

**M A S A R Y K
U N I V E R S I T Y**

FACULTY OF INFORMATICS

**Modern visualization of partial
atomic charges in Mol***

Bachelor's Thesis

DOMINIK TICHÝ

Brno, Spring 2023

**M A S A R Y K
U N I V E R S I T Y**

FACULTY OF INFORMATICS

**Modern visualization of partial
atomic charges in Mol***

Bachelor's Thesis

DOMINIK TICHÝ

Advisor: RNDr. Tomáš Raček, Ph.D.

Department of Computer Systems and Communications

Brno, Spring 2023



Declaration

I declare that I have worked on this thesis independently, using only the primary and secondary sources listed in the bibliography.

Dominik Tichý

Advisor: RNDr. Tomáš Raček, Ph.D.

Acknowledgements

TODO

Abstract

TODO

Keywords

Molstar, Mol*, Atomic Charge Calculator 2, ACC2, AlphaCharges, α Charges, SB NCBR, molecular visualization, molecular graphics, partial atomic charges, structural biology

Contents

Introduction	1
1 Theory	2
1.1 Molecular structure	2
1.1.1 Atoms	2
1.1.2 Residues	3
1.1.3 Chains	3
1.2 Chemical file formats	3
1.2.1 SDF	3
1.2.2 MOL2	3
1.2.3 PDB	4
1.2.4 mmCIF	4
1.3 Alternative conformations	4
1.4 Partial atomic charges	4
1.5 Color interpolation	5
1.6 Provider pattern	5
2 Visualizing molecular data	6
2.1 Types of visualizations	6
2.1.1 Ball and stick	6
2.1.2 Surface	6
2.1.3 Cartoon	6
2.1.4 Coloring of molecular visualizations	7
2.2 Visualization software	7
2.2.1 Litemol	7
2.2.2 Mol*	8
3 Mol* partial charges extension	9
3.1 Requirements	9
3.2 Custom mmCIF categories	9
3.3 How Mol* works	11
3.4 Implementation	11
3.4.1 Property provider	11
3.4.2 Color theme provider	13
3.4.3 Labels	13
3.5 Controls	15

3.6	Mol* integration	15
4	Mol* viewer plugin	17
4.1	Requirements	17
5	Atomic Charge Calculator II	18
5.1	Extension of ChargeFW2	18
5.2	Multicharge support	20
5.2.1	Frontend changes	21
5.2.2	Backend changes	21
5.3	Mol* viewer integration	22
6	AlphaCharges	24
6.1	Extension to the Mol* viewer functionality	24
6.1.1	AlphaFold confidence score coloring	25
6.1.2	Focus on problematic atoms	25
6.2	Mol* viewer integration	26
6.3	Problematic atoms webpage	26
	Conclusion	28
	Bibliography	29

List of Tables

5.1	ChargeFW2 output file formats.	18
5.2	Updated ChargeFW2 output file formats.	20

List of Figures

3.1	Diagram of custom mmCIF categories.	11
3.2	Interface of the custom model property for storing partial atomic charges.	12
3.3	Partial charges color theme for different visualization types.	14
3.4	Extension controls.	15
5.1	Updated setup site for the ACC2 application.	21
5.2	Result site for the ACC2 application with the Mol* viewer.	23
6.1	Explanation for error of atom <i>GLN 33 CD</i>	25
6.2	Focus on the atom <i>GLN 33 CD</i> , one of two problematic atoms in the structure with UniProt code Q55GB6.	27

Introduction

1 Theory

Theoretical concepts are fundamental to the study of computational chemistry, providing a framework for analyzing molecular structures and properties. This chapter focuses on four key areas of theory, beginning with an overview of molecular structure in Section 1.1. Section 1.2 provides an in-depth examination of chemical file formats, including the advantages and disadvantages of different file formats for storing molecular data. Partial atomic charges are explored in section 1.3, including their importance in the analysis of molecular structures and the various methods used to compute them. Finally, section 1.4 delves into color interpolation, a critical technique for visualizing partial atomic charges in molecular structures.

By providing a comprehensive overview of these theoretical concepts, this chapter provides a strong foundation for the subsequent chapters, which will focus on the implementation and analysis of the Mol* extension for visualizing partial atomic charges in molecular structures.

1.1 Molecular structure

Molecular structure refers to the arrangement of atoms and chemical bonds in a molecule. The three main components of molecular structure are atoms, residues, and chains. In this section we will look at

1.1.1 Atoms

Atoms are the basic building blocks of matter. Atoms are composed of protons, neutrons, and electrons. The number of protons determines the element, while the number of neutrons determines the isotope. The number of electrons determines the charge of the atom. Atoms are the smallest unit of matter that can take part in a chemical reaction.

Bonds are the connections between atoms that hold molecules together. There are different types of bonds, e.g. covalent bonds, ionic bonds, and hydrogen bonds.

1.1.2 Residues

A residue refers to a specific building block that remains after a chemical modification or enzymatic reaction. In proteins, residues refer to amino acids, which are connected through peptide bonds to form polypeptide chains. Residues are crucial in biochemistry because they determine the structure and function of biological molecules. The sequence of residues in a protein or nucleic acid, for instance, determines its three-dimensional structure and ultimately its biological activity.

1.1.3 Chains

Polymer chains (chains) are sequences of residues that are linked together. In the context of biomolecules, chains can be either polypeptide chains in proteins or polynucleotide chains in nucleic acids. The sequence and structure of these chains are crucial for understanding the function and properties of the biomolecules.

