Table of Contents

2. Code [page]
- 2.1 AIzymes [page]
  - 2.1.1 AIzymes_MAIN [page]
  - 2.1.2 __init__ [page]
  - 2.1.3 setup [page]
  - 2.1.4 initialize [page]
  - 2.1.5 controller [page]
  - 2.1.6 plot [page]
- 2.2 design_ESMfold [page]
  - 2.2.1 prepare_ESMfold [page]
- 2.3 design_LigandMPNN [page]
  - 2.3.1 prepare_LigandMPNN [page]
- 2.4 design_match [page]
- 2.5 design_ProteinMPNN [page]
  - 2.5.1 prepare_ProteinMPNN [page]
- 2.6 design_RosettaDesign [page]
  - 2.6.1 prepare_RosettaDesign [page]
- 2.7 design_RosettaRelax [page]
  - 2.7.1 prepare_RosettaRelax [page]
- 2.8 helper [page]
  - 2.8.1 run_command [page]
  - 2.8.2 get_PDB_in [page]
  - 2.8.3 save_cat_res_into_all_scores_df [page]
  - 2.8.4 reset_to_after_index [page]
  - 2.8.5 wait_for_file [page]
- 2.9 main_design [page]
- 2.10 main_running [page]
  - 2.10.1 start_controller [page]
  - 2.10.2 check_running_jobs [page]
  - 2.10.3 update_potential [page]
  - 2.10.4 update_scores [page]
  - 2.10.5 boltzmann_selection [page]
  - 2.10.6 check_parent_done [page]
  - 2.10.7 start_parent_design [page]
  - 2.10.8 start_calculation [page]
  - 2.10.9 create_new_index [page]
- 2.11 main_scripts [page]
- 2.12 main_startup [page]
- 2.13 plotting [page]
  - 2.13.1 plot_interface_v_total_score_selection [page]
  - 2.13.2 plot_interface_v_total_score_generation [page]
  - 2.13.3 plot_stacked_histogram_by_cat_resi [page]
  - 2.13.4 plot_stacked_histogram_by_cat_resn [page]
  - 2.13.5 plot_stacked_histogram_by_generation [page]
- 2.14 scoring_efields [page]
  - 2.14.1 prepare_efields [page]
  - 2.14.2 update_efieldsdf [page]
- 2.15 setup_system [page]

1. Introduction

1.1 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.

Fig. 1 | General Flow Chart of AI.zymes. 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).

1.2 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.

Fig. 2 | Available packages. Implemented packages are depicted in blue, packages for future implementation are depicted in grey and white based on urgency.

1.2.1 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 must be supplied. Finally, the matcher requires the definition of the central ligand atom {LIGAND}.central and various parameters.
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.

1.2.2 RosettaDesign

1.2.3 ESMfold_RosettaRelax

1.3 Basic concepts

1.3.1 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.

1.3.2 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 structureâ€™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.
combined_potential=âˆ›(total_potential Ã—interface_potentialÃ—catalytic_potential) Eq. 1

1.3.3 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.

1.3.4 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.

1.3.5 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).

2. Code

2.1 AIzymes

AIzymes Project Main Workflow

This script defines the main AIzymes workflow, including setup, initialization, control, and plotting functions. It manages the primary processes and configurations required to execute AIzymes functionalities.

Classes

AIzymes_MAIN	Manages the main workflow for AIzymes, including setup, initialization, and various control functions.

Functions

__init__()	Initializes an instance of the AIzymes_MAIN class.
setup()	Sets up the AIzymes project environment with specified parameters.
initialize()	Initializes AIzymes with provided configurations.
controller()	Controls the AIzymes project based on scoring and normalization parameters.
plot()	Generates various plots based on AIzymes data.

2.1.1 AIzymes_MAIN

Main class for managing AIzymes workflow, including setup, initialization, control, and plotting functions.

2.1.2 __init__

Initializes an instance of the AIzymes_MAIN class.

2.1.3 setup

Sets up the AIzymes project environment with specified parameters.

