки и обработващи процедури.

Обектите в реалния свят рядко са неделими. Дори и в случай, че те са монолитни, човек моделира като съставени от части, всяка от които има определена функционалност. Човешкото тяло е неделимо, се описва като структура от ръце, крака, глава и т.н. Особено ясно йерархичното разделяне на части се вижда в обектите, които човек сам създава. В проектирането тази йерархия е най-добре обособена, като там дори всяко ниво от йерархията носи отделно име: детайл, възел, агрегат. Използването на йерархично структурирани компоненти в един модел дава редица предимства *(Sharpe et al., 2012):*

- възможност за създаване на всеки от компонентите поотделно и независимо един от друг;
- сложните обекти се създават чрез просто сглобяване на съставни компоненти;
- всеки компонент може да, бъде описан само веднъж, а включен като съставна част на много други обекти;
- възможност да се управлява изменението на обектите чрез изменението на техните компоненти.

Всичко може да се представи за нуждите на визуализацията чрез съвкупност от графични примитиви – отсечки и многоъгълници. Всеки от компонентите е сравнително независим и неговото описание в графични примитиви не зависи от конкретното му местоположение.

В йерархичните модели всеки обект има една дефиниция, но може да бъде разположен на много места едновременно. Всяко

отделно включване в ново местоположение се нарича екземпляр на обекта. Съставните компоненти на йерархията са именно екземплярите на някакъв набор от обекти *(Dallas, 2000).*

Една от основните задачи на графичната система е да осигури независимост на създаваните приложни програми от наличните физически устройства. Това важи както за изходните устройства - дисплей, плотер, принтер и др., така и за входните диалогови устройства. Поради по-голямото им разнообразие от това на изходните устройства, породено от различните по тип интерактивни дейности, които се обслужват, те са групирани в няколко класа. Графичните системи предлагат на приложните програми възможност за работа с няколко типа виртуални входни устройства, които съответстват именно на споменатите класове.

Всяко абстрактно входно устройство съответства точно на определена интерактивна дейност. Пълният набор от тези дейности е следният:

- **позициониране:** задаване на координати, местоположение на обект в някаква координатна система: потребителска, чертожна, нормирана, двумерна, тримерна и др. Съответното абстрактно устройство носи името локатор (locator);
- **избор от възможности:** определяне на една от определен брой възможности, осъществявано от абстрактното устройство избор (choice) *(Fryer, 2000);*
- **определяне на стойност:** задаване на стойност от непрекъснат интервал от реални числа. Съответното абстрактно устройство носи име валюатор (valuator);
- **въвеждане на текстов низ:** абстрактното устройство е добре познатата клавиатура (keyboard);
- **посочване:** избор на един от създадените до този момент обекти, които са предмет на обслужваната приложна графична програма. Съответното абстрактно устройство

носи името селектор (pick);

- **задаване на поредица от позиции:** задаване на произволно дълга последователност от координати. В много от системите тази дейност се осъществява с устройството локатор, но в някои системи като GKS и PHIGS съществува отделно абстрактно устройство – поредица (stroke).

Различните графични системи предлагат различен брой входни абстрактни устройства, както и различни правила за тяхното дефиниране. Докато в някои системи се среща само подмножество от целия набор абстрактни устройства, в други е възможно дори дефинирането едновременно на няколко устройства от един н същ абстрактен тип – GKS и PHIGS. Системи от типа на X Window System са малко по-зависими от конкретните физически устройства, предоставяйки на приложния програмист възможност да използва повече спецификата на конкретната конфигурация. Това естествено е за сметка на по-голяма обвързаност, по-голяма сложност на приложните програми.

З А К Л Ю Ч Е Н И Е

Във фокуса на този монографичен труд е да се покаже приложението на 3D моделите в научните изследвания. Изследването на протеиновата структура и прогнозирането на техните триизмерни (3D) структури е един от ключовите изследователски проблеми в структурната биоинформатика. Предвиждането на триизмерната структура на протеин, който няма шаблони в Protein Data Bank е много трудна и понякога практически неразрешима задача. През последните години са разработени много изчислителни методи, системи и алгоритми с цел решаване на този сложен проблем. Проблемът обаче все още предизвиква биолозите, биоинформатиците, химиците, компютърните учени и математиците поради сложността и високата размерност на пространството за търсене на протеинови конформации. В рамките на този огромен проблем се разглеждат два основни подпроблема, като акцента пада върху първия: (1) търсене на оптимална конформация при процеса на нагъване на протеини и (2) генериране на хидрофобно ядро при процеса на нагъване. В контекста на проблеми за търсене на оптимална конформация при процеса на нагъване на протеин, са проучени обещаващи евристични стратегии за генериране на конформации, които водят до ефективно решение на задачи свързани със сложни енергийни взаимодействия, като проблема за нагъване на протеини и проблема за намиране на самоизбягващи се пътища в граф.

За оценяване на изпълнението на описаните в монографията нови алгоритми за нагъване на протеини са използвани емпирични техники. Освен това, се прави опит да се разбере поведението и свойствата на алгоритмите чрез изучаване на влиянието на различните параметри, а именно различната дължина на сегментите, на които се дели протеиновата последователност и различния размер на решетката, в която се извършва нагъването с

цел да се изследва производителността и качеството на алгоритмите.

Основната цел на това проучване е да се разбере процеса на нагъване на протеини чрез прилагане и изучаване на различни модели и алгоритми. Въпреки, че за постигането на тази цел основно се използва силно опростения 2D и 3D HP модел, а не нещо по-реалистично, ясно се подчертава изключителната сложност на актуалния проблем. Концепцията на разделяне на последователност на сегменти позволява да се премине от решетъчни модели към извън решетъчни модели, които са обект на бъдещи разработки. Един пример за извън решетъчен модел е така наречения 8-ъглов извън решетъчен модел. При този модел 8-те ъгъла (45°, 90°, 135°, 180°, 225°, 270°, 315° и 360°) представляват възможните посоки за аминокиселините в протеиновата последователност.

Прогнозирането на протеиновата структура е много труден проблем и предстои да се направят допълнителни изследвания. Определено е необходимостта от разработването на нови стратегии, адаптирането и изследването на нови методи и комбинацията от съществуващи и съвременни изчислителни методи и техники. Разбирането на това как експерименталните данни могат да бъдат по-добре използвани в комбинация с Ab initio техники е друг открит изследователски въпрос. В обобщение, в тази област трябва да се проучат няколко възможности за научни изследвания и пътища със съответните мултидисциплинарни приложения в областта на компютърните науки, биоинформатиката, химията, биохимията и медицинските науки.

Пред нас стой предизвикателството да приложим тези алгоритми и модели в други типове решетъчни и извън решетъчни модели за нагъване протеини в 2D и 3D HP модела, да се генерират нагъвания за дълги протеинови последователности и разработените алгоритми да се приложат над реални протеинови структури, съхранявани в PDB.

ИЗПОЛЗВАНА И ЦИТИРАНА ЛИТЕРАТУРА

1. **Айала** Ж., Джон А. Кигер (1987) Съвременна генетика. Земиздат.
2. **Сигал**, И. & Иванова, А., (2002). Введение в прикладное дискретное программирование: модели и вычислительные алгоритмы. Москва: ФИЗМАТЛИТ.
3. **Abdel-Sayed** G., B. Bakht, L.G. Jaeger, (1990). Soil-steel Bridges: Design and Construction, McGraw-Hill Inc., New York.
4. **Agarwal**, P., Mustafa, N. & Wang, Y.,(2007). Fast molecular shape matching using contact maps. Journal of Computational Biology, 14, 131-143.
5. **Altman**, R.B., Dugan, J.M., (2005). Defining Bioinformatics and Structural Bioinformat- ics. John Wiley and Sons, Inc., Hoboken.
6. **America's** Army News. (2014). America's Army medic training helps save a life. http://news.americasarmy.com/americas-army-medictraining-helps-save-a-life/. Accessed Dec 31, 2014.
7. **Anfinsen** C. (1973). Principles that Govern the Folding of Protein Chains New Series, 181, No. 4096 (Jul. 20, 1973), 223-230.
8. **Anfinsen** C. et al. (1961). The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. Proc Natl Acad Sci U S A. 47(9): 1309–1314.
9. **Bailly,** F.; Maurin, C.; Teissier, E.; Vezin, H.; Cotelle, P. (2004) .Antioxidant Properties of 3-Hydroxycoumarin Derivatives. Bioorg. Med. Chem., 12, 5611−5618.
10. **Bales** F.B., (1985). Close-range photogrammetry for bridge measurement, in: Transportation Research Record: Journal of the Transportation Research Board, No. 950, TRB, National Research Council, Washington, DC, 39–44.
11. **Balreira** Dennis G., Marcelo Walter, Dieter W. (2018). Fellner, A survey of the contents in introductory Computer Graphics courses, Computers & Graphics, Volume 77, Pp. 88-96.

12. **Barzegar,** A.; Davari, M. D.; Chaparzadeh, N.; Zarghami, N.; Pedersen, J. Z.; Incerpi, S.; Saso, L.; Moosavi-Movahedi, A. A. (2011). Theoretical and Experimental Studies on the Structure-Antioxidant Activity Relationship of Synthetic 4-Methylcoumarins. J. Iran. Chem. Soc. 8, 973−982.
13. **Becke**, A. D. (1988). Density-Functional Exchange-Energy Approximation with Correct Asymptotic Behavior. Phys. Rev. A, 38, 3098−3100.
14. **Berger**, B. & Leighton, T., (1998). Protein folding in the hydrophobichydrophilic (HP) model is NP-complete. Journal of Computational Biology, 5(1), 27-40.
15. **Birney**, E. & Ponting, C., 2000. Identification of domains from protein sequences. Protein Structure Prediction: Methods and Protocols, pp. 53-70.
16. **Bouvier**, D. J. (2002). From pixels to scene graphs in introductory computer graphics courses, Computers & Graphics, Volume 26(4), 34-36.
17. **Branden**, C., Tooze, J., (1998). Introduction to Protein Structure, 2nd ed. Garland Pub- lishing Inc, New York.
18. **Burtch** R., (2004). History of Photogrammetry, Center for Photogrammetric Training, Ferris State University, Big Rapids, Michigan.
19. **Buss**, S., & Fillmore, (2001). J. Spherical averages and applications to spherical splines and interpolation. ACM Transactions on Graphics (TOG), 20(2), 95–126.
20. **Cain** J, Conway JM, Divall MV, et al. (2015). Report of the 2013-2014 Academic Affairs Committee. Am J Pharm Educ.
21. **Cain J**eff, Peggy Piascik, (2015). Are Serious Games a Good Strategy for Pharmacy Education? American Journal of Pharmaceutical Education, 79 (4) 23-31.
22. **Cañellas** Barrera, M., Franconetti Manchado, J., & Melo Pereira, L. (2019). Cultural tourism in the balearic islands: a case of efficient use of cultural historical heritage. Journal of Tourism

and Heritage Research, 2(2), 37-48. Retrieved from http://www.jthr.es/index.php/journal/article/view/40