1.2 Chemical file formats

TODO: introduction

1.2.1 SDF

Structure-data file (SDF) is a widely used chemical file format for representing molecular structures and their associated properties. It is a text-based format that describes the atoms, bonds, and atomic coordinates of a molecule.

1.2.2 MOL2

The Mol2 file format is another text-based format for storing molecular structures and their associated properties. It can store multiple conformations of a molecule and is commonly used in molecular modeling and cheminformatics applications. The Mol2 format provides more flexibility and additional features compared to the SDF format, such as support for multiple substructures and atom types.

1.2.3 PDB

The Protein Data Bank (PDB) file format is a widely used format for storing three-dimensional structures of proteins, nucleic acids, and other macromolecules. PDB files contain information about the atomic coordinates, secondary structure, and other important details required for understanding macromolecular structures. The PDB format has been widely adopted in structural biology, bioinformatics, and related fields.

1.2.4 mmCIF

The macromolecular Crystallographic Information File (mmCIF) format is an extension of the CIF format, specifically designed for macromolecular structures. It is a text-based format that provides a more comprehensive and flexible representation of macromolecular crystallography data compared to the PDB format. One of the most important features of the mmCIF format is its support for data dictionaries. This allows users to define new data items and integrate additional information. In contrast to other formats, the mmCIF format does not impose limits on column width and entry count, making it more flexible and accommodating for storing large amounts of data.

TODO: add example image + better explanation

1.3 Alternative conformations

Alternative conformations, also referred to as conformational isomerism, are different conformations of the same molecule. They can be used to represent different states of a molecule, such as different protonation states or different ligand binding modes. Alternative conformations are important for understanding the structure and function of biomolecules, as they provide insight into how they interact with other molecules.

1.4 Partial atomic charges

Partial atomic charges are a measure of the distribution of electronic charge within a molecule. These charges are important for under-

standing and predicting molecular interactions, including hydrogen bonding, electrostatic interactions, and solvation effects.

TODO: mention chargefw2 and alphacharges

1.5 Color interpolation

Color interpolation is the process of creating new colors by mixing two or more colors together. It is a common technique used in computer graphics and digital image processing to create smooth transitions between colors.

Color interpolation works by calculating the intermediate colors between two or more given colors. This is typically done by taking a weighted average of the red, green, and blue values of the colors being interpolated.

TODO: add math equation + image of red,white and white,blue interpolation

1.6 Provider pattern

2 Visualizing molecular data

This chapter will discuss various types of visualizations, the role of coloring in molecular representations, and some commonly used software tools for creating these visualizations.

2.1 Types of visualizations

There are several methods to represent molecular data, each with its own benefits and drawbacks. The methods most relevant to this work are the following three types: ball and stick, surface, and cartoon.

2.1.1 Ball and stick

The ball and stick model represents atoms as spheres and bonds as cylindrical connections between these spheres. This model provides a simple and intuitive visualization of a molecule's atomic structure. It highlights individual atoms and their bonds, including their bond types. However, it may not accurately represent the spatial relationships between atoms in larger molecules or macromolecular complexes.

2.1.2 Surface

Surface representations depict the three-dimensional shape of a molecule by displaying its solvent-accessible surface. This model provides a more accurate representation of the molecule's overall shape and size, making it especially useful for studying macromolecular interactions and the binding of small molecules. For example, surface visualization can be used to identify potential binding sites on a protein surface, which can then be targeted by drug molecules.

2.1.3 Cartoon

Cartoon representations simplify the molecular structure by focusing on the secondary structure elements of proteins and nucleic acids, such as alpha helices, beta sheets, and loops. Alpha helices are often

depicted as a spiral-like structures, whereas beta sheets as arrows. This type of visualization is particularly useful for visualizing large macromolecular complexes, as it highlights the overall organization and topology of the molecule without the clutter of atomic details. The simplification of the structure also makes it easier to understand the folding and dynamics of the molecule.

2.1.4 Coloring of molecular visualizations

Coloring is an essential aspect of molecular visualization, as it can provide additional information and help to emphasize specific features or properties of the molecule. Some common coloring schemes include:

- By element: Atoms are colored according to their chemical element (e.g., carbon in grey, oxygen in red, nitrogen in blue).
- By partial atomic charge: Atoms or residues are colored according to their charge or charge sum. Negative charges are depicted in red, positive charges in blue.

2.2 Visualization software

There are numerous software tools available for visualizing molecular data, with varying levels of complexity, customization, and features. Two widely used tools are LiteMol and Mol*.

2.2.1 Litemol

LiteMol is an open-source, web-native molecular visualization tool that supports various file formats and offers a user-friendly interface for creating visualizations. LiteMol provides essential visualization types, including ball and stick, surface, and cartoon representations, as well as options for customizing colors, lighting, and other display settings. The web-based nature of LiteMol makes it easily accessible and platform-independent.

The LiteMol suite is a freely available tool for visualizing large macromolecular structure datasets, which consists of three components: data delivery services, a compression format, and a lightweight 3D molecular viewer. It enables fast delivery and visualization of large datasets and is compatible with modern web browsers and mobile

devices, making it accessible to users with and without structural biology expertise. The tool addresses the challenges of delivering and visualizing large structural data sets, which are becoming increasingly available due to advances in electron microscopy and other techniques.

(1)

2.2.2 Mol*

(2)

Mol* ([/molstar/](http://molstar/)) is another web-native molecular visualization tool, developed as part of the wwPDB OneDep system for macromolecular structure deposition and validation. Mol* offers a wide range of visualization options, including advanced features such as electron density maps and validation reports. Mol* supports many file formats, including PDB, mmCIF, and PDBx/mmJSON. Like LiteMol, Mol* is platform-independent and can be accessed from any web browser.

