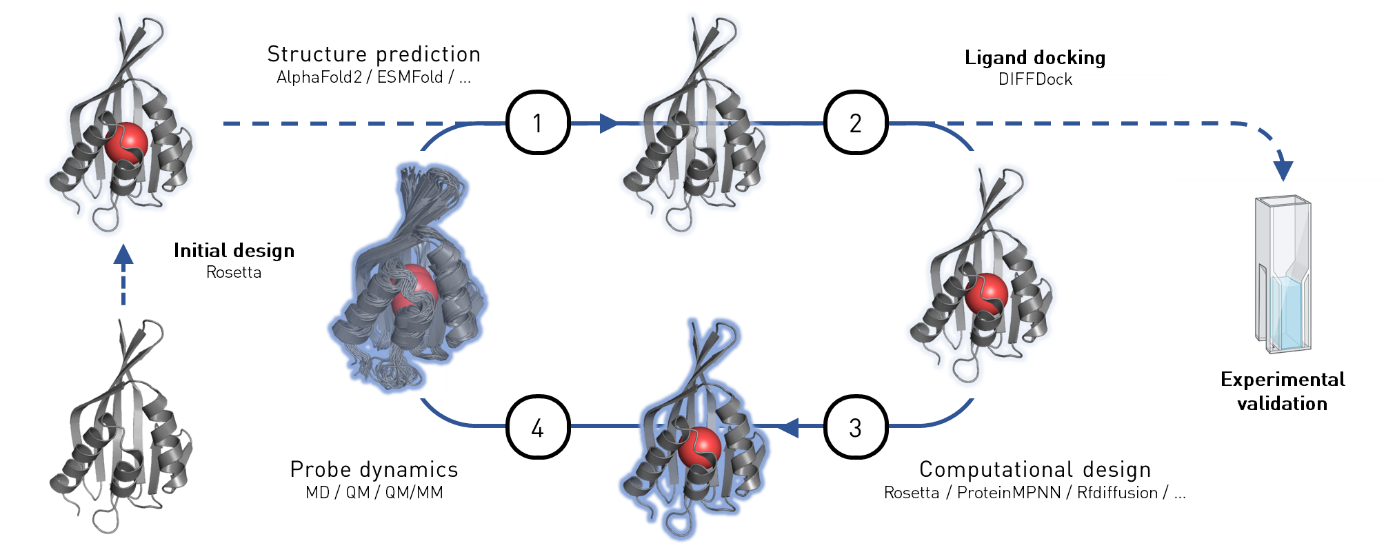
AI.zymes

A Computational Pipeline Combining AI and Conventional Protein Design Methods to Create New Enzymes

Writing convention

* **Variables** in bold green
* ToDos/ Current limitations as footnotes in red



Jannik Neumann, Hans Adrian Bunzel

Department of Biosystems Science and Engineering, ETH Zurich, Basel, Switzerland

Contents

[1 Introduction 2](#_Toc157664233)

[1.1 Coding philosophy 2](#_Toc157664234)

[1.2 Design Packages 3](#_Toc157664235)

[1.2.1 RosettaMatch 3](#_Toc157664236)

[1.2.2 ESMfold\_RosettaRelax 3](#_Toc157664237)

[1.3 Basic concepts 4](#_Toc157664238)

[1.3.1 System startup based on input settings 4](#_Toc157664239)

[1.3.2 Scoring 4](#_Toc157664240)

[1.3.3 Databases 4](#_Toc157664241)

[1.3.4 Controller 4](#_Toc157664242)

[1.3.5 Globally stored variables 5](#_Toc157664243)

[2 Running AI.zymes 6](#_Toc157664244)

[2.1 AI.zymes code 6](#_Toc157664245)

[3 Current State and Future Directions 7](#_Toc157664246)

[3.1 Benchmarking (HAB) 7](#_Toc157664247)

[3.2 Rosetta Match / Combs2 (HAB) 7](#_Toc157664248)

[3.3 ProteinMPNN or all-atom ProteinMPNN 7](#_Toc157664249)

[3.4 Molecular Dynamics 7](#_Toc157664250)

[3.5 Electric fields 7](#_Toc157664251)