23. **Caprara**, A. & Lancia, G., (2002). Structural alignment of large-size proteins via lagrangian relaxation. 23, 100-108.
24. **Castro**-Sa´nchez E, Charani E, Moore L, Gharbi M, Holmes A. (2014). "On call: antibiotics"-development and evaluation of a serious antimicrobial prescribing game for hospital care. Games for Health 2014: Springer; 1-7.
25. **Chikenji** Y.,, G., Takada, S., (2006). Simfold energy function for de novo protein structure prediction: consensus with rosetta. Proteins: Struct. Funct. Gen. 62 (2), 381.
26. **Clote,** P., Backofen, R., (2000). Computational Molecular Biology: An Introduction, 1st ed. John Wiley and Sons Inc., West Sussex.
27. **Cooper** M.A.R., S. Robson, (2000). Theory of close range photogrammetry, in: Close Range Photogrammetry and Machine Vision, Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, KW5 6DW, Scotland, UK, 9–50.
28. **Craddock**, T. J. A., Friesen, D., Mane, J., Hameroff, S., & Tuszynski, J. A. (2014). The feasibility of coherent energy transfer in microtubules. Journal of the Royal Society, Interface / the Royal Society , 11 (100), 20140677.
29. **Crescenzi,** P., Goldman, D., Papadimitriou, C., Piccolboni, A., and Yannakakis, M., (1998). On the complexity of protein folding, Journal of Computational Biology, 5(3), 423–466.
30. **Dallas** R.W.A, (2000). Architectural and archaeological photogrammetry, in: Close Range Photogrammetry and Machine Vision, Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, KW5 6DW, Scotland, UK, 283–302.
31. **Debevec,** P. E., Taylor, C. J., & Malik, J. (1996). Modeling and rendering architecture from photographs: a hybrid geometry-and image-based approach. In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, ACM, pp. 11–20.

32. **Dhingra**, P., Jayaram, B., (2013). A homology/ab initio hybrid algorithm for sampling near-native protein conformations. J. Comput. Chem. 34 (22).
33. **Di Giulio**, M. and Medugno, M. (1998) The historical factor: the biosynthetic relationships between amino acids and their physicochemical properties in the origin of the genetic code. J. Mol. Evol. 46, 615–621.
34. **Dill**, K. et al., (1995). Principles of protein folding - a perspective from simple exact models. Protein Sci, Том 4, pp. 561-602.
35. **Dill**, K., (1990). Dominant forces in protein folding. Biochemistry, 29, 7133-7155.
36. **Dill**, K., Banu Ozkan, S., Scott Shell, M. & Weikl, T., (2008). The protein folding problem. Annual Review of Biophysics, Том 37, pp. 289-316.
37. **Dorn** et al., (2014). Three-dimensional protein structure prediction: Methods and computational strategies. Computational Biology and Chemistry 53, 251–276.
38. **Dorn**, M., Breda, A., Norberto de Souza, O., (2008a). A hybrid method for the protein structure prediction problem. Lect. Notes Bioinf. 5167, 47.
39. **Dorn**, M., Norberto de Souza, O., (2008b). CReF: A Central-residue-fragment-based Method for Predicting Approximate 3-D Polypeptides Structures. ACM, New York.
40. **Dorsey**, J., Rushmeier, H., AND Sillion, F., (2008). Digital Modeling of Material Appearance. Morgan Kaufmann,
41. **Dummer**, A., Poelma, C., DeRuiter, M. C., Goumans, M.-J. T. H., & Hierck, B. P. (2016). Measuring the primary cilium length: improved method for unbiased high-throughput analysis. Cilia, 5 , 7.
42. **Dunker**, A., Silman, I., Uversky, V., Sussman, J., (2008). Function and structure of inher- ently disordered proteins. Curr. Opin. Struct. Biol. 18 (6), 756.

43. **Eastman** (2004). Kodak Company, Kodak Professional DCS Pro SLR/n Digital Cameras User's Guide, Rochester, NY.
44. **Empler** T. (2018) 3D Modeling of an Archeological Area: The Imperial Fora in Rome. In: Rossi M., Buratti G. (eds) Computational Morphologies. Springer, Cham 185 -895.
45. **Evans**, C.; Scaiano, J. C.; Ingold, K. U. Absolute kinetics of hydrogen abstraction from .alpha.-tocopherol by several reactive species including an alkyl radical. J. Am. Chem. Soc. 1992, 114, 4589−4593.
46. **Fan**, H., Mark, A., (2004). Refinement of homology-based protein structures by molecular dynamics simulation techniques. Protein Sci. 13(1), 211.
47. **Farin** G. (1983). Algorithms for rational Bézier curves, Computer-Aided Design, 15(2), 73-77, ISSN 0010-4485.
48. **Feynman**, R. (1963). The Feynman Lectures on Physics: Volume 1, vol. 3 of The Feynman Lectures on Phsyics. Addison-Wesley.
49. **Foresman** J. B. and E. Frish, (1996). Exploring Chemistry with electronic Structure Methods, Guassian, Inc., Pittsburgh, Second edition.
50. **Fraenkel**, A.S., (1993). Complexity of protein folding. Bull. Math. Biol. 55 (6), 1199. Fujitsuka.
51. **Fraser** C .S., (2000). Industrial measurement applications, in: Close Range Photogrammetry and Machine Vision, Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, KW5 6DW, Scotland, UK, 329–361.
52. **Freitas**, R., & Campos, P. (2008). SMART: a System of augmented reality for teaching 2nd grade students. Proceedings of the 22nd British Computer Society Conference on Human-Computer Interaction (HCI 2008), 27-30. Liverpool John Moores University, UK.
53. **Fryer** J.G, (2000). Introduction, in: Close Range Photogrammetry and Machine Vision, Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, KW5 6DW, Scotland, UK, 1–7.

54. **Gallier,** J. (2001). Geometric Methods and Applications: for Computer science and Engineering. Springer-Verlag.
55. **Garey,** M., Johnson, D., (1979). Computers and Intractability: A Guide to the Theory of NP-Completeness, 1st ed. W.H. Freeman, New York.
56. **GE,** Q. J., & Ravani, B. (1994). Computer aided geometric design of motion interpolants. Journal of Mechanical Design, 116(3), 756–762.
57. **Georgiev** G., (2019). Digitalization of Bulgarian Cultural Heritage, Journal of Economic Development, Environment and People, 8(1) 6-17.
58. **Gibas**, C., Jambeck, P., (2001). Developing Bioinformatics Computer Skills, 1st ed.
59. **Goldman**, D., 2000. Algorithm aspects of protein folding and protein structure similarity. н.м.:U.C. Berkeley.
60. **Goldman**, D., Istrail, S. & Papadimitriou, C., (1999). Algorithmic aspects of protein structure similarity. н.м., IEEE Computer Society Press.
61. **Gopakumar**, O., (2012). Bioinformatics: Sequence and Structural Analysis. Alpha Sci- ence Intl Ltd, Oxford.
62. **Gortler** Steven J. (2015). Foundations of 3D Computer Graphics, Foundations of 3D Computer Graphics, by A K Peters/CRC Press.
63. **Graafland** M, Schraagen JM, Schijven MP. (2012). Systematic review of serious games for medical education and surgical skills training. Br J Surg. 99(10), 1322-1330.
64. **Gruen** A., (2000). Development of digital methodology and systems, in: Close Range Photogrammetry and Machine Vision, Whittles Publishing, Roseleigh House, Latheronwheel, Caithness, KW5 6DW, Scotland, UK, 78–104.
65. **Guarnieri** et al. (2009). An open source application for interactive exploration of cultural heritage 3d models on the web.
https://www.researchgate.net/profile/Francesco_Pirotti/publ

ication/228410186_AN_OPEN_SOURCE_APPLICATION_FOR_INTERACTIVE_EXPLORATION_OF_CULTURAL_HERITAGE_3D_MODELS_ON_THE_WEB/links/02e7e5231601ad84bc000000/AN-OPEN-SOURCE-APPLICATION-FOR-INTERACTIVE-EXPLORATION-OF-CULTURAL-HERITAGE-3D-MODELS-ON-THE-WEB.pdf

66. **Gunasekaran**, K., Tsai, C., Kumar, S., Zanuy, D., Nussinov, R., (2003). Extended disor- dered proteins: targeting function with less scaffold. Trends Biochem. Sci. 28 (2), 81.
67. **Guntert,** P., (2004). Automated nmr structure calculation with cyana. Methods Mol. 23, 278-353.
68. **Gurwitsch**, A. (1922). Über den Begriff des Embryonalen feldes. Wilhelm Roux' Archiv fur Entwicklungsmechanik der Organismen , 51 (1), 383–415.
69. **Guschin** et al. (2018). On the function of DNA magnetism, http://vixra.org/abs/1803.0075, Biol. 278, 353.
70. **Hameroff,** S. (2008). That's Life!—The Geometry of π Electron Cloudsл Quantum aspects of life,
71. **Hameroff** S., Alex Nip, Mitchell Porter, Jack Tuszynski, (2002) Conduction pathways in microtubules, biological quantum computation, and consciousness, Biosystems, Volume 64, Issues 1–3, 149-168.
72. **Hameroff**, S. (2008). That's Life!—The Geometry of π Electron Clouds. In *Quantum Aspects of Life* (pp. 403–434). World Scientific.
73. **Hameroff**, S. R. (1994). Quantum coherence in microtubules: A neural basis for emergent consciousness? *Journal of Consciousness Studies* , *1* (1), 91–118.
74. **Hameroff**, S., Nip, A., Porter, M., & Tuszynski, J. (2002). Conduction pathways in microtubules, biological quantum computation, and consciousness. *Bio Systems* , *64* (1-3), 149–168.
75. **Hanisch** F., Wolfgang Straßer, (2005). How to include visuals and interactivities in an educational computer graphics repository, Computers & Graphics, Volume 29, Issue 2.

76. **Haozhao** Liang, Jie Meng, Shan-Gui Zhou, (2015). Hidden pseudospin and spin symmetries and their origins in atomic nuclei, Physics Reports, Volume 570, , Pages 1-84, ISSN 0370-1573.
77. **Hart**, W. & Istrail, S., (1995). Fast protein folding in the Hydrophobic-Hydrophilic model within three-eighths of optimal (extended abstract). н.м., н.а., pp. 157-168.
78. **Hart,** W. & Istrail, S., (1997). Lattice and off-lattice side chain models of protein folding: Linear time structure prediction better than 86. Journal of Computational Biology, Том 4, pp. 241-259.
79. **Hart**, W., Istrail, S., (1997). Robust proofs of np-hardness for protein folding: general lattices and energy potentials. J. Comput. Biol. 4 (1), 78 -82.
80. **Heckbert**, P. S. Fundamentals of texture mapping and image warping. Tech. Rep. UCB/CSD-89-516, EECS Department, University of California, Berkeley, Jun 1989.
81. **Hering** L. Closed (1983). (C2- and C3-continuous) Bézier and B-spline curves with given tangent polygons. Computer-Aided Design https://doi.org/10.1016/S0010-4485(83)80042-6.
82. **Ishima**, R., Torchia, D. (2000). Protein dynamics from NMR. *Nat Struct Mol Biol* **7,** 740–743 https://doi.org/10.1038/78963
83. **Jauregui** D.V., K.R. White, (2010). Bridge inspection using virtual reality and photogrammetry, in: Inspection and Monitoring Techniques for Bridges and Civil Structures, Woodhead Publishing Limited, Abington Hall, Abington, Cambridge CB 1 6AH, England, 216–246.
84. **Jianga** Ruinian, David V.Jáureguib Kenneth R.Whiteb (2008). Close-range photogrammetry applications in bridge measurement: Measurement Volume 41(8), 823-834.
85. **Johnson**, L., Levine, A., Smith, R., & Stone, S. (2010). Simple augmented reality. The 2010 Horizon Report, 21-24. Austin, TX: The New Media Consortium.