Mol* emphasizes interactivity and offers various tools for manipulating and analyzing the molecular structure, such as distance and angle measurements, selection and display of specific residues, and custom coloring schemes. Additionally, Mol* provides integration with external databases and services, such as UniProt, PDBe, and RCSB PDB, enabling users to quickly access related information and resources.

3 Mol* partial charges extension

Visualizing partial atomic charges in molecules is an essential aspect of computational chemistry research, aiding in analyzing complex molecular structures. Mol* provides an extensive range of features for users to explore molecular structures. However, the tool lacks the functionality to color and label atoms and residues based on their partial atomic charges. This can be a considerable limitation for researchers. In response to this need, we have created an extension to Mol* that addresses this limitation.

It should be noted that the Mol* viewer predecessor, Litemol, supported this functionality. However, since Litemol is no longer supported, we saw the need to bring this functionality to Mol*.

This chapter describes the requirements for the extension, the custom mmCIF categories necessary for storing the partial atomic charges, and the implementation of the extension itself.

3.1 Requirements

TODO: describe what residue charges are

TODO: describe what each representation should visualize when colored using partial charges coloring

Firstly, the extension should enable the coloring of atoms and residues based on their partial atomic charges. Secondly, it should describe the charge values of the atoms and residues. Thirdly, the extension should allow the user to provide multiple charge sets for a single structure and select which one to display. Finally, the extension should be seamlessly integrated into the Mol* library, facilitating access to its features and functionality.

3.2 Custom mmCIF categories

To store partial atomic charges within a single file, we developed custom categories within the mmCIF format. The mmCIF format was chosen because it is widely used in the field of structural biology and

offers several advantages over other formats, as discussed in 1.2.4. The custom categories allow us to store information about the partial atomic charges separately from the other structural data, while still being able to access it within the same file. Storing all data in one file was important as it allowed for easier management and distribution of the data. If the charges were stored separately, we would have to provide the charge data to Mol* in a different way e.g. through custom import controls.

We used two separate categories for this purpose: one to store the partial charge values for each atom in the structure, and another to store metadata about the charge sets.

The category `_sb_ncbr_partial_atomic_charges` maps together the atoms of the structure and their charges. The category has three attributes:

- `type_id` - pointer to the `_sb_ncbr_partial_atomic_charges_meta.id` item
- `atom_id` - pointer to the `_atom_site.id` item described in 1.2.4
- `charge` - partial charge value for the atom

The category `_sb_ncbr_partial_atomic_charges_meta` is dedicated to storing metadata about the charge sets. The metadata category has the following attributes:

- `id` - unique identifier for the charge set
- `type` - type of the calculation method (e.g. 'empirical', 'quantum')
- `method` - computation method used to calculate the charge set (e.g. 'EQeq', 'EEM/Racek 2016 (ccd2016_npa)')

Figure 3.1 provides a detailed illustration of the custom mmCIF categories and their relationships.

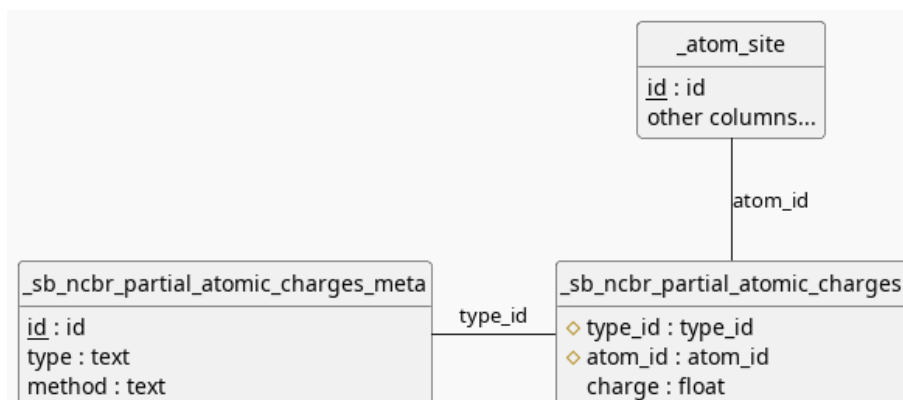


Figure 3.1: Diagram of custom mmCIF categories.

3.3 How Mol* works

3.4 Implementation

This section will detail the implementation of the extension. The extension consists of multiple providers. Each provider serves a distinct functionality, such as supplying the partial charge data and coloring the structural elements based on their charges. The providers will be described in detail in the following subsections.

The extension was created using TypeScript, a superset of JavaScript that adds static typing and other features to the language. The Mol* library is also written in TypeScript, so the extension was written in the same language to ensure compatibility.

3.4.1 Property provider

In order to retrieve the charges from the mmCIF file, it is necessary to parse the file. This is done by the Mol* library, which parses the mmCIF file and provides the parsed mmCIF file data in the form of a `MmcifFormat` object. The purpose of this provider is to process the charge data from this object and supply the charge data to the rest of the extension providers through a custom property. The interface of this property is depicted in Figure ??.

The atom charges are stored in the `typeIdToAtomIdToCharge` map. The map is indexed by the charge set (`typeId`) and the atom id. The atom id is a pointer to the `atom_site.id` item in the mmCIF file. The atom charges are retrieved from the mmCIF file by iterating over the `atom_site.id` category and retrieving the charge values for each atom. The charge values are then stored in the `typeIdToAtomIdToCharge` map.

The residue charges are calculated by summing the charges of the atoms that make up the residue. The residue charge is then stored in the `typeIdToResidueIdToCharge` map.

