Welcome

Contents

This is the root of the book.

Here is a list:

- A
- B
- C
    1. 1
    2. 2
- D

Declaration of authorship

I hereby declare that the bachelor thesis submitted is my own work. It was made unaided by third parties and only using the quoted sources and tools. All sources used are labelled as references. The thesis, in equal or similar form, was not previously presented to another examination board. Alle exemplare übereinstimmen.

Abstract

here

English title

This dataset builds the master thesis with the title ‘From raw data to kinetic parameters: an EnzymeML-based workflow for reproducible enzyme kinetics’. The thesis contains four different research scenarios, in which kinetic parameter were estimated by a reproducible modeling workflow based on EnzymeML.

German title

Acknowledgements

Danke Merkel.

Abbreviations

:Alignment 1: If the field body starts on the first line...

```
        Then the entire field body must be indented the same.
```

:Alignment 2: If the field body starts on a subsequent line...

Introduction

- Reproducibility crisis
- Kinetic parameter estimation of enzyme reactions
- Biology and big data

Methods

1. EnzymeML

In its core, EnzymeML is a data model, structuring data and metadata of biocatalytic reactions. Thereby, information on reaction conditions, substrate and product measurement data, as well as estimated kinetic parameters are documented [Pleiss, 2021]. Additionally, the enzyme and the reactants are specified by protein sequence or InChI respectively, enabling clear description of all involved species in an enzyme reaction. EnzymeML is based on the ontology of Systems Biology Markup Language [Hucka *et al.*, 2003] and adheres to STRENDA guidelines [Tipton *et al.*, 2014], which define minimal reporting standards for enzymatic experiments. Hence, EnzymeML serves as an exchange format for biocatalytic data between experimentalist, modelers and database providers, which is compliant with FAIR data principles [Wilkinson *et al.*, 2016].
EnzymeML documents can be read, edited, and written via the Python API PyEnzyme, providing the possibility to integrate PyEnzyme in Python data analysis workflows.

1.1 Creation of EnzymeML documents

EnzymeML documents were created using the EnzymeML Excel template, in combination with the `.fromTemplate()` method of PyEnzyme. Within the spreadsheet, data and metadata of an experiment can be be filled in the respective sheets. Alternatively, only metadata was entered to the spreadsheet, whereas measurement data was parsed by a custom Python function from the output file of the analytical device to the measurement data section of the EnzymeML document.

CaliPytion

CaliPytion was developed to provide an easy way to use linear and non linear calibration equations to calculate concentrations of analytical raw data based on a standard measurements.

Data model for calibration data

The standard curve functionality of CaliPythion is based on a data model, structuring data and metadata of calibration measurements. Thereby, calibration conditions like temperature and pH, and information of the analytical device can be provided. Furthermore, the name as well as an ID can be specified for the analyzed substance. All of this information is stored in the `Calibration` root object. Additionally, the root object can contain a `Standard` and a `Spectrum`. A standard contains measurements of multiple predefined concentrations and a wavelength at which the measurement was conducted, in the case of spectrophotometry. A `Spectrum` can be defined by providing measurement data as well as the respectuve wavelengths at which the data was measured.

Linear and non-linear fitting of calibration equations

Finds optimal model parameters through non-linear least squares fitting of experimental data to models
Since no true model from the relation between ..., and calibrations ar known to not always be linear

the following equations were implemented

$$A = ax \tag{1}$$

$$A = ax^2 + bx \tag{2}$$

$$A = ax^3 + bx^2 + cx \tag{3}$$

$$A = ae^{\frac{b}{x}} \tag{4}$$

(4)

$$A = \frac{ax}{b + x} \tag{5}$$

When a `StandardCurve` is created, the measurement data, which is defined in `Calibration` is used to fit the listed calibration model equations to the data. The model, which represents the relation between analytical signal and concentration the best, is determined based on the lowest Akaike information criterion (AIC). Concentrations are calculated by calculating the root of the fitted calibration model.
After `StandardCurve` initialization, concentrations can be calculated by calling the `get_concentration()` method. Alternatively, a `StandardCurve` can be directly applied to an `EnzymeMLDocument` by calling the `apply_to_EnzymeML()` method

```
product_standard = StandardCurve()
enzmldoc_absorption = EnzymeMLDocument()

enzmldoc_concentration = product_standard.apply_to_EnzymeML(
    enzmldoc=enzmldoc_absorption,
    species_id="s1")
```

EnzymePynetics

Overview

EnzymePynetics is a a python package, for kinetic parameter estimation of single-substrate enzyme reactions, which was developed during this thesis. The `ParameterEstimator` of EnzymePynetics estimates the kinetic parameters k_{cat} and K_m by fitting time-course measurement data of enzyme reactions to different Michaelis-Menten models. Thereby, the residuals between measurement data and integrated Michaelis-Menten rate equations are minimized through a non-linear least-squares algorithm. Additionally, the inhibition constant K_i can be assessed for potential substrate or product inhibition. Furthermore, the inhibition constant of an enzyme inhibitor can be determined.

Data model

Data models build the backbone of applications, by defining the relations between informations in an hierarchical manner. EnzymePynetics is based on a data model, resembling the experimental design of an enzyme kinetics assay. Thereby, all relevant data and meta data of an kinetic experiment are ordered in the base object `EnzymeKineticsExperiment`. On the metadata side, the base object consists of the attributes temperature with its respective unit, pH, and the name of the measured substance. Additionally, it can be specified whether the measurement data originates from substrate or product measurements. On the data side, `EnzymeKineticsExperiment` contains one or multiple `Measurements`. Each measurement stores the information of an experimental condition, to which the enzyme was subjected. Therefore, each `Measurement` contains information on the initial substrate concentration, enzyme concentration, and inhibitor concentration, if present, along with the respective concentration units. Each `Measurement` contains the measured data, which itself consist of one or multiple replicates of the respective experimental condition. The data model was was generated using sdRDM, a python tool allowing the creation and versioning of data models.

An extensive documentation of the data model can be accessed in the specifications of the the software package.

3.3 Kinetic models

Besides the irreversible Michaelis-Menten rate equation (Eq. 1) inhibition models for competitive (Eq. 2), uncompetitive (Eq. 3), and non-competitive) inhibition (Eq. 4) were implemented. Thereby, S, E, and I denote the concentration of substrate, enzyme, and inhibitor, respectively. In therms of kinetic parameters, k_{cat} denotes the turnover number, K_m the Michaelis-Menten constant of the substrate, whereas K_{ic} and K_{iu} describe the competitive and uncompetitive inhibition constant, respectively.

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m + S} \tag{6}$$

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m * (1 + \frac{I}{K_{ic}}) + S} \tag{7}$$

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m * (1 + \frac{I}{K_{iu}}) * S} \tag{8}$$

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m * (1 + \frac{I}{K_{ic}}) + (1 + \frac{I}{K_{iu}}) + S} \tag{9}$$

By default, E is assumed to be constant throughout the reaction. If required, the kinetic models can be extended by the parameter K_{inact}, which describes the time-dependent inactivation rate of the enzyme. The decrease in active enzyme is modeled by Eq. 5:

$$\frac{dE}{dt} = -K_{inact} * E \tag{10}$$

3.4 Initialization

Whereas the `EnzymeKineticsExperiment` object solely serves a The `ParameterEstimator` harbors the functionalities for parameter estimation. Data can be provided by passing data as an `EnzymeKineticsExperiment` object. Alternatively, an EnzymeML documents can be provided as the data source via PyEnzyme software.
Initially, product concentration is calculated, if the input data is from substrate measurements. Substrate is calculated, if product data was provided. The calculation is based on the initial substrate concentration, which needs to be provided for all measurements. The parameter estimation is then based on substrate data.
In the background, rough estimates for k_{cat}, K_m, and K_i are calculated based on the fastest reaction rate in the provided dataset. Initial parameter estimates are needed as a starting point for the solver, whereas the parameter space is limited to ±1000-fold its estimated value for each parameter.

```
from EnzymePynetics.tools.parameterestimator import ParameterEstimator
from EnzymePynetics.core.enzymekineticsexperiment import EnzymeKineticsExperiment
import pyenzyme

# Define data
experimental_data = EnzymeKineticsExperiment(
    pH=7,
    reactant_name="test substance",
    ...)

enzymeml_document = pyenzyme.EnzymeMLDocument.fromFile("enzymeML_dataset.omex")

# Initialize the parameter estimator from an 'EnzymeKineticsExperiment' instance
estimator = ParameterEstimator(data=experimental_data)

# ... or directly from an EnzymeML document.
estimator = ParameterEstimator.from_EnzymeML(
    enzmldoc=enzymeml_document,
    reactant_id="s1",
    measured_species="product",
    inhibitor_id="s2")

# Fit experimental data to kinetic models and get the fit report of all models
estimator.fit_models()

# Visualize data with the best fitting model
estimator.visualize()
```

3.5 Model selection

3.6 Visualization

Akaike information criterion

Results

Scenario-driven workflow development

The development of the workflow was driven by different research scenarios of EnzymeML project partners. The goal of all project was to reliably estimate kinetic parameters of enzyme reactions. The collaboration consisted of multiple rounds of lab experiments and kinetic modeling of the respective data. Each round consisted of (i) wet lab experiments, (ii) kinetic modeling, (iii) design of follow-up experiments, and (iv) discussion of results with project partners. Hence, a short feedback loop between lab experiments and modeling-based experimental suggestions was established.
In parallel, the python modules CaliPytion and EnzymePynetics were developed to provide a workflow from analytical raw data to kinetic parameter estimates. Thereby, the individual requirements of each research

scenario, fostered the implementation of different features. Hence, a generic workflow for parameter estimation of enzyme kinetics was established. The developed workflow is schematically visualized in figure 1.

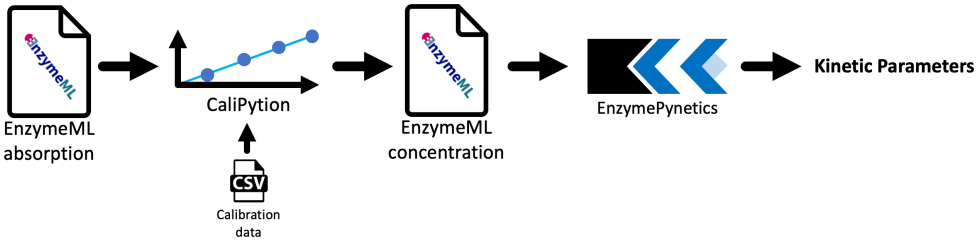


Fig. 1: Workflow for kinetic parameter estimation of enzyme reactions.

The workflow was designed around EnzymeML, whereas it's data model serves

- Data model vs. format
- FAIR

The following chapters show the developed workflow applied in different research scenarios. Each each of the following sections is an executable JupyterNotebook. Hence all figures for data visualization were generated at runtime of the analysis. Each notebook consists of a short description of the project's background and methodology carried out by the project partners in the wet lab. Additionally kinetic modeling steps as well as the results are shown. Lastly, project specific results are discussed.

Scenario A:
Chymotrypsin inhibiton by a *in silico* designed albumin fusion protein

Data provided by Marwa Mohamed (Institute of Cell Biology and Immunology, University of Stuttgart, Germany)

Project background

In this scenario, the binding of an *in silico* designed protein to an enzyme was assessed by determining the inhibitory constant K_i. Thereby, the efficiency of the newly developed approach for computational design of protein binders to deliberately chosen targets was demonstrated. This was done, by comparing K_i of the designed human serum albumin variant (HSA(M3)) to the wild-type (HSA(M3)) by respectively applying the proteins to chymotrypsin enzyme reactions. Both designed proteins were Both HSAs were individually dimerized into fusion proteins through a huFc.

Experimental design

Enzyme activity was monitored by measuring the product formation of p-Nitroanilin (p-NA) photometrically at 410 nm for 30 min at 30°C. Therefore, Succinyl-gly-gly-phe-p-nitroanilide (SGGPpNA) was applied as substrate in a concentration range of 0.25 - 2 mM. For concentration calculations, p-NA standard was prepared in the range of 0 - 0.3 mM in duplicates. K_i of HSA(WT)-huFc and HSA(M3)-huFc on chymotrypsin were investigated in independent experiments. Each experiment consisted of enzyme reactions with and without the respective HSA variant. Each enzyme reaction contained 0.2 µM of enzyme and 26.88 µM of the respective HSA variant, if inhibitor was applied. All enzyme reactions were prepared in duplicates.

Data management

Experimental data and meta data was filled in EnzymeML Excel templates for each of the two inhibition experiments respectively. Calibration data was stored as Excel files. Measurement data was already blanked.

Data preparation

Imports

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import pyenzyme as pe
import copy
import string
from IPython.display import display
from EnzymePynetics.tools.parameterestimator import ParameterEstimator
from CaliPytion.tools.standardcurve import StandardCurve

import warnings
warnings.filterwarnings('ignore')
```

Concentration calculation

Product standard data was imported directly from an Excel file. Then, a standard curve was created.

```
product_standard = StandardCurve.from_excel(
    path="../../data/chymotrypsin_inhibition/pNA-standard.xlsx",
    reactant_id="s1",
    wavelength=410,
    sheet_name="csv",
    concentration_unit="mmole / l",
    temperature=30,
    temperature_unit="C")

product_standard.visualize()
```

```
Calibration data was automatically blanked.
```

	AIC
Quadratic	-161
3rd polynominal	-161
Linear	-143
Rational	-141
Exponential	-17

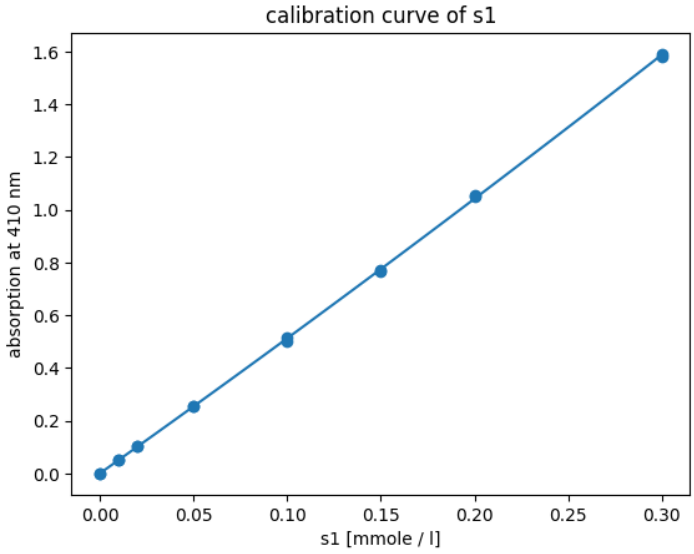


Fig. XXX: Fitted quadratic calibration model to standard data of p-NA.

Based on the Akaike information criterion (AIC), the relation between concentration and absorption is best described by a quadratic function. The calibration measurements and the fitted calibration model are shown in Fig. XXX.

Experimantal data

The EnzymeML documents of each experiment were loaded and the standard curve was applied to the absorption data to calculate concentrations.

```
# Load data from
chymo_HSAwt =
pe.EnzymeMLDocument.fromTemplate("../../data/chymotrypsin_inhibition/chymo_HSAwt.x
lsx")
chymo_HSAM3 =
pe.EnzymeMLDocument.fromTemplate("../../data/chymotrypsin_inhibition/chymo_HSA(M3)
.xlsx")

# Apply standard curve to 'EnzymeMLDocument'
chymo_HSAwt = product_standard.apply_to_EnzymeML(chymo_HSAwt, "s1")
chymo_HSAM3 = product_standard.apply_to_EnzymeML(chymo_HSAM3, "s1")
```

Comparability of the experiments

Since experimental data with HSA(WT)-huFc and HSA(M3)-huFc originate from independent experiments, the control reactions without the respective inhibitor were compared by performing a parameter estimation. Thereby, catalytic efficiency $\frac{k_{cat}}{K_m}$ was used to assess comparability between the data sets, since k_{cat} and K_m were highly correlated (corr > 0.98). High correlations between parameters, indicate that the parameters cannot be determined independently with certainty. In this case, the highest initial substrate concentration is presumably too low, compared to the true K_m of the enzyme under the given experimental conditions. However, higher substrate concentration were not applied for multiple reasons. On the one hand dimethyl sulfoxide (DMSO) was used as a co-solvent of the substrate, which inhibits enzyme activity Busby *et al.* [1999]. Hence, higher initial substrate concentrations would have led to higher enzyme inhibition, which would have distorted the assessment of K_i. On the other hand, high substrate viscosity denied the application of higher concentrations without sacrificing pipetting precision.