86. **Jones**, D., (2001). Predicting novel protein folds by using fragfold. Proteins: Struct. Funct. Gen. 45 (S5), 127.
87. **Karastoyanov** D., Doukovska L., Angelova G., Yatchev I. (2020). Intelligent Approach for Analysis of 3D Digitalization of Planer Objects for Visually Impaired People. In: Jardim-Goncalves R., Sgurev V., Jotsov V., Kacprzyk J. (eds) Intelligent Systems: Theory, Research and Innovation in Applications. Studies in Computational Intelligence, vol 864. Springer, Cham.
88. **Kerfoot** BP & Kissane N. (2014). The use of gamification to boost residents' engagement in simulation training. JAMA Surg. 149 (11):1208-1209.
89. **Kim** B.G., (1989). Development of a photogrammetric system for monitoring structural deformations of the sturgeon bay bridge, PhD Dissertation, University of Wisconsin, Madison.
90. **Kittel** C.. (1996). Introduction to Solid State Physics. John Wiley & Sons, Inc., Singapore, seventh edition.
91. **Knight**, R.D. Freeland S.J., F. Landweber (1999). Selection, history and chemistry: the three faces of the genetic code. Trends Biochem. Sci. 24, 241–247.
92. **Kolodziej**, H.; Kayser, O.; Woerdenbag, H. J.; Uden, W. v.; Pras, N. (1997). Structure -Cytotoxicity Relationships of a Series of Natural and Semi-Synthetic Simple Coumarins as Assessed in Two Human Tumour Cell Lines. Z. Naturforsch., C: J. Biosci. 52, 240−244.
93. **Kondo,** T. (2006). Augmented learning environment using mixed reality technology, Proc. E-Learn, 83-88.
94. **Kostova**, I. (2005). Synthetic and Natural Coumarins as CytotoxicAgents. Curr. Med. Chem. Anti Cancer Agents 5, 29−46.
95. **Kundrot**, C., Ponder, J., Richards, F., (1991). Algorithms for calculating excluded volume and its derivatives as a function of molecular conformation and their use in energy minimization. J. Comput. Chem. 12 (3), 402.

96. **Lancia**, G., Carr, R., Walenz, B. & Istrail, S., (2001). 101 optimal PDB structure alignments: A branch-and-cut algorithm for the maximum contact map overlap problem. RECOMB, Volume 1, pp. 201–211.
97. **Laskowiski,** R., Watson, J., Thornton, J., (2005a). Profunc: a server for predicting protein functions from 3d structure. Nucleic Acids Res. 33, 89.
98. **Laskowiski,** R., Watson, J., Thornton, J., (2005b). Protein function prediction using local 3d templates. J. Mol. Biol. 351, 614.
99. **Lathrop**, R., (1994). The protein threading problem with sequence amino acid interaction preferences is NP-complete. Protein Engineering, Том 7, pp. 1059-1068.
100. **Le Grand**, S., Merz, K.J., (1993). The application of the genetic algorithm to the minimization of potential energy functions. J. Global Optim. 3 (1), 49.
101. **Lee Kangdon** (2012). Augmented Reality in Education and Training TechTrends Volume 56, Number 2, 13-21.
102. **Lee**, D., amd Redfern, O., Orengo, C., (2007). Predicting protein function from sequence and structure. Nat. Rev. Mol. Cell Biol. 8, 995.
103. **Lee,** J., Kim, S., Joo, K., Kim, I., Lee, J., (2004). Prediction of protein tertiary structure using profesy, a novel method based on fragment assembly and conformational space annealing. Proteins: Struct. Funct. Gen. 56 (4), 704.
104. **Lee**, J., Pillardy, J., Czaplewski, C., Arnautova, Y., Ripoll, D., Liwo, A., Gibson, K.D., Wawak, R., Scheraga, H., (2000). Efficient parallel algorithms in global optimization of potential energy functions for peptides, proteins, and crystals. Comput. Phys. Commun. 128 (1-2), 399.
105. **Lee,** J., Ripoll, D., Czaplewski, C., Pillardy, J., Wedemeyer, W., Scheraga, H., (2001). Optimization of parameters in macromolecular potential energy functions by conformational space annealing. J. Phys. Chem. B 105 (30), 7291.

106. **Lee**, J., Scheraga, H., (1999). Conformational space annealing by parallel computa- tions: extensive conformational search of met-enkephalin and of the 20-residue membrane-bound portion of melittin. Int. J. Quantum Chem. 75, 255.
107. **Lensink**, Marc & Nadzirin, Nurul & Velankar, Sameer & Wodak, Shoshana. (2019). Modeling protein-protein, protein-peptide and protein-oligosaccharide complexes: CAPRI 7th edition. Proteins: Structure, Function, and Bioinformatics. 10.1002/prot.25870.
108. **Levinthal,** C., (1968). Are there pathways for protein folding?. J. Chem. Phys., Том 65, pp. 44-45.
109. **Liljas**, A., Liljas, L., Pskur, J., Lindblom, G., Nissen, P., Kjeldgaard, M., (2001). Textbook of Structural Biology. World Scientific Printers, Singapore.
110. **Lippincott**-Schwartz, J., Snapp, E. & Kenworthy, A. (2001). Studying protein dynamics in living cells. Nat Rev Mol Cell Biol 2, 444–456. https://doi.org/10.1038/35073068
111. **Lopes** R. O. et al., (2019). Exploring Digital Architectural Heritage in Brunei Darussalam: Towards Heritage Safeguarding, Smart Tourism, and Interactive Education, 2019 IEEE Fifth International Conference on Multimedia Big Data (BigMM), Singapore, Singapore, 383-390.
112. **Luce** Henry R. (Editor). December 23, (1966). Special Double Issue: Photography. Life Magazine (December 23, 1966), 112–113.
113. **Lyu**, P., Liff, M., Marky, L. & Kallenbach, N., (1990). Side chain contributions to the stability of a-helical structure in peptides. Science, 250, 669-673.
114. **Mann** Steve, Tom Furness, Yu Yuan, Jay Iorio, and Zixin Wang (2014). All Reality: Virtual, Augmented, Mixed (X), Mediated (X,Y), and Multimediated Reality HC conference, https://arxiv.org/pdf/1804.08386.pdf 2018 11-26
115. **Marković**, Z.; Tošovic, J.; Milenkovic, D.; Markovic, S., (2016). Revisiting the Solvation Enthalpies and Free Energies of the

Proton and Electron in Various Solvents. Comput. Theor. Chem. 1077, 11−17.

116. **Matheson**, Granville & Sigray, Pontus & Louzolo, Anaïs & Borg, Jacqueline & Farde, Lars & Petrovic, Predrag & Cervenka, Simon. (2018). Dopamine D1 receptor availability is not associated with delusional ideation measures of psychosis proneness. 10.1101/321646.
117. **McQuarrie** D. A, (1983). Quantum Chemistry, University Science Books, Sausalito.
118. **Mendes**, J., Baptista, A., Carrondo, M. & Soares, C., (1999). Improvement of side-chain modeling in proteins with the self-consistent mean field theory method based on an analysis of the factors influencing prediction. Biopolymers, 50, 111-131.
119. **Michalewicz**, Z. & Fogel, B., (2004). How to Solve It: Modern Heuristics. Berlin: Springer.
120. **Micron**, CMOS image sensors. <http://www.micron.com/products/imaging>. (accessed July 2007).
121. **Mikhail** E. M., J.S. Bethel, J.C. McGlone, (2001). Introduction to Modern Photogrammetry, John Wiley & Sons, Inc., New York.
122. **Milgram** P. & Fumio KISHINO (1994). A Taxonomy of Mixed Reality Visual Displays. IEICE TRANSACTIONS on Information and Systems Vol.E77-D No.12 pp.1321-1329 http://vered.rose.toronto.edu/people/paul_dir/SPIE94/SPIE94.full.html
123. **Milin**, M., von Oertzen, W., (2002).Search for molecular bands in 13C. Eur Phys J A 14, 295–307 https://doi.org/10.1140/epja/i2001-10199-6
124. **Mills** J., D. Barber, (2004). Geomatics techniques for structural surveying, Journal of Surveying Engineering 130 (2) 56–64.
125. **Misra** J. C. & Mukherjee. (2004). A mathematical model for enzymatic action on DNA knots and links, Math. Computer Modelling, 39, 1423-1430.

126. **Miyazawa**, S. & Jernigan, R., (1985). Estimation of effective interresidue contact energies from protein crystal structures: Quasi-chemical approximation. 18, 534-552.
127. **Moult,** J., Fidelis, K., Kryshtafovych, A., Schwede, T., Tramontano, A., (2014). Critical assessment of methods of protein structure prediction (casp) – round x. Proteins: Struct. Funct. Bioinf. 82, 1–6.
128. **Moult,** J., Fidelis, K., Kryshtafovych, A., Tramontano, A., (2011). Critical assessment of methods of protein structure prediction (casp) – round ix. Proteins: Struct. Funct. Bioinf. 79 (S10), 1.
129. **Murzina**, A., Brennera, S., Hubbarda, T. & Chothia, C., (1995). SCOP: A structural classification of proteins database for the investigation of sequences and structures. J. Mol. Biol., 247, 536-540.
130. **Ngo,** J., Marks, J., Karplus, M., (1997). The protein folding problem and tertiary structure prediction. In: Merz Jr., K.J., Grand, S. (Eds.), Computational complexity, protein structure prediction and the Levinthal Paradox. Birkhauser, Boston, p. 435.
131. **Nicholls,** M. (2017) Classics and 3D digital modelling at the University of Reading. In: Fung, D. (ed.) A connected curriculum for higher education. UCL Press, London, pp. 52-53. ISBN 9781911576334.
132. **Ofran**, Y. & Margalit, H., (2006). Proteins of the same fold and unrelated sequences have similar amino acid composition. Proteins: Structure, Function and Genetics, 64, 275-279.
133. **Orts** E. (2014.). Tragedy of the tuna. http://simulations.wharton.upenn. edu/solutions/tragedy-of-the-tuna/. Accessed Sept 2, 2014.
134. **Pappa** P.S. et al., (2001). Photogrammetry of a 5 m inflatable space antenna with consumer-grade digital cameras, Experimental Techniques. 25(4), 21–29.