The maximum absolute charge values of the atoms and residues are calculated and stored in the `maxAbsoluteAtomCharges` and `maxAbsoluteResidueCharges` maps. These maps are used in the color theme provider to normalize the charges to the range of 0 to 1. Additionally, the maximum absolute charge values are used to calculate the color interpolations in the color theme provider. Additionally, the maximum absolute charge of both atoms and residues is calculated and stored in the `maxAbsoluteChargesAll` map.

Lastly, the method name used to calculate the charges of a given charge set is stored in the `typeIdToMethod` map. This map is used to display the method name in the UIs.

```
type TypeId = number;
type IdToCharge = Map<number, number>;
export interface SBNcbrPartialChargeData {
  typeIdToMethod: Map<TypeId, string>;
  typeIdToAtomIdToCharge: Map<TypeId, IdToCharge>;
  typeIdToResidueToCharge: Map<TypeId, IdToCharge>;
  maxAbsoluteAtomCharges: IdToCharge;
  maxAbsoluteResidueCharges: IdToCharge;
  maxAbsoluteAtomChargeAll: number;
  params: PartialChargesPropertyParams;
}
```

Figure 3.2: Interface of the custom model property for storing partial atomic charges.

3.4.2 Color theme provider

TODO: mention the color parameters

This provider serves as the central component of the extension, with its primary function being to assign colors to atoms and residues based on their charges. It achieves this by using the ColorTheme API provided by Mol*. The ColorTheme API is a mechanism for assigning colors to structural elements of a molecule. These structural elements can be atoms, residues, bonds, and so on. The API is based on the concept of a ColorTheme object, which is a collection of color assignments for structural elements. The ColorTheme object is then used by the Mol* library to color the structural elements of the molecule.

For the purposes of this extension, it was necessary to color two structural elements - atoms and residues. For both of these structural elements the charges were retrieved from the provider described in the previous section ??, which provided charges for atoms and residues.

To establish the color for a given charge, two color interpolations are employed: one for negative charges and another for positive charges. Atoms with positive charges receive a color from a white-to-blue color interpolation, while atoms with negative charges are assigned a color from a white-to-red color interpolation. These color interpolations are highlighted in Figure ??.

3.4.3 Labels

TODO: add a figure of the labels (screenshot from Mol*)

Having colored the structural elements, it was also necessary to create a label provider, which would assign labels that describe the charge of the structural element. In order to determine which element is highlighted, Mol* uses the object Loci. A Loci object is utilized for general selections and highlights. Consequently, it is essential to first extract the location from the Loci object in order to obtain the atom ID. The charge is acquired from the property provider, and the label is an HTML string that conveys the charge of the atom or residue. An example of the label can be seen in the right-hand corner in figure ??.

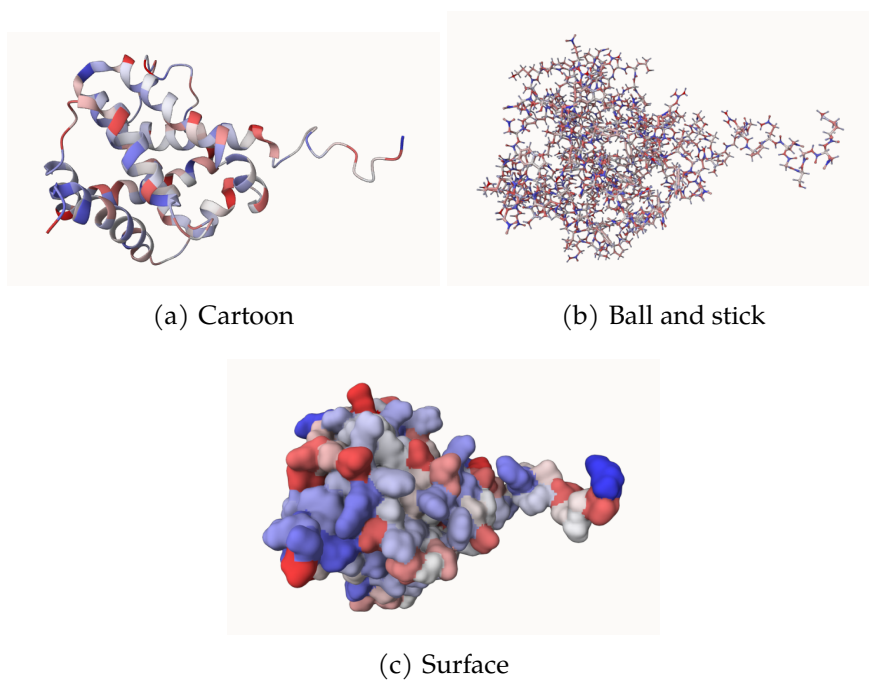


Figure 3.3: Partial charges color theme for different visualization types.

3.5 Controls

The controls are implemented automatically by the Mol* library based on the parameters of the providers. The user has access to controls of the charge set and the color theme. The charge set controls allow the user to select the charge set to display. The color theme controls allow the user to specify the following parameters:

- **Charge Range:** Sets the range of the color interpolation
- **Use Range:** Toggles whether the range of the color interpolation is automatically calculated or manually specified.
- **Charge Type:** Selects whether to display the partial atomic charges or the partial residue charges.

The controls are depicted in Figure 3.4.