```
# Create copys of the data sets and delete measuremnts with inhibitor.
wt_control = copy.deepcopy(chymo_HSAwt)
del wt_control.measurement_dict["m4"]
del wt_control.measurement_dict["m5"]
del wt_control.measurement_dict["m6"]
del wt_control.measurement_dict["m7"]

m3_control = copy.deepcopy(chymo_HSAM3)
del m3_control.measurement_dict["m4"]
del m3_control.measurement_dict["m5"]
del m3_control.measurement_dict["m6"]
del m3_control.measurement_dict["m7"]

# Estimate kinetic parameters of the control reactions of the HSA wild-type data
set.
kinetics_wt_control = ParameterEstimator.from_EnzymeML(wt_control, "s1",
"product")
kinetics_wt_control.fit_models(stop_time_index=-1, display_output=False)
print("Kinetic parameters of HSA(wt) chymotrypsin control reactions:")
display(kinetics_wt_control.result_dict\
    .style.set_table_attributes('style="font-size: 12px"'))


# Estimate kinetic parameters of the control reactions of the HSA(M3) data set.
kinetics_m3_control = ParameterEstimator.from_EnzymeML(m3_control, "s1",
"product")
kinetics_m3_control.fit_models(stop_time_index=-1, display_output=False)
print("\nKinetic parameters of HSA(M3) chymotrypsin control reactions:")
display(kinetics_m3_control.result_dict\
    .style.set_table_attributes('style="font-size: 12px"'))


fig, axes = plt.subplots(1,2, figsize=(12.8,4.8), sharey=True, sharex=True)
for e, (doc, ax, title) in enumerate(zip([kinetics_wt_control
,kinetics_m3_control], axes.flatten(), ["chymotrypsin control reactions HSA(wt)",
"chymotrypsin control reactions HSA(M3)"])):
    doc.visualize(ax=ax, title=title)
    ax.set_ylabel("4-nitroanilin [mM]")
    ax.set_xlabel("time after reaction start [min]")
    ax.set_xticks([5, 10, 15, 20])
    ax.text(0, 1.1, string.ascii_uppercase[e], transform=ax.transAxes,
            size=20, weight='bold')

handles, labels = ax.get_legend_handles_labels()

fig.legend(handles, labels, loc="lower center", ncol=4, title="initial SGGPpNA
[mM]", bbox_to_anchor=(0.5,-0.15))
plt.tight_layout()
```

Kinetic parameters of HSA(wt) chymotrypsin control reactions:

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
irreversible Michaelis Menten	-1857	56.440 +/- 4.52%	2.226 +/- 7.28%	25.353 +/- 8.57%	-	-
competitive product inhibition	-1856	60.225 +/- 9.01%	2.342 +/- 9.88%	25.714 +/- 13.37%	0.725 +/- 112.65%	-
uncompetitive product inhibition	-1855	57.529 +/- 8.30%	2.276 +/- 10.65%	25.272 +/- 13.50%	-	3.175 +/- 428.92%
substrate inhibition	-1855	56.585 +/- 29.14%	2.233 +/- 35.51%	25.339 +/- 45.93%	-	997.715 +/- 12275.11%
non-competitive product inhibition	-1854	60.110 +/- 9.50%	2.331 +/- 10.00%	25.787 +/- 13.79%	0.702 +/- 117.81%	179.118 +/- 358.38%

Kinetic parameters of HSA(M3) chymotrypsin control reactions:

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
irreversible Michaelis Menten	-549	48.996 +/- 2.23%	1.978 +/- 3.74%	24.776 +/- 4.35%	-	-
competitive product inhibition	-549	51.227 +/- 4.39%	2.043 +/- 4.76%	25.077 +/- 6.47%	0.908 +/- 81.63%	-
uncompetitive product inhibition	-547	49.384 +/- 5.33%	1.996 +/- 6.71%	24.743 +/- 8.57%	-	7.445 +/- 631.41%
non-competitive product inhibition	-547	51.241 +/- 5.62%	2.044 +/- 6.62%	25.073 +/- 8.69%	0.914 +/- 83.56%	121.556 +/- 2156.85%
substrate inhibition	-545	50.368 +/- 3.78%	2.046 +/- 5.12%	24.623 +/- 6.36%	-	87.549 +/- 121.66%

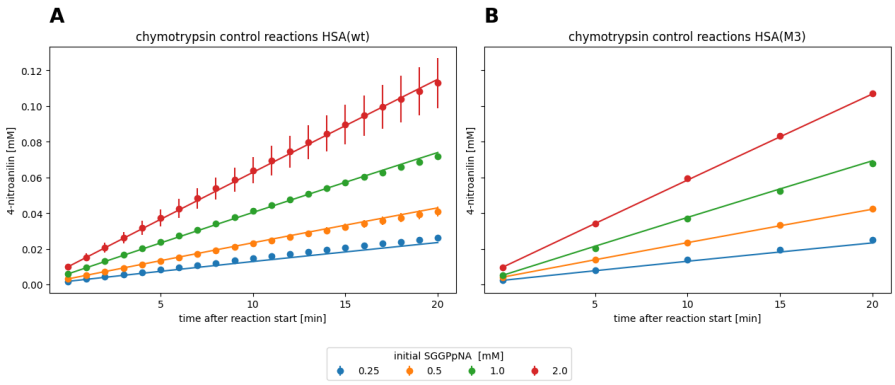


Fig. XXX: Measurement data and fitted irreversible Michaelis-Menten model for chymotrypsin reactions without HSA inhibitor.

The two dataset were fitted against kinetic models, which are displayed in the tables above. Each dataset is best described by the irreversible Michaelis-Menten model in therms of AIC and standard deviation on the estimated parameters. Models with product or substrate inhibition resulted in large uncertanties above 80 % on the parameter estimates. Therefore, irreversible Michaelis-Menten model was utilized for comparison of parameters. $\frac{k_{cat}}{K_m}$ was estimated to be 25.353 $min^{-1}mM^{-1}$ ± 8.57% for the control reaction of the HSA(WT)-huFc data set and 24.776 $min^{-1}mM^{-1}$ ± 4.35% for the HSA(M3)-huFc data set. As a result, the two experiments showed to be comparable, since the catalytic efficiency differs less than 3 % between the two data sets.

Determination and comparison of K_i

Experimental data of chymotrypsin inhibition by HSA(M3)-huFc contained negative absorption values for the first measurement point. Presumably sourcing from an incorrect blank measurement. Therefore, only measurement data from the second data point (minute 5 and onward) was considered for parameter estimation. Parameter estimates for all applied kinetic models are displayed in the output below.

```
# Parameter estimation for HSA(wt) data set
kinetics_HSAwt = ParameterEstimator.from_EnzymeML(chymo_HSAwt, reactant_id="s1",
inhibitor_id="s2", measured_species="product")
kinetics_HSAwt.fit_models(initial_substrate_concs=[0.25, 0.5, 1, 2],
stop_time_index=-1, start_time_index=5, display_output=False)
print("Kinetic parameters estimates for all models of chymotrypsin inhibition by
HSA(WT)-huFc:")
display(kinetics_HSAwt.result_dict.drop(columns=["kcat [1/min]", "Km [mmole /
l]"])\
    .style.set_table_attributes('style="font-size: 12px"'))

# Parameter estimation for HSA(M3) data set
kinetics_HSAM3 = ParameterEstimator.from_EnzymeML(chymo_HSAM3, reactant_id="s1",
inhibitor_id="s3", measured_species="product")
kinetics_HSAM3.fit_models(initial_substrate_concs=[0.25, 0.5, 1, 2],
stop_time_index=-1, start_time_index=1, display_output=False)
print("\nKinetic parameters estimates for all models of chymotrypsin inhibition by
HSA(M3)-huFc:")
display(kinetics_HSAM3.result_dict.drop(columns=["kcat [1/min]", "Km [mmole /
l]"])\
    .style.set_table_attributes('style="font-size: 12px"'))

# Visualize experimental data and fitted models
fig, axes = plt.subplots(1,2, figsize=(12.8,4.8), sharey=True, sharex=True)
for e, (doc, ax, title) in enumerate(zip([kinetics_HSAwt ,kinetics_HSAM3],
axes.flatten(), ["chymotrypsin inhibition by HSA(WT)-huFc", "chymotrypsin
inhibition by HSA(M3)-huFc"])):
    doc.visualize(ax=ax, title=title)
    ax.set_ylabel("4-nitroanilin [mM]")
    ax.set_xlabel("time after reaction start [min]")
    ax.set_xticks([5, 10, 15, 20])
    ax.text(0, 1.1, string.ascii_uppercase[e], transform=ax.transAxes,
            size=20, weight='bold')

handles, labels = ax.get_legend_handles_labels()

fig.legend(handles, labels, loc="lower center", ncol=2, title="initial SGGPpNA
[mM]", bbox_to_anchor=(0.5,-0.2))
plt.tight_layout()
```

```
Kinetic parameters estimates for all models of chymotrypsin inhibition by HSA(WT)-
huFc:
```

	AIC	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
competitive inhibition	-3100	22.253 +/- 6.97%	0.460 +/- 27.24%	-
non-competitive inhibition	-3098	22.403 +/- 7.00%	0.449 +/- 27.64%	79.193 +/- 516.52%
irreversible Michaelis Menten	-3088	21.622 +/- 7.15%	-	-
uncompetitive inhibition	-3087	21.631 +/- 7.63%	-	0.696 +/- 88.99%
partially competitive inhibition	-3084	21.643 +/- 8.23%	796.196 +/- 1075.02%	992.685 +/- 48.74%

```
Kinetic parameters estimates for all models of chymotrypsin inhibition by HSA(M3)-
huFc:
```

	AIC	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
competitive inhibition	-808	23.031 +/- 10.48%	0.059 +/- 8.51%	-
non-competitive inhibition	-806	23.002 +/- 10.89%	0.060 +/- 9.04%	44.965 +/- 302.77%
uncompetitive inhibition	-751	19.297 +/- 21.77%	-	0.035 +/- 21.33%
irreversible Michaelis Menten	-713	18.512 +/- 24.41%	-	-
partially competitive inhibition	-709	18.544 +/- 27.64%	569.354 +/- 57.96%	925.096 +/- 11733.22%

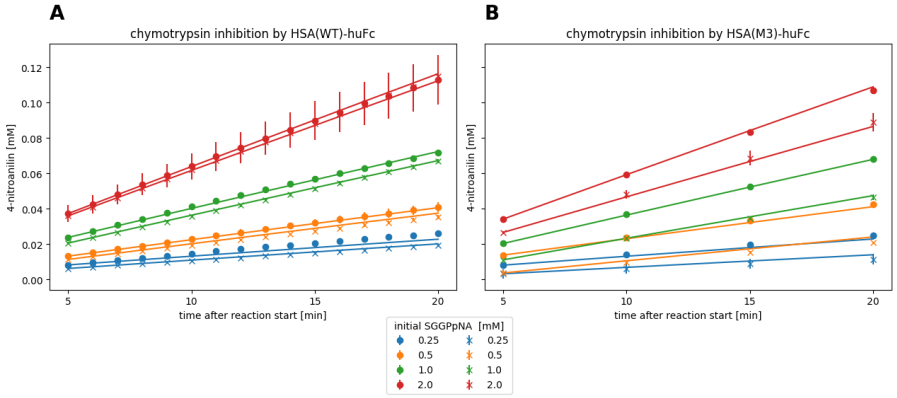


Fig. XXX: Measurement data and fitted product inhibition model for chymotrypsin reactions with respective HSA inhibitior.

Both reaction systems are best described by the competitive inhibition model, which is indicated by the lowest AIC and standard deviation on the estimated parameters. Thereby, a K_i of 0.460 mM ± 27.24% was estimated for HSA(WT)-huFc and 0.059 mM ± 8.51% for HSA(M3)-huFc. This resembles a roughly 7-fold increase in affinity of HSA(M3) to the enzyme compared to the HSA(wt). Since the competitive inhibition model describes the data the best, HSA(M3)-huFc presumably interacts with the enzyme in the active site region.

Scenario B:
α-glucosidase inhibition by fucoidan

Data provided by Chantal Daub (Biochemistry, Rhodes University, Makhanda, South Africa)

Project background

In this scenario, the inhibitory properties of fucoidan on α-glucosidase from *Saccharomyces cerevisiae* was investigated. Fucoidan is a sulfated polysaccharide found in various brown algae. The polysaccaride is investigated as a potential active compound in the fields of anit-cancer, anti-inflammation, and anti-coagulate research, among others (Li *et al.* [2008]). Recently, Daub *et al.* [2020] proposed the application of fucoidan as a drug for diabetes mellitus treatment, since fucoidan showed to effectively inhibit α-glucosidase. In the corresponding study, fucoidan from *E. maxima* showed an almost 2-fold lower IC_{50} value, compared to established diabetes drug acarbose.
In the following analysis, α-glucosidase was exposed to fucoidan from *Ecklonia maxima*, *Ecklonia radiata*, *Fucus vesiculosus*, and *Schimmelmannia elegans* to test their respective capability of α-glucosidase inhibiton by assessing their respective K_i values.

Experimental design

α-glucosidase reactions, catalyzing the hydrolysis of p-nitrophenyl glucopyranoside (p-NPG) to p-nitrophenol were conducted with and without fucoidan from each seaweed species as well as acarbose. Thereby, fucoidan was applied in two different concentrations. p-NPG was applied in a range from 0.1 mM to 5 mM, to enzyme reactions containing 9.19 µM α-glucosidase Product formation was recorded photometrically at 405 nm and 37°C for 20 min. Product concentrations were calculated utilizing a photometric p-NP standard. Additionally, control reaction without enzyme were prepared to subtract the absorption contribution of the respective inhibitor, buffer, enzyme and substrate from each measurement.

Data preparation

Imports and parser function

```
from typing import Dict
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors
import re
import os
import pyenzyme as pe
from IPython.display import display
from CaliPytion.tools.standardcurve import StandardCurve
from EnzymePynetics.tools.parameterestimator import ParameterEstimator


import warnings
warnings.filterwarnings('ignore')

colors = list(mcolors.TABLEAU_COLORS.values())

# Parser
def measurement_data_to_EnzymeML(
    template_path: str,
    measurement_data: np.ndarray,
    species_id: str,
    time: np.ndarray,
    data_unit: str,
    time_unit: str
    ) -> pe.EnzymeMLDocument:

    enzmldoc: pe.EnzymeMLDocument =
pe.EnzymeMLDocument.fromTemplate(template_path)

    for IDs, concentration in zip(enzmldoc.measurement_dict.keys(),
measurement_data):
        for counter, replicate in enumerate(concentration):

            enzmldoc.getMeasurement(IDs).addReplicates(pe.Replicate(
                id=f"Measurement{counter}",
                species_id=species_id,
                data=list(replicate),
                data_unit=data_unit,
                time=list(time),
                time_unit=time_unit), enzmldoc)

    return enzmldoc

# Ignore hidden files in file stystem
def listdir_nohidden(path):
    for f in os.listdir(path):
        if not f.startswith('.'):
            yield f
```

Measurement data was provided as an Excel file, whereas metadata was filled in EnzymeML Excel templates for each fucoidan seaweed species and acarbose respectively. In preliminary experiments, p-NPG showed to slightly absorb at the product detection wavelength. Therefore, the absorbance contribution of substrate at the product detection wavelength was subtracted as well as the the contributions of enzyme, buffer and inhibitor. Then, the blanked absorbance data was written to the EnzymeML documents by a parser function.

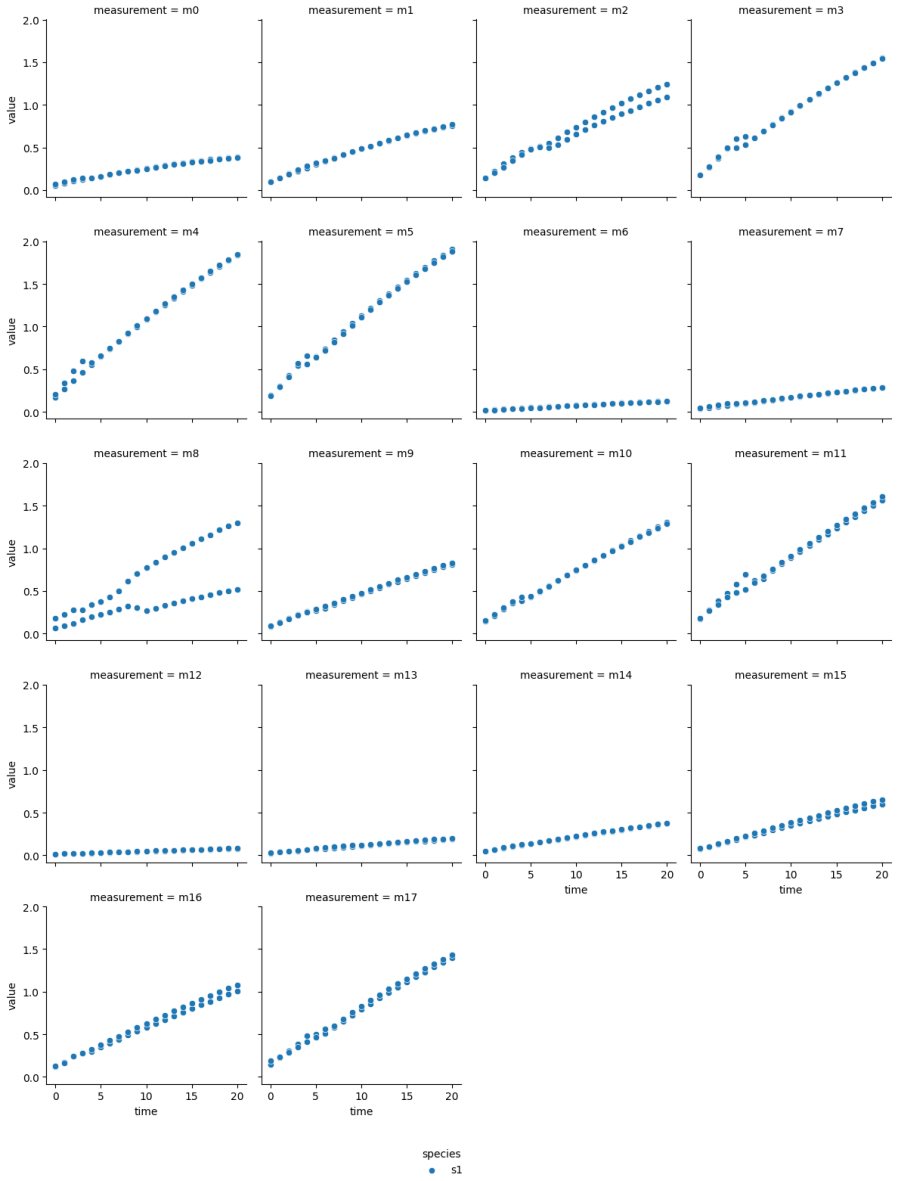
```
dataset_path = "../../data/glucosidase_inhibition/experimental_data_real.xlsx"
template_directory = "../../data/glucosidase_inhibition/EnzymeML_templates"

enzml_docs = []
# Load experimental data from Excel
excel_sheets = sorted(pd.ExcelFile(dataset_path).sheet_names)
inhibitors = excel_sheets[:-1]
substrate_controls = excel_sheets[-1]
initial_substrates = [0.1, 0.25, 0.5, 1, 2.5, 5] # mM