135. **Parilov**, E., & Zorin, D. (2008). Real-time rendering of textures with feature curves. ACM Transactions on Graphics (TOG) 27, 1, 3:1–3:15.
136. **Patricia** S. Moyer-Packenham, Christina W. Lommatsch, Kristy Litster, Jill Ashby, Emma K. Bullock, Allison L. Roxburgh, Jessica F. Shumway, Emily Speed, Benjamin Covington, Christine Hartmann, Jody Clarke-Midura, Joel Skaria, Arla Westenskow, Beth MacDonald, Jürgen Symanzik, Kerry Jordan, (2019) How design features in digital math games support learning and mathematics connections, Computers in Human Behavior, 91, 34-45.
137. **Paul Milgram**. (1994). Augmented Reality: A Class Of Displays On The Reality-Virtuality Continuum.
138. **Pevzner**, P.A., (2000). Computational Molecular Biology: An Algorithmic Approach, 1st ed. The MIT Press, Cambridge.
139. **Pilar** F.L, (1968). Elementary Quantum Chemistry, Mc Graw-Hill Book Company, New York.
140. **Poshusta,** R. D., V. P. Agrawal and W. D. Moseley, 1975, Intrenational Journal of Quantum Chemistry, 9, 635-647.
141. **Qiu,** D., Shenkin, P., Hollinger, F., Still, W., (1997). The gb/sa continuum model for solvation. a fast analytical method for the calculation of approximate born radii. J. Phys. Chem. A 101, 3005.
142. **Rackovsky**, S., (2010). Global characteristics of protein sequences and their implica- tions. Proc. Natl. Acad. Sci. U. S. A. 107 (19), 8623.
143. **Ren**, Z., Zhang, S., Zhao, P. et al. (2019). Stability of the linear chain structure for 12C in covariant density functional theory on a 3D lattice. Sci. China Phys. Mech. Astron. 62, 112062 https://doi.org/10.1007/s11433-019-9412-3
144. **Rentzsch**, R., Orengo, C., 2009. Protein function prediction – the power of multiplic- ity. Trends Biotechnol. 27 (4), 210.
145. **Reva,** B., Finkelstein, A. & Skolnick, J., (1998). A self-consistent field optimization approach to build energetically and

geometrically correct lattice models of proteins. н.м., ACM Press.

146. **Reva**, B., Finkelstein, A., Rykunov, D. & Olson, A., (1996). Building self-avoiding lattice models of proteins using a self-consistent field optimization. Proteins: Structure, Function and Genetics, 26, 1-8.
147. **Richardson**, J., (1977). Beta-sheet topology and the relatedness of proteins. Nature, 268, 495-500.
148. **Richardson**, J., (1981). The anatomy and taxonomy of protein structure. Advances in Protein Chemistry, 34, 167-339.
149. **Rodriguez**-Tello, Eduardo & Lardeux, Frédéric & Duarte, Abraham & Narvaez-Teran, Valentina. (2018). Alternative evaluation functions for the cyclic bandwidth sum problem. European Journal of Operational Research. 10.1016/j.ejor.2018.09.031.
150. **Rushdi**, Ahmad. (2010). A mathematical model of DNA replication. International Magazine on Advances in Computer Science and Telecommunications. 1. 23-30.
151. **Sali**, A., Shakhnovich, E. & Karplus, M., (1994a). Kinetics of protein folding a lattice model study of the requirements for folding to the native state. J. Mol. Biol., 235, 1614-1636.
152. **Sančanin**, B., Perić, G., & Stojiljković, M. (2019). Cultural-historical resources as initiators of tourism development in Sremski Karlovci. Hotel and Tourism Management, 7(2), 77-85. https://doi.org/10.5937/menhottur1902077S
153. **Sanchez**, R. & Sali, A., (2000). Comparative protein structure modeling: Introduction and practical examples with modeller. Protein Structure Prediction: Methods and Protocols, 97–129.
154. **Scheicher** R.H., (2004). The Role of the Impurities in the Molecular solids and the Biological Materials, , Ph. D. thesis submitted to the Department of Physics, University at Albany, State University of New York.
155. **Schindler** W., (1997). On the efficient simulation of a particular class of random rotations relevant to computer graphics,

Journal of Computational and Applied Mathematics, Volume 81,(1), 107-114.

156. **Schouten** B, Fedtke S, Bekker T, Schijven M, Gekker A. (2013). Games for Health: Proceedings of the 3rd Conference on Gaming and Playful Interaction in Health Care.
157. **Sharpe** Jason et al., (2012) Silico: 3D Animation and Simulation of Cell Biology with Maya and MEL. Morgan Kaufmann Publishers is an imprint of Elsevier.
158. **Sheldrake** R. (2009). Morphic Resonance: The Nature of Formative Causation. Inner Traditions/Bear & Co.
159. **Shmygelska**, A. & Hoos, H., (2005). An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC Bioinformatics, 6, 30-52.
160. **Shoemake**, K. Animating rotation with quaternion curves. ACM SIGGRAPH Computer Graphics 19, 3 (1985), 245–254.
161. **Skolnick**, J. & Kolinski, A., (1990). Simulations of the folding of a globular protein. Science, Том 250, pp. 1121-1125.
162. **Slavova-Kazakova**, V. D.;, A. K.; Angelova, S. E.; Singh, S. K.; Malhotra, S.; Singh, B. K.; Saso, L.; Prasad, A. K.; Parmar, V. S. (2017) Protective Effects of 4-Methylcoumarins and Related Compounds as Radical Scavengers and Chain-Breaking Antioxidants. Biochimie, 140, 133−145.
163. **Snickars** P. (2019). Metamodeling : 3D-(re)designing Polhem's Laboratorium mechanicumn. In: Der Modelle Tugend 20 : Digitale 3D-Rekonstruktion als virtueller Raum der architekturhistorischen Forschung [Internet]. Heidelberg: arthistoricum.net; 509–28. Available from: http://urn.kb.se/resolve?urn=urn:nbn:se:umu:diva-164711
164. **Song**, J., Cheng, J. & Zheng, T., (2006). Protein 3D HP model folding simulation based on ACO. н.м., н.а., 410-415.
165. **Sternberg**, M. & Thornton, J., (1977). On the conformation of proteins: The handedness of the connection between parallel beta strands. J. Mol. Biol., 110, 269-283.

166. **Strickland**, D., Barnes, E. & Sokol, J., (2005). Optimal protein structure alignment using maximum cliques. Operation Research, Том 53, pp. 389-402.
167. **Stump** D.M., P.J. Watson & W.B. Fraser (2000). Mathematical modelling of interwound DNA supercoil , J. of Biomechanics, 33, 407-413.
168. **Sutherland,** I. (1968). A head-mounted three-dimensional display. Proceedings of Fall Joint Computer Conference, 757-764.
169. **Swigon** D., D.B. Coleman and I. Tobias (1998). The elastic rod model for DNA and its application to the tertiary structure of DNA minicircles in mononucleasomes, Biophysical J., 74, 2515-2530
170. **Szabo** Attila and Neil S. Ostlund, (1996). Modern Quantum Chemistry, Introduction to Advanced Electronic Structure Thoery, Dover Publications Inc, New York, second edition.
171. **Taylor,** W., (2000). Protein structure comparison using sap. Protein Structure Prediction: Methods and Protocols, pp. 19-32.
172. **Thijssen** J.M., (1999). Computational Physics, Cambridge University Press, Cambridge, first edition.
173. **Thilagavathi**, N. & Amudha, T., (2015). Aco-metaheuristic for 3D-hp protein folding optimization. ARPN Journal of Engineering and Applied Sciences, 10, 4948-4953.
174. **Tompa**, P., 2002. Intrinsically unstructured proteins. Trends Biochem. Sci. 27 (10), 527.
175. **Tompa**, P., Csermely, P., (2004). The role of structural disorder in the function of RNA and protein chaperones. FASEB J. 18 (11), 1169.
176. **Tramontano**, A., (2006). Protein Structure Prediction, 1st ed. John Wiley and Sons, Inc., Weinheim.
177. **Traykov**, M., Angelov, S. & Yanev, N., (2016). A new heuristic algorithm for protein folding in the hp model. J. Comp. Biol., 23, 662-668.

178. **U.S. Army** and Project (2014). Lead the Way partner on technology education program. http://www.prnewswire.com/newsreleases/us-army-and-project-lead-the-way-partner-on-technologyeducation-program-65124492.html. Accessed Dec 31, 2014.
179. **Veland,** I. R., Montjean, R., Eley, L., Pedersen, L. B., Schwab, A., Goodship, J., … Christensen, S. T. (2013). Inversin/Nephrocystin-2 is required for fibroblast polarity and directional cell migration. *PloS One* , *8* (4), e60193.
180. **Vinson**, Valda. (2017). Pulling apart protein unfolding. Science. 355. 920.16-922. 10.1126/science.355.6328.920-p.
181. **Voet,** D. & Voet, J., (1995). Biochemistry, 2nd Edition:John Wiley & Sons.
182. **Wang**, Zhijun & Wieder, Benjamin & Li, Jian & Yan, Binghai & Bernevig, B.. (2019). Higher-Order Topology, Monopole Nodal Lines, and the Origin of Large Fermi Arcs in Transition Metal Dichalcogenides X Te 2 (X = Mo , W). Physical Review Letters. 123. 10.1103/PhysRevLett.123.186401.
183. **Whisstock**, J., Lesk, A., (2003). Prediction of protein function from protein sequence and structure. Q. Rev. Biophys. 36 (3), 307.
184. **Whitford**, D., (1995). Proteins: Structure and Function. Chichester, West Sussex, England: JohnWiley & Sons.
185. **Wiers**, Vincent & de Kok, Ton. (2018). Vendor Selection. 10.1007/978-3-319-65055-5_6.
186. **Wind**, Yoram & Robinson, Patrick. (1968). The Determinants of Vendor Selection: The Evaluation Function Approach. Journal of Purchasing and Materials Management. 4. 29-41. 10.1111/j.1745-493X.1968.tb00592.x.
187. **Wright,** P., Dyson, H., (1999). Intrinsically unstructured proteins: re-assessing the protein structure–function paradigm. J. Mol. Biol. 293 (2), 321.