Args

FOLDER_HOME (str)	Path to the main folder.
FOLDER_PARENT (str)	Path to the parent folder.
CST_NAME (str)	Constraint name.
WT (str)	Wild type information.
LIGAND (str)	Ligand data.
DESIGN (str)	Design specifications.
MAX_JOBS (int)	Maximum number of jobs to run concurrently.
N_PARENT_JOBS (int)	Number of parent jobs.
MAX_DESIGNS (int)	Maximum number of designs.
KBT_BOLTZMANN (list)	Boltzmann constant values.
CST_WEIGHT (float)	Constraint weight.
ProteinMPNN_PROB (float)	Probability parameter for ProteinMPNN.
ProteinMPNN_BIAS (float)	Bias parameter for ProteinMPNN.
LMPNN_PROB (float)	Probability parameter for LMPNN.
FOLDER_MATCH (str)	Path to match folder.
ProteinMPNN_T (str)	Temperature for ProteinMPNN.
LMPNN_T (str)	Temperature for LMPNN.
LMPNN_BIAS (float)	Bias parameter for LMPNN.
SUBMIT_PREFIX (str)	Submission prefix.
SYSTEM (str)	System information.
MATCH (str)	Match specifications.
EXPLORE (bool)	Whether to explore parameter space.
FIELD_TARGET (str)	Target atoms at which to calculate electric field.
LOG (str)	Logging level.
PARENT_DES_MED (str)	Parent design method.

2.1.4 initialize

Initializes AIzymes with given parameters.

Args

FOLDER_HOME (str)	Path to the main folder.
UNBLOCK_ALL (bool)	Flag to unblock all processes.
PRINT_VAR (bool)	Flag to print variables.
PLOT_DATA (bool)	Flag to plot data.
LOG (str)	Logging level.

2.1.5 controller

Controls the AIzymes project based on scoring and normalization parameters.

Args

HIGHSCORE (float)	High score threshold for evaluation.
NORM (dict)	Normalization values for different scores.

2.1.6 plot

Generates plots based on AIzymes data, including main, tree, and landscape plots.

Args

main_plots (bool)	Flag to generate main plots.
tree_plot (bool)	Flag to generate tree plot.
landscape_plot (bool)	Flag to generate landscape plot.
print_vals (bool)	Flag to print values on plots.
NORM (dict)	Normalization values for different scores.
HIGHSCORE_NEGBEST (dict)	High score and negative best score for different metrics.

2.2 design_ESMfold

Design ESMfold Module

Manages the structure prediction of protein sequences using ESMfold within the AIzymes project.

Functions

- prepare_ESMfold	Prepares commands for ESMfold job submission.

2.2.1 prepare_ESMfold

Predicts structure of sequence in {index} using ESMfold.

Parameters: index (str): The index of the protein variant to be predicted. cmd (str): Growing list of commands to be exected by run_design using submit_job.

Returns

cmd (str)	Command to be exected by run_design using submit_job.

2.3 design_LigandMPNN

NOTE: NOT WORKING YET!

Integrates LigandMPNN for generating protein sequences adapted to specific ligand contexts.

Functions

prepare_LigandMPNN	Executes LigandMPNN steps to adapt protein designs with ligand binding considerations.

Modules Required helper_001

2.3.1 prepare_LigandMPNN

Executes the LigandMPNN pipeline for a given protein-ligand structure and generates new protein sequences with potentially higher functional scores considering the ligand context.

Parameters: - parent_index (str): The index of the parent protein variant. - new_index (str): The index assigned to the new protein variant. - all_scores_df (DataFrame): A DataFrame containing information for protein variants.

2.4 design_match

Design Match Module

Provides functionalities for matching protein designs with specific constraints and requirements in the AIzymes workflow.

Functions

- prepare_LigandMPNN	Executes the LigandMPNN pipeline for protein-ligand structure adaptation.

Modules Required - helper_001

2.5 design_ProteinMPNN

Manages ProteinMPNN design steps to generate protein sequences tailored for specific functional and structural properties in the AIzymes project.
This function assumes the ProteinMPNN toolkit is available and properly set up in the specified location. It involves multiple subprocess calls to Python scripts for processing protein structures and generating new sequences.

Functions

prepare_ProteinMPNN	Sets up commands for ProteinMPNN job submission.

Modules Required helper_001

2.5.1 prepare_ProteinMPNN

Executes the ProteinMPNN pipeline for a given protein structure and generates new protein sequences with potentially higher functional scores.

Args

new_index (str)	The index of the designed variant.
cmd (str)	Growing list of commands to be exected by run_design using submit_job.

Returns

cmd (str)	Command to be exected by run_design using submit_job

2.6 design_RosettaDesign

Handles RosettaDesign steps to optimize protein design scores and enhance stability within the AIzymes project.

Functions

prepare_RosettaDesign	Prepares RosettaDesign commands for job submission.

Modules Required helper_001

2.6.1 prepare_RosettaDesign

Designs protein structure in {new_index} based on {parent_index} using RosettaDesign.

Args

parent_index (str)	Index of the parent protein variant to be designed.
new_index (str)	Index assigned to the resulting design.
input_suffix (str)	Suffix of the input structure to be used for design.