# Blank data
## Absorption contribution from substrate
substrate_absorption_data = pd.read_excel(dataset_path,
sheet_name=substrate_controls).set_index("time")
buffer_enzyme_absorption = np.mean(substrate_absorption_data.iloc[:,0])
substrate_absorptions =
substrate_absorption_data.subtract(buffer_enzyme_absorption).drop(columns=
["Buffer+ Enzyme"])
substrate_absorptions = substrate_absorptions.values.T.reshape(2,6,21)
substrate_absorptions = np.mean(substrate_absorptions, axis=0)
substrate_absorptions = np.mean(substrate_absorptions, axis=1)
mapper_substrate_enzyme_absorption = dict(zip(initial_substrates,
substrate_absorptions))

for inhibitor, template in zip(sorted(inhibitors),
sorted(listdir_nohidden(template_directory))):
    df = pd.read_excel(dataset_path, sheet_name=inhibitor).set_index("time")
    time = df.index.values
    inhibitor_controls = df.iloc[:,:4]
    inhibitor_concs = np.unique([float(conc.split(" ")[-2]) for conc in
inhibitor_controls.columns])
    inhibitor_absorptions = inhibitor_controls.values.T.reshape(2,2,21)
    inhibitor_absorptions = np.mean(inhibitor_absorptions, axis=1)
    inhibitor_absorptions = np.mean(inhibitor_absorptions, axis=1)
    mapper_inhibitor_absorption = dict(zip(inhibitor_concs,
inhibitor_absorptions))
    mapper_inhibitor_absorption[0.0] = buffer_enzyme_absorption
    df = df.iloc[:,4:]
    for column in df.columns:
        init_substrate = float(column.split(" ")[4])
        inhibitor_conc = float(column.split(" ")[1])
        df[column] = df[column] -
mapper_substrate_enzyme_absorption[init_substrate]
        df[column] = df[column] - mapper_inhibitor_absorption[inhibitor_conc]

    data = df.values.T.reshape(3,2,6,21)
    data = np.moveaxis(data,1,2).reshape(18,2,21)

    # Parse measurement data to EnzymeML documents
    enzml_docs.append(measurement_data_to_EnzymeML(
        template_path=f"{template_directory}//{template}",
        measurement_data=data,
        time=time,
        species_id="s1",
        data_unit="mmole / l",
        time_unit="min"
    ))
```

Data quality

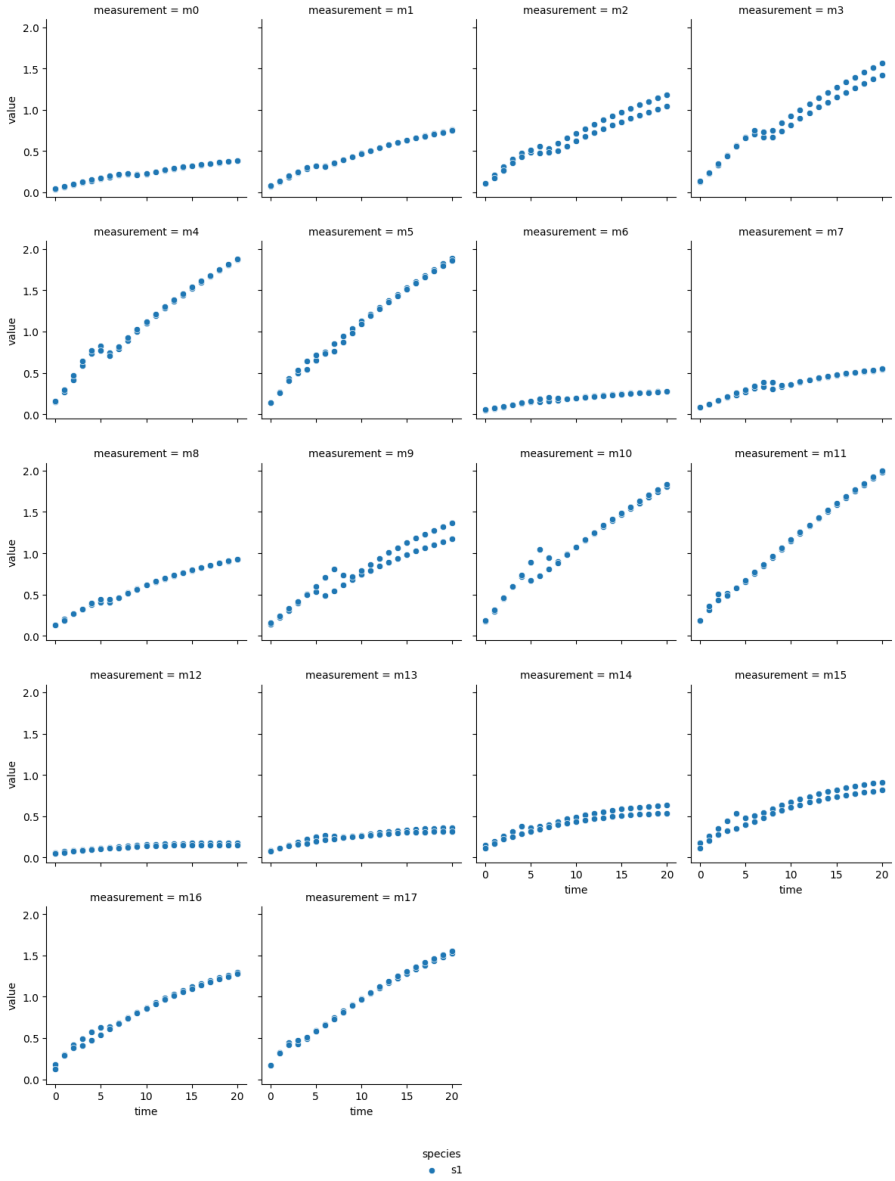
In the next cell, the blanked absorption data of each EnzymeML document is visualized with the `.visualize()`-method of `PyEnzyme` for quality control.

```
for doc in enzml_docs:
    print(doc.name)
    doc.visualize()
    plt.show()
```

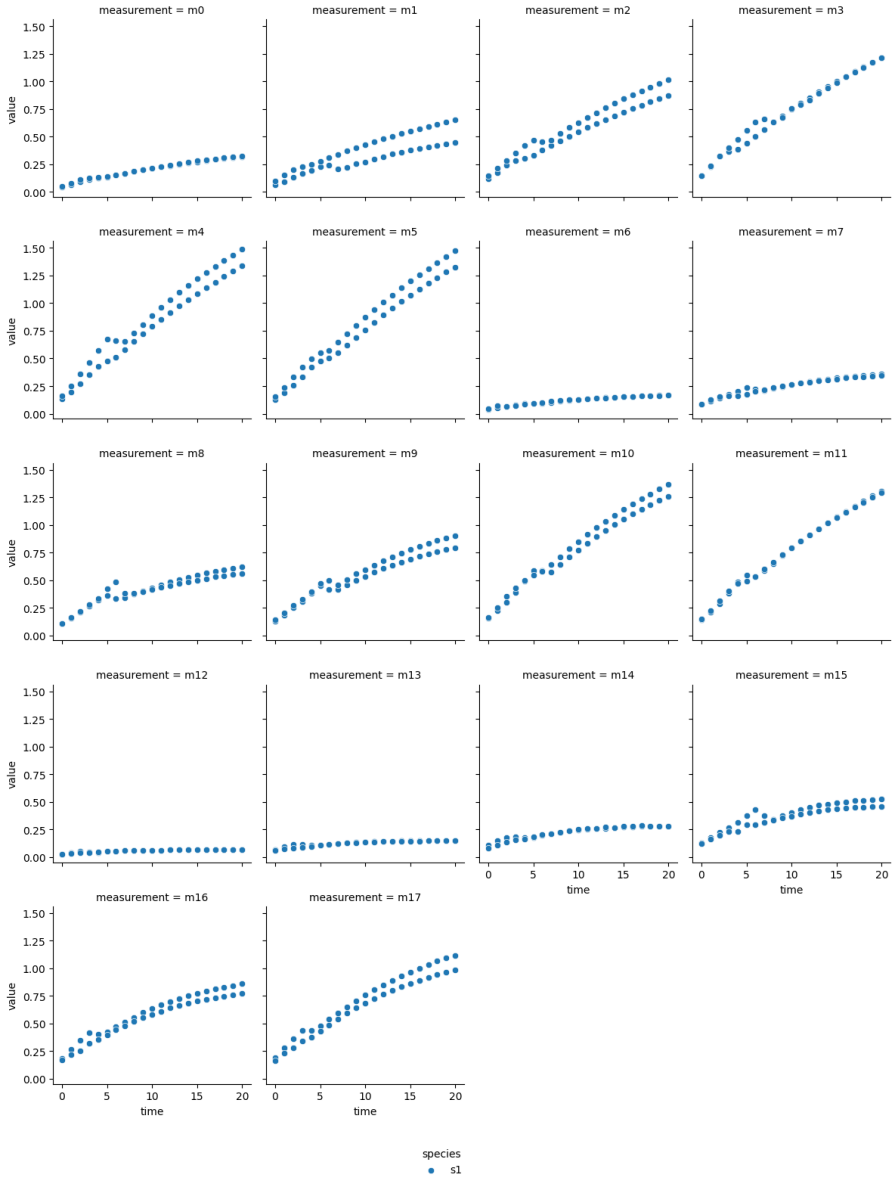
a-glucosidase inhibition by acarbose



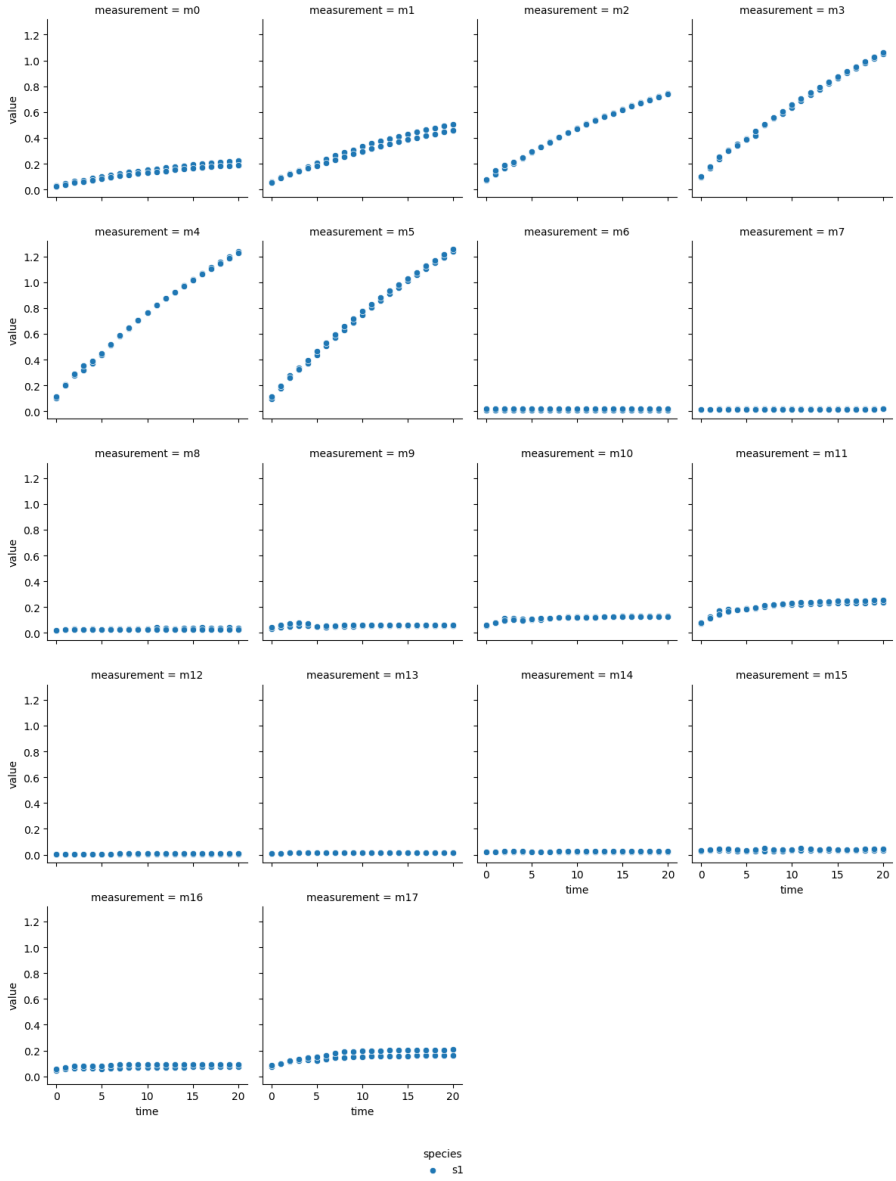
a-glucosidase inhibition by fucoidan from E. maxima



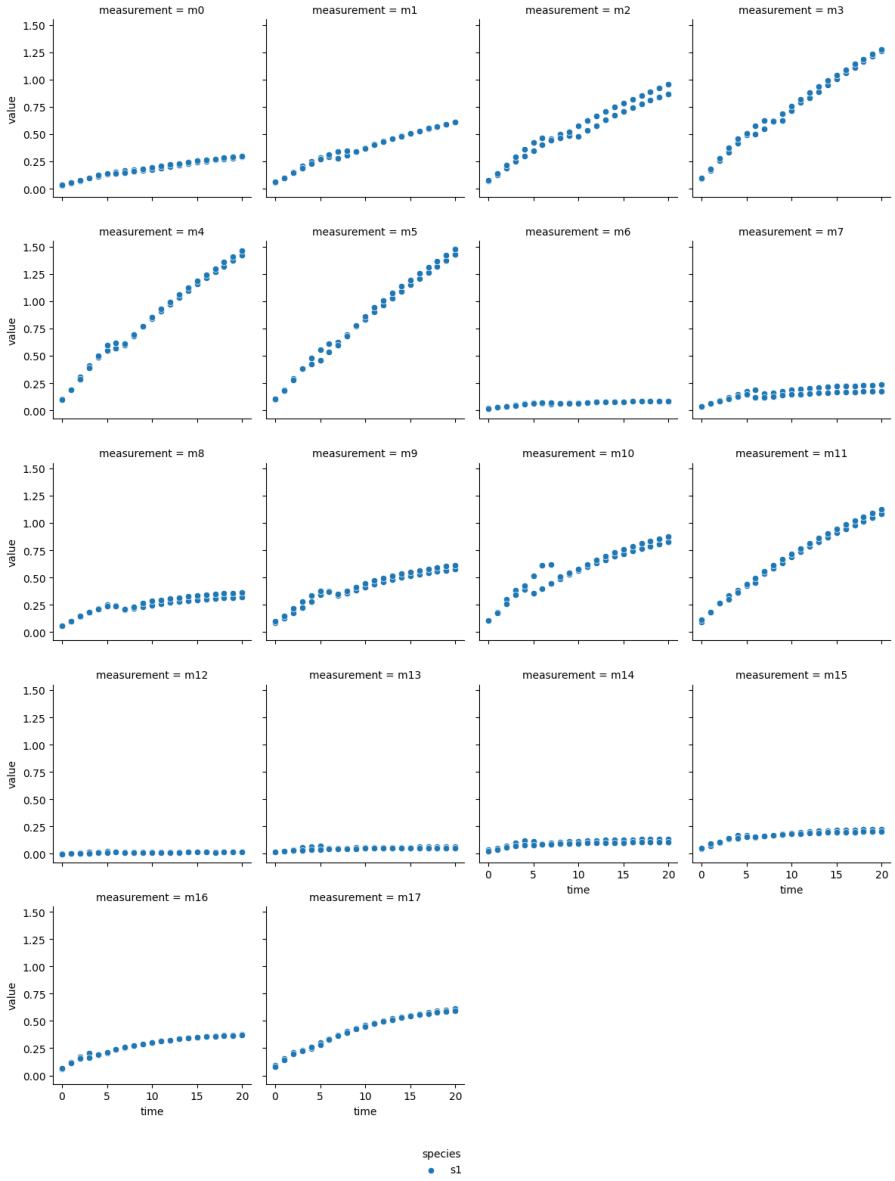
a-glucosidase inhibition by fucoidan from E. radiata



a-glucosidase inhibition by fucoidan from F. vesiculosus



a-glucosidase inhibition by fucoidan from S. elegans



The technical output of the cell above visualizes the blanked product absorbance data of each dataset. Therein, the individual measurements are labeled from m0 - m17, which represent individual experimental conditions. Thereby, measurements m0 - m5 are from reactions without inhibitor, m6 - m11 reactions with the lower inhibitor concentration, and m12 - m17 originate from reactions with the higher inhibitor concentration. Each subplot contains the data of two experimental repeats.
In most reactions a decrease followed by an increase of reaction rate is visible around minute 5. Since each dataset originates from a continuous photometric measurement carried out on a single MTP, the observed behavior likely sources from an analytical device malfunction. Additionally, one repeat of measurement ‘m8’ from the acarbose dataset shows low absorption, which is inconsistent with comparable measurements. Therefore, ‘m8’ was excluded from the dataset.

```
del enzml_docs[0].measurement_dict["m8"].getReactant("s1").replicates[0]
```

Concentration calculation

Standard data of p-NP was loaded from an excel file, and a standard curve was created. Then, the standard curve was applied to the EnzymeML documents.