[3.6 AI-based hypercontroller (TSC) 7](#_Toc157664252)

[4 Detailed Code Architecture **Error! Bookmark not defined.**](#_Toc157664253)

[4.1 Global variables 8](#_Toc157664254)

[4.2 Global files 8](#_Toc157664255)

[4.3 Local files 8](#_Toc157664256)

[4.4 Functions 8](#_Toc157664257)

[4.4.1 Main Functions - Running 8](#_Toc157664258)

[4.4.2 Main Functions - Design 9](#_Toc157664259)

[4.4.3 Helper Functions 10](#_Toc157664260)

[4.4.4 Plotting Functions 10](#_Toc157664261)

[5 Contributions / History 11](#_Toc157664262)

# Introduction

## Coding philosophy

AI.zymes is a modular program that seamlessly combines computational methods for design, structure prediction, and machine learning in a coherent enzyme design workflow (Fig. 1). At the core, AI.zymes employes the **controller** that decides what action to take and assures that the maximum number of design jobs are running in parallel. The **controller** collects information from the designs and stores them in a shared database, selects which variants to submit for design and decides what type of design or structure prediction to perform with the selected variant.

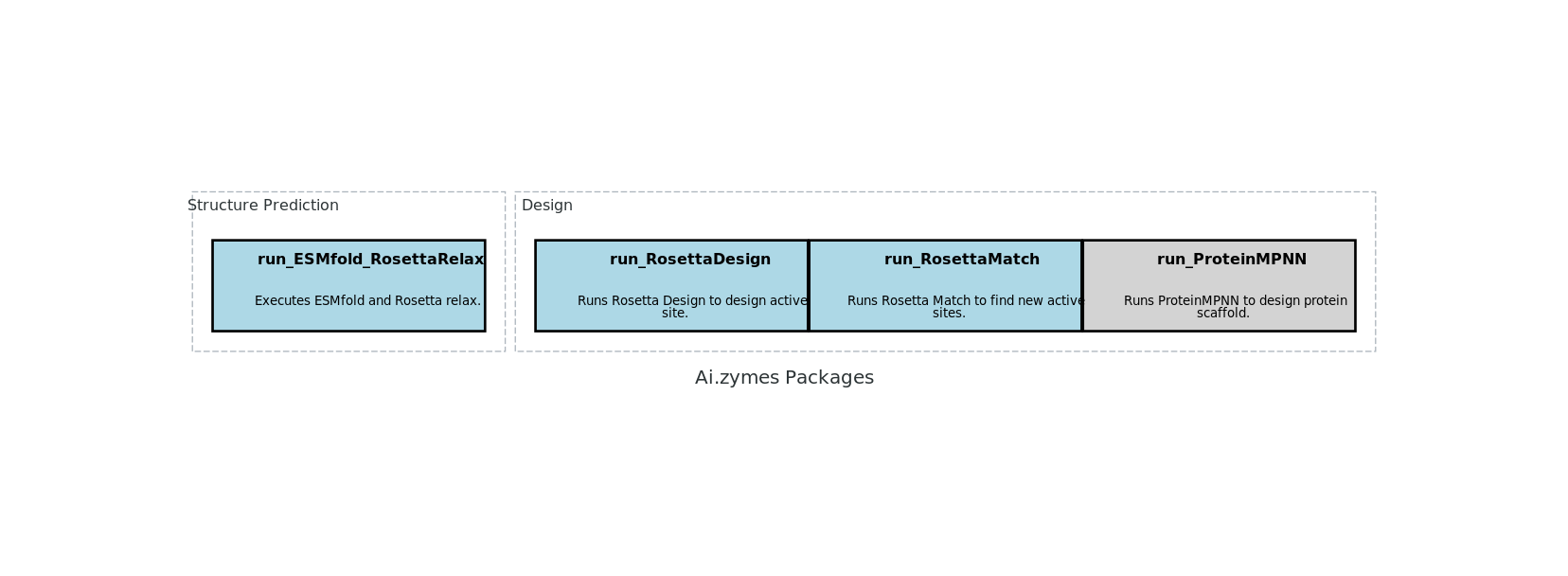
A diagram of a diagram

Description automatically generated with medium confidence

**Fig. 1 | General Flow Chart of AI.zymes.[[1]](#footnote-1)** Based on a set of input variables (grey), AI.zymes will be set up and started (yellow). The main program of AI.zymes is the controller (blue), which controls the overall workflow, decides what action to take (salmon, red), and writes information into the databases (green).