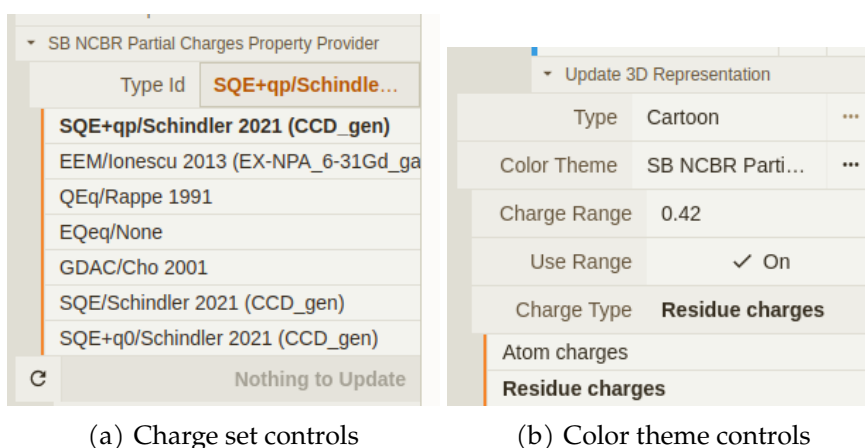


Figure 3.4: Extension controls.

3.6 Mol* integration

After creating the extension, it was integrated into the Mol* library. The extension is integrated in a way that allows the user to simply upload a mmCIF file containing the custom charge categories to the

3. MOL* PARTIAL CHARGES EXTENSION

Mol* viewer and the extension will automatically detect the custom categories and display the partial atomic charges. By integrating it into the Mol* library the extension was made freely available to anyone to use

4 Mol* viewer plugin

TODO: maybe make this a section in chapter about ACC2

In addition to creating the partial atomic charges extension for Mol*, it was necessary to create a custom Mol* viewer instance to facilitate custom functionality not present in the official Mol* viewer instance deployed at molstar.org/viewer. This chapter focuses on describing the implementation and functionality of the custom Mol* viewer instance for the web applications discussed in chapters 5 and 6.

4.1 Requirements

The viewer needs extended functionality

5 Atomic Charge Calculator II

Atomic Charge Calculator II (ACC2) is a web application for calculating partial atomic charges for structure files. The application is built using Flask for the backend and Javascript with Bootstrap for the frontend. The core of the ACC2 application is the ChargeFW2 program, which is used to calculate the partial atomic charges. For visualizing the calculation results, the application uses the Litemol software. (3) However, since the Litemol software is no longer supported, it was necessary to replace it with its modern counterpart, Mol*.

This chapter describes the changes made to the ACC2 application. We first discuss the modifications to the ChargeFW2 program, then explain the changes made to the backend and frontend to support calculations of multiple charge sets, and lastly we describe the integration of the Mol* viewer into the ACC2 application.

5.1 Extension of ChargeFW2

As discussed in ??, ChargeFW2 is a C++ program for computing partial atomic charges. The program supports the following input file formats: SDF, MOL2, PDB, and mmCIF, all of which are described in Section 1.2. The program parses the necessary atom and bond data for each molecule in the input file and stores it in a `Molecule` object. These objects are then stored in a `MoleculeSet` object over which the program then iterates and calculates the partial atomic charges for each molecule.

The program outputs the charge results in multiple file formats. The following table 5.1 lists the output file formats generated by ChargeFW2 for the corresponding input file formats.

Table 5.1: ChargeFW2 output file formats.

Input format	Output formats
SDF, MOL2	TXT, MOL2
PDB	TXT, PQR
mmCIF	TXT, PQR, mmCIF

As can be seen in Table 5.1, ChargeFW2 already outputs a mmCIF file. However, the format of this mmCIF file stores the partial atomic charges in a custom `_atom_site.fw2_charge` item, and therefore, it is not compatible with the Mol* viewer, which expects the partial atomic charges to be stored in the custom mmCIF categories described in Section 3.2.

The following subsections describe in detail how ChargeFW2 was modified to produce the custom mmCIF file format for all the input files, regardless of the input file format.

mmCIF

For mmCIF input files, the program simply appends the custom categories to the input file. To add the categories, the program uses the GEMMI library (4) to parse the input mmCIF file into a `Block` object, which provides access to all mmCIF categories in the file. Before appending the charge categories, the program first removes alternative conformations from the `Block` object. This is necessary because the Mol* viewer does not support alternative conformations. The program then appends the custom charge categories to the `Block` object and writes the `Block` object to a new mmCIF file.

PDB

The PDB input files are first converted to mmCIF format using the GEMMI library. To achieve this, the PDB file is first parsed into a `Structure` object, which is then converted into a `Block` object. After converting the PDB file to mmCIF, it was necessary to remove the `_chem_comp` category from the `Block` object. This category is generated by GEMMI by default, however, it is not necessary for the Mol* viewer and its presence causes the viewer to visualize the structure incorrectly. The program then proceeds in the same way as for the mmCIF input files.

SDF and MOL2

Both SDF and MOL2 input files are converted to mmCIF format using the atom and bond data parsed from the input file. The program

first creates an empty Block object, which it then populates with the `_atom_site` and `_chem_comp_bond` categories. The program then proceeds in the same way as for the mmCIF input files.

Summary

The ChargeFW2 was modified to produce a mmCIF file for each calculation, which is compatible with the Mol* viewer. The Table 5.2 lists the updated output file formats generated by ChargeFW2 for the corresponding input file formats.

Table 5.2: Updated ChargeFW2 output file formats.