```
path_calibration_data = "../../data/glucosidase_inhibition/p-NP_standard.xlsx"

product_standard = StandardCurve.from_excel(
    path=path_calibration_data,
    reactant_id="s1",
    sheet_name="csv",
    wavelength=405,
    concentration_unit = "mmole / l",
    cutoff_absorption=2)

product_standard.visualize()

# Apply calibration curves to absorption EnzymeML documents
for enzmldoc in enzml_docs:
    product_standard.apply_to_EnzymeML(enzmldoc, "s1")
```

```
Calibration data was automatically blanked.
```

	AIC
3rd polynominal	-135
Quadratic	-135
Rational	-135
Linear	-125
Exponential	-11

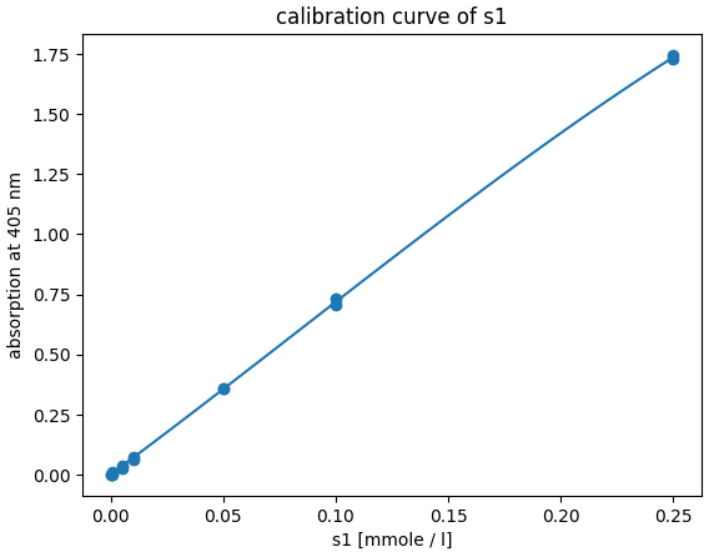


Fig. XXX: Fitted 3rd-degree polynomial calibration model to standard data of p-NP (s1).

Based on AIC, 3rd-degree polynomial, quadratic and rational calibration models describe the relation between absorption to concentration ration of p-NP equally. 3rd-degree polynomial model was used to calculate p-NP concentrations.

Parameter estimation

Due to the mentioned issue with the measurement data after 5 minutes, kinetic parameters were estimated based on data of the initial 2 minutes only. Thus, different inhibition models cannot be compared, since information provided by the initial slopes is sufficient to distinguish kinetic inhibition mechanisms by modeling. In consequence, kinetic parameters were estimated based on initial rates, assuming competitive inhibition.
Additionally, the faulty measurement with an initial substrate concentration of 0.5 mM from the 'a-glucosidase inhibition by acarbose' data set was excluded from parameter estimation. Based on an initial parameter estimation run with subsequent visual analysis, severals measurements from the *E. maxima* dataset were excluded. Thereby, measurements from both applied inhibitor concentrations and an initial substrate concentration of 2.5 mM and 5 mM showed similar activity compared to reactions without inhibitor. In contrast, the reactions with inhibitor present showed inhibition, according to the difference in slope.

```
# Discard measurement with 2.5 mM and 5 mM and lower inhibition concentration from
E. maxima dataset
del enzml_docs[1].measurement_dict["m10"]
del enzml_docs[1].measurement_dict["m11"]
del enzml_docs[1].measurement_dict["m16"]
del enzml_docs[1].measurement_dict["m17"]
```

```
# Run parameter estimation for all data sets
results=[]
for enzmldoc in enzml_docs:
    result = ParameterEstimator.from_EnzymeML(enzmldoc=enzmldoc, reactant_id="s1",
inhibitor_id="s2", measured_species="product")
    result.fit_models(enzyme_inactivation=False, display_output=False,
stop_time_index=3)
    results.append(result)

# Display results for competitive inhibition model
labels = np.array(["Acarbose", "Fucoidan (E. maxima)", "Fucoidan (E. radiata)",
"Fucoidan (F. vesiculosus)", "Fucoidan (S. elegans)"])
df = pd.concat([result.result_dict.loc["competitive inhibition"].to_frame().T for
result in results],\
    keys=labels)\
        .reset_index()\
        .drop(columns=["level_1", "AIC", "Ki uncompetitive [g / l]"])\
        .set_index("level_0")\
        .rename_axis("Inhibitor", axis = 0)
print("Kinetic parameters based on competitive inhibition model:")
display(df.style.set_table_attributes('style="font-size: 12px"'))
```

```
Kinetic parameters based on competitive inhibition model:
```

Inhibitor	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [g / l]
Acarbose	1.886 +/- 2.71%	0.326 +/- 10.73%	5.779 +/- 11.06%	0.112 +/- 10.77%
Fucoidan (E. maxima)	2.305 +/- 2.09%	0.390 +/- 7.56%	5.910 +/- 7.84%	0.858 +/- 13.98%
Fucoidan (E. radiata)	1.347 +/- 2.69%	0.173 +/- 14.49%	7.765 +/- 14.74%	0.028 +/- 19.09%
Fucoidan (F. vesiculosus)	1.482 +/- 2.02%	0.405 +/- 7.28%	3.656 +/- 7.55%	0.003 +/- 7.75%
Fucoidan (S. elegans)	1.487 +/- 2.39%	0.243 +/- 10.73%	6.115 +/- 10.99%	0.016 +/- 11.83%

The output above displays estimated kinetic parameters for all data sets, assuming competitive inhibition between enzyme and inhibitor. As a result k_{cat} was estimated between 2.305 min^{-1} ± 2.09% and 1.347 min^{-1} ± 2.69%, whereas K_m was estimated between 0.405 mM ± 7.28% and 0.173 mM ± 14.49%. Since all measurements were conducted under equal conditions, k_{cat} and K_m should be similar across experiments. Deviations in parameter values might therefore result from pipetting or deviations in reaction temperature.

```
# Get K_i parameter results
def get_parameter(result, model_name: str, parameter: str):
    return result.models[model_name].result.params[parameter]

ki = []
ki_stderr = []
for result in results:
    ki.append(get_parameter(result, "competitive inhibition", "K_ic").value)
    ki_stderr.append(get_parameter(result, "competitive inhibition",
"K_ic").stderr)
ki = np.array(ki)
ki_stderr = np.array(ki_stderr)

# Visualize
kis = dict(zip(labels,ki))
f, (a0, a1) = plt.subplots(1, 2, gridspec_kw={'width_ratios': [3,1.4]})
a0.bar(labels[[0,2,3,4]], ki[[0,2,3,4]], yerr=ki_stderr[[0,2,3,4]],
align='center', capsize=3)
a0.set_xticklabels(labels[[0,2,3,4]], rotation = 90)
a1.bar(labels[:2], ki[:2], yerr=ki_stderr[:2], align='center', capsize=3)
a1.set_xticklabels(labels[:2], rotation = 90)
a0.set_ylabel("$K_{i}$ [$mg$ $ml^{-1}$]")
plt.tight_layout()
```

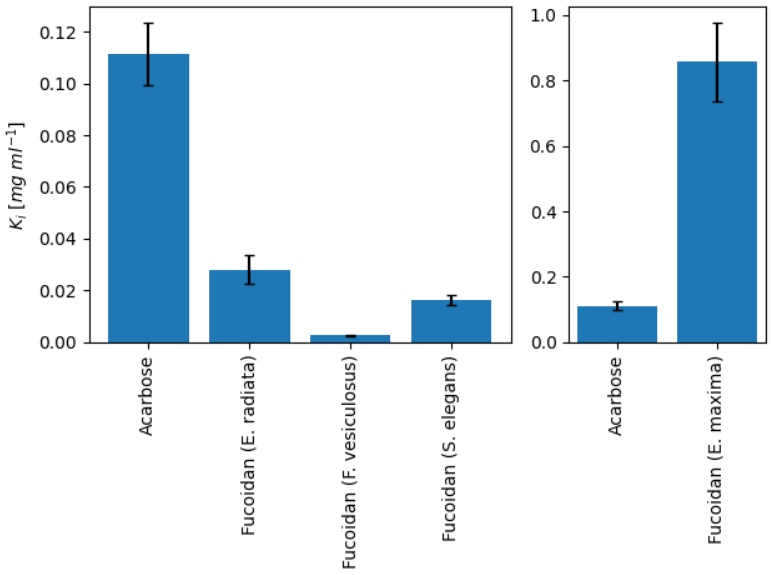


Fig. XXX: Estimated inhibitory constants for α-glucosidase inhibition by fucoidan from different algae species and acarbose

Fucoidan from *E. radiata*, *F. vesiculosus*, and *S. elegans* all showed an lower K_i (28, 3, and 16 µg ml^{-1} respectively) compared to the acarbose reference (112 µg ml^{-1}). This resembles a 37-fold higher binding affinity of fucoidan from *F. vesiculosus* to α-glucosidase compared to acarbose. Opposing to the previous study, fucoidan from *E. maxima* showed a more than 7-fold higher K_i (858 µg ml^{-1}) compared to acarbose. This likely originates from inconsistencies during experimental preparation. Additionally, 4 of 10 measurements with applied inhibitor were excluded from parameter estimation, since they showed higher activity compared to the respective reactions without inhibitor. Therefore, this measurement should be repeated.

In general, the estimated parameters are not reliable on a quantitative level, since the malfunction of the analytical device undermines the viability of the whole dataset. Therefore, it cannot be ruled out, that some of the calculated concentrations deviate from the real concentrations. Furthermore, all estimated parameters are only estimated based on three time points of measure. In addition each experimental condition was applied as duplicate repeats, which additionally showed deviations. In conclusion the estimated parameters qualitatively confirm that fucoidan from different host organism, inhibits α-glucosidase more strongly, compared to acarbose.

Scenario C:
SLAC characterization

Data provided by Alaric Prins (Biocatalysis and Technical Biology, Cape Peninsula University of Technology, Capetown, South Africa)

Project background

Laccases find industral application in pulp and paper industry. Thereby, the enzyme is used for its polymerization or depolymerization capabilities through oxidation of phenoclic compounds [Widsten and Kandelbauer, 2008]. Something about SLAC...

Experimental design

In this scenario, the catalytic properties of the small laccase from *Streptomyces coelicolor* (SLAC) were investigated. Therefore, the enzymatic oxidation of 2,2′-Azino-bis(3-ethylbenzothiazoline-6-sulfonic acid) (ABTS) to its radical form $ABTS^{+\cdot}$ was studied in the pH range of pH 3 - pH 5.5 and temperature range of 25°C - 45°C. In total 30 kinetic enzyme assays in a substrate range between 0 - 200 uM of ABTS were conducted. Additionally, for each enzyme reaction with a given initial substrate concentration, a control reaction without enzyme was prepared.
For each pH - temperature condition an individual ABTS standard curve and absorption spectrum was recorded to account for varying ABTS absorption properties due to reaction conditions. Each enzyme reaction was followed for 15 min photometrically at two wavelengths, measuring substrate depletion and

product accumulation simultaneously. Preliminary experiments confirmed, that ABTS absorbs at 340 nm, whereas the $ABTS^{+\cdot}$ absorbs at 420 nm in contrast to the substrate. Furthermore, cross absorbance of product at the substrate detection wavelength and vice versa was ruled out.

Data management

Overall, the dataset consists of more than 100 000 individual absorbance reads. Thus, data preparation was automated by custom parser functions, which were tailored to the output of the used spectrophotometer. Consequently, the tedious and error-prone manual copying of raw data was avoided. Information on the involved reactants, and the enzyme was filled in an EnzymeML Excel spreadsheet, which served as a meta data container. All other information was parsed from the output of the spectrophotometer. Figure 1 illustrates the schematic data flow of this project.

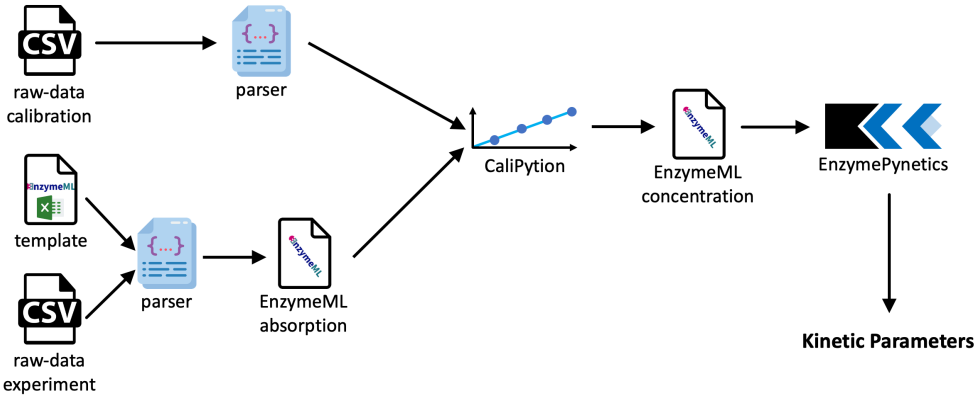


Fig. 1: Schematic data pipeline of the SLAC characterization.

Data preparation

Imports

```
from typing import Dict, List
import pyenzyme as pe
import numpy as np
import pandas as pd
import os
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import copy
import string
from lmfit import Parameters, minimize
from scipy.stats import linregress, pearsonr
from joblib import Parallel, delayed
from CaliPytion.tools.standardcurve import StandardCurve
from EnzymePynetics.tools.parameterestimator import ParameterEstimator

# Custom functions for data mapping
from parser_functions import measurement_data_to_EnzymeML, plot
from parser_functions import read_measurement_data, read_calibration_data

import warnings
warnings.filterwarnings('ignore')
```

Experimantal data

Data from SLAC reactions was loaded from the output files of the photometer and written to individual EnzymeML documents. Then, information of the control reactions was used to subtract the absorption contribution from enzyme and buffer from the substrate and product signal.

```
# Specify the location of the data sets
directory_measurement_data =
"../../data/SLAC_kinetic_characterization/TimeCourseData"
directory_standard_data = "../../data/SLAC_kinetic_characterization/StandardData"
directory_spectrum_data = "../../data/SLAC_kinetic_characterization/SpectrumData"
path_EnzymeML_templates =
"../../data/SLAC_kinetic_characterization/EnzymeML_templates"

# Define IDs for species, listed in the EnzymeML Excel template
substrate_id = "s0"
product_id = "s1"
substrate_control_id = "s2"
product_control_id = "s3"
species_ids = [substrate_id, product_id, substrate_control_id, product_control_id]


# Parse measurement data from photometer output
raw_data_dict = {}
for path in os.listdir(directory_measurement_data):
    data = read_measurement_data(f"{directory_measurement_data}/{path}")
    pH = data["pH"]
    temp = data["temperature"]
    raw_data_dict[f"{pH} {temp}"] = data

EnzymeML_template_dict = {
    3.0: "EnzymeML_SLAC_pH3.xlsm",
    3.5: "EnzymeML_SLAC_pH3_5.xlsm",
    4.0: "EnzymeML_SLAC_pH4.xlsm",
    4.5: "EnzymeML_SLAC_pH4_5.xlsm",
    5.0: "EnzymeML_SLAC_pH5.xlsm",
    5.5: "EnzymeML_SLAC_pH5_5.xlsm",
}

# Write absorption data to EnzymeMLDocuments
absortion_enzymemldocs: List[pe.EnzymeMLDocument] = []
for name, data in raw_data_dict.items():
    pH = data["pH"]
    absortion_enzymemldocs.append(measurement_data_to_EnzymeML(
        template_path=f"{path_EnzymeML_templates}/{EnzymeML_template_dict[pH]}",
        measurement_data=data,
        species_ids=species_ids,
        data_unit="umole / l",
        time_unit="s"))

# Sort documents by ascending pH and temperature
absortion_enzymemldocs = sorted(absortion_enzymemldocs, key=lambda x:
(x.getReaction("r0").ph, x.getReaction("r0").temperature))

# Blanc measurement data
for enzmldoc in absortion_enzymemldocs:
    blanc_measurement =
enzmldoc.measurement_dict["m0"].getReactant("s0").replicates
    blanc = np.mean([repeat.data for repeat in blanc_measurement])

    for id, measurement in enzmldoc.measurement_dict.items():
        for rep, replicate in enumerate(measurement.getReactant("s0").replicates):
            blanced_data = [value - blanc for value in replicate.data]
            enzmldoc.measurement_dict[id].getReactant("s0").replicates[rep].data =
blanced_data
        for rep, replicate in enumerate(measurement.getReactant("s2").replicates):
            blanced_data = [value - blanc for value in replicate.data]
            enzmldoc.measurement_dict[id].getReactant("s2").replicates[rep].data =
blanced_data

        for rep, replicate in enumerate(measurement.getReactant("s1").replicates):
            blanced_data = [value - blanc for value in replicate.data]
            enzmldoc.measurement_dict[id].getReactant("s1").replicates[rep].data =
blanced_data
        for rep, replicate in enumerate(measurement.getReactant("s3").replicates):
            blanced_data = [value - blanc for value in replicate.data]
            enzmldoc.measurement_dict[id].getReactant("s3").replicates[rep].data =
blanced_data

    # Delete control measurement 'm0'
    del enzmldoc.measurement_dict["m0"]
```

Concentration calculation

Calibration data was loaded and converted into individual instances of the calibration data model. Some meta data of the calibration needed to be provided to the custom `read_calibration_data` function, since the output of the used spectrophotometer only contained a minimum of information. Thereafter, a StandardCurve was created for each calibration data set. Thereby, only absoption values below 3.2 were considered, since higher absorption values could not be converted into concentration accurately. Lastly, the fit of each standard curves was visualized.