## Available Packages

Currently established design packages include **run\_RosettaMatch** and **run\_RosettaDesign**. For structure prediction, **run\_ESMfold\_RosettaRelax** has been established. The next step will be to establish **run\_ProteinMPNN** for protein design and **run\_ElectricFields** to generate an additional scoring metric based on electrostatic stabilization of the TS. In the future, additional AI and MD packages should be implemented to guide the controller and augment scoring.



*A close-up of a label

Description automatically generated*

**Fig. 2 | Available packages.[[2]](#footnote-2)** Implemented packages are depicted in blue, packages for future implementation are depicted in grey and white based on urgency.

### RosettaMatch

RosettaMatch is an optional design step in AI.zymes that can be used to create *de novo* active sites. AI.zymes starts from all files in **FOLDER\_PARENT**. If RosettaMatch should not be run, these can also be manually supplied by the user. RosettaMatch screens an input structure for potentially binding sites using an enzdes-type CST file. To that end, the Matcher tries to find pockets that can accommodate the ligand as well as all catalytic residues defined in a constraint file. Importantly, the Matcher ignores all sidechains present in the structure and only recognizes the input structure backbone. The Matcher can thus introduce new pockets and does not rely on structures that already contain a pocket.

**run\_RosettaMatch** requires an input **WT** structure with a ligand molecule positioned roughly where the new active site is to be designed, **run\_RosettaMatch** will relax the structure using **run\_ESMfold\_RosettaRelax**, so no initial realax is required. Furthermore, an enzdes-type CST file **{LIGAND}\_enzdes.cst** and a **{LIGAND}.params** file[[3]](#footnote-3) must be supplied. Finally, the matcher requiresthe definition of the central ligand atom **{LIGAND}.central** and various parameters.[[4]](#footnote-4)

**run\_RosettaMatch** produces several Match PDB structures in **{FOLDER\_HOME}/{FOLDER\_MATCH} /matches** that contain new catalytic residues and the bound reaction transition state. These structures can be accessed by the main AI.zymes algorithm through **FOLDER\_PARENT**.

### RosettaDesign

### ESMfold\_RosettaRelax

## Basic concepts

### System startup based on input settings

The system startup involve n optional setup / reset of AI.zymes (**setup\_aizymes**) to start AI.zymes blank. In addition, a startup script is run every time the controller is started to load all necessary information for the controller to run smoothly (**startup\_controller**).

To set up the system, various global variables need to be defined (see Globally stored variables). Among others, these include the name of the protein and ligand as well as which residues can be designed. Furthermore, general settings can be set such as how many jobs may run in parallel and how many designs are to be done in total.

### Scoring

Three different scores have thus far proven valuable to identify promising enzyme designs: The total\_score corresponding to the total energy of the system, the interface\_score corresponding to the binding energy of the ligand to the protein, as well as the catalytic\_score corresponding to the score of the catalytic interaction. AI.zymes uses the concept of potential to select which variants to take forward for design. Potential is aimed to provide some predictive information on the variants. Thus, each potential value corresponds to the arithmetic average of a structure’s score, as well as of the corresponding score from all its directed descendants. To select a variant for design, the total\_potential, catalytic\_potential, and interface\_potential are normalized from 0 to 1 with 1 being the best, and the geometric mean is calculated from these potentials for each variant to give the combined\_potential (Eq. 1). Boltzmann selection is performed on the combined\_potential to finally identify the variant to be taken forward for design.

|  |  |  |
| --- | --- | --- |
|  |  | Eq. 1 |

### Databases

The ALL\_SCORES.csv database is the main file that holds all information of the AI.zymes run. Amongst others, ALL\_SCORES contains information on the parent scaffold variant, the precise design algorithm used, as well as key scoring metrics obtained from Rosetta. In addition, the BLOCKED.csv database contains a list of all structures that are currently undergoing structure prediction. These structures are excluded from Boltzmann selection, to prevent that structure prediction is needlessly performed multiple times based on the same structure.

