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, Max Häußler, hereby declare, that…

Abstract

here

English title

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 meta data of biocatalytic reactions. Thereby, properties of the reaction vessel, molecules involved in the reaction, as well as reaction properties like stoichiometry, pH, and temperature can be documented. Furthermore, properties of the protein catalyst like sequence, EC-number, UniProt ID can be stored alongside of potential reaction modifiers like inhibitory molecules. [Pleiss, 2021]

Within the measurements section of the data model, measurement data of one or multiple measurements is stored. Each measurement contains data of identical measurement conditions as well as information about these conditions. Thus, data and meta data from experiments with differing conditions can be stored in a structured way.

The data model consists of descriptions for the reaction vessel,

CaliPytion

Data model for calibration data

Linear and non-linear fitting of calibration equations

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.

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m + S} \tag{1}$$

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m * (1 + \frac{I}{K_{ic}}) + S} \tag{2}$$

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m * (1 + \frac{I}{K_{iu}}) * S} \tag{3}$$

$$\frac{dS}{dt} = -\frac{k_{cat} * E * S}{K_m * (1 + \frac{I}{K_{ic}}) + (1 + \frac{I}{K_{iu}}) + S} \tag{4}$$

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:

$$\frac{dE}{dt} = -K_{inact} * E \tag{5}$$

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 Visualization

4. Model selection

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.

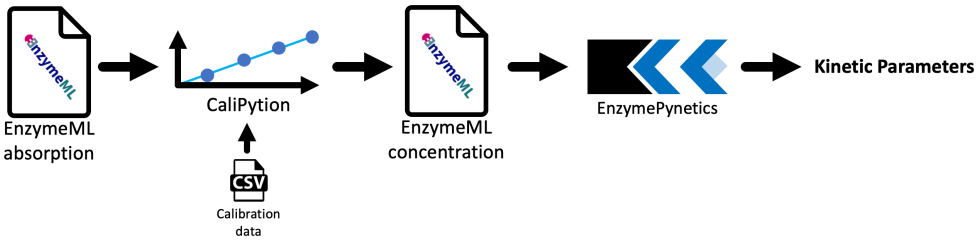


Fig. 1: Workflow for kinetic parameter estimation of enzyme reactions.

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 designed albumin fusion protein

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

Project background

In this scenaro, the inhibitory effect of a human serum albumin mutant on chymotrypsin was investigated. Thereby, the inhibition constant K_i of the HSA wild-type was compared to HSA(M3) mutant. Both HSAs were individually dimerized into fusion proteins through a huFc.
Experimental data from the HSA(wt)-huFc and HSA(M3)-huFc originate from two independent experiments. In each experiment the initial substrate concentration was varied. Once with the respective inhibitor, and once without. Therefore, K_i and K_m were determined indenpendently from each other. Additionally, each individual reaction condition was prepared in duplicates to ensure repeatability.
Since

Experimental design

In order to assess the effect of the introduced mutations to the HSA(M3) variant, K_i of the HSA wild-type was compared to HSA(M3) variant. 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) and HSA(M3) on chymotrypsin were investigated in independent experiments. Each experiment consisted of enzyme reactions with and without the respective HSA variant. The enzyme reactions contined dfhdfhdfh µM of enzyme and 26.88 µM of the respective HSA variant, if applied.

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
from IPython.display import display
from EnzymePynetics.tools.parameterestimator import ParameterEstimator
from CaliPytion.tools.standardcurve import StandardCurve

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

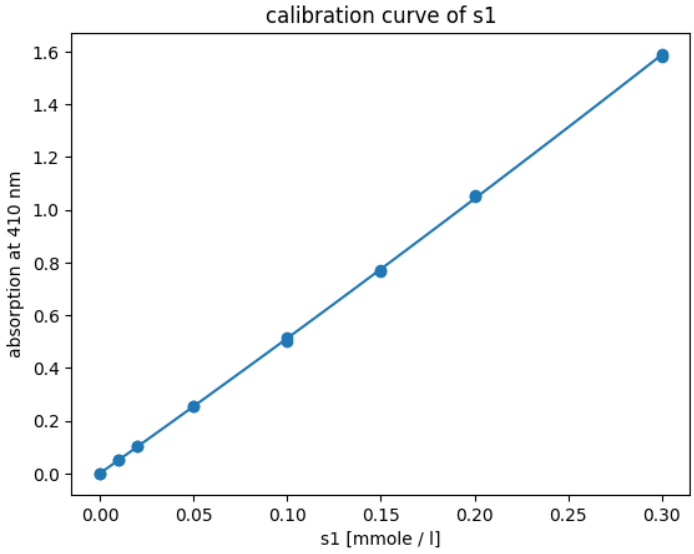
Concentration calculation

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



Based on the Akaike information criterion (AIC), the relation between concentration and absorption is best described by a quadratic function. The figure above visualizes the calibration measurements and the fitted calibration model.

Experimantal data

The EnzymeML documents were loaded and the standard curve was applied to the absorption data for concentration calculation.

```
# 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 the experimental data from the HSA wild-type and the HSA(M3) variant originate from independent experiments, the control reactions without the respective inhibitor were compared by performing a parameter estimation. Thereby, k_{cat} and K_m were highly correlated (corr > 0.98). Hence, catalytic efficiency $\frac{k_{cat}}{K_m}$ was used to assess comparability between the data sets. High correlation indicates, that the highest initial substrate concentration is too low, compared to the true K_m of the enzyme under the given experimental conditions. In this case, 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. On the other hand, high substrate viscosity denied the application of higher concentrations without sacrificing pipetting percision.

```
# 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.drop(columns=["kcat [1/min]", ]))


# 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)


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])

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	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
irreversible Michaelis Menten	-1857	2.226 +/- 7.28%	25.353 +/- 8.57%	-	-
competitive product inhibition	-1856	2.342 +/- 9.88%	25.714 +/- 13.37%	0.725 +/- 112.65%	-
uncompetitive product inhibition	-1855	2.276 +/- 10.65%	25.272 +/- 13.50%	-	3.175 +/- 428.92%
substrate inhibition	-1855	2.233 +/- 35.51%	25.339 +/- 45.93%	-	997.715 +/- 12275.11%
non-competitive product inhibition	-1854	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%

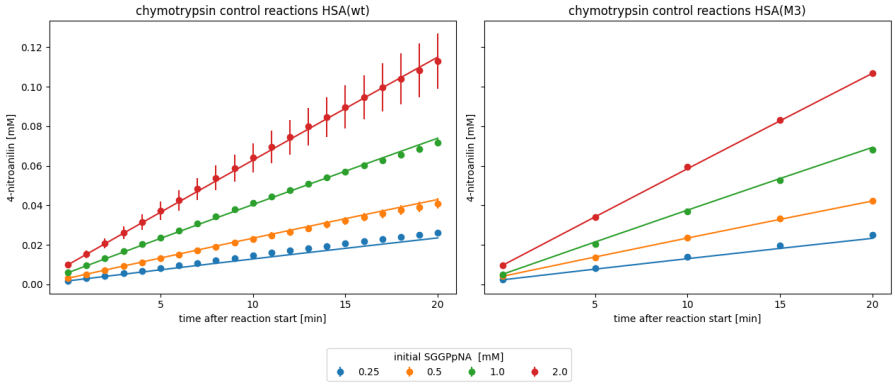


Fig. 1: Measurement data and fitted irreversible Michaelis-Menten model for chymotrypsin reactions without HSA inhibitor.

The above figure visualizes the control reactions of each HSA inhibition experiment. Each dataset was fitted against the models listed in the above table. 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. Thus, substrate and product inhibition were ruled out for the given reactions. $\frac{k_{cat}}{K_m}$ was estimated to be 25.353 $min^{-1}mM^{-1}$ ± 8.57% for the control reaction of the HSA(wt) data set and 24.776 $min^{-1}mM^{-1}$ ± 4.35% for the HSA(M3) 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

The data set of chymotrypsin inhibition by HSA(M3) contained negative absorption values for the first measurement point. Presumably from an incorrect blank measurement. Therefore, only measurement data from the second data point (minute 5 and onward) was considered for parameter estimation. Additionally, measuremet The tables below show the parameter estimastes for all applied kinetic models.

```
# 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):")
display(kinetics_HSAwt.result_dict.drop(columns=["kcat [1/min]", "Km [mmole /
l]"]))

# 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):")
display(kinetics_HSAM3.result_dict.drop(columns=["kcat [1/min]", "Km [mmole /
l]"]))

# 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(Chymo-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])

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):
```

	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):
```

	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%

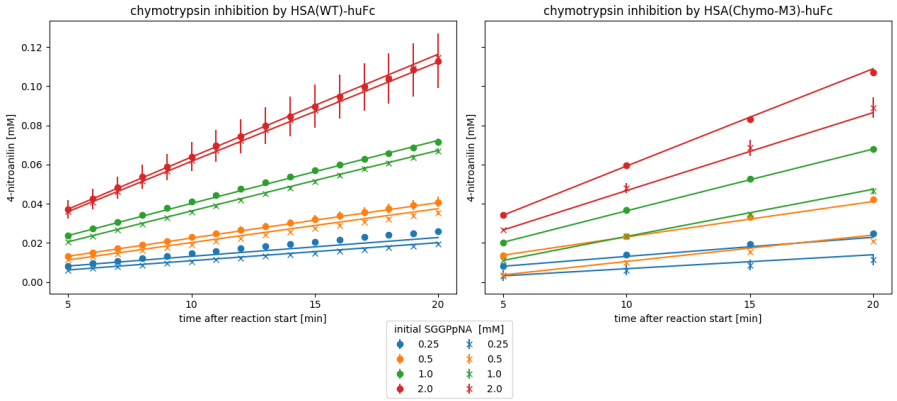
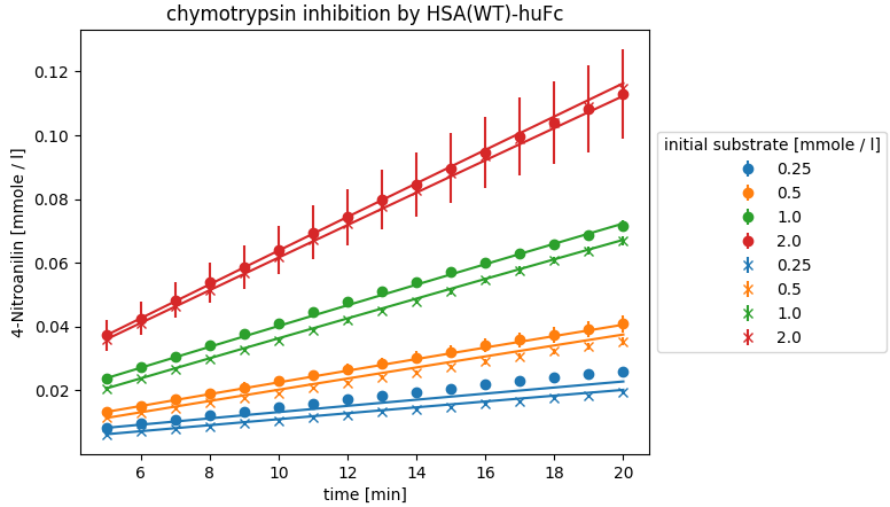


Fig. 1: Measurement data and fitted product inhibition model for chymotrypsin reactions with respective HSA inhibitior.

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) and 0.059 mM ± 8.51% for HSA(M3). 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) presumably interacts with the enzyme in the active site region.

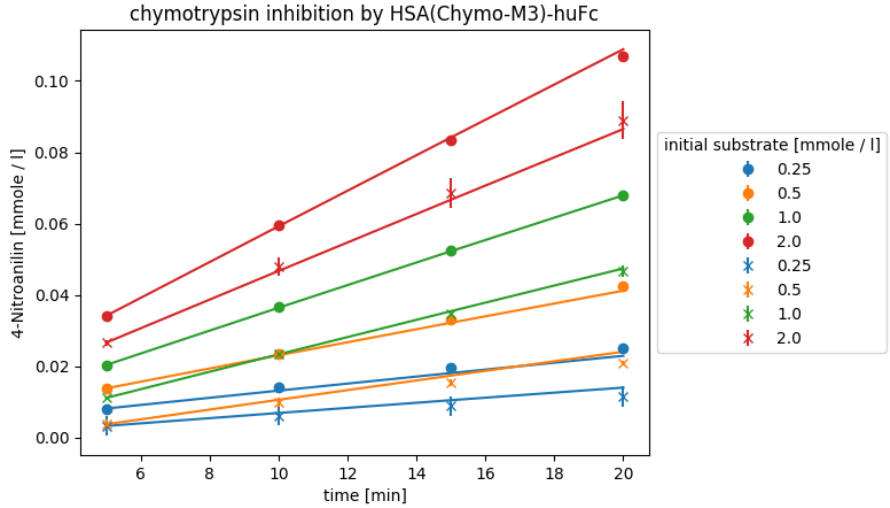
```
kinetics_HSAwt.visualize(title="chymotrypsin inhibition by HSA(WT)-huFc")
```