```
# Load calibration raw data
calibration_data = []
standard_directory = np.sort(os.listdir(directory_standard_data))
spectrum_directory = np.sort(os.listdir(directory_spectrum_data))

for standard, spectrum in zip(standard_directory, spectrum_directory):
    standard = f"{directory_standard_data}/{standard}"
    spectrum = f"{directory_spectrum_data}/{spectrum}"
    result = read_calibration_data(
        path_standard=standard,
        path_spectrum=spectrum,
        species_id=substrate_id,
        wavelengths=[340, 420],
        concentrations=[0,5,10,15,25,50,75,100,125,150,175,200],
        concentration_unit="umole / l",
        device_manufacturer="MANUFACTURER",
        device_model="SUPERMODEL",
        spectrum_reactant_concentration=69
    )
    for pH in result.keys():
        calibration_data.append(result[pH])

# Sort calibration data by ascending pH and temperature
calibration_data = sorted(calibration_data, key = lambda x: (x.pH, x.temperature))

# Generate standard curves for ABTS calibration data
standard_curves: List[StandardCurve] = []
for calibration in list(calibration_data):
    standard_curves.append(StandardCurve(calibration_data=calibration,
wavelength=340, cutoff_absorption=3.2, show_output=False))

# Sort standard curves by ascending pH and temperature.
standard_curves = sorted(standard_curves, key = lambda x: (x.calibration_data.pH,
x.calibration_data.temperature))

# Visualize all fitted standard curves
fig, axes = plt.subplots(6,5, figsize=(10, 13), sharey=True, sharex=True)
for i, (standard, ax) in enumerate(zip(standard_curves[:30], axes.flatten())):
    if not i%5:
        ax.set_ylabel("ABTS [uM]")
    standard.visualize(ax=ax)
    ax.set_title(f"pH {standard.calibration_data.pH},
{standard.calibration_data.temperature}°C")
    if i in [25,26,27,28,29]:
        ax.set_xlabel("time [s]")
plt.tight_layout()
```

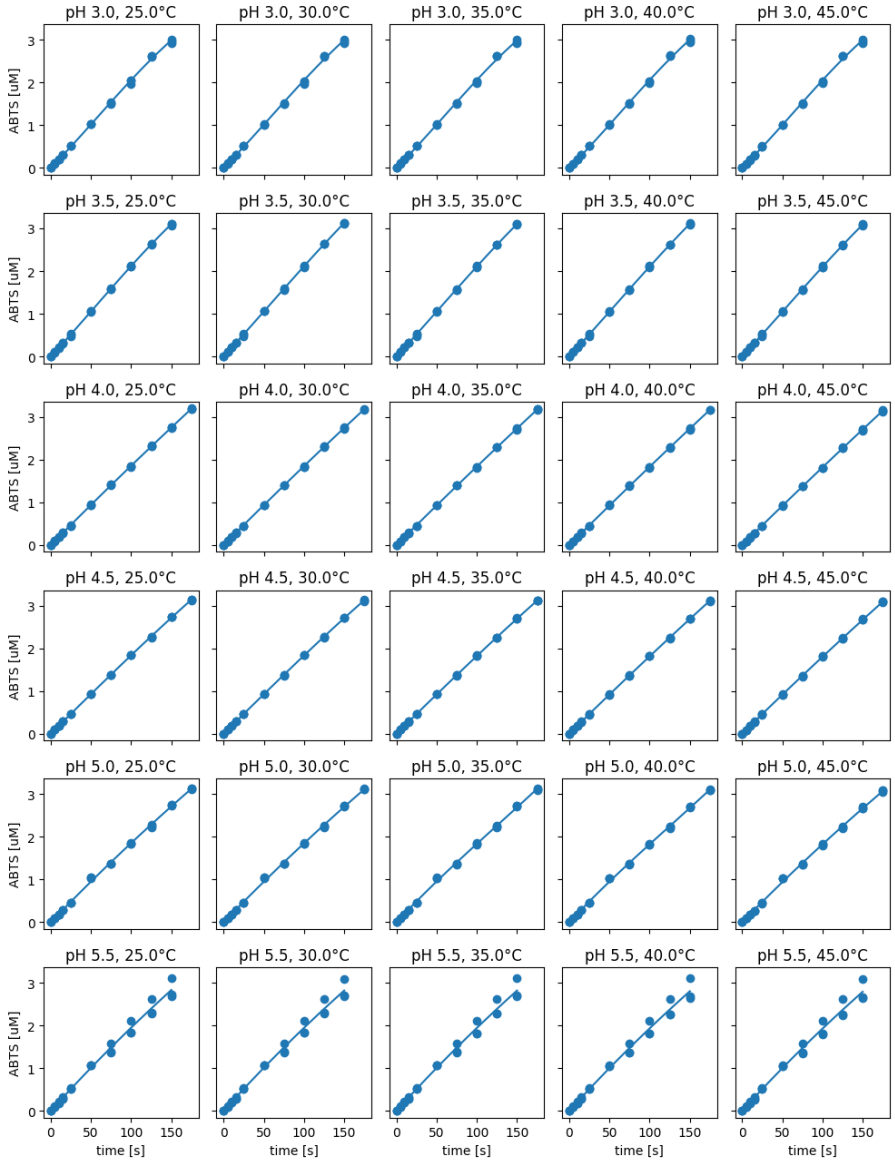


Fig. XXX: ABTS standard curves for different experimental condition.

Since the absorption characteristics of ABTS change with pH, the calibration range differs between pH values, due to the upper absorption limit (Fig. XXX). In consequence, the upper calibration limit for reactions at pH 3, pH 3.5, and pH 5.5 is at 150 uM of ABTS, whereas for all other pH values the upper limit is at 175 uM. This might source from the protonation state of ABTS, since the sulfonate groups of ABTS are deprotonated for less acidic pH values. Therefore, the absorption properties of ABTS can decrease. Calibration curve data at pH 5.5 showed larger variation between the repeats. In this case, pipetting of one of the the three repeats differs from the other two, which should be considered for kinetic parameter estimation. In contrast to pH, the temperature during calibration affected the calibration curve only marginally.

The generate standard curves were used to convert the absorption measurement data into concentration data. Thereby, the respective concentration values were only calculated, if the measured absorption was within the respective calibration bounds to avoid extrapolation.

```python
# Calculate concentrations by applying standard curves to 'EnzymeMLDocuments'.
concentration_enzymemldocs = []
for standard_curve, abso_enzmldoc in zip(standard_curves, absortion_enzymemldocs):

    # Validate, that measurement data is paired with respective standard curve
    assert standard_curve.calibration_data.pH ==
abso_enzmldoc.getReaction("r0").ph
    assert standard_curve.calibration_data.temperature ==
abso_enzmldoc.getReaction("r0").temperature

    enzmldoc = copy.deepcopy(abso_enzmldoc)
    conc_enzmldoc = standard_curve.apply_to_EnzymeML(enzmldoc, "s0",
ommit_nan_measurements=True)
    conc_enzmldoc = standard_curve.apply_to_EnzymeML(enzmldoc, "s2",
ommit_nan_measurements=True)

    concentration_enzymemldocs.append(conc_enzmldoc)
```

```
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m8']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m8']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m8']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m8']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m8']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
Measurements '['m9']' removed from document, since respective measurement values
are out of calibration range.
```

Quality control through mass balance analysis

Since product and substrate of the SLAC reaction were simultaneously recorded, mass balance analysis was conducted as a control of quality. By assuming mass conservation, the following concentration balance can be established:

$$0 = S_{(t)} + P_{(t)} - S_0 \tag{11}$$

Thereby, S_0 denotes the initial substrate concentration, whereas $S_{(t)}$ and $P_{(t)}$ describe the substrate and product concentration for each time point t. S_t and S_0 were individually measured. Thus each enzyme reaction with a given initial substrate concentration had a control reaction with identical substrate concentration. In contrast to the substrate, no calibration standard was available for the product. Therefore, an additional parameter §k§ was introduced to the mass balance equation, assuming linear relationship between the product concentration and its signal:

$$0 = S_{(t)} + P_{(t)}k - S_0 \tag{12}$$

k was determined for each data set individually by a minimization algorithm. The minimization objective was to find the optimal k, which minimizes all slopes of an experiment. Minimal slope of each reaction time-course was chosen as the target function, since the slopes should be zero under the assumption of mass conservation. Mass balances of all measurements are visualized in Fig. XXX.

```
# Defenition of parameter 'k'
params = Parameters()
params.add("k", value=30, min=0, max=200)

# Target function for the minimizer
def residual(params, x):
    k=params["k"]

    substrate, product,  control = x
    slopes = substrate[:,0] * 0.0
    for i, (s, p, c) in enumerate(zip(substrate, product, control)):
        sub = np.mean(s, axis=0)
        prod = np.mean(p, axis=0)
        cont = np.mean(c, axis=0)

        model = sub + prod*k - cont

        slopes[i] = linregress(np.arange(len(model)), model)[0]

    return slopes.flatten()


f = []
fig, axes = plt.subplots(6,5, figsize=(12.5, 15), sharey=True, sharex=True)
for e, (doc, ax) in enumerate(zip(concentration_enzymemldocs, axes.flatten())):
    substrate = []
    product = []
    initial_substrate = []
    control = []
    for measurement in doc.measurement_dict.values():
        initial_substrate.append(measurement.getReactant(substrate_id).init_conc)
        for replicate in measurement.getReactant(substrate_id).replicates:
            substrate.append(replicate.data)
        for replicate in measurement.getReactant(product_id).replicates:
            product.append(replicate.data)
        for replicate in measurement.getReactant("s2").replicates:
            control.append(replicate.data)

    time = np.array(replicate.time)

    product = np.array(product).reshape(int(len((product))/3),3,11)
    substrate = np.array(substrate).reshape(int(len((substrate))/3),3,11)
    control = np.array(control).reshape(int(len((control))/3),3,11)
    initial_substrate = np.array(initial_substrate)

    x = (substrate, product, control)

    result = minimize(residual, params, args=(x,))

    factor = result.params["k"].value
    f.append(factor)

    for i, (s, p, c, l) in enumerate(zip(substrate, product,  control,
initial_substrate)):
        init = np.mean(c, axis=0)
        sub = np.mean(s, axis=0)
        prod = np.mean(p, axis=0)
        cont = np.mean(c, axis=0)
        balance = (sub + prod*factor - cont)

        ax.plot(doc.getMeasurement("m1").getReactant("s0").replicates[0].time,
balance, label = l)
        if not i%5:
            ax.set_ylabel("mass balance [uM]")
        if i in [24, 25, 26, 27, 28, 29]:
            ax.set_xlabel("time [s]")
        pH = doc.getReaction("r0").ph
        temp = doc.getReaction("r0").temperature
        ax.set_title(f"pH {pH}, {temp}°C")#, factor: {factor:.0f}")
    if e == 0:
        handles, labels = ax.get_legend_handles_labels()
        fig.legend(handles, labels, loc="lower center", ncol=len(labels),
title="initial ABTS [uM]", bbox_to_anchor=(0.5,-0.03))
plt.tight_layout()
```

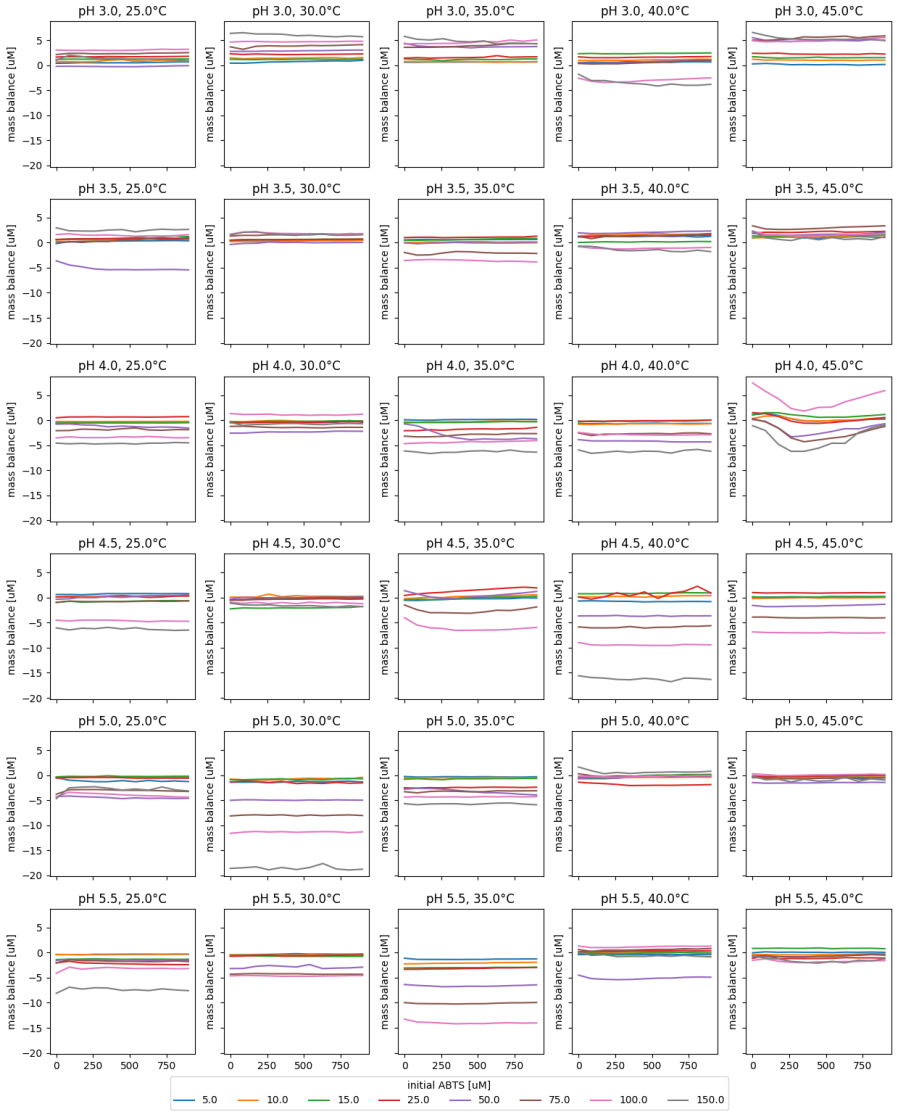


Fig. XXX: Quality control of each experiment through modeled mass balance.

For all experiments, except for pH 4, 45°C and pH 4.5, 35°C, the product and substrate is in balance over the reaction time-course. Mass balances, differing from 0 uM originate from differing substrate concentration in the enzyme reaction and the substrate control. In the reactions at pH 4, 45°C and pH 4.5, 35°C the mass balance slopes are not linear. This indicates issues with the measurement. Therefore, the respective measurements likely result in wrong parameter estimations.

Kinetic parameter estimation

Choice of kinetic model

Model selection is vital for parameter estimation. Thus, different settings for the parameter estimator were tested. Firstly, product inhibition models were excluded, since in some of the experiments the measured initial substrate concentration was higher than the specified one, which is used to calculate product concentrations. This resulted in calculated product concentrations with negative values.
Secondly, substrate inhibition models were excluded, since the model was not able to describe the observed reaction kinetics. This was evident from a higher AIC as well as more than 100 % standard deviation on the estimated parameters.
Lastly, the irreversible Michaelis-Menten model with and without time-dependent enzyme inactivation was compared, since enzyme inactivation was observed in previous experiments. Estimated parameters and the fitted models are shown for the experimental data at pH 3 and 25°C. Once without enzyme inactivation and once with.

```
fig, axes = plt.subplots(1,2, figsize=(10,5), sharey=True, sharex=True)
for i, ax in enumerate(axes.flatten()):
    ax.text(0, 1.1, string.ascii_uppercase[i], transform=ax.transAxes,
            size=20, weight='bold')
    if i == 0:
        print("Estimated parameters without time-dependent enzyme inactivation:")
        kinetics = ParameterEstimator.from_EnzymeML(concentration_enzymemldocs[0],
"s0", "substrate")
        kinetics.fit_models(only_irrev_MM=True)
        kinetics.visualize(ax=ax, title="without enzyme inactivation")
        ax.set_ylabel("ABTS [uM]")
        print("\n")
    else:
        print("Estimated parameters with time-dependent enzyme inactivation:")
        kinetics = ParameterEstimator.from_EnzymeML(concentration_enzymemldocs[0],
"s0", "substrate")
        kinetics.fit_models(enzyme_inactivation=True, only_irrev_MM=True)
        kinetics.visualize(ax=ax, title="with enzyme inactivation")
    ax.set_xlabel("time [s]")
handles, labels = ax.get_legend_handles_labels()
fig.legend(handles, labels, loc="lower center", ncol=len(labels), title="initial
ABTS [uM]", bbox_to_anchor=(0.5,-0.09))
plt.tight_layout()
```

```
Estimated parameters without time-dependent enzyme inactivation:
Fitting data to:
 - irreversible Michaelis Menten model
```

	AIC	Km [umole / l]	kcat / Km [1/s * 1/umole / l]	kcat [1/s]
irreversible Michaelis Menten	-162	35.473 +/- 5.52%	0.009 +/- 5.89%	0.308 +/- 2.07%

```
Estimated parameters with time-dependent enzyme inactivation:
Fitting data to:
 - irreversible Michaelis Menten model
```

	AIC	Km [umole / l]	kcat / Km [1/s * 1/umole / l]	kcat [1/s]	ki time-dep enzyme-inactiv. [1/s]
irreversible Michaelis Menten	-476	35.137 +/- 3.00%	0.013 +/- 3.58%	0.460 +/- 1.95%	0.001 +/- 4.47%

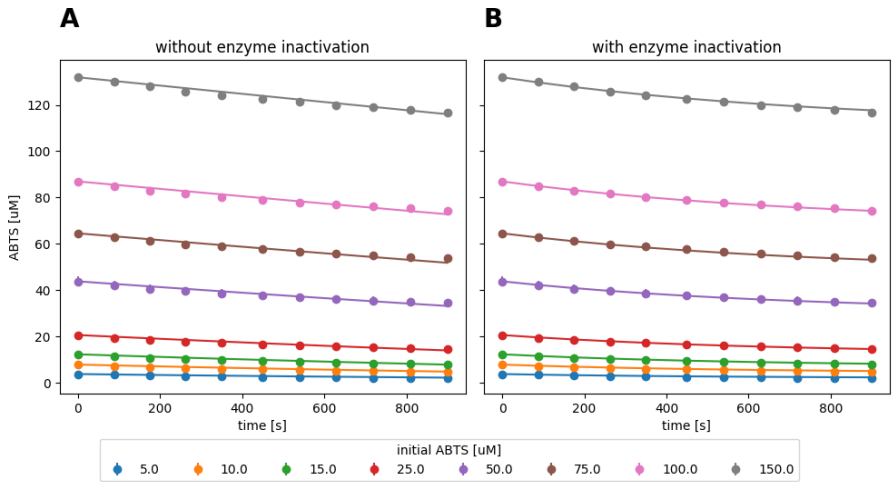


Fig. XXX: Fitted irreversible Michaelis-Menten model to experimental data of SLAC reaction at pH 3 and 25°C with and without considering time-dependent enzyme inactivation.

Visually, the model with enzyme inactinvation describes the data better, since the model intersects the data points more directly, compared to the model without enzyme inactivation (Fig. XXX). Statistically, this is characterized through a lower AIC, as well as lower standard deviations on the estimated parameters. In absolute therms, the estimated K_m of both models is approximately identical, whereas the k_{cat} estimate is approximately 33% lower for the model without enzyme inactivation. Hence, the model without enzyme inactivation underestimates the turnover number of the enzyme. As a result, irreversible Michaelis-Menten model with time-dependent enzyme inactivation was selected for the parameter estimation for all data sets.