188. **Xu**, J., Li, M. & Xu, Y., (2004). Protein threading by linear programming: theoretical analysis and computational results. Journal of Combinatorial Optimization, 8, 403-418.
189. **Yamaguchi** H., D. Kahl, S. Hayakawa, Y. Sakaguchi, K. Abe, T. Nakao, T. Suhara, N. Iwasa, A. Kim, D.H. Kim, S.M. Cha, M.S. Kwag, J.H. Lee, E.J. Lee, K.Y. Chae, Y. Wakabayashi, N. Imai, N. Kitamura, P. Lee, J. (2017). Moon, K.B. Lee, C. Akers, H.S. Jung, N.N. Duy, L.H. Khiem, C.S. Lee, 92017, Experimental investigation of a linear-chain structure in the nucleus 14C, Physics Letters B, 766, 11-16.
190. **Yanev**, N., Andonov, R., Veber, P. & Balev, S., (2008). Lagrangian Approaches for a class of Matching Problems in Computational Biology. Comput. Math. Appl., 55, 1054-1067.
191. **Yanev**, N., Milanov, P. & Mirchev, I., (2011). Integer programming approaches to HP folding. Serdica J. Computing, 5, pp. 359-366.
192. **Yanev,** N., Traykov, M., Milanov, P. & Yurukov, B., (2017). Protein folding prediction in a cubic lattice in hydrophobic-polar model. J. Comp. Biol., 24, 412-421.
193. **Yoon,** H., (2006). Optimization Approaches to Protein Folding. Georgia: School of Industrial and System Engineering, Institute of Technology.
194. **Yoshida** Shinsuke, Masahiro Shin, Taichi Kin, Nobuhito Saito. (2020). 3-dimensional computer graphics model elucidating involvement of intraparenchymal venous malformation in trigeminal nucleus of brachium pontis with intractable trigeminal neuralgia, Journal of Clinical Neuroscience.
195. **Yu,** Y., (2002). Coiled-coils: Stability, specificity and drug delivery potential. Advanced Drug Delivery Reviews, 54, 1113-1129.
196. **Zhang,** Pengcheng & Wang, Lin & Huang, Zhiwei & Yu, Jipan & Li, Zijie & Deng, Hao & Yin, Taiqi & Li, Jun & Gibson, John & Mei, Lei & Zheng, Lirong & Wang, Hongqing & Chai, Zhifang & Shi, Weiqun. (2020). Aryl Diazonium-Assisted

Amidoximation of MXene for Boosting Water Stability and Uranyl Sequestration via Electrochemical Sorption. ACS Applied Materials & Interfaces. XXXX. 10.1021/acsami.0c00861.

197. **Zhang**, Q., Veretnik, S., Bourne, P.E., (2005). Overview of Structural Bioinformatics.
198. **Zhang**, Y. & Skolnick, J., (2005). TASSER: An automated method for the prediction of protein tertiary structures in CASP6. Proteins, 61, 91-98.
199. **Zhour** et al., (2008). Mathematical Properties of DNA Structure in 3-Dimensional Space Int. J. Open Problems Compt. Math., 1(3), 23-45.

ПОСЛЕСЛОВ

БЕЛЕЖКИ НА РЕЦЕНЗЕНТИТЕ

ПРИЛОЖЕНИЕ НА 3D МОДЕЛИ В БИОИНФОРМАТИЧНИ ИЗСЛЕДВАНИЯ: АНАЛИЗ НА БИОЛОГИЧНИТЕ СВОЙСТВА

Темата на монографичния труд „Приложение на 3D модели в биоинформатични изследвания“ с автор Иван Тренчев отговаря на актуална изследователска проблематика, свързана с изследване на биологичните свойства на макромолекулите, представени във вид на големи масиви от данни чрез in silico изследвания и модели.

Биоинформатиката, възникваща и развиваща се като ново научно-практическо направление, съчетава в себе си математика, информатика и биология и помага да се намерят отговори на актуални теми и въпроси в съвременната биологична наука.

Терминът „биоинформатика“ е използван за първи път през 1968 г., а неговото изчерпателно енциклопедично определение е дадено за първи път през 1978 г. Биоинформатиката също се нарича „изчислителна биология“. Въпреки това, строго погледнато, изчислителната биология се занимава основно с моделиране на биологични системи.

Основните компоненти на биоинформатиката са:

- разработване на софтуерни инструменти и алгоритми и
- анализ и интерпретация на биологични данни чрез използване на различни софтуерни инструменти и конкретни алгоритми.

Един от основните изследователски проблеми в структурната биоинформатика е задачата за предсказване на пространствена структура на протеините. Те са „дълги“ последователности, образувани от 20 различни аминокиселинни остатъци, които във физиологични условия приемат уникална 3D структура.

Познаването на протеиновата структура позволява изследването на биологичните свойства на изследваните протеини.

В настоящия монографичен труд са разгледани два подхода за предсказване на 3D структурата на протеините:

- евристичен и
- комбинаторен.

Разгледани са конкретни примери като е направен сравнителен анализ между реалната структура на съществуващ протеин и компютърно генерирана 3D структура, чрез описаните в монографията алгоритми.

Предимство на настоящия труд е използването на съвременни квантово-химични методи в изчислителния процес. Изследвано е влиянието на магнитното поле върху някои свойства на ДНК молекулата.

Без да се твърди за изчерпателност на изследванията са направени изводи, че магнитното поле на ДНК зависи от нуклеотидната последователност на ДНК. Може да се направи и изводът, че промените на магнитното поле пораждат резонанси между подобни последователности в генома. Друг аспект на тези колебания е да възникнат резонанси между подобни геномни последователности. Бихме могли да формулираме и противоположната хипотеза, че модели, формирани от резониращи последователности с различна първична последователност, могат да породят един и същи резонанс.

В структурно отношение монографията съответства на поставената цел и произтичащите от нея задачи. Възприетият топологичен структурен модел включва увод, две глави, заключение, библиография и приложения.

Още във въведението доц. Тренчев подчертава, че повечето изследвани биоинформатични проблеми са с голяма изчислителна сложност. Анализът на сложността на проблемите и представените различни концепции за тяхното решаване прави тематиката още по-актуална. Настоящото изследване на автора внася добавена

стойност в изследването на различните 3D модели, които се прилагат в биоинформатичните изследвания.

В процеса на разработката на изследването са проучени трудовете на учени по биоинформатика, математика, химия и физика. Използваната литература е оформена в библиографски списък, структуриран в последователност: източници на латиница и източници на кирилица.

Монографията на доц. д-р Иван Тренчев е с комплексен, интердисциплинарен характер. Основните изводи и заключения, формулирани в разработения труд, произтичат от представеното изложение и се базират на използваната методология.

Ето защо считам, че това изследване на доц. Тренчев има практико-приложен характер и ще бъде от полза за широк кръг специалисти, свързани с биоинформатичните изследвания.

Проф. д.ик.н. Стоян Денчев
Председател на Общото събрание на УниБИТ
Директор на Института за информация
и сигурност

ПРИЛОЖЕНИЕ НА 3D МОДЕЛИ В БИОИНФОРМАТИЧНИ ИЗСЛЕДВАНИЯ. ИЗПОЛЗВАНЕ НА КВАНТОВО-ХИМИЧНИ ПОДХОДИ ЗА АНАЛИЗ НА СЛОЖНИ БИОМОЛЕКУЛИ

Основните проблеми, към които се насочва авторът, са свързани с въпросите на изчислителната биология, по-конкретно приложението на 3D моделите в биоинформатичните изследвания. Разглежданата проблематика представлява интерес за изследователи и практици, както от страните членки на Европейския съюз, така и от всички други страни по целия свят.

Подходите на биоинформатични изследвания са в търсенето на някои биологични свойства на биомолекули чрез комбинация от молекулярни, химични и информатични техники и подходи. Важните критерии за молекулярните подходи включват филогенетична разделителна способност и потенциал за мащабен скрининг. Прилагането на сравнителен анализ на последователността на генома е от съществено значение за по-доброто разбиране на генетичните и епигенетични компоненти на различни бактериални таксони.

С увеличения брой на напълно секвенирани микробни геноми, включително тези на добре известни бактерии и микроорганизми, става ясно, че геномният и метаболитен капацитет на тези микроорганизми е много по-висок от първоначално предвидения. Това се дължи на откриването на „мълчаливи“ или „криптични“ вторични метаболитни генни клъстери, които кодират производството на допълнителни, неидентифицирани съединения.

Настоящият монографичен труд представя информационни алгоритми и подходи за изследване на 3D структурата на

протеините, визуализация на макромолекули и използване на квантово-химични подходи за изследване и анализ на сложни биомолекули. От най-простите аминокиселини природата надгражда основополагащата структура по увлекателни начини. Разбирането на тази организация е от решаващо значение за разбирането на това как работи живата материя и как тези операции могат да бъдат представени и визуализирани с помощта на компютри.

Предсказването на пространствената структура на протеините е скъпо струващ процес. Първата стъпка е синтезирането на протеина, следващата е кристалография и рентгенов анализ. Ако предсказването на пространствената структура на протеините може да бъде заменено от компютърни симулации, то този процес ще подпомогне синтеза на нови лекарства и изучаване на свойствата на сложни макромолекулни комплекси.

Тези компютърни изчисления са полезни при сравнителни анализи на геноми за идентифициране на гени. Това дава възможност за детайлен анализ на еволюционния, структурен и функционален аспект на производството на природни продукти въз основа на сравнението на молекулни последователности, молекулярно моделиране и симулация. Друг аспект на тези изследвания е да се улесни идентифицирането на молекулните компоненти на производството на природни продукти, както и реконструкцията на пътищата за синтез на тези продукти.

Тезата, от която се ръководи доц. Тренчев, е, че приложението на описаните в монографичния труд модели влияе върху развитието на биоинформатиката като цяло. В настоящото изследване авторът определя като негов предмет разкриването и описанието на математическите 3D модели и алгоритми и приложението им в биологията. Обект на изследването са предсказване на 3D структурата на протеините, влиянието на магнитното поле върху ДНК, визуализацията на макромолекули и други.

Намирам за отличен подход фактът, че авторът въвежда още в

началото на своето научно изследване понятия като протеин, ДНК и генетичен код. Монографичният труд по своя характер има интердисциплинарен характер. Макар че математическите описания в научната разработка преобладават, тя ще бъде от полза и за нематематици.

Авторът е представил съдържанието по изключително професионален начин и настоящият труд ще бъде полезен за специалисти и неспециалисти в областта. Поздравявам автора, че е дръзнал да се гмурне в такава трудна материя. Това е една тежка и актуална тематика, за която се изискват много богат опит и натрупани знания и практика.

Проф. д-р Костадин Ангелов
Министър на здравеопазването

ПРИЛОЖЕНИЯ

ПРИЛОЖЕНИЕ № 1

ОПИСАНИЕ НА СОРС КОДА

Този софтуер има графичен потребителски интерфейс (GUI – Graphical User Interface) и е обектно ориентиран (като програмна парадигма). Състои се от 5 пакета:

ПАКЕТ PROTEIN

Този пакет е най-съществения пакет в този софтуер. В него се съдържат 4 класове: Protein.java, Protein1.java, Protein2.java и Protein3.java. С тези класове всъщност се описват моделите на протеините, които описахме в глава първа, техните атрибути и методи. Protein1.java, Protein2.java и Protein3.java са дъщерни класове на класа Protein.java. Това е така, защото тези 3 класове имат много подобни свойства и притежават абсолютно същите атрибути. Единствената разлика в тях са само две функции, които ще опишем по-долу. По този начин се спестява код и самият код става по-гъвкав.