```
Fit report for competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 36
    # data points      = 256
    # variables        = 3
    chi-square         = 0.00137490
    reduced chi-square = 5.4344e-06
    Akaike info crit   = -3100.44585
    Bayesian info crit = -3089.81032
[[Variables]]
    k_cat:  68.6263364 +/- 2.69078924 (3.92%) (init = 32.216)
    Km:     3.08387875 +/- 0.17764014 (5.76%) (init = 0.32216)
    K_ic:   0.45950740 +/- 0.12515388 (27.24%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, Km) = 0.983
    C(Km, K_ic)  = 0.152
```




```
kinetics_HSAM3.visualize(title="chymotrypsin inhibition by HSA(Chymo-M3)-huFc")
```

```
Fit report for competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 59
    # data points      = 64
    # variables        = 3
    chi-square         = 1.9077e-04
    reduced chi-square = 3.1274e-06
    Akaike info crit   = -808.292970
    Bayesian info crit = -801.816321
[[Variables]]
    k_cat:  56.8160006 +/- 3.20785889 (5.65%) (init = 25.93928)
    Km:     2.46689202 +/- 0.21789814 (8.83%) (init = 0.2593928)
    K_ic:   0.05940519 +/- 0.00505739 (8.51%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, Km)   = 0.982
    C(Km, K_ic)    = 0.388
    C(k_cat, K_ic) = 0.273
```



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, a polysaccharide found in brown algae, on α-glucosidase from *Saccharomyces cerevisiae* was investigated. Fucoidans are actively investigated in the fields of anit-cancer, anti-inflammation, and anti-coagulate, to name a fiew (Li *et al.* [2008]). In a previous study (Daub *et al.* [2020]), fucoidan from *E. maxima* showed an almost 2-fold lower IC_{50} value, compared to acarbose. Thus, fucoidan is a potential antidiabetic drug candidate for the treatment of diabetes mellitus. In the following analysis, fucoidan from the brown algae species *Ecklonia maxima*, *Ecklonia radiata*, *Fucus vesiculosus*, S. CYM???????, and *Schimmelmannia elegans* were investigated for their inhibition constant K_i for α-glucosidase inhibition.

Experimental design

Extracted fucoidan from the mentioned brown algea species was applied in two different concentrations to enzyme reactions. Additionally, control reactions without inhibitor were performed. For the enzyme reactions, *p*-nitrophenyl-α-D-glucopyranoside (pNPG) was applied as a substrate in a concentration range from 0.1 - 5 mM. Product accumulatin was followed photometrically in a micro titer plate at 405 nm for 20 min.

Data preparation

Imports

```
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 CaliPytion.tools.standardcurve import StandardCurve
from EnzymePynetics.tools.parameterestimator import ParameterEstimator


import warnings
warnings.filterwarnings('ignore')

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

Experimental data

Time-course data of all kinetic experiments was collected in an Excel file, whereas the meta data was specified in individual EnzymeML Excel templates for each origin species of fucoidan. In the following cell the experimental data is loaded and blanked to subtract the absorbance contribution of enzyme, buffer, and the respective inhibitor for each measurement. Thereafter, the data is written to EnzymeML documents.

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

inhibitors = sorted(pd.ExcelFile(dataset_path).sheet_names)
initial_substrates = [0.1, 0.25, 0.5, 1, 2.5, 5]


data_dict = {}
for inhibitor in inhibitors:
    df = pd.read_excel(dataset_path, sheet_name=inhibitor).set_index("time")
    blanc_no_inhibitor = df["buffer + enzyme"].mean()
    blank_low_inhibitor = df[df.columns[[1,2]]].values.mean()
    blank_high_inhibitor = df[df.columns[[3,4]]].values.mean()
    df = df.iloc[:,17:]
    keys = sorted(df.columns)
    df_no_inhibitor = df[keys[:12]]
    df_low_inhibitor = df[keys[12:24]]
    df_high_inhibitor = df[keys[24:]]

    df_no_inhibitor = df_no_inhibitor.subtract(blanc_no_inhibitor)
    df_low_inhibitor = df_low_inhibitor.subtract(blank_low_inhibitor)
    df_high_inhibitor = df_high_inhibitor.subtract(blank_high_inhibitor)


    data = []
    data.append(df_no_inhibitor.values.T)
    data.append(df_low_inhibitor.values.T)
    data.append(df_high_inhibitor.values.T)

    data_dict[inhibitor] = np.array(data).reshape(18,2,20)

time = df.index.values

# Parse measurement data to EnzymeML documents
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

enzml_docs = []
datas = list(data_dict.values())
for file, data in zip(sorted(os.listdir(template_directory)), datas):
    enzml_docs.append(measurement_data_to_EnzymeML(
        template_path=f"{template_directory}/{file}",
        measurement_data=data,
        time=time,
        species_id="s1",
        data_unit="mmole / l",
        time_unit="min"
    ))
```

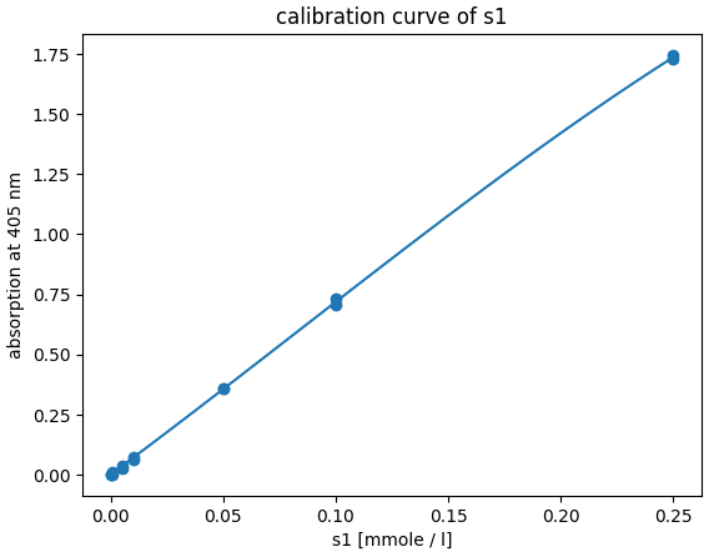
Concentration calculation

Standard data of the product was loaded from an excel file, and a standard curve was created. Then, the standard curve was applied to the EnzymeML documents, containing the absorption measurements for concentration calculation.

```
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



Parameter estimation

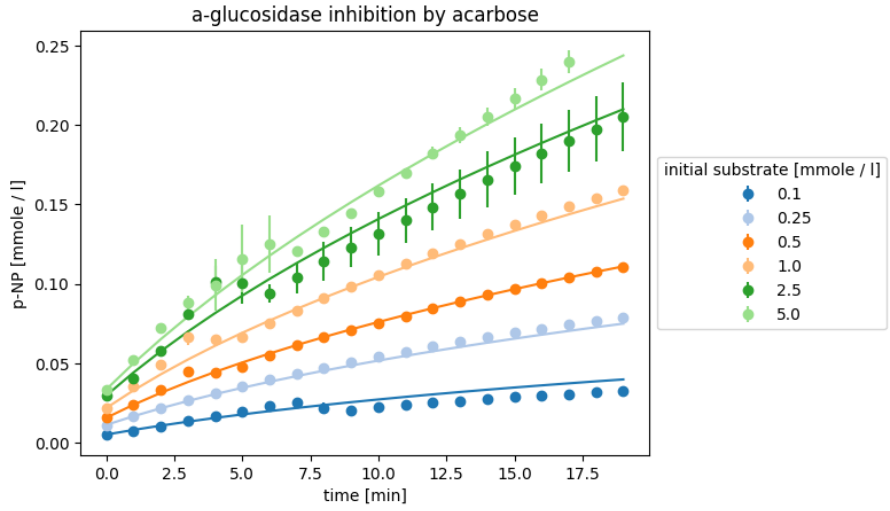
Parameter estimation was perfomed with EnzymePynetics. Thereby, each data set was fitted to competitive, uncompetitive, and non-competitive inhibition models.

```
kinetics = []
for enzmldoc in enzml_docs:
    result = ParameterEstimator.from_EnzymeML(enzmldoc=enzmldoc,reactant_id="s1",
inhibitor_id="s2", measured_species="product")
    result.fit_models()
    kinetics.append(result)
    result.visualize(plot_means=True)
    plt.show()
```

```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive product inhibition model
 - uncompetitive product inhibition model
 - non-competitive product inhibition model
 - substrate inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
non-competitive product inhibition	-2257	2480.867 +/- 10.41%	0.545 +/- 19.96%	4554.048 +/- 22.51%	0.040 +/- 24.85%	0.249 +/- 34.07%
competitive product inhibition	-2243	1756.430 +/- 2.72%	0.286 +/- 15.37%	6132.817 +/- 15.61%	0.027 +/- 25.68%	-
uncompetitive product inhibition	-2187	2912.596 +/- 14.85%	1.432 +/- 17.61%	2034.270 +/- 23.03%	-	0.108 +/- 29.22%
irreversible Michaelis Menten	-2146	1429.827 +/- 1.55%	0.603 +/- 4.92%	2372.038 +/- 5.16%	-	-
substrate inhibition	-2139	1455.278 +/- 1.73%	0.623 +/- 5.31%	2334.831 +/- 5.59%	-	248.634 +/- 61.85%

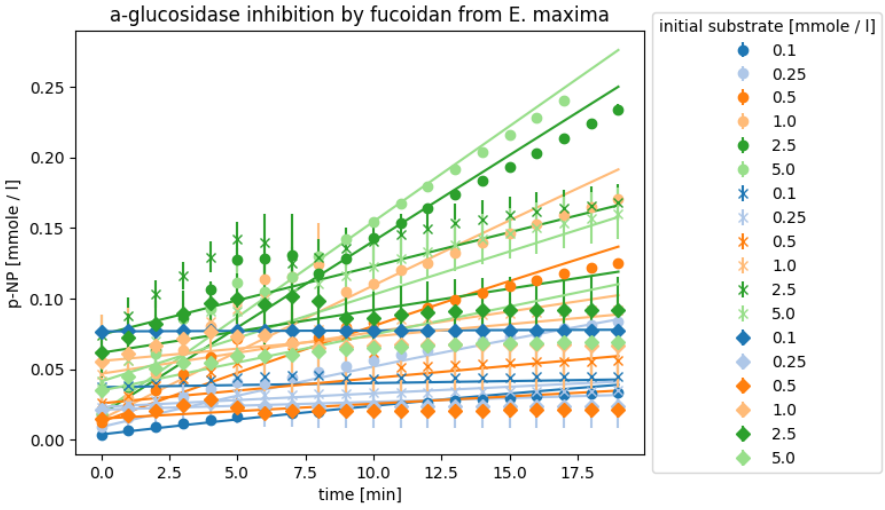
```
Fit report for non-competitive product inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 54
    # data points      = 237
    # variables        = 4
    chi-square         = 0.01676347
    reduced chi-square = 7.1946e-05
    Akaike info crit   = -2256.91734
    Bayesian info crit = -2243.04510
[[Variables]]
    k_cat:  2480.86683 +/- 258.151536 (10.41%) (init = 3652.667)
    Km:     0.54476083 +/- 0.10872800 (19.96%) (init = 1.6784)
    K_iu:   0.24899154 +/- 0.08483063 (34.07%) (init = 0.1)
    K_ic:   0.04004974 +/- 0.00995050 (24.85%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, K_iu) = -0.964
    C(Km, K_iu)    = -0.808
    C(Km, K_ic)    = 0.771
    C(k_cat, Km)   = 0.731
    C(K_iu, K_ic)  = -0.324
    C(k_cat, K_ic) = 0.150
```