```
# Run parameter estimator for all datasets, utilizing multi-processing.
def run_ParameterEstimator(enzmldoc: pe.EnzymeMLDocument):
    kinetics = ParameterEstimator.from_EnzymeML(enzmldoc, "s0", "substrate")
    kinetics.fit_models(enzyme_inactivation=True, only_irrev_MM=True,
display_output=False)
    return kinetics

results = Parallel(n_jobs=8)(delayed(run_ParameterEstimator)(enzmldoc) for
enzmldoc in concentration_enzymemldocs)
results = sorted(results, key=lambda x: (x.data.pH, x.data.temperature))

# Visualize all fitted models
fig, axes = plt.subplots(6,5, figsize=(12.5, 15), sharey=True, sharex=True)
for i, (doc, ax) in enumerate(zip(results, axes.flatten())):
    ph = doc.data.pH
    if not i%5:
        ax.set_ylabel("ABTS [uM]")
    doc.visualize(ax=ax, title = f"pH {doc.data.pH}, {doc.data.temperature}°C")
    if i in [25,26,27,28,29]:
        ax.set_xlabel("time [s]")
    if i == 0:
        handles, labels = ax.get_legend_handles_labels()
        fig.legend(handles, labels, loc="lower center", ncol=len(labels),
title="initial ABTS [uM]", bbox_to_anchor=(0.5,-0.04))
plt.tight_layout()
```

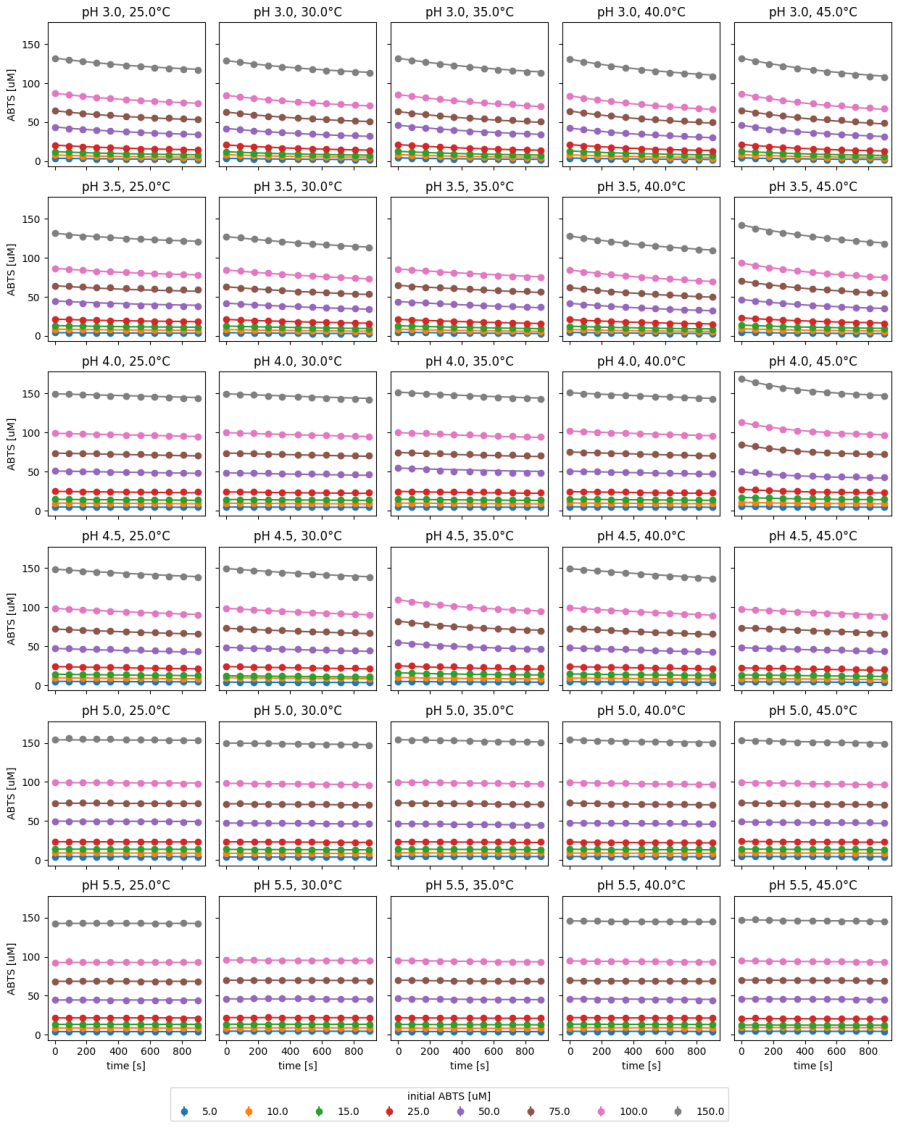


Fig. XXX: Experimental data and fitted irreversible Michaelis-Menten model with time-dependent enzyme inactivation of SLAC reactions under various conditions.

Experimental data as well as the fitted model are visualized in Fig XXX. Based on the reaction slopes, no catalytic activity was observed for reactions at pH 5 or higher. As in the mass balance analysis, the reactions at pH 4, 45°C and pH 4.5, 35°C differed from other experiments with identical pH. In both cases, all applied substrate concentration were higher than intended in the design of the experiment. Furthermore, the resulting parameter estimates have a high uncertainty. Therefore, the respective measurements were

excluded from further analysis. Additionally, the results from pH 5.5, 35°C were excluded, since the uncertainty of the parameters could not be estimated. All calculated kinetic parameters are listed in the table below.

```
# Extract kinetic parameters of all datasets
kcat = []
kcat_std = []
Km = []
Km_std = []
ki = []
ki_std = []
pH = []
temperature = []
corr_kcat_km = []
for result in results:
    params = result.get_parameter_dict()

    kcat.append(params["k_cat"].value)
    kcat_std.append(params["k_cat"].stderr)

    Km.append(params["Km"].value)
    Km_std.append(params["Km"].stderr)

    ki.append(params["K_ie"].value)
    ki_std.append(params["K_ie"].stderr)

    correlation = params["k_cat"].correl
    if correlation == None:
        corr_kcat_km.append(float("nan"))
    else:
        corr_kcat_km.append(correlation["Km"])

    pH.append(result.data.pH)
    temperature.append(result.data.temperature)

# Organize kinetic and experimantal parameters in a 'DataFrame'
df = pd.DataFrame.from_dict({
    'pH':pH,
    "temperature [C]":temperature,
    'kcat [1/s]':kcat,
    'kcat stderr':kcat_std,
    'Km [uM]':Km,
    'Km stderr':Km_std,
    "Enzyme inactivation [1/s]":ki,
    "Enzyme inactivation std":ki_std,
    "correlation kcat/Km":corr_kcat_km})

df["kcat/Km [1/s * uM]"] = df["kcat [1/s]"] / df["Km [uM]"]
kcat_km_stderr =((df["kcat stderr"]/df["kcat [1/s]"])**2+(df["Km stderr"]/df["Km
[uM]"])**2)**0.5 * df["kcat/Km [1/s * uM]"]
df["kcat/Km stderr"] = kcat_km_stderr

# Exclued results from failed experiments
df = df.drop(index=25) # reached parameter boundaries --> inactive
#df = df.drop(index=12) #Km really high stddev
df = df.drop(index=17) #kcat outlier, mass balance outlier
df = df.drop(index=14) #kcat outlier, mass balance outlier
df["zeros"] = np.zeros(27)

# Calculate halflife of enzyme
def calulate_halflife(x):
    return np.log(2)/x

df["half life [min]"] = df["Enzyme inactivation
[1/s]"].apply(calulate_halflife)/60
df["half life std"] = df["Enzyme inactivation std"] / df["Enzyme inactivation
[1/s]"] * df["half life [min]"]

# Delete inactivation rates
df = df.drop(columns=["Enzyme inactivation [1/s]", "Enzyme inactivation std"])
df
```

	pH	temperature [C]	kcat [1/s]	kcat stderr	Km [uM]	Km stderr	correlation kcat/Km	kcat/Km [1/s * uM]	kcat/Km stderr	zeros	half life [min]	half life
0	3.0	25.0	0.459962	0.008966	35.137357	1.053968	0.535141	0.013090	0.000468	0.0	9.139562	0.4085
1	3.0	30.0	0.470366	0.009855	32.114083	1.037088	0.522963	0.014647	0.000564	0.0	9.522170	0.476
2	3.0	35.0	0.546029	0.016598	37.454496	1.765891	0.558865	0.014578	0.000818	0.0	10.304462	0.776
3	3.0	40.0	0.712488	0.021754	46.441229	2.155285	0.608947	0.015342	0.000852	0.0	9.127765	0.600
4	3.0	45.0	0.815724	0.022451	47.967746	2.047138	0.623270	0.017006	0.000864	0.0	9.087343	0.530
5	3.5	25.0	0.420480	0.045642	89.752447	13.848584	0.746575	0.004685	0.000884	0.0	8.824513	1.680
6	3.5	30.0	0.399649	0.007455	61.224383	1.674235	0.666682	0.006528	0.000216	0.0	14.545045	0.804
7	3.5	35.0	0.246773	0.009247	28.237534	1.572755	0.554406	0.008739	0.000587	0.0	18.269277	2.766
8	3.5	40.0	0.650031	0.012397	79.432405	2.189303	0.731830	0.008183	0.000274	0.0	11.026573	0.453
9	3.5	45.0	0.978983	0.026556	103.784356	4.040850	0.774126	0.009433	0.000448	0.0	9.110325	0.425
10	4.0	25.0	0.115133	0.008752	65.852866	2.476592	0.247717	0.001748	0.000148	0.0	111.702355	222.064
11	4.0	30.0	0.192374	0.020265	95.314777	5.569713	-0.718208	0.002018	0.000243	0.0	23.819987	10.874
12	4.0	35.0	0.282991	0.029279	100.647793	11.576143	0.608286	0.002812	0.000435	0.0	14.929497	4.974
13	4.0	40.0	0.225429	0.012766	97.966631	1.677116	0.168743	0.002301	0.000136	0.0	29.441552	12.288
15	4.5	25.0	0.368730	0.025801	110.902372	0.830267	-0.736267	0.003325	0.000234	0.0	22.536023	6.247
16	4.5	30.0	0.486402	0.036084	148.193267	3.440830	0.816213	0.003282	0.000255	0.0	19.046651	4.109
18	4.5	40.0	0.549232	0.021351	178.481651	2.544896	0.084106	0.003077	0.000127	0.0	34.946634	11.931
19	4.5	45.0	0.235716	0.013338	64.718239	0.037679	-0.113239	0.003642	0.000206	0.0	108.053681	160.970
20	5.0	25.0	0.008085	0.006871	15.067578	20.447918	0.276523	0.000537	0.000859	0.0	115.522134	2593.086
21	5.0	30.0	0.040665	0.005659	50.772187	3.784251	0.009319	0.000801	0.000126	0.0	115.512956	453.016
22	5.0	35.0	0.061677	0.006158	68.874565	0.267258	0.299400	0.000895	0.000089	0.0	115.519891	286.705
23	5.0	40.0	0.131196	0.017131	91.314407	2.756719	-0.426692	0.001437	0.000193	0.0	8.399863	2.519
24	5.0	45.0	0.129399	0.014577	96.735544	2.562607	-0.696719	0.001338	0.000155	0.0	11.438272	2.986
26	5.5	30.0	0.014374	0.007402	96.125085	0.393705	-0.200038	0.000150	0.000077	0.0	115.523248	1471.972
27	5.5	35.0	0.124370	0.023968	81.996103	22.257271	0.771045	0.001517	0.000505	0.0	5.841192	1.400
28	5.5	40.0	0.055892	0.016036	48.740460	19.833207	0.552798	0.001147	0.000571	0.0	6.132460	2.957
29	5.5	45.0	0.075249	0.015052	115.390172	27.541849	0.604743	0.000652	0.000203	0.0	10.834329	5.343

High correlation between k_{cat} and K_m

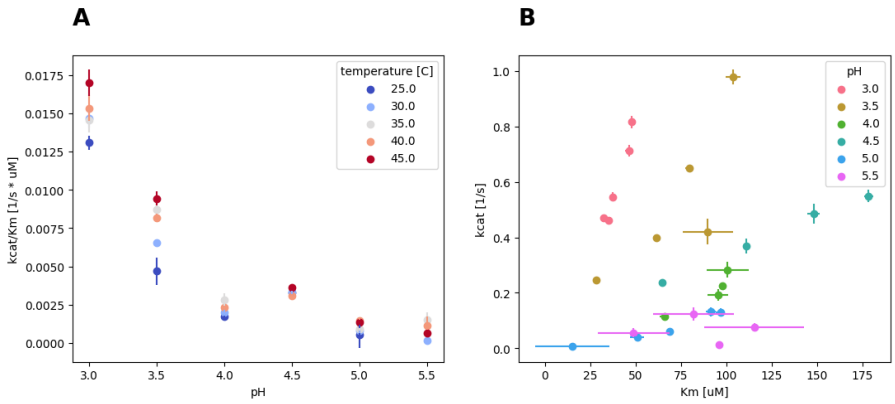
The parameter estimates for k_{cat} and K_m showed correlation. For experiments at pH 3 and pH 3.5, the parameters were correlated between 0.5 - 0.85. This indicates, that the highest initial substrate concentration, which was applied for parameter estimation was not sufficiently high. Ideally, the highest initial substrate concentration applied for a kinetic experiment should be 10-fold higher than K_m to allow independent estimates for k_{cat} and K_m [Tipton *et al.*, 2014]. In this scenario, only reactions with an initial substrate concentration up to 175 uM were used for parameter estimation, due to the limited photometric detection range of ABTS. Hence, ABTS was only applied 1.5-fold to 6-fold of the estimated K_m.
For reactions at pH 4 and above, positive as well as negative correlations were observed. This might result from wrong parameter estimated, due to minimal enzyme activity at the given reaction conditions.

k_{cat} over K_m for all experimental conditions is visualized in Fig. XXXB, whereas the pH is color-coded. For reactions with identical pH values, k_{cat} and K_m both increase with temperatures. This might be attributed to the high correlation between the parameters. Thus, the true change of k_{cat} or K_m through pH or temperature cannot be assessed with certainty. Therefore, the catalytic efficiency $\frac{k_{cat}}{K_m}$, was a better measure to assess the enzyme activity under different reaction conditions.

For reactions at pH 5 and above, almost no catalytic activity was observed. Therefore, data of these experimental conditions is excluded from further analysis.

Parameter estimates for k_{cat} and K_m

```
# Visualize estimated parameters
fig, axes = plt.subplots(1,2, figsize=(12.8, 4.8), sharey=False, sharex=False)
for i, ax in enumerate(axes.flatten()):
    ax.text(0, 1.1, string.ascii_uppercase[i], transform=ax.transAxes,
            size=20, weight='bold')
    if i==1:
        plot(df, xdata="Km [uM]", ydata="kcat [1/s]", xerror="Km stderr",
yerror="kcat stderr", colors="pH", ax=ax)
    else:
        plot(df, xdata="pH", ydata="kcat/Km [1/s * uM]", xerror="zeros",
yerror="kcat/Km stderr", colors="temperature [C]", ax=ax)
```



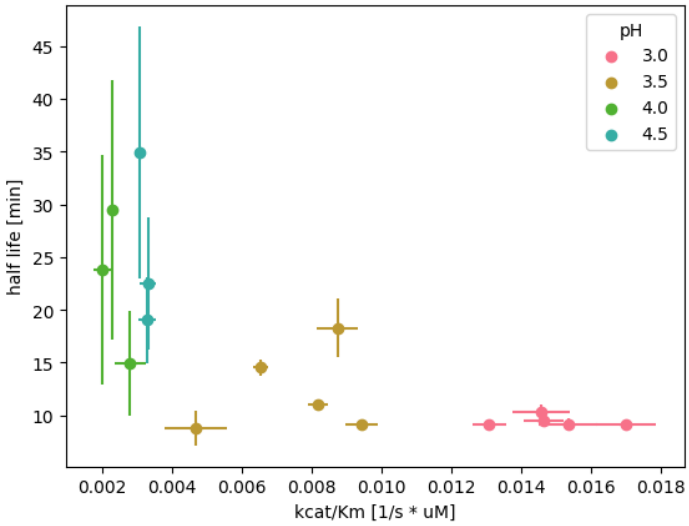
$\frac{k_{cat}}{K_m}$ in relation to reaction pH is visualized in Fig. XXXA. The highest catalytic efficiency was observed at pH 3 and 45°C. Increasing the pH by 0.5 reduced $\frac{k_{cat}}{K_m}$ approximately by half. For higher pH values, $\frac{k_{cat}}{K_m}$ is even more decreased. The catalytic efficiency is therefore highly sensitive to the pH. This might source from changed protonation state of either substrate or enzyme, hindering the efficient formation of the enzyme-substrate complex. In terms of temperature, higher $\frac{k_{cat}}{K_m}$ was achieved at higher temperatures. SLAC might even be more active above 45°C and under pH 3.

Time-dependent enzyme inactivation

The half life of SLAC was calculated with equation (13) [Buxbaum, 2015].

$$t_{\frac{1}{2}} = \frac{ln(2)}{k_{inact}} \tag{13}$$

```
# Exclued inactive enzyme reactions above pH 4.5 and discard measurements with
more than 100 % standard deviation on enzyme inactivation parameter
df = df.loc[:19]
df = df.drop(index=10) # more than 100 % standard deviation on enzyme inactivation
parameter
df = df.drop(index=19) # more than 100 % standard deviation on enzyme inactivation
parameter

# Plot half life over catalytic efficiency
plot(df, xdata="kcat/Km [1/s * uM]", ydata="half life [min]", xerror="kcat/Km
stderr", yerror="half life std", colors="pH")
plt.show()
```



The enzyme's calculated half life was between 8 - 18 min for reactions at pH 3 and pH 3.5, whereas enzyme reactions at higher pH values showed a half life between 15 - 35 min. Generally, reactions with higher catalytic efficiency showed a shorter half life compared to reactions with lower catalytic efficiency In order to check for corrections between k_{inact}, k_{cat}, K_m, $\frac{k_{cat}}{K_m}$, as well as the reaction temperature and pH, a correlation analysis was conducted across all datasets.