Input format	Output formats
SDF, MOL2	TXT, mmCIF, MOL2
PDB, mmCIF	TXT, mmCIF, PQR

5.2 Multicharge support

The ACC2 application allows the user to select the method and parameters that will be used to calculate the partial atomic charges for the user's input file. Once the user confirms their selection, the application sends it to the backend, which runs the ChargeFW2 program with the chosen method and parameters. The application then displays the results of the calculation to the user. However, the application only supports one calculation per request and does not support the visualization of multiple charge sets. This limitation was previously caused by the Litemol viewer, which only supported one charge set per structure. However, since the Mol* viewer supports multiple charge sets, the ACC2 application was modified to support multiple charge sets as well.

The following subsections describe the necessary changes made to the ACC2 application to facilitate multiple calculations per one request and the visualization of multiple charge sets.

5.2.1 Frontend changes

As previously explained in section 5.2, the ACC2 application used to limit the user to selecting only one combination of method and parameters. This limitation was resolved by introducing new controls to the calculation setup.

Figure 5.1 shows the updated setup site for the ACC2 application. The site now contains a list of calculations, where each calculation represents a combination of method and parameters. The list is initially empty and the user can add a new calculation to the list by selecting the desired method and parameters from the drop-down menus and clicking the *Add to calculation* button. The user can also remove any calculation from the list by clicking the cross button next to the calculation name. The list of calculations is then sent to the backend where it is used to generate the desired charge sets for the user's input file.

The screenshot displays the ACC2 application's setup interface. It features two main selection areas: 'Method' and 'Parameters'. The 'Method' dropdown is set to 'SQE+qp', and the 'Parameters' dropdown is set to 'Schindler 2021 (CCD_gen)'. Below these is a green 'Add to calculation' button. A section titled 'Will compute the following charges:' contains a table with four entries, each with a cross icon for removal:

SQE+qp (Schindler 2021 (CCD_gen))	✕	EEM (Racek 2016 (ccd2016_npa))	✕
QEq (Rappe 1991)	✕	EQeq (No parameters)	✕

At the bottom, there is a green 'Compute' button and a blue 'Back to main page' button. The 'Full name' and 'Publication' sections on the right provide details for the selected method and parameters, including a citation for Schindler et al. (2021).

Figure 5.1: Updated setup site for the ACC2 application.

5.2.2 Backend changes

The Flask backend of the ACC2 application was modified to support multiple calculations per request. The backend now receives a list

of calculations from the frontend, which it uses to generate the desired charge sets for the user's input file. It does this by running the ChargeFW2 program for each calculation in the list. For each calculation the ChargeFW2 generates the output file formats described in Table 5.2. The backend then parses the charge values from the output TXT file and stores them in a dictionary, creating a mapping between the charge set name and the charge values.

After all the calculations have been completed, the dictionary is used to generate the mmCIF file. Firstly, the backend parses one of the output mmCIF files generated by ChargeFW2 into a Block object using the GEMMI library. The backend then iterates over dictionary and appends the charge values to the Block object under the custom charge categories. Lastly, the backend writes the Block object to a new mmCIF file, which is then sent to the frontend, where it is loaded into the Mol* viewer.

5.3 Mol* viewer integration

Figure 5.2 shows the result site for the ACC2 application with the Mol* viewer. The Mol* viewer is integrated into the result site by initializing the custom viewer plugin, loading the mmCIF file generated by the backend, and configuring the viewer to visualize the charge values.

Additionally, the site allows the user to change the coloring and visualization of the structure. The structure can be visualized as a ball-and-stick model, as a surface model, or if applicable as a cartoon model. The user has also the option to choose the coloring. The structure can be colored by partial atomic charges or by the element symbol of each atom. The user can also set the maximum charge value that will be used for the coloring or reset the maximum charge value to the default value.

The user can download all the output files generated by the backend by clicking the *Download charges* button. This button downloads a ZIP archive containing all the output files, which are organized into folders for each file format.

Atomic Charge Calculator II

Computation results

Method: SQE+qp
Parameters: Schindler 2021 (CCD_gen)

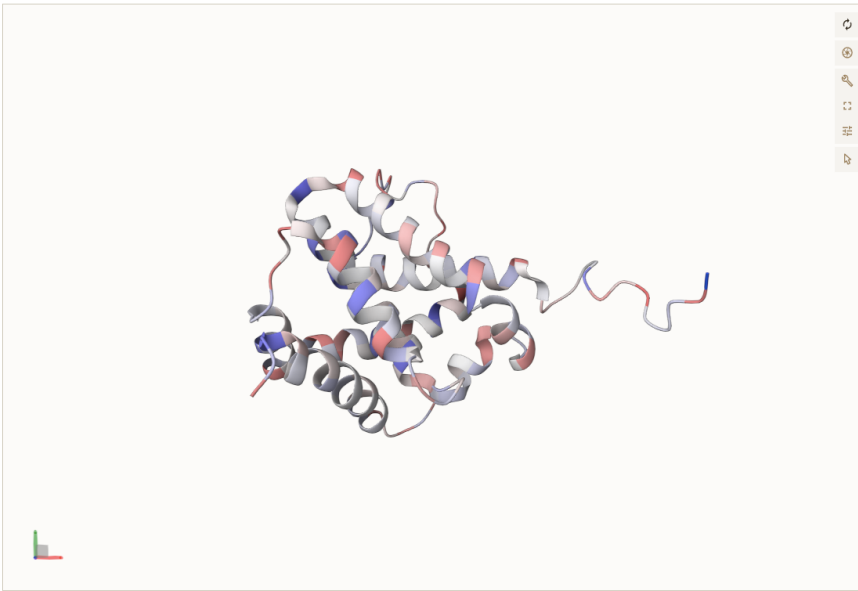
Structure
1F16

Charge set
SQE+qp/Schindler 2021 (CCD_gen)

View
☒ Cartoon ☐ Balls and sticks ☐ Surface

Coloring
☐ Structure
☒ Charges (relative) ☐ Charges (absolute)

Max value: 1.0825 [Reset](#)



[Download charges](#) [Back to setup](#) [Back to main page](#)

© 2021 Tomáš Raček | [Terms of Use](#) & [GDPR](#)

Figure 5.2: Result site for the ACC2 application with the Mol* viewer.