```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive inhibition model
 - uncompetitive inhibition model
 - non-competitive inhibition model
 - partially competitive inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [g / l]	Ki uncompetitive [g / l]
non-competitive inhibition	-6142	1639.222 +/- 1.65%	0.552 +/- 5.44%	2971.298 +/- 5.68%	0.109 +/- 10.27%	0.602 +/- 10.61%
competitive inhibition	-6055	1610.235 +/- 1.73%	0.526 +/- 5.80%	3062.265 +/- 6.05%	0.052 +/- 5.48%	-
partially competitive inhibition	-6053	1610.208 +/- 1.74%	0.526 +/- 5.80%	3062.405 +/- 6.05%	0.052 +/- 5.75%	1000.000 +/- 3817.44%
uncompetitive inhibition	-5819	1746.747 +/- 2.21%	0.742 +/- 6.49%	2353.001 +/- 6.86%	-	0.214 +/- 4.57%
irreversible Michaelis Menten	-4753	945.417 +/- 5.07%	0.809 +/- 15.18%	1168.173 +/- 16.00%	-	-

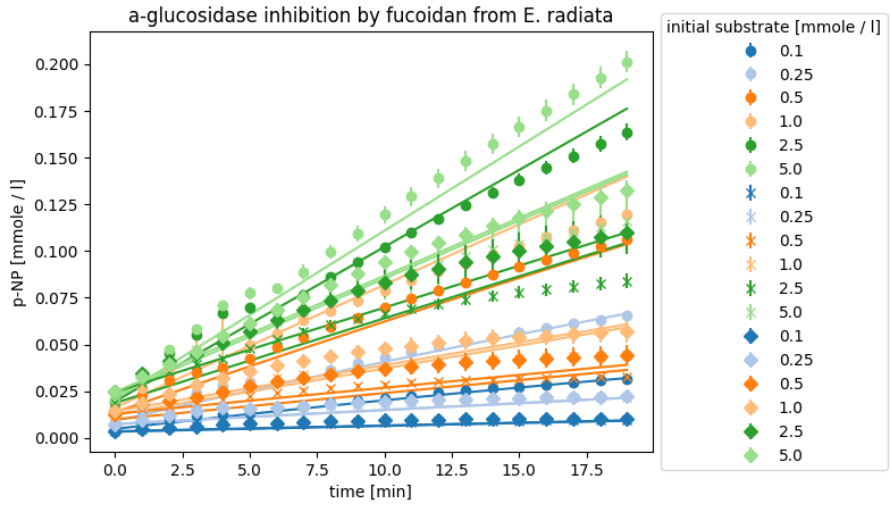
```
Fit report for non-competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 31
    # data points      = 717
    # variables        = 4
    chi-square         = 0.13495950
    reduced chi-square = 1.8928e-04
    Akaike info crit   = -6142.32304
    Bayesian info crit = -6124.02274
[[Variables]]
    k_cat:  1639.22182 +/- 27.1273263 (1.65%) (init = 4965.792)
    Km:     0.55168535 +/- 0.03000408 (5.44%) (init = 2.281781)
    K_iu:   0.60181800 +/- 0.06386257 (10.61%) (init = 0.1)
    K_ic:   0.10943768 +/- 0.01123454 (10.27%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, Km)   = 0.822
    C(K_iu, K_ic)  = -0.765
    C(Km, K_ic)    = 0.469
    C(k_cat, K_ic) = 0.322
    C(k_cat, K_iu) = -0.306
    C(Km, K_iu)    = -0.253
```




```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive inhibition model
 - uncompetitive inhibition model
 - non-competitive inhibition model
 - partially competitive inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [g / l]	Ki uncompetitive [g / l]
partially competitive inhibition	-7058	1075.485 +/- 1.11%	0.478 +/- 3.97%	2252.123 +/- 4.12%	0.000 +/- 271.89%	0.001 +/- 728.83%
non-competitive inhibition	-6658	1059.810 +/- 1.56%	0.483 +/- 5.51%	2194.677 +/- 5.73%	0.017 +/- 7.98%	0.789 +/- 48.63%
competitive inhibition	-6656	1046.201 +/- 1.46%	0.464 +/- 5.32%	2253.506 +/- 5.52%	0.015 +/- 5.33%	-
uncompetitive inhibition	-6260	1291.208 +/- 2.47%	1.151 +/- 6.12%	1121.648 +/- 6.60%	-	0.065 +/- 5.45%
irreversible Michaelis Menten	-5818	921.806 +/- 2.90%	1.070 +/- 8.06%	861.615 +/- 8.57%	-	-

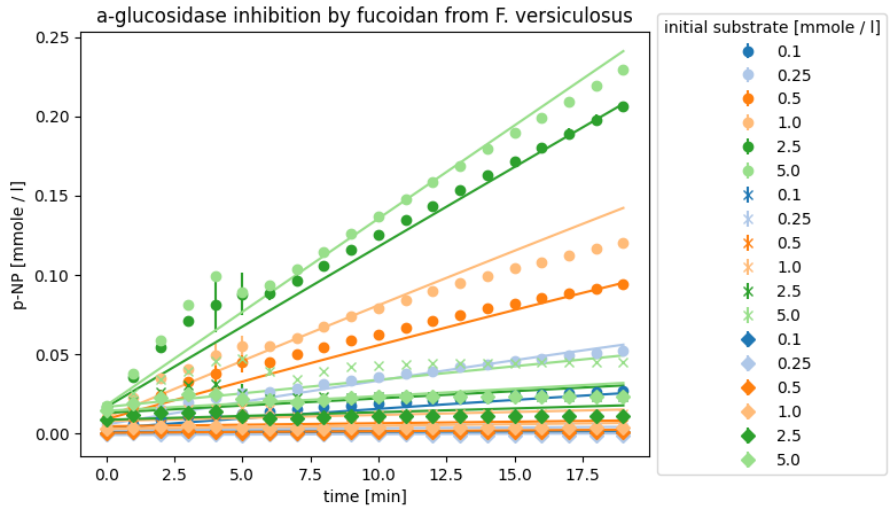
```
Fit report for partially competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 272
    # data points      = 720
    # variables        = 4
    chi-square         = 0.03936609
    reduced chi-square = 5.4981e-05
    Akaike info crit   = -7058.15327
    Bayesian info crit = -7039.83626
[[Variables]]
    k_cat:  1075.48537 +/- 11.8902583 (1.11%) (init = 1620.85)
    Km:     0.47754289 +/- 0.01896377 (3.97%) (init = 0.7447807)
    K_iu:   6.1448e-04 +/- 0.00447849 (728.83%) (init = 0.1)
    K_ic:   1.0000e-04 +/- 2.7189e-04 (271.89%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(K_iu, K_ic) = -1.000
    C(k_cat, Km)  = 0.779
```




```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive inhibition model
 - uncompetitive inhibition model
 - non-competitive inhibition model
 - partially competitive inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [g / l]	Ki uncompetitive [g / l]
competitive inhibition	-7326	1527.487 +/- 0.99%	0.938 +/- 2.87%	1628.994 +/- 3.04%	0.001 +/- 4.85%	-
non-competitive inhibition	-7324	1527.536 +/- 0.99%	0.938 +/- 2.88%	1628.803 +/- 3.04%	0.001 +/- 14.89%	15.494 +/- 36228.16%
partially competitive inhibition	-7324	1527.908 +/- 0.99%	0.939 +/- 2.88%	1627.963 +/- 3.04%	0.001 +/- 7.82%	999.141 +/- 139876.39%
uncompetitive inhibition	-7028	1531.314 +/- 1.22%	0.947 +/- 3.53%	1616.521 +/- 3.74%	-	0.002 +/- 7.52%
irreversible Michaelis Menten	-4775	667.025 +/- 10.33%	1.455 +/- 26.07%	458.453 +/- 28.04%	-	-