```
# Calculate correlations
df_corr = df.drop(columns=["kcat stderr", "Km stderr", "correlation kcat/Km",
"kcat/Km stderr", "zeros", "half life std"])
rho = df_corr.corr()

# Visualize correlation heatmap
#sns.heatmap(rho,
#        xticklabels=rho.columns,
#        yticklabels=rho.columns,
#        cmap = sns.color_palette("vlag", as_cmap=True))
#plt.show()

# Calculate confidence intercal for correlations
pval = df_corr.corr(method=lambda x, y: pearsonr(x, y)[1]) - np.eye(*rho.shape)
p = pval.applymap(lambda x: ''.join(['*' for t in [.05] if x<=t]))
rho.round(2).astype(str) + p
```

	pH	temperature [C]	kcat [1/s]	Km [uM]	kcat/Km [1/s * uM]	half life [min]
pH	1.0***	-0.15	-0.39	0.88***	-0.89***	0.8***
temperature [C]	-0.15	1.0***	0.6*	0.07	0.27	0.03
kcat [1/s]	-0.39	0.6*	1.0***	-0.03	0.58*	-0.49
Km [uM]	0.88***	0.07	-0.03	1.0***	-0.75***	0.68**
kcat/Km [1/s * uM]	-0.89***	0.27	0.58*	-0.75***	1.0***	-0.71**
half life [min]	0.8***	0.03	-0.49	0.68**	-0.71**	1.0***

The above table shows correlation between between kinetic parameters and experimental conditions. Thereby, significant ($p < 0.05$) correlations are labeled with '*'. The half life of SLAC is significantly correlated to $\frac{k_{cat}}{K_m}$ as well as to pH. Due to correlation between k_{cat} and K_m, the enzyme's half life is also correlated to K_m. Multiple reasons might be the reason for this observation. On the one hand, the observed correlation might be of technical origin. Hence the system of ordinary differential equations, describing the change in substrate and enzyme concentration, are not an accurate model for the reaction system. Therefore, high cross-correlation occurs, since individual observations cannot be attributed to individual parameters. On the other hand, enzyme activity and enzyme inactivation might be causally related. In this case, the formed ABTS radical might inactivate the enzyme by potential suicide inhibition, or the enzyme deteriorates through catalysis.

Conclusion

Data management
The established data pipeline allows scalable data preparation and analysis of enzyme kinetics experiments. In following experiments this workflow could be used to further expand the experimental parameter space. As a result, different enzymes with different substrates in different buffers at different temperatures and pH values could be analyzed.

Correlation between parameters
The highest catalytic efficiency of SLAC was observed at pH 3 at 45°C and therefore on the edge of the investigated parameter space. Optimal reaction conditions might therefore be at an even lower pH and higher temperature. k_{cat} and K_m were correlated, which is likely the result of a too low initial substrate concentration in relation to the true K_m of the enzyme at a given experimental condition. Hence, in adjoining experiments the product signal could be used for parameter estimation, since the concentration to absorbance ratio is lower compared to the one of the product. However, a reliable method for product quantification would need to be established first.

Enzyme inactivation
SLAC showed a short half life of approximately 10 min at reaction conditions with the highest catalytic efficiency, which was correlated to the catalytic efficiency. Further experiments are required to investigate the observed inactivation.

Scenario D:
Time-dependent enzyme inactivation

Data provided by Paulo Durão (Microbial & Enzyme Technology, Instituto de Tecnologia Química e Biológica, Oeiras, Portugal)

Project background

All investigated enzyme reaction without inhibitor applied to the reaction showed progress curve behavior, which were not explainable by irreversible Michaelis-Menten kinetics. All experiments had in common, that enzyme reactions were carried out in 96-well polystyrene micro titer plates (MTP), whereas the change in substrate or product absorption was monitored photometrically. In theory, the observed time-dependent decrease in enzyme activity can either be explained through enzyme inactivation or product inhibition. In case of time-dependent enzyme inactivation, the catalytic activity decreases, since the catalyst becomes inactive. Thereby, the decrease in relative reaction rate is independent from product concentration. In case of product inhibition, the formed product reduces the reaction rate. Hence, apparent enzyme inactivation is dependent on product concentration. Since both phenomena express themselves in an apparent decrease of reaction rate, they bare the potential to be confused

One hypothesis for the suspected enzyme inactivation is hydrophobic interaction between the enzyme and the MTP surface. Thereby, hydrophobic regions of the enzyme's surface might interact with the polystyrene reaction vessel, potentially preventing substrate access to the active site of the enzyme. In order to test the hypothesis, adsorption experiments in which enzyme was incubated in MTP wells prior to reaction start was performed. Thereby, the enzyme activity should decrease depending on prior incubation time. If the hypothesis is correct, the calculated half life from the adsorption experiment should match with the half life of an enzyme kinetics experiment which was conducted in parallel.

Show results of enzyme inactication across projects #TODO

All investigated enzyme reactions in this thesis showed a time-dependent decrease in catalytic activity, which was not explainable with the irreversible Michaelis-Menten model.

Experimental design

Determination of enzyme inactivation trough adsorption
The inactivation experiment was conducted by incubating CotA laccase from *Bacillus subtilis* in individual MTP wells up to 1 h prior to reaction start. Then, enzyme reactions were started in 10 min intervals by transferring 2 µL of incubated enzyme in individual MTP wells. Each proceeding enzyme reaction contained 256 nM CotA, 1 mM ABTS and was buffered in acetate buffer at pH 4. Product formation was followed photometrically at 420 nm and 25°C for 5 min, whereas concentrations were calculated assuming an extinction coefficient of ε = 36000 $M^{-1}cm^{-1}$ for the ABTS radical product. Each experiment was performed in technical triplicates.

Enzyme kinetics experiment

Enzymatic oxidation of ABTS to its radical form was followed photometrically at 420 nm at 25°C for 70 min. Thereby, ABTS was applied in a range from 0.01 mM - 2 mM. Each proceeding enzyme reaction contained 256 nM CotA, and was buffered in acetate buffer at pH 4. Each experimental condition was repeated as technical quadruplicates. Concentrations were calculated via the provided extinction coefficient of the ABTS radical (ε = 36000 $M^{-1}cm^{-1}$).

Experimental data

Measurement data was provided as an Excel file, containing time-course absorption data. Meta data was written into the EnzymeML Excel template. Then, the experimental data was written to an EnzymeML document by a parser function.

Adsorption of enzyme to micro titer plate surface

Imports

```
import numpy as np
import pandas as pd
from scipy.stats import linregress
import pyenzyme as pe
from EnzymePynetics.tools.parameterestimator import ParameterEstimator
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from IPython.display import display
import string


#TODO warning in EnzymePynetics

import warnings
warnings.filterwarnings('ignore')
```

Calculation of inactivation rate

Experimental data was loaded into a Pandas DataFrame and the slopes of each incubation condition were calculated through linear regression. The resulting initial rates were then used to calculate the rate by which the enzyme activity decreases. Lastly the half life of the enzyme was calculated with equation (13).

```
# Load excel
path = '../../data/enzyme_inactivation/Slide 2 - Activity effect of incubating
CotA in MTP.xlsx'
df = pd.read_excel(path, sheet_name='csv').set_index('time (min)')

# replace values of '0**' with nan-values, since the measurement is incorrect
df['0**'] = np.nan



# Get data from Excel file
columns = [int(x) for x in list(df.columns) if str(x).endswith("0")]
time = df.index.values
absorption = df.values.T.reshape(7,3,22)

# Calculate concentrations
extinction_coefficient = 36 # (1/mM * 1/cm)
optical_length = 0.65 # cm

def absorption_to_concentration(abso):
    return abso / (extinction_coefficient*optical_length)

concentration = absorption_to_concentration(absorption)
concentration_mean = np.nanmean(concentration, axis = 1)
concentration_std = np.nanstd(concentration, axis = 1)

# Linear regression
slopes = []
intercepts = []
stderrs = []
for time_set in concentration:
    mask = ~np.isnan(time_set)
    time_regression = np.tile(time, 3).reshape(mask.shape)[mask].flatten()
    data_regression = time_set[mask].flatten()
    slope,intercept,_,_,stderr = (linregress(time_regression, data_regression))
    slopes.append(slope)
    intercepts.append(intercept)
    stderrs.append(stderr)

# Regression of the slopes between 10 and 60 mincolumns
slope_of_slopes, intercept_of_slopes,_,_,_ = linregress(columns[1:], slopes[1:])

# Half life in hours
t12 = (np.log(2)/-slope_of_slopes)/60

# Plot results
print(f"Calculated enzyme half life based on regression slope: {t12:.2f} h")

colors = cm.tab10(np.linspace(0, 1, len(absorption)))
fig, axes = plt.subplots(1,2, figsize=(12.8,4.8), sharey=False, sharex=False)

for mean, std, label, slope, intercept, color in zip(concentration_mean,
concentration_std, columns, slopes, intercepts, colors):
    axes[0].errorbar(time, mean, std, fmt='o', label=label, alpha = 0.4,
color=color)
    axes[0].plot(np.array(time), np.array(time)*slope+intercept, '--',
color=color)
    axes[0].legend(title="CotA incubation\nin MTP-well [min]")
    axes[0].set_ylabel('ABTS radical [mM]')
    axes[0].set_xlabel('time [min]')
    axes[0].set_title('Experimental data with calculated reaction rate by linear
regression')
    axes[0].text(-0.1, 1.1, string.ascii_uppercase[0],
transform=axes[0].transAxes,
            size=20, weight='bold')


axes[1].errorbar(columns, slopes, stderrs, fmt='o', label="initial rates within
the first 5 min")
axes[1].set_xlabel("CotA incubation time\nin MTP-well [min]")
axes[1].set_ylabel('initial rate [mM min$^{-1}$]')
axes[1].plot(np.array(columns[1:]),
slope_of_slopes*np.array(columns[1:])+intercept_of_slopes, "--", color=colors[0],
label=f"regression")
axes[1].legend()
axes[1].set_title('Initial rates of CotA reaction')
axes[1].text(1.1, 1.1, string.ascii_uppercase[1], transform=axes[0].transAxes,
        size=20, weight='bold')

plt.show()
```

```
Calculated enzyme half life based on regression slope: 693.73 h
```

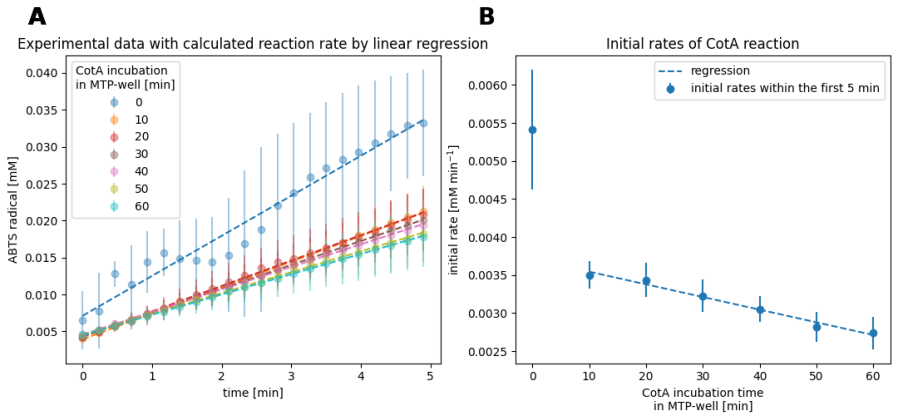


Fig. XXX: Experimental data and regression results of CotA reaction with different enzyme incubation times.

Enzyme reactions without prior incubation showed the fastest reaction rate, whereas all reactions with prior incubation showed an approximately halved reaction rate (Fig. XXXA). Thereby, reactions with longer incubation time showed gradually reduced reaction rates with increased incubation time. Enzyme reactions, especially reactions without incubation, showed large deviations across replicates. This presumably resulted from inhomogeneous mixing or deviating enzyme concentration due to the small volume of enzyme solution, which was used to start the reactions.
Since the true reaction rate of reactions without prior incubation was highly uncertain and showed mixing issued, the respective data was not used to estimate the enzyme's time-dependent inactivation rate. Based on the remaining initial reaction rates the enzymes, the half life was estimated at approximately 28 days (Fig. XXXB).

Enzyme kinetics experiment

Experimental data

Experimental data was provided as an Excel file, whereas meta data of the experiment was filled in to an EnzymeML Excel spreadsheet. Measurement data was written to the EnzymeML document by a parser function Kinetic parameters were estimated with and without considering enzyme inactivation.

```
# Load experimental data from excel file
df = pd.read_excel("../../data/enzyme_inactivation/Repetition CotA ABTS kinetics
higher volumes 2nd time.xlsx", sheet_name="csv").set_index("Time(min)")
data = df.values.T.reshape(8,4,72)
time = df.index.values

# Calculate concentrations
extinction_coefficient = 36 # (1/mM * 1/cm)
optical_length = 0.65 # cm

def absorption_to_concentration(abso):
    return abso / (extinction_coefficient*optical_length)

concentration_data = absorption_to_concentration(data)


# Parser function
def data_to_EnzymeML(
    template_path: str,
    measurement_data: np.ndarray,
    species_id: str,
    data_unit: str,
    time_unit: str
    ) -> pe.EnzymeMLDocument:

    enzmldoc = pe.EnzymeMLDocument.fromTemplate(template_path)
    for IDs, concentration in zip(enzmldoc.measurement_dict.keys(),
measurement_data):
        for counter, replicate in enumerate(concentration):
            rep = pe.Replicate(
                id=f"Measurement{counter}",
                species_id=species_id,
                data=list(replicate),
                data_unit=data_unit,
                time=list(time),
                time_unit=time_unit)
            enzmldoc.getMeasurement(IDs).addReplicates(rep, enzmldoc)
    return enzmldoc

# Write experimantal data to EnzymeML document vis parser function
enzmldoc = data_to_EnzymeML(
    template_path="../../data/enzyme_inactivation/EnzymeML_CotA.xlsm",
    measurement_data=concentration_data,
    species_id="s1",
    data_unit="mmole / l",
    time_unit="min")

# Visualize experimantal data
fig, axes = plt.subplots(1,2, figsize=(12.8, 4.8), sharey=False, sharex=False)
for measurement in enzmldoc.measurement_dict.values():
    concentration = []
    product = measurement.getReactant("s1")
    init_substrate=measurement.getReactant("s0").init_conc
    for replicate in product.replicates:
        concentration.append(replicate.data)
    axes[0].errorbar(time, np.mean(concentration, axis=0), np.std(concentration,
axis=0), label=init_substrate,\
        fmt=".", alpha=0.5)
    axes[1].errorbar(time, np.mean(concentration, axis=0), np.std(concentration,
axis=0), label=init_substrate,\
        fmt="o")

axes[0].legend(title="Initial ABTS [mM]")
axes[1].legend(title="Initial ABTS [mM]")
axes[0].set_ylabel("ABTS radical [mM]")
axes[0].set_title("ABTS radical concentration over CotA reaction time-course ")
axes[1].set_title("ABTS radical concentration within the first 5 min")
axes[1].set_ylabel("ABTS radical [mM]")
axes[0].set_xlabel("time [min]")
axes[1].set_xlabel("time [min]")
axes[1].set_xlim([-0.2,5.2])
axes[1].set_ylim([0,0.01])
axes[0].text(-0.1, 1.1, string.ascii_uppercase[0], transform=axes[0].transAxes,
        size=20, weight='bold')
axes[1].text(1.1, 1.1, string.ascii_uppercase[1], transform=axes[0].transAxes,
        size=20, weight='bold')

fig.show()
```

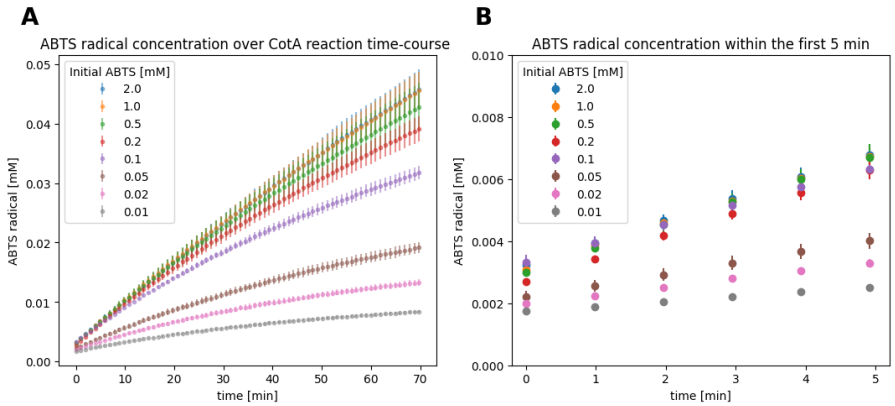


Fig XXX. ABTS radical concentration over the time-course of CotA reaction.

The provided reaction data showed large deviations between individual repeats of reactions with identical initial ABTS concentrations. Especially for initial ABTS concentrations above 0.1 mM (Fig. XXXA). Furthermore, visual analysis of the initial measurement points indicated that reactions with an initial substrate concentration of 0.1 mM had an increased enzyme concentration compared to others. This is evident, because the mentioned reactions show the highest concentration at the initial measurement time (Fig. XXXB). The order of initial measurement points is expected to reflect the initial substrate concentrations and since the substrate slightly absorbs at the product detection wavelength. The substrate of the mentioned reaction was likely applied correctly, since the final concentration at 70 min is between reactions with the next lower and next higher initial substrate concentration.

Kinetic parameter estimation

Quality control through modeling

An initial parameter estimation considering enzyme inactivation was conducted in order to check further systematic deviations. As a result, reactions with an initial ABTS concentration of 0.05 mM showed to deviate from all other reactions, since the respective product concentrations is lower than the model predicted. The mentioned measurements were the only measurements where the measured concentration was distinctly and systematically below that of the model (Fig. xxxA). Hence indicating that either less substrate or enzyme than defined in the protocol was applied to the respective reactions. In consequence, data from reactions with an initial substrate concentration of 0.05 and 0.1 mM were excluded from parameter estimation, because the identified systematic deviations in reaction conditions would distort the results.