### Controller

The controller is the central program of AI.zymes. It constantly cycles between three different scripts. The first scrip (update\_scores) checks all designs in ALL\_SCORES.csv that do not yet contain any scores. If it finds a finished design, it will update the scores of that design. update\_scores also unblocks all indices for which the structure prediction runs are completed. Subsequently, the controller will find the next scaffold for design by Boltzman selection. Only unblocked indices will go into the selection algorithm and selection will be based on the combined\_potential. Once an index is selected, the control starts the calculation. To that end, it checks if there is a structure of the selected design that went through ESMfold\_RosettaRelax. If not, the controller will start the structure prediction and block the selected index. If there is a relaxed structure, the controller will generate a new index into which the design will be stored. This involved creating a folder for the new design and appending the ALL\_SCORES.csv file with the selected index. Finally, the controller will check how many jobs are currently running. It will wait until the number of running jobs is lower than the maximum number of jobs to restart the controller cycle.

### Globally stored variables

Various key variables controlling the behavior of AI.zymes are stored in variables.json. Variables that keep track of the system include the name of the parent structure (PARENT), the name of the bound ligand (LIGAND), a list of residue numbers to repack, design, and restrict (REPACK, DESIGN, RESTRICT), and the remark line that should be added on top of the PDB to define catalytic interactions (REMARK). The overall design flow is controlled by the maximum number of jobs that can run in parallel (MAX\_JOBS), the number of jobs that should be run with the parent structure before the selection of the designed structure kicks in (N\_PARENT\_JOBS) and the maximum number of designs to be performed (MAX\_DESIGNS). In addition, specific variables controlling the behavior of specific programs are set, including the Boltzmann temperature used during selection (KBT\_BOLTZMANN), the constraint weight biasing design towards the parent sequence (CST\_WEIGHT) and the probability to run ProteinMPNN instead of RosettaDesign (ProteinMPNN\_PROB) as well as the temperauter used for ProteinMPNN ('ProteinMPNN\_T'). Several other variables control the overall file architecture, including the current design folder (DESIGN\_FOLDER), the path to Rosetta (ROSETTA\_PATH), whether or not to run design in a quick testing mode (EXPLORE), the prefix used for job submission to identify the AI.zymes jobs (SUBMIT\_PREFIX), and the identity of the cluster currently used (BLUEPEBBLE or GRID).

# Running AI.zymes

See <https://wiki-bsse.ethz.ch/display/DBSSEPANKE/Jupyter+Notebook+Setup+on+Euler+and+CSCS> on instructions on how to connect to the Grid.

## AI.zymes code

At some point, the whole AI.zymes code will be stored at GitHub. The repo is periodically updated by HAB. Please ask to be added.

<https://github.com/bunzela/AIzymes>

In the meantime, there will be a Polybox folder accessible to this link. For now, this is read-only.

<https://polybox.ethz.ch/index.php/s/Na33zctTh9L3grZ>

# Current State and Future Directions

AI.zymes is currently benchmarked. To that end, the promiscuous Kemp eliminase activity of ketosteroid isomerase (KSI) is being targeted by design. At the moment, design only involved a Rosetta design step targeting the active site and an ESMfold structure prediction step. Benchmarking involves finding the ideal settings of CST\_WEIGHT and KBT\_BOLTZMANN. Once benchmarking is completed, de novo design with Rosetta Match or Combs2 will be attempted. If successful, this will allow swift publication of the initial version of AI.zymes. Key steps in the development process are listed below:

## Benchmarking (HAB)

* What are the best settings for CST\_WEIGHT and KBT\_BOLTZMANN

## Rosetta Match / Combs2 (HAB)

* Implement Rosetta Match
* Implement Combs2 – superior matching algorithm
* De novo design based on Match / Combs2

## ProteinMPNN or all-atom ProteinMPNN

* We want to use ProteinMPNN to design the environment around the active site.
* Alternatively, all-atom ProteinMPNN was recently published. HAB is keen on testing this next.