Returns

cmd (str)	Command to be exected by run_design using submit_job.

2.7 design_RosettaRelax

Executes RosettaRelax to refine protein structures and improve stability in the AIzymes project.

Functions

prepare_RosettaRelax	Sets up commands for RosettaRelax job submission.

Modules Required helper_001

2.7.1 prepare_RosettaRelax

Relaxes protein structure in {index} using RosettaRelax.
Args index (str): The index of the protein variant to be relaxed. cmd (str): collection of commands to be run, this script wil append its commands to cmd
Optional parameters: PreMatchRelax (bool): True if ESMfold to be run without ligand (prior to RosettaMatch).

2.8 helper

Contains utility functions and supporting routines used across multiple modules within the AIzymes project.
Functions - normalize_scores - one_to_three_letter_aa - run_command - get_PDB_in - load_main_variables - save_main_variables - submit_job - sequence_from_pdb - generate_remark_from_all_scores_df - save_cat_res_into_all_scores_df - reset_to_after_parent_design - reset_to_after_index - save_all_scores_df - get_best_structures - remove_intersection_best_structures - trace_mutation_tree - print_average_scores - wait_for_file - hamming_distance - exponential_func

---

Modules Required setup_system_001

2.8.1 run_command

Wrapper to execute .py files in runtime with arguments, and print error messages if they occur.
Parameters: command: The command to run as a list of strings. cwd: Optional; The directory to execute the command in. capture_output: Optional; If True, capture stdout and stderr. Defaults to False (This is to conserve memory).

2.8.2 get_PDB_in

Based on index, find the input PDB files for the AIzymes modules
Paramters: - index: The indix of the current design
Output: - PDBfile_Design_in: Input file for RosettaDesign - PDBfile_Relax_in: Input file for RosettaRelax - PDBfile_Relax_ligand_in: Input file for Ligand to be used in RosettaRelax

2.8.3 save_cat_res_into_all_scores_df

Finds the indices and names of the catalytic residue from Saves indices and residues into in row as lists. To make sure these are saved and loaded as list, ";".join() and .split(";") should be used If information is read from an input structure for design do not save cat_resn

2.8.4 reset_to_after_index

This function resets the run back to a chosen index. It removes all later entries from the all_scores.csv and the home dir. index: The last index to keep, after which everything will be deleted.

2.8.5 wait_for_file

Wait for a file to exist and have a non-zero size.

2.9 main_design

Main Design Module
Coordinates various design steps, managing the workflow of Rosetta, ProteinMPNN, and other modules within the AIzymes project.
Functions

- get_ram	Determines RAM allocation for design steps.
- run_design	Runs the selected design steps based on configuration.

Modules Required - helper_001, design_match_001, design_ProteinMPNN_001, design_LigandMPNN_001, design_RosettaDesign_001, design_ESMfold_001, design_RosettaRelax_001

2.10 main_running

main_running_001.py

This module contains the main control functions for managing the AIzymes workflow, including job submission, score updating, and Boltzmann selection. The functions in this file are responsible for the high-level management of the design process, interacting with the AIzymes_MAIN class to initiate, control, and evaluate design variants.

Classes None

Functions start_controller(self) check_running_jobs(self) update_potential(self, score_type, index) update_scores(self) boltzmann_selection(self) check_parent_done(self) start_parent_design(self) start_calculation(self, parent_index) create_new_index(self, parent_index)

2.10.1 start_controller

Runs the main loop to manage the design process until the maximum number of designs is reached.
Continues submitting jobs, monitoring running processes, and updating scores based on Boltzmann selection until `MAX_DESIGNS` is achieved. The function pauses or starts new designs based on system resources.
Parameters: self: An instance of the AIzymes_MAIN class with setup attributes and properties.

2.10.2 check_running_jobs

Checks the current number of running jobs based on the system type.
Depending on the value of `SYSTEM`, this function counts the active jobs in GRID, BLUEPEBBLE, BACKGROUND_JOB, or ABBIE_LOCAL systems.

Returns

int	Number of running jobs for the specific system.

2.10.3 update_potential

Updates the potential file for a given score type at the specified variant index.
Creates or appends to a `_potential.dat` file in `FOLDER_HOME/`, calculating and updating potentials for the parent variant if necessary.
Parameters: score_type (str): Type of score to update (e.g., total, interface, catalytic, efield). index (int): Variant index to update potential data.