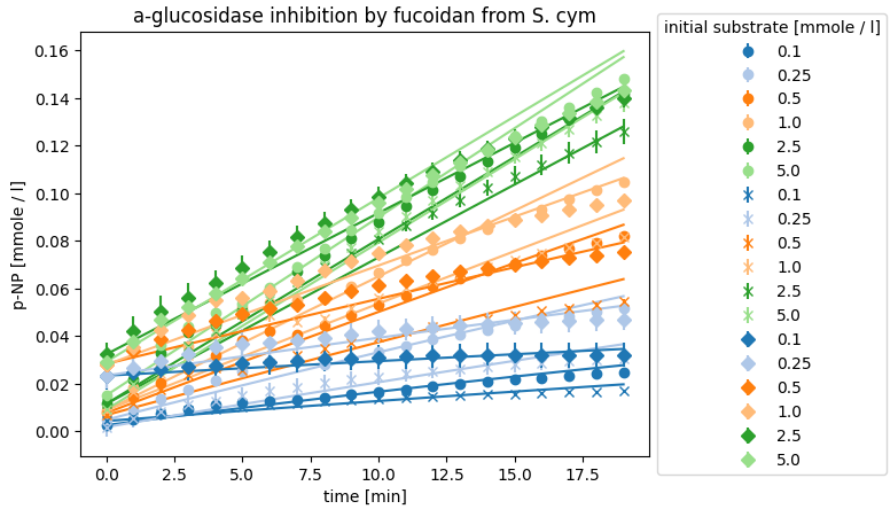
```
Fit report for competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 34
    # data points      = 720
    # variables        = 3
    chi-square         = 0.02722244
    reduced chi-square = 3.7967e-05
    Akaike info crit   = -7325.73480
    Bayesian info crit = -7311.99704
[[Variables]]
    k_cat:  1527.48678 +/- 15.1025514 (0.99%) (init = 2691.849)
    Km:     0.93768696 +/- 0.02693290 (2.87%) (init = 1.236905)
    K_ic:   0.00134723 +/- 6.5379e-05 (4.85%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, Km)    = 0.865
    C(Km, K_ic)     = 0.407
    C(k_cat, K_ic) = 0.294
```




```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive inhibition model
 - uncompetitive inhibition model
 - non-competitive inhibition model
 - partially competitive inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [g / l]	Ki uncompetitive [g / l]
partially competitive inhibition	-7611	885.193 +/- 0.65%	0.431 +/- 2.76%	2051.891 +/- 2.83%	0.175 +/- 28.00%	0.452 +/- 36.06%
non-competitive inhibition	-7574	900.852 +/- 0.92%	0.471 +/- 3.29%	1914.077 +/- 3.41%	0.966 +/- 7.73%	21.519 +/- 37.84%
competitive inhibition	-7568	885.729 +/- 0.67%	0.448 +/- 2.79%	1979.019 +/- 2.87%	0.841 +/- 5.36%	-
uncompetitive inhibition	-7324	975.489 +/- 1.06%	0.710 +/- 2.69%	1373.348 +/- 2.89%	-	3.737 +/- 6.80%
irreversible Michaelis Menten	-7096	874.452 +/- 0.94%	0.652 +/- 3.01%	1340.611 +/- 3.16%	-	-

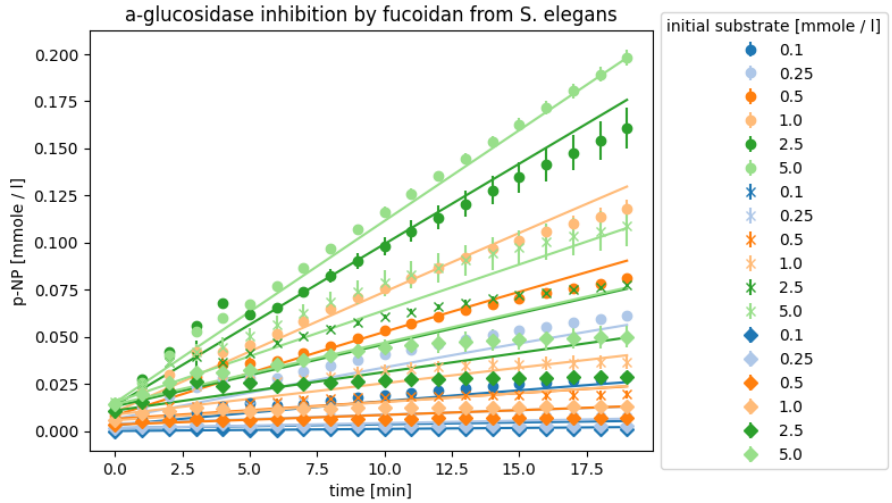
```
Fit report for partially competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 124
    # data points      = 720
    # variables        = 4
    chi-square         = 0.01825580
    reduced chi-square = 2.5497e-05
    Akaike info crit   = -7611.41698
    Bayesian info crit = -7593.09998
[[Variables]]
    k_cat:  885.192687 +/- 5.78448532 (0.65%) (init = 1279.274)
    Km:     0.43140343 +/- 0.01189835 (2.76%) (init = 0.5878264)
    K_iu:   0.45219052 +/- 0.16305078 (36.06%) (init = 0.1)
    K_ic:   0.17451878 +/- 0.04887071 (28.00%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(K_iu, K_ic) = 0.994
    C(k_cat, Km)  = 0.671
    C(Km, K_ic)   = 0.146
```




```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive inhibition model
 - uncompetitive inhibition model
 - non-competitive inhibition model
 - partially competitive inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [g / l]	Ki uncompetitive [g / l]
non-competitive inhibition	-7110	1196.091 +/- 1.22%	0.693 +/- 3.86%	1724.867 +/- 4.04%	0.008 +/- 6.67%	0.272 +/- 39.90%
competitive inhibition	-7106	1191.620 +/- 1.21%	0.686 +/- 3.85%	1736.014 +/- 4.04%	0.007 +/- 3.65%	-
partially competitive inhibition	-7104	1191.646 +/- 1.21%	0.686 +/- 3.86%	1735.964 +/- 4.04%	0.007 +/- 9.27%	999.732 +/- 110814.71%
uncompetitive inhibition	-6381	1334.869 +/- 2.31%	1.095 +/- 6.22%	1218.643 +/- 6.63%	-	0.021 +/- 4.53%
irreversible Michaelis Menten	-5320	837.262 +/- 6.08%	1.586 +/- 14.95%	527.980 +/- 16.14%	-	-

```
Fit report for non-competitive inhibition model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 48
    # data points      = 720
    # variables        = 4
    chi-square         = 0.03661395
    reduced chi-square = 5.1137e-05
    Akaike info crit   = -7110.33552
    Bayesian info crit = -7092.01851
[[Variables]]
    k_cat:  1196.09080 +/- 14.5632835 (1.22%) (init = 1686.748)
    Km:     0.69343953 +/- 0.02673319 (3.86%) (init = 0.7750608)
    K_iu:   0.27178749 +/- 0.10844409 (39.90%) (init = 0.1)
    K_ic:   0.00750637 +/- 5.0042e-04 (6.67%) (init = 0.1)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, Km)   = 0.834
    C(K_iu, K_ic)  = -0.825
    C(Km, K_ic)    = 0.477
    C(k_cat, K_ic) = 0.334
    C(k_cat, K_iu) = -0.159
    C(Km, K_iu)    = -0.135
```



k_{cat} and K_m

```
# Get kinetic parameters of all datasets
kcat = []
kcat_std = []
Km = []
Km_std = []
corr_kcat_km = []
for result in kinetics:
    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)


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


df = pd.DataFrame.from_dict({
    'kcat [1/min]':kcat,
    'kcat stderr':kcat_std,
    'Km [mM]':Km,
    'Km stderr':Km_std,
    "correlation kcat/Km":corr_kcat_km})

df
```

	kcat [1/min]	kcat stderr	Km [mM]	Km stderr	correlation kcat/Km
0	1650.543562	68.567687	0.537270	0.080309	-0.434760
1	1768.422752	102.033022	1.461047	0.200855	0.912771
2	1089.309474	15.733598	0.676981	0.031014	0.826747
3	1474.870342	11.446200	1.260928	0.025848	0.893857
4	824.901555	5.351402	0.510622	0.012484	0.722688
5	1163.327292	17.444942	0.916960	0.039877	0.861308

```
enzmldoc = enzml_docs[0]

del enzmldoc.measurement_dict["m6"]
del enzmldoc.measurement_dict["m7"]
del enzmldoc.measurement_dict["m8"]
del enzmldoc.measurement_dict["m9"]
del enzmldoc.measurement_dict["m10"]
del enzmldoc.measurement_dict["m11"]
del enzmldoc.measurement_dict["m12"]
del enzmldoc.measurement_dict["m13"]
del enzmldoc.measurement_dict["m14"]
del enzmldoc.measurement_dict["m15"]
del enzmldoc.measurement_dict["m16"]
del enzmldoc.measurement_dict["m17"]
```

```
kinetics = ParameterEstimator.from_EnzymeML(
    enzmldoc=enzmldoc,
    reactant_id="s1",
    measured_species="product")

kinetics.fit_models(enzyme_inactivation=True)
```

```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive product inhibition model
 - uncompetitive product inhibition model
 - non-competitive product inhibition model
 - substrate inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	ki time-dep enzyme-inactiv. [1/min]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
competitive product inhibition	-2257	1879.010 +/- 2.96%	0.398 +/- 11.56%	0.023 +/- 24.42%	4716.196 +/- 11.93%	0.065 +/- 32.40%	-
non-competitive product inhibition	-2255	2477.180 +/- 24.37%	0.544 +/- 30.17%	0.000 +/- 3906.23%	4554.158 +/- 38.78%	0.040 +/- 39.86%	0.251 +/- 99.55%
irreversible Michaelis Menten	-2231	1883.439 +/- 2.90%	0.582 +/- 4.10%	0.043 +/- 10.25%	3234.668 +/- 5.03%	-	-
uncompetitive product inhibition	-2229	1883.604 +/- 3.08%	0.582 +/- 4.11%	0.043 +/- 10.34%	3234.343 +/- 5.14%	-	844.290 +/- 4979.17%
substrate inhibition	-2227	1892.295 +/- 3.08%	0.587 +/- 4.17%	0.043 +/- 11.03%	3222.920 +/- 5.18%	-	999.940 +/- 156.07%

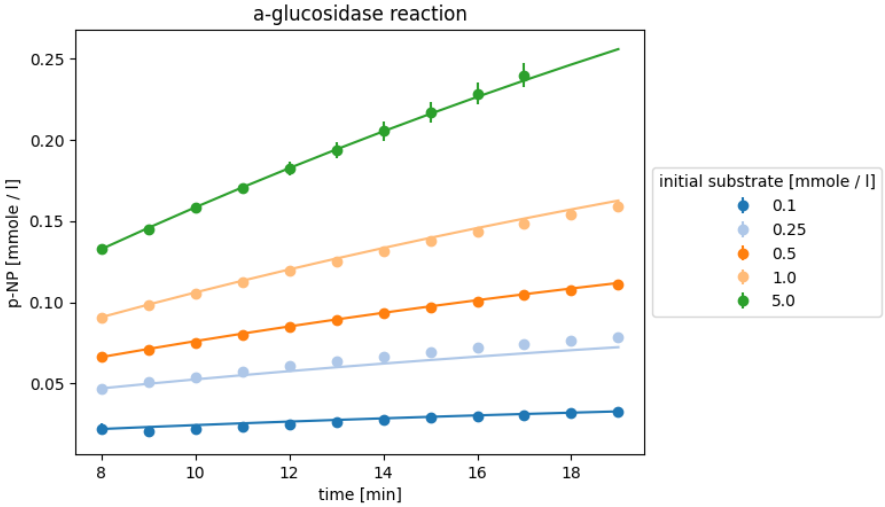
```
kinetics.fit_models(
    enzyme_inactivation=True,
    start_time_index=8,
    initial_substrate_concs=[0.1, 0.25, 0.5, 1, 5])
```