Класът Protein.java притежава следните методи и атрибути:

Protein.java
protected String q protected int N protected int n protected int s protected char [ ] a
public Protein() public Protein(String St) public boolean IsCorrect() public boolean IsCorrectBinary() public boolean IsCorrectHP() public boolean ConvertHPtoBinary() public void Convert() public static double [ ] Coordinate3D(double arg, String protein) public static double [ ] Coordinate2D(double arg, String protein) public static int CountH(String protein) public static int CountP(String protein) public static void Ploting3D(double [ ] Solution, String protein) public static void Ploting2D(double [] Solution, String protein) public void CountParity() public int getN() public int getn() public String getq() public int gets() public char [ ] geta() public void setN(int N2) public void setQ(String Q) public void setn(int n2) public void setS(int S) public void setA(char[ ] A)

Фиг. 1. Класът Protein.java

Всички атрибути (String q, integer N, integer n, integer s, char [] a) са protected, което означава, че достъп до тях имат само методите от класа

Protein.java и неговите дъщерни класове. С това се запазва енкапсулацията на данните от външни класове (освен от тези, които наследява, само те имат достъп до тези данни). Променливата String q е всъшност стрингът, в който се съхранява стойността на протеина (неговата аминокиселинна секвенция в бинарен формат). Бинарният формат е една абстрактна представа на HP формата на протеините, където хидрофобните аминокиселини (H) се записват с 1, а полярните (P) с 0. Това означава, че q на практика е бинарен стринг. В низа char [], а също така се записва протеинът както и в q, само е един масив, състоящ се от знаци (0 и 1), с цел по-лесно извършване на операциите върху протеина. Променливите n, N и s съхраняват информации за протеина q, които са описани в точка 3.

Този клас притежава един конструктор по подразбиране (default constructor). Това означава, че всеки път, когато се декларира нов обект без допълнителни параметри в тялото на конструктора, ще се изпълни тялото на този конструктор. Кодът на този конструктор е:

```
public Protein() {
q = "";
N = 0;
n = 1;
s = q.length();
a = null;
}
```

От кода е ясно, че при извикване на този конструктор q получава стойност празен стринг, N получава стойност 0, n – 1, s – 0 а a – null.

Също така има и един конструктор с приемащ параметър стринг:

```
public Protein(String St){
q = St;
s = q.length();
n = (int)Math.sqrt(s) + 1;

N = n*n*n;
a = q.toCharArray();
}
```

От тук се вижда, че приемащият параметър St е всъшност протеинът. S

получава стойност, която е дължината на St; $n = \sqrt[3]{s} + 1$; $N = n^3$, което означава, че с този конструктор протеинът се инициализира за 3D структура на протеина. Доколкото след това се окаже, че са нужни операции за 2D структура, N в кода си променя стойността в n^2 , което ще видим по-късно; а получава стойностите от S, само са представени като масив.

Методът boolean isCorrect () проверява дали протеинът притежава коректните α стойности. Връща true, ако стойностите са коректно въведени, и връща false, ако не са:

```
public boolean isCorrect (){
//проверка за голямите букви
for (int i=0; i<a.length; i++){
if ((a[i] == 'B') || (a[i] == 'J') || (a[i] == 'O') ||
   (a[i] == 'P') || (a[i] == 'U') || (a[i] == 'X') ||
   (a[i] == 'Z') || (a[i] == 'b') || (a[i] == 'j') ||
   (a[i] == 'o') || (a[i] == 'p') || (a[i] == 'u') ||
   (a[i] == 'x') || (a[i] == 'z'))
   return false;
}

//проверка за ASCII интервала [0,64]
for (int i=0; i<a.length; i++)
if ((((int) a[i])>=0) && (((int)a[i]) <= 64))
   return false;

//проверка за ASCII интервалите [0,64], [91,96], [123,127]
for (int i=0; i<a.length; i++)
if ( ((((int) a[i])>=0) && (((int)a[i]) <= 64)) ||
    ((((int) a[i])>=91) && (((int)a[i]) <= 96)) ||
    ((((int) a[i])>=123) && (((int)a[i]) <= 127))
)
   return false;

return true;
  }
```

Методът boolean isCorrectBinary() прави същата проверка както и предишният метод, с такава разлика, че не проверява за α стойности, а за бинарни стойности, т.е. връща true, ако всички знаци в стринга са 1 или 0, и връща false в другия случай:

```
  public boolean isCorrectBinary(){
boolean b = true;
```

```
for (int i=0; i<a.length; i++)
if ((a[i]!='0') && (a[i]!='1'))
   b = false;

return b;
  }
```

Подобен е и методът boolean isCorrectHP(), който проверява за HP стойностите (case insensitive):

```
        public boolean isCorrectHP(){
boolean b = true;

for (int i=0; i<a.length; i++)
if ((a[i]!='h') && (a[i]!='p') && (a[i]!='H') && (a[i]!='P'))
   b = false;

return b;
        }
```

С метода void ConvertHPtoBinary() се извършва конвертиране на аминокиселината от HP в бинарен формат:

```
        public void ConvertHPtoBinary(){
for (int i=0; i<a.length; i++){
if ((a[i]=='h') || (a[i]=='H'))
 a[i] = '1';
else
   a[i] = '0';
}

q = String.valueOf(a);
System.out.println("q= " + q);
System.out.print("a[]= ");
for (int i=0; i<a.length; i++)
```

```
System.out.print(a[i]);
        }
```

С метода void Convert() се извършва конвертиране на аминокиселина от α формат в бинарен стринг:

```
public void Convert(){
 for (int i=0; i<a.length; i++){
 if (a[i] == 'A')
    a[i] = '0';
 else if (a[i] == 'R')
    a[i] = '1';
 else if (a[i] == 'N')
    a[i] = '1';
 else if (a[i] == 'D')
    a[i] = '1';
 else if (a[i] == 'C')
    a[i] = '1';
 else if (a[i] == 'E')
    a[i] = '1';
 else if (a[i] == 'Q')
    a[i] = '1';
 else if (a[i] == 'G')
    a[i] = '1';
 else if (a[i] == 'H')
    a[i] = '1';
 else if (a[i] == 'I')
    a[i] = '0';
 else if (a[i] == 'L')
    a[i] = '0';
 else if (a[i] == 'K')
    a[i] = '1';
 else if (a[i] == 'M')
    a[i] = '0';
 else if (a[i] == 'F')
    a[i] = '0';
 else if (a[i] == 'P')
    a[i] = '0';
 else if (a[i] == 'S')
    a[i] = '1';
 else if (a[i] == 'T')
    a[i] = '1';
 else if (a[i] == 'W')
    a[i] = '0';
 else if (a[i] == 'Y')
    a[i] = '1';
 else if (a[i] == 'V')
    a[i] = '0';
 }

 q = new String(a);
        }
```

Методът tatic double [] Coordinate3D (double arg, String protein) за входни параметри приема една реална променлива arg, която всъшност е стойност от 1D координатата на някаква аминокиселина на протеина Protein, превръща я в 3D координата и я връща като резултат, който всъшност е един масив, състоящ се от 3 елемента (x,y,z). Това, че този клас е static, означава, че може да се ползва дори и да не бъде създаден обект от тип Protein:

```
public static double [] Coordinate3D (double arg, String protein){
        int p = (int) arg;
        int [] coord = new int[3];
        int x, y, z;
```

```
int nn = (int)Math.sqrt(protein.length()) + 1;

p = p-1;
z = (p/(nn*nn))+1;

p = p-((z-1)*(nn*nn));
y = (p/nn) + 1;

p = p - (y-1)*nn;
x = p + 1;

coord[0] = x;
coord[1] = y;
coord[2] = z;

double [] doubleCoord = new double[3];
System.arraycopy(coord, 0, doubleCoord, 0, 3);

return doubleCoord;
  }
```

Методът static double [] Coordinate2D (double arg, String protein) е почти същият, като и предишният. Разликата е в това, че този метод превръща 1D в 2D координата, което означава, че връща масив, който съдържа две стойности:

```
public static double [] Coordinate2D (double arg, String protein){
int p = (int) arg;
int [] coord = new int[2];
int x, y, z;

int nn = (int)Math.sqrt(protein.length()) + 1;

p = p-1;
z = (p/(nn*nn))+1;

p = p-((z-1)*(nn*nn));
y = (p/nn) + 1;

p = p - (y-1)*nn;
x = p + 1;

coord[0] = x;
coord[1] = y;

double [] doubleCoord = new double[2];
System.arraycopy(coord, 0, doubleCoord, 0, 2);

return doubleCoord;
  }
```

Методът static int CountH(String protein) брои хидрофобните аминокиселини на аргумента protein:

```
public static int CountH(String protein){
char A[] = new char [protein.length()];
```

```
A = protein.toCharArray();

int H=0;
for (int i=0; i<A.length; i++)
if (A[i]=='1')
   H++;

return H;
        }
```

Методът static int CountP(String protein) брои хидрофилните аминокиселини на аргумента protein:

```
        public static int CountP(String protein){
char A[] = new char [protein.length()];
A = protein.toCharArray();

int P=0;
for (int i=0; i<A.length; i++)
if (A[i]=='0')
   P++;

return P;
        }
```

Методът static void Ploting3D(double [] Solution, String protein) е клас, който чертае тримерната структура на протеина protein. Структурата се чертае въз основа на резултатите, получени от изчисленията от ILOG CPLEX. Solution е масив от реални числа, в който са съхранени решенията на HP модела на протеина protein. Чертаенето се овъзможва чрез ползването на библиотеката org.math.plot, която е част от API-то JMathTools.

```
public static void Ploting3D(double [] Solution, String protein){
char A[] = new char [protein.length()];
A =
```

```
protein.toCharArray();

double [][] coord = new double [Solution.length][3];
for (int i=0; i<Solution.length; i++){
coord[i] = Coordinate3D(Solution [i],protein);
}

//присвояване на стойностите на h и p броя на H и P в протеина
int h, p;
h = CountH(protein);
p = CountP(protein);

//масив съдържащ H координатите и масив съдържащ P координатите
double [][] H = new double[h][3];
double [][] P = new double[p][3];

//ind - index
//попълване на масива H с неговите координати
int ind=0;
for (int i=0; i<Solution.length; i++){
if (A[i]=='1'){
   H[ind] = coord[i];
   ind++;
}
}

//попълване на масива P с неговите координати
ind = 0;
for (int i=0; i<Solution.length; i++){
if (A[i]=='0'){
   P[ind] = coord[i];
   ind++;
}
}

// PlotPanel конструкция
        Plot3DPanel plotpanel = new Plot3DPanel();

plotpanel.addLegend("SOUTH");

plotpanel.addScatterPlot("H", H);

plotpanel.addScatterPlot("P", P);

//чертаене на решетката
plotpanel.addLinePlot(protein, Color.ORANGE, coord);

/*добавяне на лабели за точките
  @param intLabel - № на решение като целочислена променлива
  @param strLabel - № на решение като стрингова променлива
*/
int intLabel;
String strLabel;
for (int i=0; i<Solution.length; i++){
intLabel = i+1;
strLabel = " " + Integer.toString(intLabel);

plotpanel.addLabel(strLabel, Color.DARK_GRAY, coord[i]);
}

        // слагане на PlotPanel в JFrame като JPanel
        JFrame frame = new JFrame("HP модел");

frame.setSize(600, 600);
```

```
frame.setContentPane(
plotpanel);

Dimension dim =
Toolkit.getDefaultTool
kit().getScreenSize();

// нова локация на
прозореца
int width =
frame.getSize().width;
int height =
frame.getSize().height;
int x = (dim.width-
width)/2;
int y = (dim.height-
height)/2;
// помещаване на
прозорецът
frame.setLocation(x,
y);
frame.setVisible(true);
}
```