6 AlphaCharges

TODO: might want to move the description to literature since you are using a lot of citations, however, it would be weird to include it since this application does not yet exist and therefore the citations are not yet published

AlphaCharges (α Charges) is a web application for calculating partial atomic charges for structures from AlphaFoldDB, a database of protein structures predicted by the AlphaFold2 algorithm. (5) The AlphaCharges application protonates the protein structures (adds hydrogens) and calculates the partial atomic charges for the structure with the SQE+q0 method. (6, 7)

The application shares the architecture with the ACC2 application. The frontend is written in Javascript with the Bootstrap framework, the backend is written in Python and uses the Flask framework. This shared architecture meant that the AlphaCharge application could reuse most of the code from the ACC2 application, including the Mol* viewer.

This chapter describes the changes made to the Mol* viewer to support the AlphaCharges application. The chapter is divided into two sections. The first section describes the changes made to the Mol* viewer to support the AlphaCharges application. The second section describes the integration of the Mol* viewer into the AlphaCharges application.

6.1 Extension to the Mol* viewer functionality

TODO: focus only on the changes made to the viewer, not the website

Most of the features required by the AlphaCharges application were already implemented in the Mol* viewer for ACC2. However, there were some additional features that needed to be implemented to support the AlphaCharges application. These features are described in the following subsections.

6.1.1 AlphaFold confidence score coloring

6.1.2 Focus on problematic atoms

The AlphaCharges application cannot calculate the partial atomic charges for all protein structures. Some structures can be incorrectly predicted by the AlphaFold2 algorithm or the application can fail to protonate the structure. (5)

In these cases, the application redirects the user to a site with an explanation of the error. On this site the user is presented with a Mol* viewer with the problematic structure and an error message with the problematic atoms. The user has the option to hover over the problematic atoms to display an explanation for the probable cause of the error. Figure 6.1 shows the explanation provided for the atom *GLN 33 CD*.

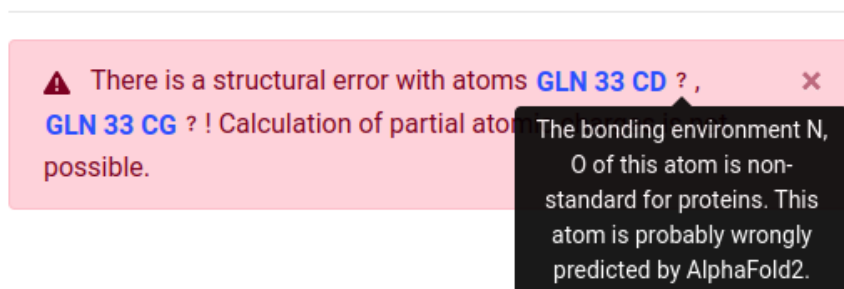


Figure 6.1: Explanation for error of atom *GLN 33 CD*.

The Mol* library provides a function to focus on a specific structure selection. The selection can be created programmatically using the Mol* query language. The query language is built using the Molscrip language (8) and allows the user to select atoms based on their properties, such as their element symbol or their residue name. To focus on the problematic atom, the query language was used to first select the atom with the residue name *GLN*, the residue number 33, and the atom name *CD*. This combination of properties uniquely identifies the problematic atom in the structure. Then the selected atom was focused using the Mol* focus function. Figure 6.2 shows the Mol* viewer with the atom *GLN 33 CD* focused

6.2 Mol* viewer integration

6.3 Problematic atoms webpage

The backend sends a JSON object to the frontend containing problematic atoms and the explanations. Figure ?? shows the structure of the JSON schema. This JSON object is then used by the frontend to display the problematic atoms and their explanations.

The problematic atoms are added to the error message as a link. When the user clicks on the link, a event listener will be trigger, which will make the Mol* viewer to visually focus (i.e. zoom in and highlight) the problematic atom. This allows the user to visually identify the problematic atom in the structure. Figure 6.2 shows the Mol* viewer with the atom *GLN 33 CD* focused.

The user can also hover over a icon adjacent to the problematic atom to display the explanation for the probable cause of the error in a pop-up text. Figure 6.1 shows the explanation provided for the atom *GLN 33 CD*.

