Module Interface Specification for SubLiMat

Uriel Garcilazo Cruz

April 3, 2025

1 Revision History

Date	Version	Notes
March 10, 2025	1.0	Document's first version
April 4th, 2025	1.1	Document's updates after changes on modular design

2 Symbols, Abbreviations and Acronyms

See SRS Documentation at SRS Documentation

Contents

3 Introduction

The following document details the Module Interface Specifications for the SubLiMatsoftware. The software is designed to evaluate the effect of given substitution matrices in the quality of the alignment of a set of DNA sequences.

Complementary documents include the System Requirement Specifications and Module Guide. The full documentation and implementation can be found at https://github.com/UGarCil/UGarcil_capstone.

Many components from the present documentation follow the template for a MIS for scientific computing software used in Patel (2023), Bicket (2017). These documentations were adapted from the MIS template.

4 Notation

This section is taken from the MIS template.

> The structure of the MIS for modules comes from Hoffman and Strooper (1995), with the addition that template modules have been adapted from Ghezzi et al. (2003). The mathematical notation comes from Chapter 3 of Hoffman and Strooper (1995). For instance, the symbol := is used for a multiple assignment statement and conditional rules follow the form $(c_1 \Rightarrow r_1 | c_2 \Rightarrow r_2 | ... | c_n \Rightarrow r_n)$....

The following table summarizes the primitive data types used by the SubLiMatsoftware.

Data Type	Notation	Description
character	char	a single symbol or digit
integer	$\mathbb{R}$	a real (i.e. non complex) number defined within (-∞, ∞), which common values between -3.0 to 3.0
natural number	$\mathbb{N}$	a number without a fractional component in [1, ∞)

The specification of SubLiMat uses some derived data types: sequences, strings, and tuples. Sequences are lists filled with elements of the same data type. Strings are sequences of characters. Tuples contain a list of values, potentially of different types. In addition, SubLiMat uses functions, which are defined by the data types of their inputs and outputs. Local functions are described by giving their type signature followed by their specification.

5 Module Decomposition

The following table is taken directly from the Module Guide document for this project.

Level 1	**Level 2**
Hardware-Hiding	
Behaviour-Hiding	Alignment (Needleman-Wunsch) Interface Module Control Manager Module
Software Decision	Substitution Matrix Module File Manager Module Sequence Data Structure Module

Table 1: Module Hierarchy

It is important to highlight the categorization for the Alignment Interface Module in the table above. While the use of the Needleman-Wunsch algorithm sits at a high level behavior that defines the main architecture of the software, it is also a Software Decision, as future releases of the software may opt for other alignment algorithms. Its placement in the documentation as Behaviour-Hiding follows a functional justification by making the Needleman-Wunsch algorithm a high-behavior decision component.

6 MIS of Alignment (Needleman-Wunsch) Module

6.1 Module

NeedlemanWunsch

6.2 Uses

- **Sequence Data Structure Module (mSDS)**: For validated biological sequence input
- **Substitution Matrix Module (mSM)**: For accessing substitution matrices and gap penalties

6.3 Syntax

6.3.1 Exported Constants

None

6.3.2 Exported Access Programs

Name	In	Out	Exceptions
`__init__`	seq_a: str, seq_b: str, gap_penalty: float = -2	NeedlemanWunsch instance	-
`__call__`	submat: dict	float (alignment score)	AlignmentError
`align`	submat: dict	float (alignment score)	AlignmentError
`get_aligned_sequences`	-	tuple (str, str)	-

6.4 Semantics

6.4.1 State Variables

- `seq_a`: First input sequence
- `seq_b`: Second input sequence
- `gap_penalty`: Gap insertion penalty score
- `submat`: Active substitution matrix
- `seq_a_aligned`: Aligned version of seq_a
- `seq_b_aligned`: Aligned version of seq_b

6.4.2 Environment Variables

None

6.4.3 Assumptions

- Input sequences contain only characters {A, T, G, C, _}
- Sequences have been pre-validated by SequenceData module
- Substitution matrix is a valid 4x4 matrix for nucleotides
- Gap penalty is a negative float value

6.4.4 Access Routine Semantics

NeedlemanWunsch.__init__(seq_a, seq_b, gap_penalty=-2):

- Initializes alignment processor with sequences and gap penalty
- Transition: Sets instance variables

NeedlemanWunsch.__call__(submat):

- Output: Alignment score (float)
- Exception: AlignmentError if sequences cannot be aligned
- Delegates to align() method

NeedlemanWunsch.align(submat):