## Molecular Dynamics

* Molecular dynamics simulations have proven powerful previously to identify promising variants from design. It would be great to include them in the pipeline as well.
* From MD, we might be able to design active sites that are highly organized and remain predominantly in a catalytically active conformation.

## Electric fields

* Natural enzymes exert high electric fields onto their substrate to promote reactions. We have previously developed an algorithm to probe these field effects. It would be great to likewise calculate fields for scoring our enzyme variants.

## AI-based hypercontroller (TSC)

* AI.zymes generates a lot of sequences. Two different AI algorithms could be developed:
  + Learn the fitness landscape from the designed sequence to guide the sequence space to be explored by AI.zymes
  + Learn which hyperparameters (CST\_WEIGHT, KBT\_BOLTZMANN, etc.) are best and adjust these during the design run.

# Manual

AIZYME\_TOOLS.py contains all functions to run AI.zymes. In the following, the input and output of each function will be briefly discussed. In addition, an overview over all files and global variables is given.

## Global variables

## Global files

## Local files

## Functions

### Main Functions - Running

controller

Main function that runs AI.zymes. The controller runs constantly. As soon as a job is completed, the controller launches a new job. To avoid errors of various jobs writing in the same file, only the controller, and functions launched of it, can write globally shared files.

RESET Deletes all files in the current design run to start the design from scratch. Default: [false]

EXPLORE Adjust the design settings to accelerate design. Useful for testing but should not be used for real design. Default: [false]

UNBLOCK\_ALL Reset the list of blocked structures. Structures are blocked for design while their structure is being predicted. Useful if AI.zymes crahsed unexpectedly. Note: Make sure to cancel all running structure prediction jobs. Default: [false]

PRINT\_VAR True

PLOT\_DATA True

BLUEPEBBLE False

GRID True

check\_running\_jobs

Description

update\_potential bla

score\_type bla

score bla

index bla

all\_scores\_df bla

update\_scores

Description

all\_scores\_df

blocked\_df

normalize\_scores

Description

all\_scores\_df

print\_norm =False

norm\_all =False

extension ="score"):

boltzmann\_selection

Description

all\_scores\_df

blocked\_df ):

start\_calculation

Description

all\_scores\_df

blocked\_df

selected\_index ):

create\_new\_index

Description

parent\_index

all\_scores\_df ):

### Main Functions - Design

run\_ESMfold\_RosettaRelax

Description

index Index of the structure on which structure prediction is to be performed

RosettaDesign Default: [false]

ProteinMPNN Default: [false]

StartupRelax Default: [false]

run\_RosettaDesign

Description

(parent\_index,

new\_index ):

run\_ProteinMPNN

(parent\_index ,

new\_index ,

bash Default: [false] ):

startup\_controller

(UNBLOCK\_ALL ,

PRINT\_VAR Default: [True],

PLOT\_DATA Default: [True]

prepare\_input\_files

Description

convert\_outputs\_to\_pdb

(outputs):

setup\_aizymes

(RESET):

### Helper Functions

submit\_job

(index,

job,

bash=False):

extract\_sequence\_from\_pdb

(pdb\_path):

### Plotting Functions

Various function to plot the progress of AI.zymes have been written. These can be accessed via **plot\_scores()**.

# Contributions / History

AIzyme\_Functions\_007 Version used for Benchmarking until 01.02.2024

AIzyme\_Functions\_008 RosettaMatch added

Now accepts multiple input structures. Run on all structures in **FOLDER\_PARENT**

Jannik Neumann, Hans Adrian Bunzel

Department of Biosystems Science and Engineering, ETH Zurich, Basel, Switzerland

The initial version of the AI.zymes code was written by HAB and JN. JN benchmarked the AI.zymes predecessor code, which has a linear script-based flow.

1. Made with the script AIzymes\_Flowchart/AIzymes\_Flowchart.ipynb [↑](#footnote-ref-1)
2. Made with the script AIzymes\_Flowchart/AIzymes\_Flowchart.ipynb [↑](#footnote-ref-2)
3. [currently user supplied] [↑](#footnote-ref-3)
4. [currently hardcoded] [↑](#footnote-ref-4)