```
Fitting data to:
 - irreversible Michaelis Menten model
 - competitive product inhibition model
 - uncompetitive product inhibition model
 - non-competitive product inhibition model
 - substrate inhibition model
```

	AIC	kcat [1/min]	Km [mmole / l]	ki time-dep enzyme-inactiv. [1/min]	kcat / Km [1/min * 1/mmole / l]	Ki competitive [mmole / l]	Ki uncompetitive [mmole / l]
competitive product inhibition	-1367	1807.855 +/- 3.09%	0.660 +/- 8.33%	0.018 +/- 42.60%	2738.363 +/- 8.88%	0.145 +/- 28.45%	-
non-competitive product inhibition	-1365	1808.173 +/- 17.82%	0.660 +/- 19.81%	0.018 +/- 54.17%	2738.303 +/- 26.65%	0.145 +/- 35.04%	954.402 +/- 86749.69%
irreversible Michaelis Menten	-1342	1716.141 +/- 3.08%	0.907 +/- 3.62%	0.032 +/- 25.03%	1892.619 +/- 4.75%	-	-
uncompetitive product inhibition	-1340	1716.422 +/- 3.14%	0.907 +/- 4.10%	0.032 +/- 26.29%	1892.485 +/- 5.17%	-	999.612 +/- 10450.38%
substrate inhibition	-1339	1725.403 +/- 4.13%	0.914 +/- 5.89%	0.032 +/- 25.24%	1886.983 +/- 7.19%	-	999.998 +/- 575.44%

```
kinetics.visualize(model_name="irreversible Michaelis Menten", title="a-glucosidase reaction")
```

```
Fit report for irreversible Michaelis Menten model
[[Fit Statistics]]
    # fitting method   = leastsq
    # function evals   = 21
    # data points      = 117
    # variables        = 3
    chi-square         = 0.00116138
    reduced chi-square = 1.0188e-05
    Akaike info crit   = -1341.87794
    Bayesian info crit = -1333.59141
[[Variables]]
    k_cat:  1716.14120 +/- 52.8030058 (3.08%) (init = 3652.667)
    Km:     0.90675489 +/- 0.03286791 (3.62%) (init = 1.6784)
    K_ie:   0.03177073 +/- 0.00795342 (25.03%) (init = 0.01)
[[Correlations]] (unreported correlations are < 0.100)
    C(k_cat, K_ie) = 0.883
    C(Km, K_ie)    = -0.210
    C(k_cat, Km)   = 0.198
```



Scenario C:
SLAC characterization

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

Project background

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 between pH 3 - 5.5 and temperature range between 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 a seperate ABTS standard curve and absorption spectrum was recorded, in order to account for varying ABTS absorption properties do to reaction conditions. Each enzyme reaction was followed for 15 min photometrically at two seperate wavelengths, measuring substrate depletion and product accumululation simultaneously. Therefore, preliminary experiments confirmed, that ABTS absorbs at 340 nm, whereas the $ABTS^{+\cdot}$ radical product absorbs at 420 nm. 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 preperation was automated by custom parser functions, which were tailored to the output of the used spectro photometer. 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 .txt output of the photometer. Figure 1 illustrates the schematic data flow of this project.

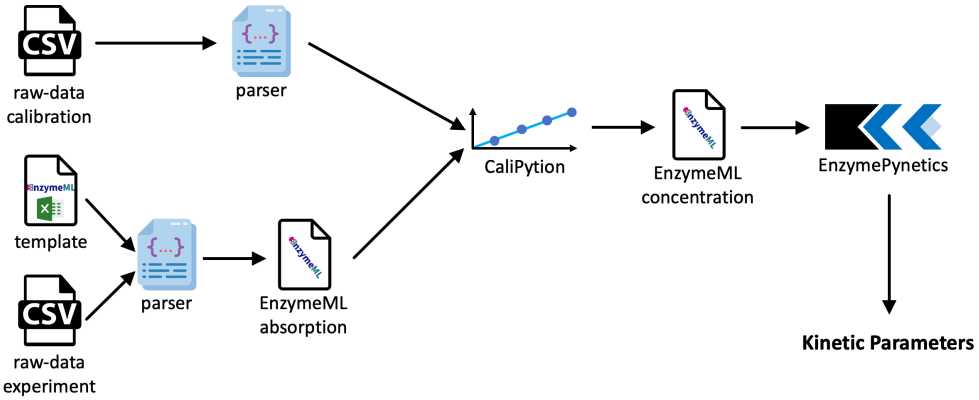


Fig. 1: Schematic data pipeline of the SLAC characterization.

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
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

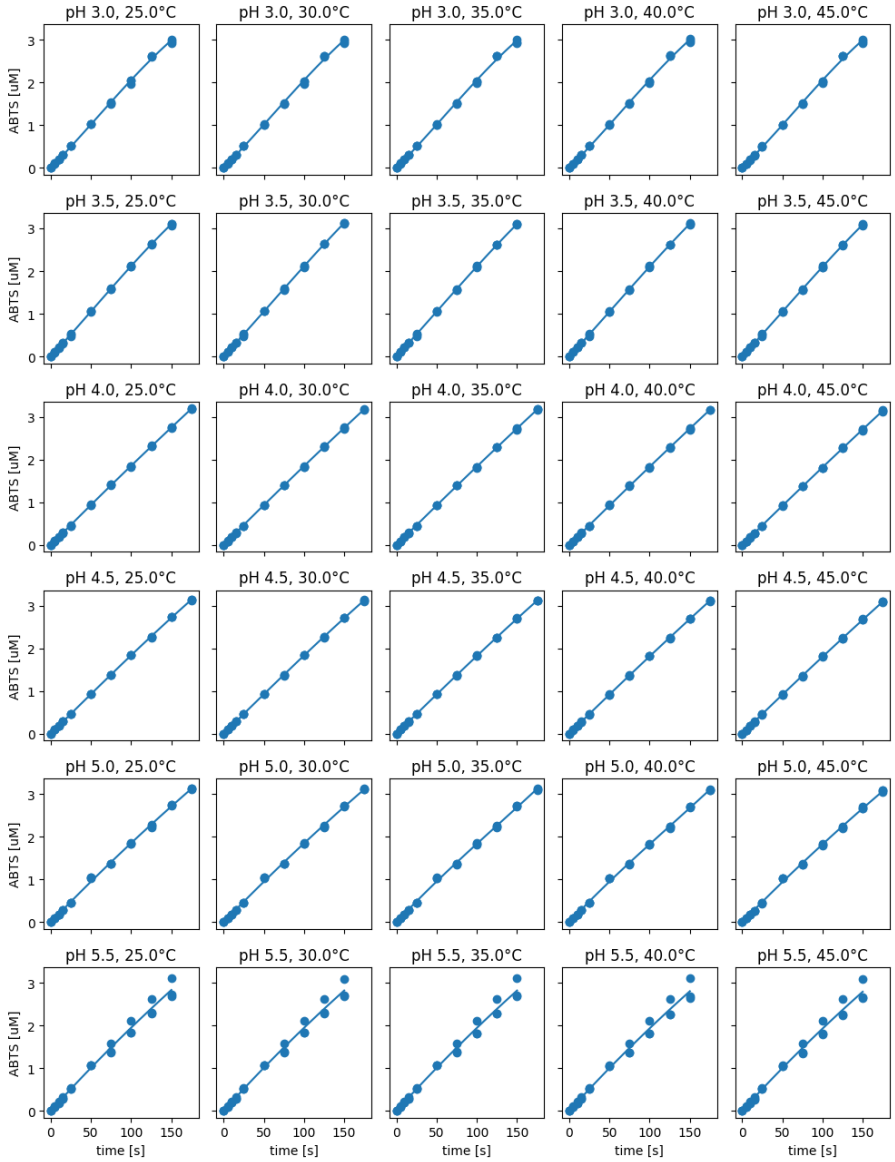
Data from the enzyme reactions was loaded from the output files of the photometer and written to individual EnzymeML documents. Then infromation of the control reactions was used to subtract the absorption, contributed from enzyme and buffer from the substrate and product signal.

```
# 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 spectro photometer contained a minimum of information only. Thereafter, a StandardCurve was created for each calibration data set. Thereby, only absoption values below 3 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()
```



The figure above shows the standard curves for all experimental conditions of this scenario. Since the absorption proterties ABTS change with pH, the calibration range differs between some pH values, due to the upper absorption limit. 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. The sulfonate groups of ABTS are deprotonated for more neutral pH values. Therefore, the absorption properties of ABTS might 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.

Nextup, 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.

```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 conductas as a control of quality. By assuming mass conservation, the following concentration balance can be established:

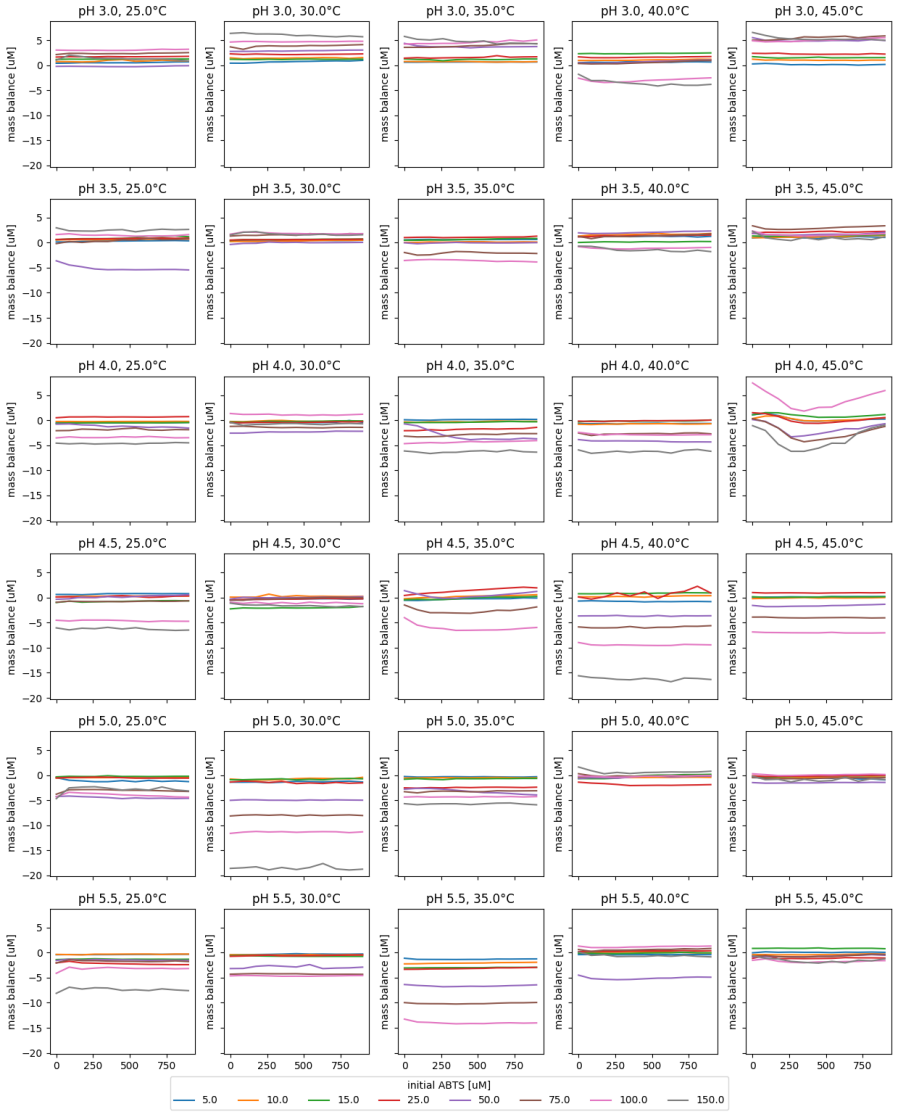
$$0 = S_{(t)} + P_{(t)} - S_0 \tag{6}$$

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 is available for the product. Therefore, an additional parameter §k§ was introdiced to the mass balance equation, assuming linear relationship between the product concentration and its signal:

$$0 = S_{(t)} + P_{(t)}k - S_0 \tag{7}$$

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 a given reaction condition. Thereafter, the mass balances of all measurements were visualized.

```
# 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()
```