2.10.4 update_scores

Updates various scores, including total, interface, catalytic, and efield scores for each design variant.
This function iterates over design variants, updating scores based on files generated by different processes. It also updates sequence information, tracks mutations, and saves the updated DataFrame.
Parameters: self: An instance of the AIzymes_MAIN class with setup attributes and properties.

2.10.5 boltzmann_selection

Selects a design variant based on a Boltzmann-weighted probability distribution.
Filters variants based on certain conditions (e.g., scores, block status), then computes probabilities using Boltzmann factors with a temperature factor (`KBT_BOLTZMANN`) to select a variant for further design steps.

Returns

int	Index of the selected design variant.

2.10.6 check_parent_done

Determines if parent designs are complete based on the number of generated designs and parent jobs.

Returns

bool	True if parent designs are complete, otherwise False.

2.10.7 start_parent_design

Initiates a new design process for a parent structure by creating a new variant entry.
Sets up the required files and configuration for designing a parent structure, then calls the design method specified in `PARENT_DES_MED`.

Parameters: self: An instance of the AIzymes_MAIN class with setup attributes and properties.

2.10.8 start_calculation

Decides the next calculation step for the specified design variant index.
Based on the current design state, this function decides to run ESMfold, RosettaRelax, or a design method for the given index.
Parameters: parent_index (int): Index of the variant to evaluate for further calculations.

2.10.9 create_new_index

Creates a new design entry in `all_scores_df` with a unique index, inheriting attributes from a parent variant.
Updates the DataFrame with new index information, saves the updated file, and sets up the directory structure for the new design variant.
Parameters: parent_index (str): The index of the parent variant or "Parent" for initial designs.

Returns

int	The newly created index for the variant.

2.11 main_scripts

2.12 main_startup

2.13 plotting

2.13.1 plot_interface_v_total_score_selection

Plots a scatter plot of total_scores vs interface_scores and highlights the points corresponding to the selected indices.
Parameters: - ax (matplotlib.axes.Axes): The Axes object to plot on. - total_scores (list or np.array): The total scores of the structures. - interface_scores (list or np.array): The interface scores of the structures. - selected_indices (list of int): Indices of the points to highlight.

2.13.2 plot_interface_v_total_score_generation

Plots a scatter plot of total_scores vs interface_scores and colors the points according to the generation for all data points, using categorical coloring. Adds a legend to represent each unique generation with its corresponding color.
Parameters: - ax (matplotlib.axes.Axes): The Axes object to plot on. - total_scores (list or np.array): The total scores of the structures. - interface_scores (list or np.array): The interface scores of the structures. - generation (pd.Series or np.array): Generation numbers for all data points.

2.13.3 plot_stacked_histogram_by_cat_resi

Plots a stacked bar plot of interface scores colored by cat_resi on the given Axes object, where each bar's segments represent counts of different cat_resi values in that bin.
Parameters: - ax (matplotlib.axes.Axes): The Axes object to plot on. - all_scores_df (pd.DataFrame): DataFrame containing 'cat_resi' and 'interface_score' columns. - color_map (dict): Optional; A dictionary mapping catalytic residue indices to colors. - show_legend (bool): Optional; Whether to show the legend. Defaults to False.

2.13.4 plot_stacked_histogram_by_cat_resn

Plots a stacked bar plot of interface scores colored by cat_resn on the given Axes object, where each bar's segments represent counts of different cat_resn values in that bin.
Parameters: - ax (matplotlib.axes.Axes): The Axes object to plot on. - all_scores_df (pd.DataFrame): DataFrame containing 'cat_resn' and 'interface_score' columns.

2.13.5 plot_stacked_histogram_by_generation

Plots a stacked bar plot of interface scores colored by generation on the given Axes object, where each bar's segments represent counts of different generation values in that bin.
Parameters: - ax (matplotlib.axes.Axes): The Axes object to plot on. - all_scores_df (pd.DataFrame): DataFrame containing 'generation' and 'interface_score' columns.

2.14 scoring_efields

2.14.1 prepare_efields

Calculate electric fields for structure in {index}.

Args

index (str)	The index of the protein variant for whcih efields are to be calculated.
cmd (str)	Growing list of commands to be exected by run_design using submit_job.

Returns

cmd (str)	Command to be exected by run_design using submit_job.

2.14.2 update_efieldsdf

Adds a new row to "{FOLDER_HOME}/electric_fields.csv" containing the electric fields generated by FieldTools.py for all residues in the protein

2.15 setup_system

Contains system specifc information. At the Moment, this is all hard-coded. In the future, this will be part of the installation of AIzymes.

set_system() contains general variables. submit_head() constructs the submission header to submit jobs.