```
# Parameter estimation considering time-dependent enzyme inactivation
CotA_kinetics_first_run = ParameterEstimator.from_EnzymeML(enzmldoc, "s1",
"product")
CotA_kinetics_first_run.fit_models(enzyme_inactivation=True, display_output=False,
initial_substrate_concs=[0.01, 0.02, 0.05, 0.1, 0.2, 0.5, 1.0, 2.0])

# Parameter estimation considering time-dependent enzyme inactivation
CotA_kinetics_with_inactivation = ParameterEstimator.from_EnzymeML(enzmldoc, "s1",
"product")
CotA_kinetics_with_inactivation.fit_models(enzyme_inactivation=True,
display_output=False, initial_substrate_concs=[0.01, 0.02, 0.2, 0.5, 1.0, 2.0])
df_inactivation = CotA_kinetics_with_inactivation.result_dict.drop(columns=["kcat
/ Km [1/min * 1/mmole / l]"])
df_inactivation.insert(1, "Enzyme inactivation model", "True")

# Visualize fit of the models
fig, axes = plt.subplots(1,2, figsize=(12.8,4.8), sharey=True, sharex=True)
CotA_kinetics_first_run.visualize("irreversible Michaelis Menten",ax=axes[0],
alpha =.2,\
    title="Parameter estimation based on full dataset")
CotA_kinetics_with_inactivation.visualize("irreversible Michaelis
Menten",ax=axes[1], alpha =.2,\
    title="Parameter estimation based on data subset")

axes[0].set_ylabel("ABTS radical [mM]")
axes[0].set_xlabel("time [min]")
axes[1].set_xlabel("time [min]")
axes[0].legend(title="initial ABTS [mM]")
axes[1].legend(title="initial ABTS [mM]")
axes[0].text(-0.1, 1.1, string.ascii_uppercase[0], transform=axes[0].transAxes,
        size=20, weight='bold')
axes[1].text(1.1, 1.1, string.ascii_uppercase[1], transform=axes[0].transAxes,
        size=20, weight='bold')
plt.tight_layout()
```

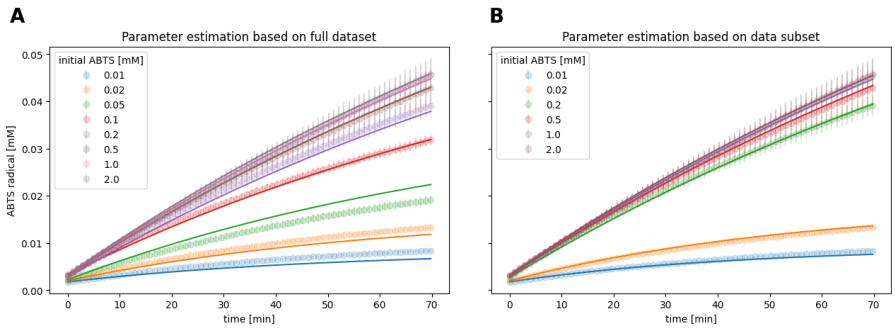


Fig XXX: Identification of systematic deviations through kinetic modeling.

By excluding the mentioned measurements, the model fit was improved (Fig xxxB). Additionally, the correlation between parameter estimates for k_{cat} and K_m was reduced from 0.518 to 0.204

Parameter estimation with and without considering enzyme inactivation

```
# Parameter estimation without time-dependent enzyme inactivation
CotA_kinetics = ParameterEstimator.from_EnzymeML(enzmldoc, "s1", "product")
CotA_kinetics.fit_models(enzyme_inactivation=False, display_output=False,
initial_substrate_concs=[0.01, 0.02, 0.2, 0.5, 1.0, 2.0])
df = CotA_kinetics.result_dict.drop(columns=["kcat / Km [1/min * 1/mmole / l]"])
df.insert(1, "Enzyme inactivation model", "False")

results = df.append(df_inactivation).sort_values("AIC")
display(results.style.set_table_attributes('style="font-size: 12px"'))
```

	AIC	Enzyme inactivation model	kcat [1/min]	Km [mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]	ki time-dep enzyme-inactiv. [1/min]
irreversible Michaelis Menten	-22202	True	3.003 +/- 0.77%	0.030 +/- 1.99%	-	-	0.007 +/- 4.48%
competitive product inhibition	-22202	True	2.999 +/- 0.78%	0.029 +/- 3.85%	0.221 +/- 92.18%	-	0.007 +/- 5.06%
substrate inhibition	-22202	True	3.021 +/- 0.86%	0.031 +/- 2.38%	-	208.577 +/- 67.12%	0.007 +/- 4.48%
uncompetitive product inhibition	-22199	True	3.012 +/- 1.23%	0.030 +/- 4.72%	-	2.096 +/- 361.49%	0.006 +/- 15.30%
non-competitive product inhibition	-22198	True	3.012 +/- 2.44%	0.029 +/- 6.56%	0.167 +/- 132.63%	1.113 +/- 466.95%	0.006 +/- 40.69%
non-competitive product inhibition	-22185	False	3.241 +/- 1.40%	0.033 +/- 5.05%	0.038 +/- 22.41%	0.073 +/- 6.86%	nan
uncompetitive product inhibition	-22156	False	3.268 +/- 1.43%	0.041 +/- 2.70%	-	0.066 +/- 6.35%	nan
competitive product inhibition	-21840	False	2.586 +/- 0.35%	0.020 +/- 6.10%	0.016 +/- 15.88%	-	nan
irreversible Michaelis Menten	-21739	False	2.529 +/- 0.27%	0.030 +/- 2.30%	-	-	nan
substrate inhibition	-21737	False	2.537 +/- 0.52%	0.030 +/- 2.71%	-	352.069 +/- 134.63%	nan

Kinetic parameters were estimated with and without considering time-dependent enzyme inactivation. Modeling results are listed in the table above. All models with an additional parameter for enzyme inactivation resulted in lower AIC values, indicating a better fit of the experimental data to inactivation models. Irreversible Michaelis-Menten model resulted in the highest AIC together with competitive product inhibition as well as substrate inhibition. Nevertheless, all inhibition models considering enzyme inactivation showed high standard deviations above 80% for the respective K_i estimates. Therefore irreversible Michaelis-Menten model with time-dependent enzyme inactivation described the measurement data the best (Fig XXXA). Thereby, k_{cat} was estimated to 3.003 min^{-1} ± 0.77% whereas 30 µM ± 1.99% was estimated for K_m. The respective half life of CotA was estimated at 95 min ± 4 min using equation (13), whereas k_{cat} and k_{inact} were highly correlated (corr > 0.95). Irreversible Michelis-Menten model without enzyme concentration did not fit the measurement data, since the reaction rate is initially underestimated, and overestimated in later stages of the reaction (Fig xxxB).

```
fig, axes = plt.subplots(2,2, figsize=(12.8,9.2), sharey=True, sharex=True)
CotA_kinetics_with_inactivation.visualize("irreversible Michaelis
Menten",ax=axes[0][0], alpha =.2,\
    title="irrev. Michaelis-Menten model with time-dependent enzyme inactivation")
CotA_kinetics.visualize("irreversible Michaelis Menten",ax=axes[0][1], alpha =.2,\
    title="irrev. Michaelis-Menten model")
CotA_kinetics.visualize("non-competitive product inhibition",ax=axes[1][0], alpha
=.2,\
    title="non-competitive product inhibition model")
CotA_kinetics.visualize("uncompetitive product inhibition",ax=axes[1][1], alpha
=.2,\
    title="uncompetitive product inhibition model")

axes[0][0].set_ylabel("ABTS radical [mM]")
axes[1][0].set_ylabel("ABTS radical [mM]")
axes[1][0].set_xlabel("time [min]")
axes[1][1].set_xlabel("time [min]")
[ax.legend(title="initial ABTS [mM]") for ax in axes.flatten()]

[ax.text(0, 1.1, string.ascii_uppercase[i], transform=ax.transAxes, size=20,
weight='bold') for i, ax in enumerate(axes.flatten())]

# axes[0]
# axes[1].text(1.1, 1.1, string.ascii_uppercase[1], transform=axes[0].transAxes,
#         size=20, weight='bold')
plt.tight_layout()
```

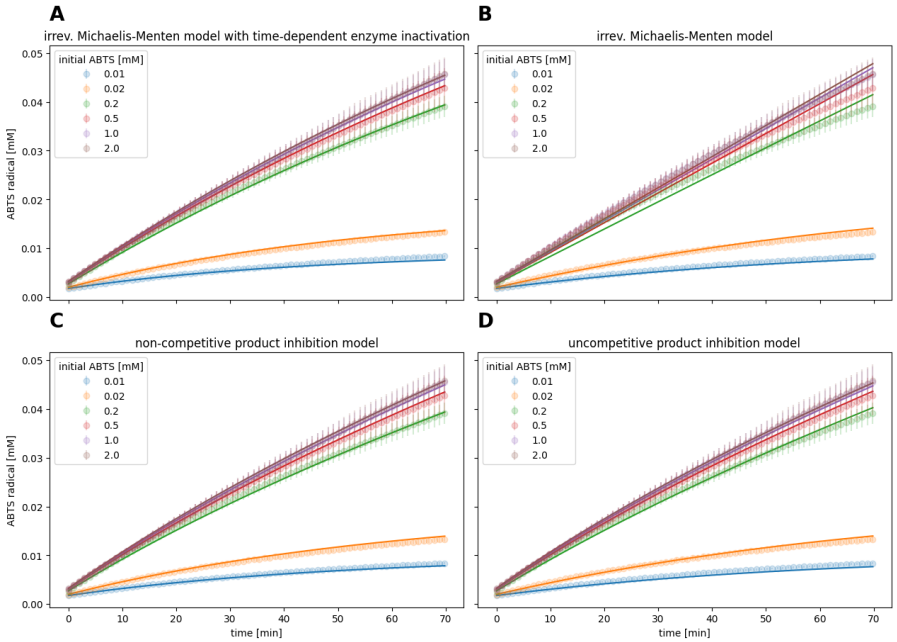


Fig. XXX: Different kinetic models fitted to CotA reaction data

Out of the models without enzyme inactivation, product inhibition models showed the lowest AIC. Qualitatively, the inhibition models described the the progress curve of the reaction (Fig. xxxC and D)

Discussion

Low data quality Both datasets, for the absorption experiment and the enzyme kinetics experiment, showed low data quality, since repeats of identical experimental conditions showed large deviations. Additionally, enzyme concentration was systematically applied in too low concentration for individual reaction conditions. Ultimately,

Enzyme inactivation The adsorption experiment resulted in an CotA half life of approximately 28 days, whereas the enzyme kinetics experiment showed an half life of 95 min. As a result, the apparent decrease in enzyme activity is likely not caused by inactivation through adsorption of the enzyme to the reaction vessel. However, low experimental precision or the design of the adsorption experiment itself might be unsuited to confirm or reject the hypothesis.

Based on kinetic modeling, time-dependent enzyme inactivation described the measured reaction progress-curve. Thereby, the estimated inactivation rate was highly correlated to the turnover number of CotA. As in the previous chapter, the correlation might be technical or causal. Nevertheless, product inhibition models were able to describe the measured progress-curves qualitatively. Due to data quality issues product inhibition cannot be completely ruled out.

Further kinetic assays with higher enzyme concentration or prolonged measurement times should be conducted to confirm or exclude product inhibition as an potential source of the apparent decrease of reaction rate. Furthermore, the pipetting volume of enzyme should be increased, to reduce variation between experimental repeats.

Discussion

Discuss both viewpoints: biologist and RSE. Documentation of experiments stops after measurement. Treatment of measurement data

Jupyter notebooks for documentation of scientific analysis

Jupyter notebooks enable enable

Reproducibility in enzyme kinetics

- Need for documentation without limiting experimental possibilities
- Strenda guidelines are insufficient??

Data management / lab digitalization

- temperature in MTPs

Jupyter notebooks and book

- great documentation (FAIR)

In this project, a robust and FAIR workflow from analytical raw-data to kinetic parameters was established. Thereby, the output of a analytical device was used directly as the input of the modeling pipeline.

Despite the robust workflow, after the data is written to an EnzymeML document, manual copying of data into the EnzymeML spreadsheet is prone to human error. Especially, for large datasets with many time points and reactants. Hence, manufacturers should enable direct access to the measurement device, for instance via an API. (Control and laer experiments)

Despite the easy applicability of the workflow, after the data is initialized,

Current state in kinetic modeling of enzyme kinetics

Model selection in enzyme kinetics: calibration and kinetic model

- correlation between parameters:
- short experimental time
- low initial substrate concenration
- unpercise data –> systematic deviations between mesaurements

pre-requires high data quality. If not, modeling can lead to wrong conclusions

- Best-practices kinetic modeling
    - initial rates vs progress-curve analysis
        - initial rates not suited to estimate conversion after 24 h.
- Cost of additional parameters
- Model selection criteria
- How to assess the quality of a model
    - what does the model represent (not newtonian mechanics)

Inactivation

- deterioration during catalysis
- adsorption to MTP surface (hydrophobic interaction)

Percision of kinetic parameters

- how percise should parameters be
- randomized order in MTPs

Conclusion

Nice.
:::{figure-md} markdown-fig 

This is a caption in **Markdown**!

References

BAC99 William F Busby, Joseph M Ackermann, and Charles L Crespi. Effect of methanol, ethanol, dimethyl sulfoxide, and acetonitrile on in vitro activities of cdna-expressed human cytochromes p-450. *Drug Metabolism and Disposition*, 27(2):246–249, 1999.

Bux15 Engelbert Buxbaum. *Enzyme Kinetics: Special Cases*, pages 185–191. Springer International Publishing, Cham, 2015. URL: https://doi.org/10.1007/978-3-319-19920-7_8, doi:10.1007/978-3-319-19920-7_8.

DMMP20 Chantal Désirée Daub, Blessing Mabate, Samkelo Malgas, and Brett Ivan Pletschke. Fucoidan from ecklonia maxima is a powerful inhibitor of the diabetes-related enzyme, α-glucosidase. *International journal of biological macromolecules*, 151:412–420, 2020.

HFS+03 Michael Hucka, Andrew Finney, Herbert M Sauro, Hamid Bolouri, John C Doyle, Hiroaki Kitano, Adam P Arkin, Benjamin J Bornstein, Dennis Bray, Athel Cornish-Bowden, and others. The systems biology markup language (sbml): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19(4):524–531, 2003.

LLWZ08 Bo Li, Fei Lu, Xinjun Wei, and Ruixiang Zhao. Fucoidan: structure and bioactivity. *Molecules*, 13(8):1671–1695, 2008.

Ple21 Jürgen Pleiss. Standardized data, scalable documentation, sustainable storage–enzymeml as a basis for fair data management in biocatalysis. *ChemCatChem*, 13(18):3909–3913, 2021.

TAB+14 Keith F Tipton, Richard N Armstrong, Barbara M Bakker, Amos Bairoch, Athel Cornish-Bowden, Peter J Halling, Jan-Hendrik Hofmeyr, Thomas S Leyh, Carsten Kettner, Frank M Raushel, and others. Standards for reporting enzyme data: the strenda consortium: what it aims to do and why it should be helpful. *Perspectives in Science*, 1(1-6):131–137, 2014.

WK08 Petri Widsten and Andreas Kandelbauer. Laccase applications in the forest products industry: a review. *Enzyme and microbial technology*, 42(4):293–307, 2008.

WDA+16 Mark D Wilkinson, Michel Dumontier, Ij J Aalbersberg, G Appleton, M Axton, A Baak, N Blomberg, JW Boiten, LB da Silva Santos, PE Bourne, and others. The fair guiding principles for scientific data management and stewardship. sci data 3: 160018. 2016.

By Max Häußler

© Copyright 2022.