- Transition:
    1. Sets active substitution matrix
    2. Initializes dynamic programming matrix
    3. Computes alignment scores
    4. Performs traceback to determine optimal alignment
- Output: Final alignment score (float)
- Exception: AlignmentError if alignment fails

NeedlemanWunsch.get_aligned_sequences():

- Output: Tuple of aligned sequences (str, str)
- Requires: align() must have been called first

6.4.5 Local Functions

- **evaluate_substitution(s1: str, s2: str) → float**:
  - Looks up substitution score in matrix
  - Handles both matrix formats (direct 4x4 or parameterized)
  - Returns score for nucleotide pair
- **compute_alignment_matrix(matrix) → List[List[float]]**:
  - Fills dynamic programming matrix
  - Applies recurrence relation:

$$F(i,j) = \max \begin{cases} F(i-1, j-1) + s(x_i, y_j) \\ F(i-1, j) + d \\ F(i, j-1) + d \end{cases} \tag{1}$$

- **backtrack(matrix) → float**:
  - Traces optimal path through DP matrix
  - Sets seq_a_aligned and seq_b_aligned
  - Returns final alignment score

6.5 Considerations

- Time Complexity: O(mn) for sequences of length m and n
- Space Complexity: O(mn) for full matrix storage
- Handles both traditional and parameterized substitution matrices
- Inserts gap penalties through constant gap penalty

7 MIS of Control Manager Module

7.1 Module

main

7.2 Uses

- **File Manager Module**: For file I/O operations
- **Alignment Module**: For sequence alignment
- **Sequence Data Structure Module**: For sequence handling
- **Substitution Matrix Module**: For matrix operations

7.3 Syntax

7.3.1 Exported Constants

None

7.3.2 Exported Access Programs

Name	In	Out	Exceptions
main	seq_data: SequenceData, matrix_bench: SubMat	list[dict]	AlignmentError

7.4 Semantics

7.4.1 State Variables

None

7.4.2 Environment Variables

None

7.4.3 Assumptions

- Input files exist and are properly formatted
- Required modules are properly initialized
- Output directory is writable

7.4.4 Access Routine Semantics

main(seq_data, matrix_bench):

- Transition:
    1. Initializes NeedlemanWunsch alignment manager
    2. Iterates through each substitution matrix
    3. Computes alignment score for each matrix
    4. Collects results

7.5 Workflow

The control flow follows this sequence:

1. Sequence data is loaded via SequenceData module
2. Substitution matrices are loaded via SubMat module
3. For each substitution matrix:
    (a) Alignment is performed via NeedlemanWunsch
    (b) Results are collected
4. Results are exported via FileManager

7.6 Considerations

- Acts as the system's main coordinator
- Stateless - maintains no internal state between runs
- Delegates all specialized operations to other modules
- Handles the sequencing of operations but not their implementation

8 MIS of Substitution Matrix Module

8.1 Module

substitution_matrix

8.2 Uses

- **File Manager Module**: For loading matrix data from files

8.3 Syntax

8.3.1 Exported Constants

- **penalizingCostOf_**: A 4x4 matrix representing the baseline penalizing costs for nucleotide comparisons.

8.3.2 Exported Access Programs

Name	In	Out	Exceptions
get_substitution_matrix	name: str	submat: dict	InvalidMatrixError
validate_benchmark	benchmark: list[dict]	-	InvalidMatrixError
get_matrix_names	-	list[str]	-

8.4 Semantics

8.4.1 State Variables

- `data`: List of matrix dictionaries containing:
  - `NAME`: String identifier
  - `PENALIZING_COSTS`: 4x4 numerical matrix

8.4.2 Environment Variables

None

8.4.3 Assumptions

- Input files follow the format:

```
>MatrixName
A,T,G,C
T,A,G,C
G,T,A,C
C,G,T,A
```

- All matrices are 4x4 and square
- Nucleotide order is fixed as [A, T, G, C] for index mapping

8.4.4 Access Routine Semantics

get_substitution_matrix(name: str) → dict:

- Retrieves matrix dictionary by name from `data`
- **Output**: Dictionary with keys:
  - `NAME`: Matrix identifier
  - `PENALIZING_COSTS`: 4x4 scoring matrix

validate_benchmark(benchmark: list[dict]):

- **Exception**: Raises `InvalidMatrixError` if:
  - Any matrix is not 4x4
  - Contains non-numeric values

  Returns boolean indicating validity

get_matrix_names() → list[str]:

- **Output**: List of all loaded matrix names

8.4.5 Local Functions

None

9 MIS of File Manager Module

9.1 Module

file_manager

9.2 Uses

- **Software Hardware Module**: For interaction with I/O services
- **Control Manager Module**: For file management operations and file path information

9.3 Syntax

9.3.1 Exported Constants

- None

9.3.2 Exported Access Programs

Name	In	Out	Exceptions
read_sequence_file	file_path: str	tuple (str, str)	ValueError
read_submat_file	file_path: str	list[dict]	ValueError
export	data: list[dict], file_path: str	-	IOError

9.4 Semantics

9.4.1 State Variables

None

9.4.2 Environment Variables

None

9.4.3 Assumptions

- Input FASTA files follow format:

```
>Sequence1
ATGC
>Sequence2
TGCA
```

- Substitution matrix files follow format:

```
>MatrixName
1.0,-0.33,-0.33,-0.33
-0.33,1.0,-0.33,-0.33
-0.33,-0.33,1.0,-0.33
-0.33,-0.33,-0.33,1.0
```

- Output directories are writable

9.4.4 Access Routine Semantics

read_sequence_file(file_path: str) → tuple:

- Reads and validates FASTA file
- **Output**: Tuple of two nucleotide sequences (str, str)
- **Exception**: Raises `ValueError` if:
  - Missing sequence headers
  - Invalid number of sequences
  - Zero-length sequences

read_submat_file(file_path: str) → list[dict]:

- Loads and validates substitution matrices
- **Output**: List of matrix dictionaries with:
  - `NAME`: String identifier
  - `PENALIZING_COSTS`: 4x4 numerical matrix
- **Exception**: Raises `ValueError` if:
  - Non-square matrices
  - Non-numeric values
  - Incomplete matrix data

export(data: list[dict], file_path: str):

- Writes benchmark results to CSV
- **Output**: None (creates file at specified path)
- **Exception**: Raises `IOError` if file cannot be written

9.4.5 Local Functions

None

9.5 Considerations

- UTF-8 encoding enforced for all text files
- CSV output includes headers: ”matrix” and ”score”
- Paths are resolved relative to user’s designated directory

10 MIS of Sequence Data Structure Module

10.1 Module

sequence_data_structure

10.2 Uses

- **File Manager Module**: For loading sequence data from files

10.3 Syntax

10.3.1 Exported Constants

None

10.3.2 Exported Access Programs

Name	In	Out	Exceptions
`__init__`	read_file_path: str	SequenceData instance	ValueError
`validate_sequence`	seq: str	-	ValueError
`get_sequences`	-	tuple (str, str)	-

10.4 Semantics

10.4.1 State Variables

- `seq_a`: First nucleotide sequence (5' to 3')
- `seq_b`: Second nucleotide sequence (5' to 3')

10.4.2 Environment Variables

None

10.4.3 Assumptions

- Sequences contain only characters {A, T, G, C, _}
- Sequences are non-empty (length ≥ 1)
- Input files contain exactly two sequences
- Underscore (_) represents gap characters

10.4.4 Access Routine Semantics

SequenceData.__init__(read_file_path: str):

- Loads sequences via `read_sequence_file()`
- Validates sequences using `validate_sequence()`
- Initializes `seq_a` and `seq_b`
- **Exception**: Raises `ValueError` for:
  - Invalid FASTA format
  - Invalid nucleotide characters
  - Unequal sequence counts

validate_sequence(seq: str):

- Checks sequence validity
- **Exception**: Raises `ValueError` if:
  - Contains non-{A,T,G,C,_} characters
  - Zero-length sequence

get_sequences() → tuple:

- **Output**: Returns (`seq_a`, `seq_b`) pair

10.4.5 Local Functions

None

10.5 Considerations

- Case-sensitive for nucleotide characters
- No support for ambiguous bases (N, R, Y, etc.)
- Gap character (_) must be explicitly included
- Validation occurs during initialization

References

Isobel Bicket. Module interface specification for spectrumimageanalysispy. Software documentation, SpectrumImageAnalysisPy, December 2017. Technical documentation, 231 KB.

Carlo Ghezzi, Mehdi Jazayeri, and Dino Mandrioli. *Fundamentals of Software Engineering*. Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 2003.

Daniel M. Hoffman and Paul A. Strooper. *Software Design, Automated Testing, and Maintenance: A Practical Approach*. International Thomson Computer Press, New York, NY, USA, 1995. URL http://citeseer.ist.psu.edu/428727.html.

Deesha Patel. Module interface specification for scec (solar cooker energy calculator). Software documentation, CAS-741-Solar-Cooker, April 2023. Technical documentation, 214 KB.