The figure above shows the mass balances for all experiments. For all reactions, 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, diverging from 0 uM indicate, result likely from pipetting errors, eighter increasing or decreasing the substrate concentration in the enzyme- or control reaction. 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, presumably from the spectro photometer. Therefore, the respective measurements might 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, substrate and product inhibition models were excluded, since in some of the experiments the actual initial substrate concentration was higher than the specified initial substrate concentration. This results calculated product concentrations with negative value. Hence, product inhibition models were excluede, since no acurate information on product concentration was available.
Secondly, substrate inhibition models were excluded, since the model was not able to describe the observed reaction kinetics. This was evident from a higher Akaike information criterion 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.

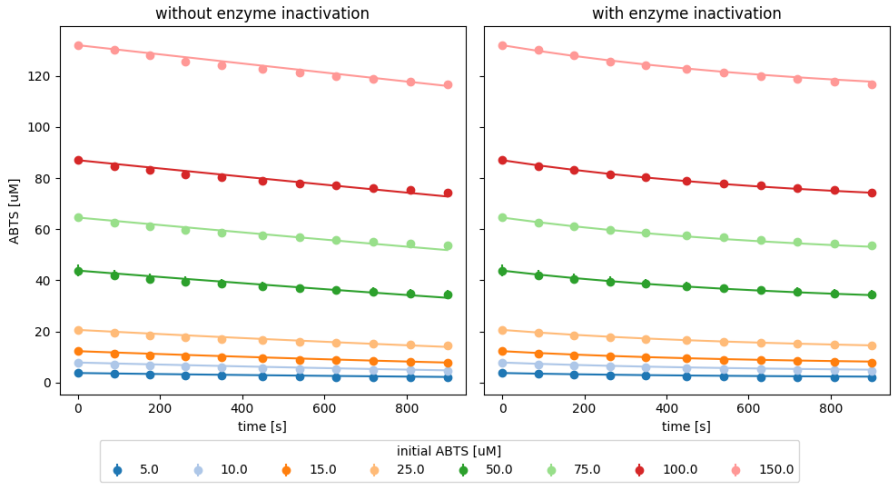
```
fig, axes = plt.subplots(1,2, figsize=(10,5), sharey=True, sharex=True)
for i, ax in enumerate(axes.flatten()):
    if i == 0:
        print("Estimated parameters without time-dependent enzyme inactivation:")
        kinetics = ParameterEstimator.from_EnzymeML(concentration_enzymemldocs[0],
"s0", "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
```

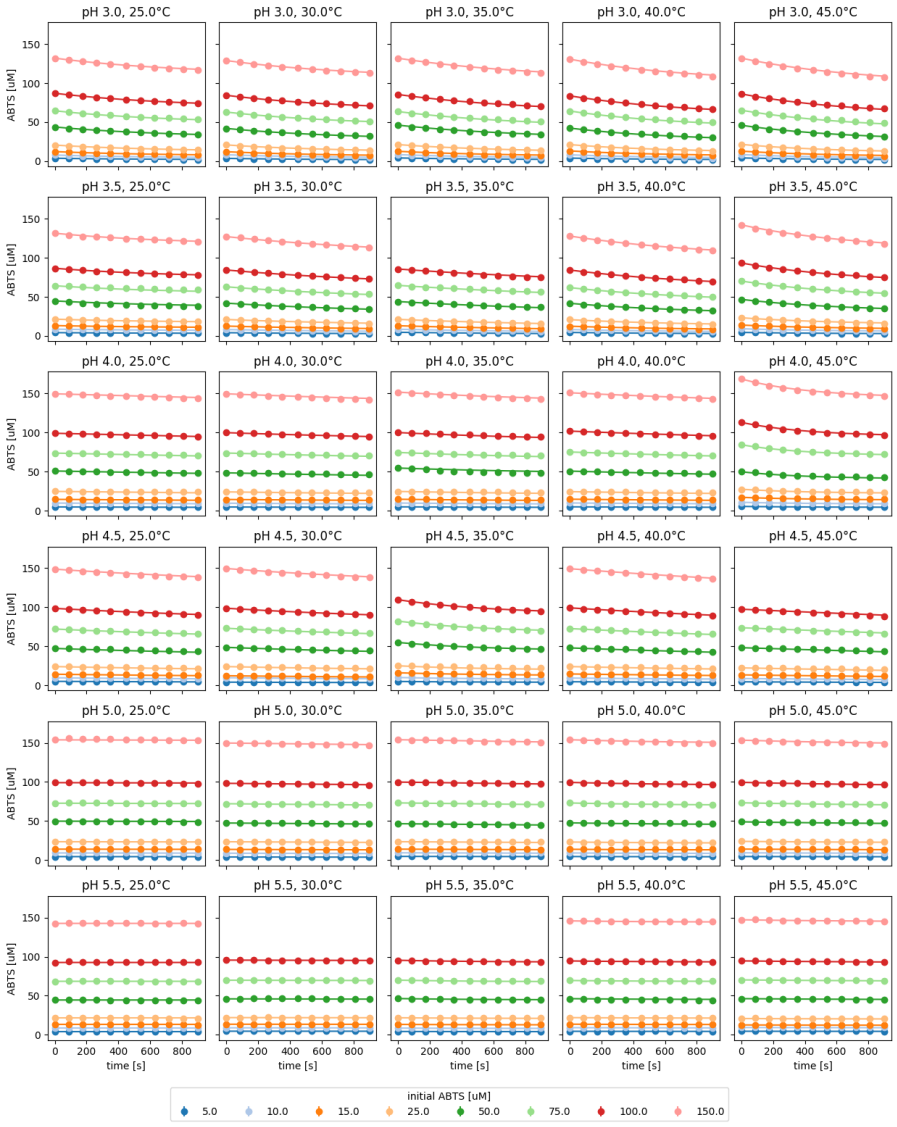
	AIC	Km [umole / l]	kcat / Km [1/s * 1/umole / l]	kcat [1/s]	ki time-dep enzyme-inactiv. [1/s]
irreversible Michaelis Menten	-476	35.137 +/- 3.00%	0.013 +/- 3.58%	0.460 +/- 1.95%	0.001 +/- 4.47%



Both plots in the above figure contain the same experimental data. The left plot shows the fitted model with enzyme inactivation and the right without enzyme inactivation, which is shown as solid lines. 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. Statistically, this is characterized through a lower Akaike information criterion, 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.

Thus, 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()
```



The above plot shows all kinetic experiments, fitted to the irreversible Michaelis-Menten model with time-dependent enzyme inactivation. Based on the reaction slopes, no catalytic avtivity is observable for reactions at pH 5 or higher. In the following cells, the resulting kinetic parameters were extracted and visualized. As in the mass balance analysis, the reactions at pH 4, 45°C and pH 4.5, 35°C differ from other experiments of the respective pH. In both cases, every applied substrate concentration is higher than

intended in the design of the experiment. Furthermore, the resulting parameter estimates have a high uncertanty. Therefore, the respective measurements are excluded from further analysis. Additionaly, the results from pH 5.5, 35°C were excluded, since the uncertainty of the parameters could not be estimated.

```
# 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

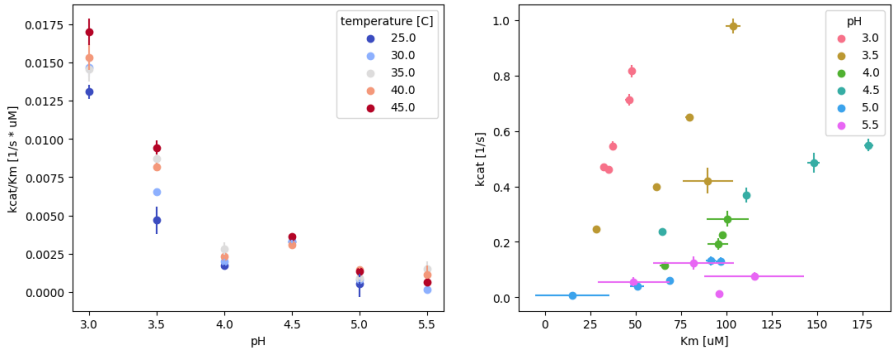
The parameter estimates for k_{cat} and K_m showed correlation. For experiments at pH 3 and pH 3.5, the parameters were correlated between 0.5 - 0.85. This indeicates, that the highest initial substrate concentration, which was applied for parameter estimation was not sufficently high. Ideally, the highest 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. In this scenario, only reactions with an initial substrate concentration of 150 uM were used for parameter estimation, due to the limited photometric measurement range of ABTS. Hence, ABTS was only applied 1.5-fold to 6-fold of the estimated K_m.
For reactions at pH values of 4 and above, positive as well as negative correlations were observed. As the figure above on the right shows, K_m was estimated in the range of 25 - 105 uM in the mentioned pH range. Low and also negative correlations were observed for reactions with pH values at and above pH 4.