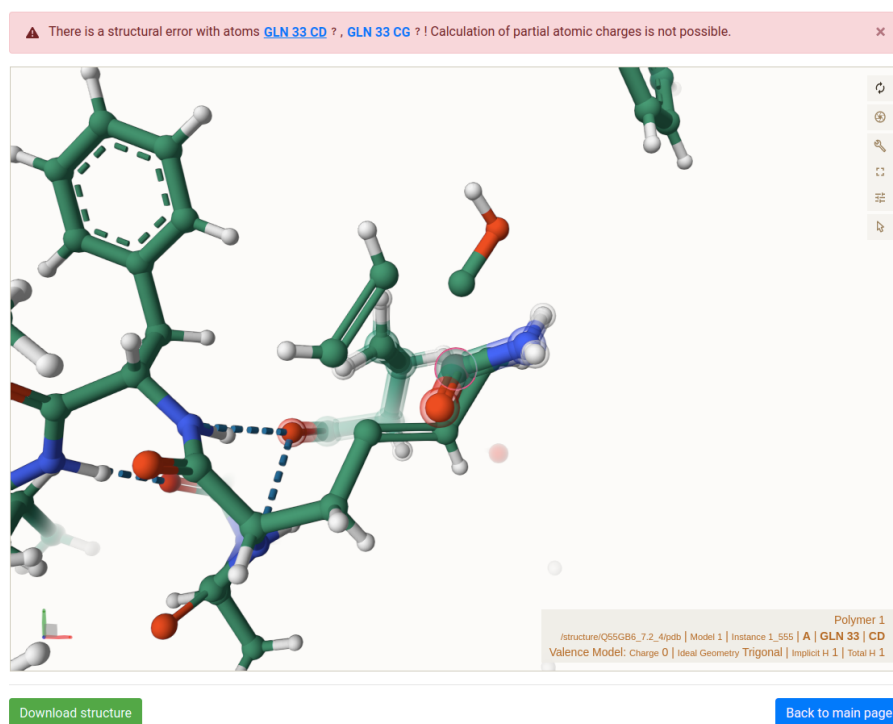


Figure 6.2: Focus on the atom *GLN 33 CD*, one of two problematic atoms in the structure with UniProt code Q55GB6.

Conclusion

TODO: try to somehow mention the publications in the conclusion

Things were created and some things even work.

Bibliography

1. SEHNAL, David; DESHPANDE, Mandar; VAŘEKOVÁ, Radka Svobodová; MIR, Saqib; BERKA, Karel; MIDLIK, Adam; PRAVDA, Lukáš; VELANKAR, Sameer; KOČA, Jaroslav. LiteMol suite: interactive web-based visualization of large-scale macromolecular structure data. *Nature Methods*. 2017, vol. 14, no. 12, pp. 1121–1122. ISSN 1548-7105. Available from DOI: 10.1038/nmeth.4499.
2. SEHNAL, David; BITTRICH, Sebastian; DESHPANDE, Mandar; SVOBODOVÁ, Radka; BERKA, Karel; BAZGIER, Václav; VELANKAR, Sameer; BURLEY, Stephen K; KOČA, Jaroslav; ROSE, Alexander S. Mol* Viewer: modern web app for 3D visualization and analysis of large biomolecular structures. *Nucleic Acids Research*. 2021, vol. 49, no. W1, W431–W437. ISSN 0305-1048. Available from DOI: 10.1093/nar/gkab314.
3. RAČEK, Tomáš; SCHINDLER, Ondřej; TOUŠEK, Dominik; HORSKÝ, Vladimír; BERKA, Karel; KOČA, Jaroslav; SVOBODOVÁ, Radka. Atomic Charge Calculator II: web-based tool for the calculation of partial atomic charges. *Nucleic Acids Research*. 2020, vol. 48, no. W1, W591–W596. ISSN 0305-1048. Available from DOI: 10.1093/nar/gkaa367.
4. WOJDYR, Marcin. GEMMI: A library for structural biology. *Journal of Open Source Software*. 2022, vol. 7, no. 73, p. 4200. Available from DOI: 10.21105/joss.04200.
5. JUMPER, John; EVANS, Richard; PRITZEL, Alexander; GREEN, Tim; FIGURNOV, Michael; RONNEBERGER, Olaf; TUNYASUVUNAKOOL, Kathryn; BATES, Russ; ŽÍDEK, Augustin; POTAPENKO, Anna; BRIDGLAND, Alex; MEYER, Clemens; KOHL, Simon A. A.; BALLARD, Andrew J.; COWIE, Andrew; ROMERA-PAREDES, Bernardino; NIKOLOV, Stanislav; JAIN, Rishub; ADLER, Jonas; BACK, Trevor; PETERSEN, Stig; REIMAN, David; CLANCY, Ellen; ZIELINSKI, Michal; STEINEGGER, Martin; PACHOLSKA, Michalina; BERGHAMMER, Tamas; BODENSTEIN, Sebastian; SILVER, David; VINYALS, Oriol; SENIOR, Andrew W.; KAVUKCUOGLU, Koray; KOHLI, Pushmeet; HASSABIS, Demis. Highly accurate protein structure prediction with AlphaFold. *Nature*. 2021, vol. 596, no.

BIBLIOGRAPHY

- 7873, pp. 583–589. ISSN 1476-4687. Available from DOI: 10.1038/s41586-021-03819-2.
6. SCHINDLER, Ondřej; BERKA, Karel; CANTARA, Alessio; KŘENEK, Aleš; TICHÝ, Dominik; RAČEK, Tomáš; SVOBODOVÁ, Radka. α Charges: partial atomic charges for AlphaFold structures in high quality. *Nucleic Acids Research*. 2023. ISSN 0305-1048. Available from DOI: 10.1093/nar/gkad349. gkad349.
 7. SCHINDLER, Ondřej; RAČEK, Tomáš; MARŠAVELSKI, Aleksandra; KOČA, Jaroslav; BERKA, Karel; SVOBODOVÁ, Radka. Optimized SQE atomic charges for peptides accessible via a web application. *Journal of Cheminformatics*. 2021, vol. 13, no. 1, p. 45. ISSN 1758-2946. Available from DOI: 10.1186/s13321-021-00528-w.
 8. KRAULIS, P. J. MOLSCRIPT: a program to produce both detailed and schematic plots of protein structures. *Journal of Applied Crystallography*. 1991, vol. 24, no. 5, pp. 946–950. Available from DOI: 10.1107/S0021889891004399.