Методът static void Ploting2D(double [] Solution, String protein) е същият като и предишният, само че се отнася на чертаенето на 2D решетката на протеина:

```
public static void
Ploting2D(double [] Solution, String
protein){
char A[] = new char [protein.length()];
A = protein.toCharArray();

double [][] coord = new double
[Solution.length][2];
for (int i=0; i<Solution.length; i++){
coord[i] =
Coordinate2D(Solution[i],protein);
}

//присвояване на стойностите на h
и p броя на H и P в протеина
int h, p;
h = CountH(protein);
p = CountP(protein);

//масив съдържащ H координатите
и масив съдържащ P координатите
double [][] H = new double[h][2];
double [][] P = new double[p][2];

//ind - index
//попълване на масива H с
неговите координати
int ind=0;
for (int i=0; i<Solution.length; i++){
if (A[i]=='1'){
   H[ind] = coord[i];
   ind++;
}
}

//попълване на масива P с неговите
координати
ind = 0;
for (int i=0; i<Solution.length; i++){
if (A[i]=='0'){
   P[ind] = coord[i];
   ind++;
}
}
```

```
// PlotPanel конструкция
        Plot2DPanel plotpanel = new Plot2DPanel();

plotpanel.addLegend("SOUTH");

        plotpanel.addScatterPlot("H", H);
plotpanel.addScatterPlot("P", P);

//чертаене на решетката
plotpanel.addLinePlot(protein, Color.ORANGE, coord);

/*добавяне на лабели за точките
 @param intLabel - № на решение като целочислена променлива
 @param strLabel - № на решение като стрингова променлива
*/
int intLabel;
String strLabel;
for (int i=0; i<Solution.length; i++){
intLabel = i+1;
strLabel = "  " + Integer.toString(intLabel);
plotpanel.addLabel(strLabel,
```

Методът int getN() връща N:

```
    public int getN(){
return N;
    }
```

Методът int getn() връща n:

```
    public int getn(){
return n;
Color.DARK_GRAY, coord[i]);
 }

        // поставяне на PlotPanel в JFrame като JPanel
        JFrame frame = new JFrame("HP модел");
        frame.setSize(600, 600);

frame.setContentPane(plotpanel);

Dimension dim = Toolkit.getDefaultToolkit().getScreenSize();

// определяне на нова локация на прозорецът
int width = frame.getSize().width;
int height = frame.getSize().height;
int x = (dim.width-width)/2;
int y = (dim.height-height)/2;

// помещаване на прозорецът
frame.setLocation(x, y);
frame.setVisible(true);
  }
```

```
}
```

Методът String getq() връща q:

```
public String getq(){
return q;
}
```

Методът int gets() връща s:

```
public int gets(){
return s;
}
```

Методът int geta() връща a:

```
public char [] geta(){
return a;
}
```

Методът void setN(int N2) редактира N, така че след изпълването на тази функция, N ще има стойността N2:

```
public void setN(int N2){
N = N2;
}
```

Методът void setQ(String Q) редактира q, така че след изпълването на тази функция q ще има стойността Q:

```
public void setQ(String Q){
q = Q;
}
```

Методът void setn(int n2) редактира n, така че след изпълването на тази функция n ще има стойността n2:

```
public void setn(int n2){
n = n2;
}
```

Методът void setS(int S) редактира s, така че след изпълването на тази функция s ще има стойността S:

```
public void setS(int S){
s = S;
}
```

Методът void setA(char[] A) редактира a, така че след изпълването на тази функция a ще има стойността A:

```
public void setA(char[] A){
a = A;
}
```

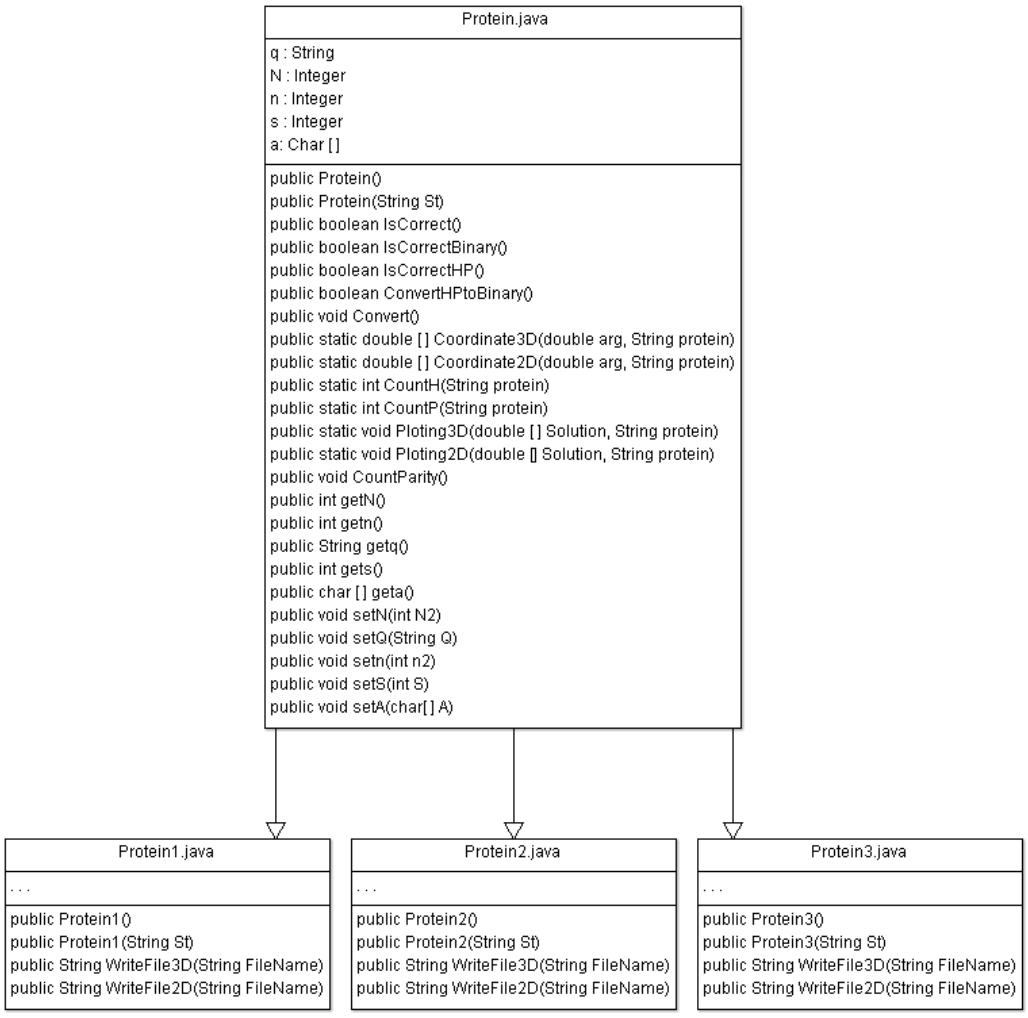
ПРИЛОЖЕНИЕ № 2

КЛАСОВЕТЕ PROTEIN1.JAVA, PROTEIN2.JAVA И PROTEIN3.JAVA

Класовете Protein1.java, Protein2.java и Protein3.java наследяват класа Protein.java. Това означава, че всички три класове (Protein1.java, Protein2.java и Protein3.java) включват в себе си същите атрибути и методи, както и Protein.java и добавят свои собствени методи, в случая това са методите public String WriteFile3D(String FileName) и public String WriteFile2D(String FileName).

Значи, както казахме по-горе, в атрибутите се съхраняват информации за протеина q, които са описани в точка 3. Методите на Protein.java се използват и от Protein1.java, Protein2.java и Protein3.java, тъкмо затова и Protein.java е надклас на тези класове. Тъй като атрибутите се ползват и от други функции на тези три класове (които са самостоятелни и Protein.java не знае, че съществуват) и плюс факта, че искаме да осигурим енкапсулация на данните – те са дефинирани като protected променливи на класа Protein.java, което означава, че само методите на Protein.java и методите на всички негови дъщерни класове имат право на достъп до тях (Фиг. 1.).

От по-горе казаното, ясно е, че класът Protein.java дефинира общите характеристики на протеините. Те са общи за всички три модела. Трите модела, обяснени в 3.2, 3.3 и 3.4, са дефинирани чрез класовете Protein1.java, Protein2.java и Protein3.java съответно. Те, освен че записват основните характеристики на един въведен протеин, те също така съдържат методи, които създават .lp файлове за CPLEX, в които са описани математическите модели на Модел №:1, Модел №:2 и Модел №:3. След това тези .lp файлове се четат чрез софтуера CPLEX и той намира оптимално решение за тях и връща някакъв резултат. Решенията от този резултат се вкарват в един масив (посредством този написан софтуер), след което това решение, получено от CPLEX, може да бъде визуализирано и този визуализиран резултат да бъде съхранен като графичен файл и също така да се запишат в текстов файл неговите координати.



Фиг. 2. Класовете Protein.java, Protein1.java, Protein2.java и Protein3.java

Този клас има два конструктора: конструктор по подразбиране и конструктор с приемащ параметър. Конструкторът по подразбиране има следната форма:

```
public Protein1(){
super();
}
```

Както може да се види, в тялото на този конструктор има само един оператор: super(). В Java това означава, че подкласът Protein1.java извиква дефиниран в надкласа конструктор. В този случай това означава, че super() всъшност извиква конструктора по подразбиране на Protein.java, а това е

Protein().

Сега ще разгледаме конструктора с приемащ параметър на Protein1.java – Protein(String St):

```
public Protein1(String St){
super(St);
}
```

В този случай super(St) извиква конструктора с приемащ параметър String на Protein.java, а това е Protein(String St).

По аналогичен начин са оформени и конструкторите на Protein2.java и Protein3.java.

В Protein1.java, Protein2.java и Protein3.java има още два допълнителни метода:

String WriteFile3D(String FileName) - този метод записва в .lp файл (със спазване на синтактичните правила за тези файлове) съответния HP модел на протеина, който е съхранен в променливите a и q за тримерното пространство. Файлът се съхранява в директорията lp files, която се намира в директорията на самия проект, и се съхранява под името FileName. Тази функция връща String.