The figure on the left shows k_{cat} over K_m for all experimental conditions, 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 correleation between the parameters. Thus, the true change of k_{cat} or K_m through pH or temperature cannot be assessed. Therefore,

For pH values of 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()):
    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)
```



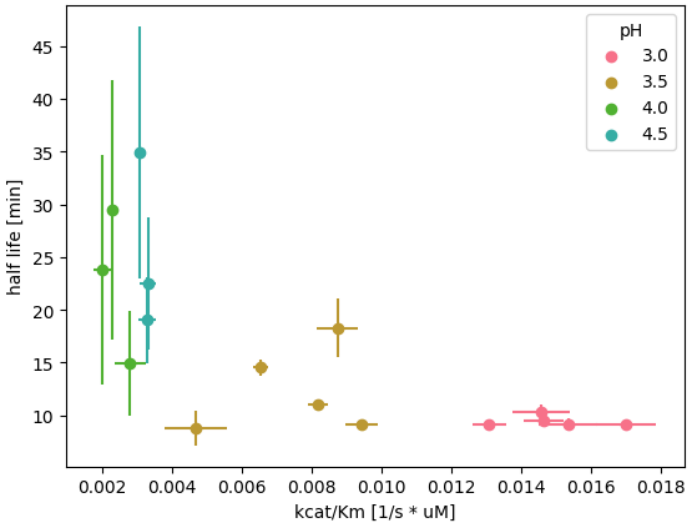
The figure on the left shows the catalytic efficiency $\frac{k_{cat}}{K_m}$ over the pH value of each reaction. The highest catalytic efficiency was observed at pH 3 and 45°C. Increasing the pH by 0.5 reduced $\frac{k_{cat}}{K_m}$ approximatel by half. For higher pH values, $\frac{k_{cat}}{K_m}$ is even more decreased. The catalytic efficiency is therefore highly sensitive to the pH. This might source from different causes. Eighter deprotonation ABTS sulfonate groups for more neutral pH values or protonation changes of the enzyme might hinder catalysis at higher pH values.
In terms of temperature, higher $\frac{k_{cat}}{K_m}$ was achieved at higher temperatures. SLAC might even be more active at higher temperatures and pH values.

Time-dependent enzyme inactivation

The time-dependent enzyme inactivaten rate k_{inact}, was estimated for each experiment, since the irreversible Michaels-Menten model did not describe the observed time-course data accurately. The halfife of the enzyme was calculated as shown in equation (8) [Buxbaum, 2015].

$$t_{\frac{1}{2}} = \frac{ln(2)}{k_{inact}} \tag{8}$$

```
# 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 at higher pH values showed a half life between 15 - 35 min. Therefore, enzyme at reaction conditions with higher catalytic efficieny showed generally a shorter half life compared to enzyme at reaction conditions 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.

```
# 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, .01, .001] 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. The half life of the enzyme is significantly ($p < 0.05$) 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 origine. 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 occures, since observations cannot be attributed to single parameter. On the other hand, enzyme activity and enzyme inactivation might might be causually related. In this case, the formed ABTS radical might in activate 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 futer experiments this workflow could be used to further expand the experimental parameter space. As a result, different enzymes with different substrates in diffrent buffers at different temperatures and pH values could be analyzed.

Correlation between parameters
The highest catalytic efficieny of SLAC was observed at pH 3 at 45°C and therefore on the edge of the investigated prameter space. Optimal reaction conditions might therefore be at an even lower pH value 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 future experiments the product signal could be used for parameter estimation, since the concentration to absorbance ratio is lower compared to the product. However, a reliable method for product quantification would need to be established first.

Enzyme inactivation
SLAC showed a low half life of approximately 10 min at reaction conditions with the highest catalytic efficiency. Potential reasons for enzyme inactivation are discussed in the next chapter.

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 showed progress curves behavior, which was not explainable by irreversible Michaelis-Menten kinetics. SUBSTRATE PRODUCT INHIBITION. All experiments had in commmon, that enzyme reactions were carried out in 96-well polystyrene micro titer plates (MTP), whereas the change in substrate and or product absorption was monitored photometrically. One hypothesis for the observed time-dependent decrease of enzyme activity is potential hydrophobic interaction between the enzyme and the MTP suface. Thereby, hydrophobic regions of the enzyme's surface might interact with the hydrophobic reaction vessle, ultimately preventing substrate access to the active site of the enzyme. In order to test the hypothesis, an absorption experiment in which enzyme was incubated in MTP wells prior to reaction start was performed. Thereby, the enzyme activity should decrease with regard to the 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.

For this experiment Co

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 by the irreversible Michaelis-Menten model.

Experimental design

CotA laccase from *Bacillus subtilis* Both experiments wer conducted with CotA from

Determination of enzyme inactivation trough adsorption
In order to test the hypothesis of time-dependent enzyme inactivation through adsorption to the MTP surface, the following experiment was conducted. Thereby, enzyme solution was incubated in individual MTP wells up to 1 h, prior to reaction start. Then, individual enzyme reactions were started in triplicates by transferring 2 µL of incubated enzyme in 10 min increments. 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 extincion coefficient of ε = 36000 $M^{-1}cm^{-1}$ for the ABTS radical product.

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. Repeats of 4.

Experimantal data

Experimantal data was provided as an Excel file, containing time-course absorption data. Meta data was written into the EnzymeML Excel template. Then, the experimantal data was written to an EnzymeML document by a parser function. Concentrations were calculated via the provided extinction coefficient of the ABTS radical (ε = 36000 $M^{-1}cm^{-1}$).

Adsorption of enzyme to micro titer plate surface

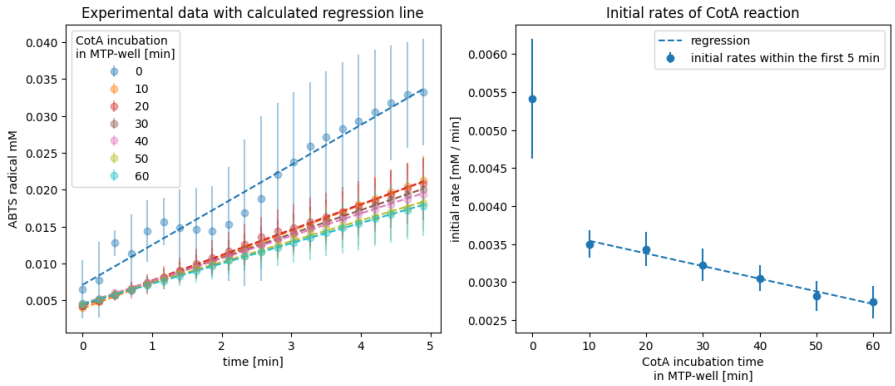
Imports

```
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


#TODO warning in EnzymePynetics

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

Experimantal data

Experimental data was provided as an Excel file. Data was loaded into an pandas DataFrame and the slopes of each incubation condition were calculated through linear regression. The resulting initial rates were used to calculate by which the enzyme activity decreases in regard to the incubation time tin the reaction vessel. Lastly the half life of the enzyme was calculated with equation (8).

```
# 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
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 regression line')


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]')
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')

plt.show()

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




```
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 periods.

The left plot of figure XXX the change in product concentration is shown over the first 5 minutes of the enzyme reaction. Therein, different enzyme incubation times prior to reaction start are color-coded. The first reaction without prior incubation shows large standard deviations between the experimental repeats. This might be the result of insufficient mixing or inconsistent ammount of enzyme between the repeats. All other slopes showed a gradually decreasing slope for increasing incubation times.
The plot on the right of figure XXX shows the initial rates of the enzyme reactions in relation to prior enzyme incubation time. Based on the calculated rate of deactivation, the half life of the enzyme is approximately 28 day. Due to the high standard deviation between the repeats of the reaction withour prior incubation. The experiments were not considered for the calculation of the inactivation rate.

Enzyme kinetics experiment

Experimental data

Experimental data was provided as excel files, 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, whereas concentrations were calculated via the provided extinction coefficient.
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])
fig.show()
```

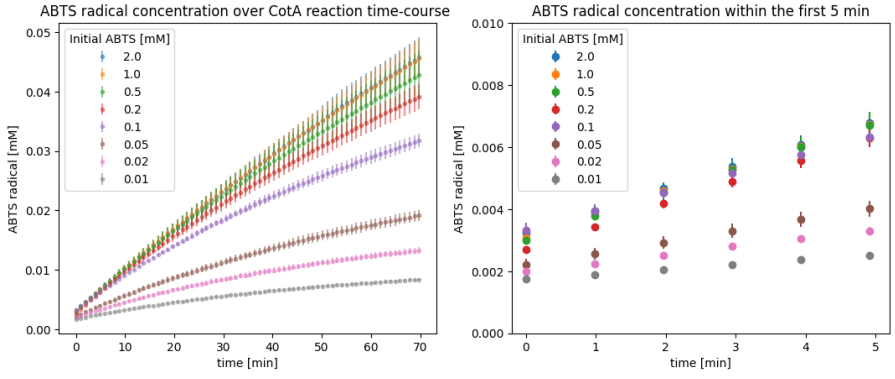


Fig XXX. ABTS radical concentration over the time-course of CotA reaction.

The provided reaction data showed large deviations between individual repeats of reactions with identical initial ABTS concentrations. Expecially for initial ABTS concentrations above 0.1 mM (Fig. XXX, left). Additionally, measurement data was not blanked, since the first measured data points at 0 min have too high concentration values in relation to their respective slopes. Furthermore, the order of initial measurement points is expected to reflect the initial substrate concentrations (Fig. XXX right). Thereby, higher initial substrate concentrations should also show a slightly increased product concentration, since ABTS marginally absorbs at the product detection wavelength.

Kinetic parameter estimation

Kinetic parameters were estimated for all models 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)
df = CotA_kinetics.result_dict.drop(columns=["kcat / Km [1/min * 1/mmole / l]"])
df.insert(1, "Enzyme inactivation model", "False")

# 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)
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")

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]
substrate inhibition	-29337	True	3.209 +/- 0.90%	0.047 +/- 1.73%	-	33.750 +/- 13.18%	0.007 +/- 4.12%
competitive product inhibition	-29300	True	3.058 +/- 0.80%	0.037 +/- 3.55%	0.074 +/- 26.83%	-	0.006 +/- 5.27%
non-competitive product inhibition	-29296	True	3.077 +/- 2.06%	0.037 +/- 5.14%	0.064 +/- 40.49%	0.984 +/- 297.88%	0.006 +/- 29.07%
irreversible Michaelis Menten	-29287	True	3.095 +/- 0.77%	0.043 +/- 1.30%	-	-	0.007 +/- 4.18%
uncompetitive product inhibition	-29283	True	3.102 +/- 0.99%	0.043 +/- 2.20%	-	3.259 +/- 198.94%	0.007 +/- 5.60%
non-competitive product inhibition	-29276	False	3.312 +/- 1.52%	0.040 +/- 4.92%	0.024 +/- 12.73%	0.074 +/- 7.44%	nan
uncompetitive product inhibition	-29157	False	3.413 +/- 1.61%	0.062 +/- 2.46%	-	0.060 +/- 6.63%	nan
competitive product inhibition	-28969	False	2.638 +/- 0.33%	0.023 +/- 5.39%	0.013 +/- 11.76%	-	nan
substrate inhibition	-28778	False	2.664 +/- 0.59%	0.046 +/- 1.96%	-	37.058 +/- 16.47%	nan
irreversible Michaelis Menten	-28741	False	2.579 +/- 0.29%	0.042 +/- 1.48%	-	-	nan

Kinetic parameteres were estimated with and without considering time-dependent enzyme inactivation. Modeling results are listed in table XXX. All models with an additional parameter for enzyme inactivation resulted in lower AIC values, indicating a better fit of the experimantal data to the respective models. Estimates for k_{cat} ranged from 2.58 min^{-1} to 3.41 min^{-1}, where K_m estimates ranged from 23 µM to 62 µM depending on the kinetic model. Based on the resulting k_{inact}, the enzyme's half life was estimated to 95 min, using equation xxx. Since AIC does not consider uncertanties of parameteres, they need to be assessed for model evaluaton.

Kinetic models with enzyme inactivation

For models with enzyme inactivation, the substrate inhibiton model showed the lowest AIC and low standard deviations on the estimated parameters. The irreversible Michaelis-Menten model had the least ammount of parameters and showed even lower standard deviations on the parameter estimates, compared to the substrate inhibition model. Product inhibition models showed generally higher standard deviation on the estimated inhibition parameters. Thus, the two models shown in figure xxx describe the data the best in therms of kinetic parameter standard deviations.