String WriteFile2D(String FileName) - този метод записва в .lp файл (със спазване на синтактичните правила за тези файлове) съответния HP модел на протеина, който е съхранен в променливите a и q за двумерното пространство. Файлът се съхранява в директорията lp files, която се намира в директорията на самия проект, и се съхранява под името FileName. Тази функция връща String.

Тези три класове съдържат тези две функции. Те са също дефинирани със също име, същ тип и същ приемащ параметър. Обаче те извършват различни функции. Това зависи от кой клас се извикват. Например нека да имаме този код:

```
Protein1 P1 = new Protein1(“1110101011”);
Protein2 P2 = new Protein2(“1110101011”);
Protein3 P3 = new Protein3(“1110101011”);
```

P1.WriteFile3D(“1110101011_Model1_3D.lp”);
P2.WriteFile3D(“1110101011_Model2_3D.lp”);
P3.WriteFile3D(“1110101011_Model3_3D.lp”);

Създават се обектите P1, P2, P3, които са от тип Protein1, Protein2, Protein3 съответно. След това се извиква функцията WriteFile3D от Protein1, която записва първия HP модел, описан в 3.2 за теоретичния протеин 1110101011 в тримерното пространство в .lp файл с наименование 1110101011_Model1_3D.lp. След това, извиква се функцията WriteFile3D от Protein2, която записва втория HP модел, описан в 3.3 за теоретичния протеин 1110101011 в тримерното пространство в .lp файл с наименование 1110101011_Model2_3D.lp. И най-после, извиква се функцията WriteFile3D от Protein3, която записва третия HP модел, описан в 3.4 за теоретичния протеин 1110101011 в тримерното пространство в .lp файл с наименование 1110101011_Model2_3D.lp.

ПРИЛОЖЕНИЕ № 3

ПАКЕТ GUI

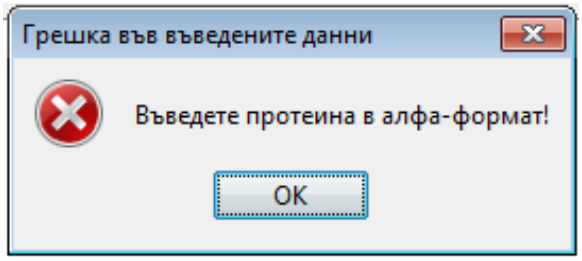
Пакет Images

Този пакет всъшност се състои само от картини, които се използват в пакета GUI. Тези картини се картините на бутоните, менютата, формите, които са сложени с цел поясняване на всеки визуален обект в графичния потребителски интерфейс.

Пакет Exceptions

В себе си този пакет съдържа java класове, които са изключения, които могат да се появят с ползването на софтуера. Този пакет съдържа следващите класове:

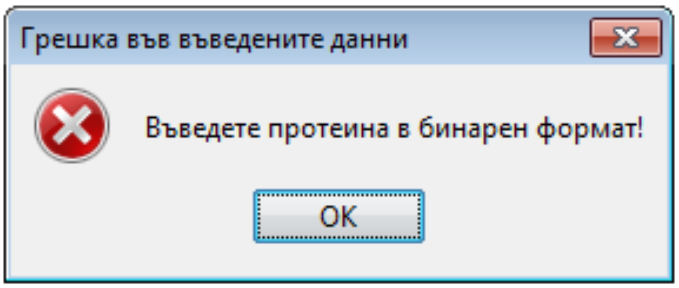
AlphaInputException.java – изключение, което се хвърля, когато има грешка във въведените данни (въведения протеин), т.е. когато протеинът не е въведен с неговите аминокиселини, които трябва да бъдат въведени в техния $^{\alpha}$ формат. Именно, когато трябва да се въведе протеинът в неговия $^{\alpha}$ формат, активира се това изключение, което предупреждава потребителя да въведе протеина правилно. Това предупреждаване идва във форма на диалог, който изглежда така:



Фиг. 3. AlphaInputException dialog

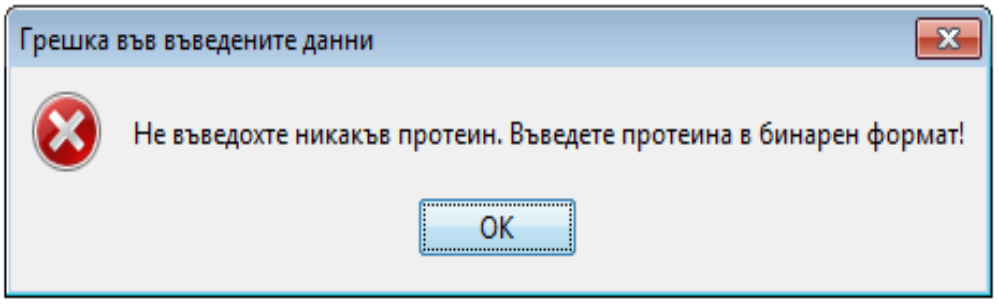
BinaryInputException.java – изключение, което се хвърля, когато има грешка във въведените данни (въведения протеин), т.е. когато протеинът не е въведен с неговите аминокиселини, които трябва да бъдат въведени в техния бинарен формат. Именно, когато трябва да се въведе протеинът в неговия бинарен формат, активира се това изключение, което предупреждава потребителя да въведе протеина правилно. Това предупреждаване идва във

форма на диалог, който изглежда така:



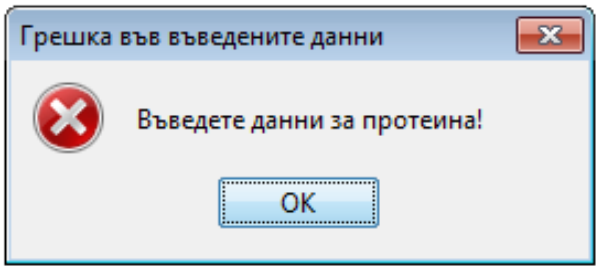
Фиг. 4. BinaryInputException dialog

EmptyBinaryFieldException.java – изключение, което се хвърля, когато има грешка във въведените данни (въведения протеин), т.е. когато за протеина не е въведена никаква стойност, а се очаква да е въведен неговият бинарен формат. Именно, когато трябва да се въведе протеинът в неговия бинарен формат, а въобще не е въведена никаква стойност, активира се това изключение, което предупреждава потребителя да въведе протеина правилно. Това предупреждаване идва във форма на диалог, който изглежда така:



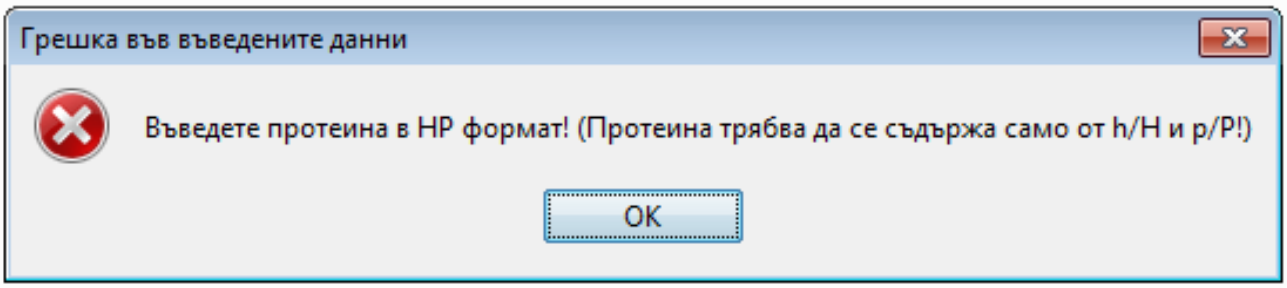
Фиг. 5. EmptyBinaryFieldException dialog

EmptyFieldException.java – изключение, което се хвърля, когато има грешка във въведените данни (въведения протеин), т.е. когато за протеина не е въведена никаква стойност. Именно, когато трябва да се въведе протеинът в неговия изискван формат, а въобще не е въведена никаква стойност, активира се това изключение, което предупреждава потребителя, че не е въведена никаква стойност за протеина. Това предупреждаване идва във форма на диалог, който изглежда така:



Фиг. 6. EmptyFieldException dialog

- HPInputException.java – изключение, което се хвърля, когато има грешка във въведените данни (въведения протеин), т.е. когато протеинът не е въведен с неговите аминокиселини, които трябва да бъдат въведени в техния HP формат. Именно, когато трябва да се въведе протеинът в неговия HP формат, активира се това изключение, което предупреждава потребителя да въведе протеина правилно. Това предупреждаване идва във форма на диалог, който изглежда така:



Фиг. 7. PInputException dialog

ИВАН ТРЕНЧЕВ

ПРИЛОЖЕНИЕ НА 3D МОДЕЛИ В БИОИНФОРМАТИЧНИ ИЗСЛЕДВАНИЯ

Монография

Българска
Първо издание

Научни рецензенти
проф. д.ик.н. Стоян Денчев
проф. д-р Костадин Ангелов

Графичен дизайн на корицата
Камелия Планска-Симеонова

Коректор
Евелина Здравкова-Величкова

Формат 60/90/8

Академично издателство „За буквите – О писменехъ“
ISBN 978-619-185-444-8

София, 2020 г.

Иван Тренчев е доцент в Университета по библиотекознание и информационни технологии от 2019 г. в катедра „Информационни системи", работи като доцент и в катедра „Електротехника, електроника и автоматика" на Югозападен университет „Неофит Рилски" (2000 г. -). Завършва математика в Югозападен университет „Неофит Рилски" (1992-1997 г.). Защитава дисертационен труд на тема: „Математически модели за изследване на оптималността на съвременния генетичен код" (2006 г.) в ЮЗУ „Неофит Рилски".

Иван Тренчев е един от създателите и дългогодишен ръководител на магистърска програма по Биоинформатика в ЮЗУ „Неофит Рилски". В периода 2011 - 2014 г. е заместник-главен редактор на списание "Scientific Research Journal of South-West University".

От 2013 до 2018 г. е зам.-ръководител на Център по съвременни биоинформатични изследвания в ЮЗУ „Неофит Рилски".

Член е на Съюза на математиците в България, Съюза на учените, Биоинформатичен свят и други български и чуждестранни организации.

Доц. Тренчев е участвал като организатор и координатор в повече от пет национални и международни проекта. Ръководител е на успешно приключилия научен проект „Изследвания на биоинформатиката: сгъване на протеини, скачване и прогнозиране на биологична активност", финансиран от Фонд „Научни изследвания" към Министерство на образованието и науката.

Чете лекции в УниБИТ и ЮЗУ „Неофит Рилски" по дисциплините: „Математически основи на киберсигурността", „Разработване на компютърни игри", „Операционни системи", „Обектно ориентирано програмиране", „Компютърен дизайн", „Програмиране на Java" и др.

Иван Тренчев е треньор на студентския отбор по програмиране при ЮЗУ „Неофит Рилски" в периода 2008 – 2018 г., през 2015 г. е председател на Организационния комитет на XXVII Републиканска студентска олимпиада по програмиране, провела се в същия университет.

ISBN 978-619-185-444-8