```
fig, axes = plt.subplots(1,2, figsize=(12.8,4.8), sharey=True, sharex=True)
CotA_kinetics_with_inactivation.visualize("substrate inhibition",ax=axes[0], alpha
=.2,\
    title="Substrate inhibition model with enzyme inactivation")
CotA_kinetics_with_inactivation.visualize("irreversible Michaelis
Menten",ax=axes[1], alpha =.2,\
    title="irreversible Michaelis–Menten model with enzyme inactivation")

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]")
plt.tight_layout()
```

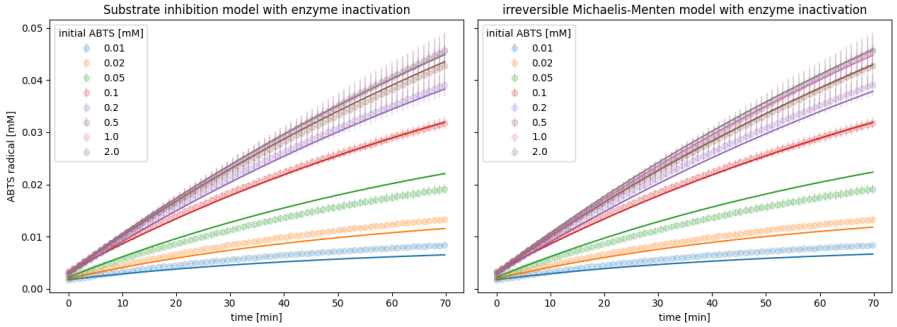


Fig. XXX: Substrate inhibition and irreversible Michealis-Menten model fitted to CotA reaction data

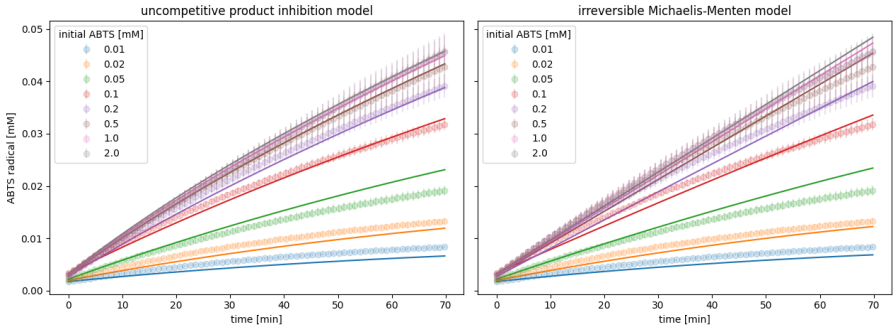
Visually, both models in figure xxx describe the observed concentration time-course of ABTS radical formation. For both models, measurement data from enzyme reactions with intial substrate concentrations below 0.1 mM deviate more strongly from the model compared to higher concentrations. Since the measured absorption values were not blanked, all calculated concentrations are falsely increased. This effect is more pronounced for low initial substrate concentrations, since the absoprtion contribution from buffer is lager compared to the accumulating product for low concentrations.
For reactions with initial substrate concentrations above 0.1 mM, both models describe the progress curve of the reaction. Due to high standard deviation between the prepeat, the true reaction rate at high substrate concentrations is unknown. Therefore, neighter substrate inhibition nor irreversible Michaelis-Menten model were able to clearly represent the experimental data.

Kinetic models without enzyme inactivation

In therms of AIC, product inhibition models without considering enzyme inactivation described the data the best, whereas substrate inhibition and irreversible Michaelis-Menten model showed resulted in the highest AIC of all models. In therms of parameter standard deviation, irreversible Michaelis-Menten model showed the lowest relative errors Non-competitive product inhibition and irreversible Michaelis-Menten model both showed low standerd deviations on the estimated parameters, whereas product inhibition models showed higher uncertanties on the inhibition parameters. Visualizations of the fitted

All models, except for non-competitive and uncompetitice product inhibition with enzyme inactivation resulted in parameter uncertanties of more than 100 % of the estimated value.

```
fig, axes = plt.subplots(1,2, figsize=(12.8,4.8), sharey=True, sharex=True)
CotA_kinetics.visualize("uncompetitive product inhibition",ax=axes[0], alpha =.2,\
    title="uncompetitive product inhibition model")
CotA_kinetics.visualize("irreversible Michaelis Menten",ax=axes[1], alpha =.2,\
    title="irreversible Michaelis–Menten model")

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]")
plt.tight_layout()
```



Conclusion

Low data quality

Experimental data of the data set for parameter estimation showed low quality. This was indicated by high deviations between experimental repeats as well as

Enzyme inactivation Parameter estimation with kinetic models considering enzyme inactivation leads to opposing conclusion of the kinetic mechanism. Not considerin enzyme inactivation would lead to the assumption of product inhibition

Effect of calibration equation on kinetic parameters

In this analysis the influence of calibration model selection on the resulting parameters should was investigated. Each parameter estimation scenario in this work relies on the conversion from absorption data into concentration data. Therefore, a standard of the analyte, which was used to measure the enzyme activty, was provided with each kinetic experiment. Usually, a linear relationship between the absobance signal and concentration is assumed by experimentalists. Therefore, a linear calibration is applied.

Experimentalists often restrict themselfs to only use the so called ‘linear’ range of the photometer below an absorption of 1. Thus, sacrificing a significant ammount of the measurement range of the analytical device.

Data

- chymotrypsin
- SLAC
- trametes
- glucosidase
-

Calibration

Inactivation

Effekt of calibration temperature and pH on kinetic parameters

```
enzmldoc = enzmldoc_absorption_dict["3.0 45.0"]

standards_variations = list(filter(lambda x: x.calibration_data.pH == 3.0,
standard_list))
[standards_variations.append(x) for x in list(filter(lambda x:
x.calibration_data.temperature == 45 and x.calibration_data.pH < 6,
standard_list))];
```

```
kinetics = []
for standard in standards_variations:
    print(copy.__file__)
    doc = copy.deepcopy(enzmldoc)
    standard.apply_to_EnzymeML(doc, species_id="s0", ommit_nan_measurements=True)
    kinetic = (ParameterEstimator.from_EnzymeML(doc, reactant_id="s0",
measured_species="substrate"))
    kinetic.fit_models(enzyme_inactivation=True, only_irrev_MM=True)
    kinetics.append(kinetic)
```

Using linear calibration instead of 3rd polynominal model

```
models = [None, "Linear"]
kinetics=[]
for model in models:
    print(copy.__file__)
    doc = copy.deepcopy(enzmldoc)
    standards_variations[4].apply_to_EnzymeML(doc, species_id="s0",
ommit_nan_measurements=True, model_name=model)
    kinetic = (ParameterEstimator.from_EnzymeML(doc, reactant_id="s0",
measured_species="substrate"))
    kinetic.fit_models(enzyme_inactivation=True, only_irrev_MM=True)
    kinetics.append(kinetic)

# Get kinetic parameters of all datasets
kcat = []
kcat_std = []
Km = []
Km_std = []
ki = []
ki_std = []
temperature = [45,45]
pH = [3,3]
for result in kinetics:
    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)


df = pd.DataFrame.from_dict({
    'pH':pH,
    "temperature":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})

df["kcat/Km"] = 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"]
df["kcat/Km stderr"] = kcat_km_stderr

ph_map = sns.color_palette("husl")
ph_colors = [ph_map[(x)] for x in np.arange(6)]
pH_dict = dict(zip([3.0, 3.5, 4.0, 4.5, 5.0, 5.5], ph_colors))
pH_tuples = [pH_dict[x] for x in df["pH"].values]

temperature_map = matplotlib.cm.get_cmap('coolwarm')
colors = [temperature_map(x) for x in np.linspace(0,1,5)]
color_dict = dict(zip([25, 30, 35, 40 ,45], colors))
color_tuples = [color_dict[x] for x in df["temperature"]]

ax = plt.gca()
for i, (km, kcat, km_error, kcat_error, color, label) in enumerate(zip(df["Km
[uM]"], df["kcat [1/s]"], df["Km stderr"], df["kcat stderr"], pH_tuples,
df["pH"].values)):

    ax.scatter(km, kcat, color = color, label = label if not i % 5 else "")
    ax.errorbar(km, kcat, yerr=kcat_error,xerr=km_error, fmt=".", ecolor=color,
markersize=0)
plt.legend(title = "pH")
plt.ylabel("kcat [1/s]")
plt.xlabel("Km [uM]")
plt.xlim([0,200])
```

```
subset_enzymeml = [copy.deepcopy(x) for x in enzmldoc_absorption_dict.values() if
x.getReaction("r0").temperature == 45]
models = [None, "Linear"]
calibration_data = [x for x in standard_curve_dict.values() if
x.calibration_data.temperature == 45 and x.calibration_data.pH < 6]
applied_model = {"best model": None, "Linear": "Linear"}

kinetics = []
model_names = []
for doc, standard in zip(subset_enzymeml, calibration_data):
    for name, value in applied_model.items():
        d = copy.deepcopy(doc)
        standard.apply_to_EnzymeML(d, species_id="s0",
ommit_nan_measurements=True, model_name=value)
        kinetic = (ParameterEstimator.from_EnzymeML(d, reactant_id="s0",
measured_species="substrate"))
        kinetic.fit_models(enzyme_inactivation=True, only_irrev_MM=True);
        kinetics.append(kinetic)
        kinetic.visualize()
        model_names.append(name)



model_names
```

```
kcat = []
kcat_std = []
Km = []
Km_std = []
ki = []
ki_std = []
temp = []
calibration_model = np.tile(np.array(["best model", "linear calibration"]), 6)
pH = np.repeat(np.array([3,3.5,4,4.5,5,5.5]), 2)
for result in kinetics:
    params = result.get_parameter_dict()
    temp.append(result.data.temperature)

    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)


df = pd.DataFrame.from_dict({
    'pH':pH,
    "calibration model":calibration_model,
    'kcat [1/s]':kcat,
    'kcat stderr':kcat_std,
    'Km [uM]':Km,
    'Km stderr':Km_std,
    "Enzyme inactivation [1/s]":ki,
    "temperature": temp,
    "Enzyme inactivation std":ki_std})

df["kcat/Km"] = 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"]
df["kcat/Km stderr"] = kcat_km_stderr

df
```

```
ph_map = sns.color_palette("husl")
ph_colors = [ph_map[(x)] for x in np.arange(6)]
pH_dict = dict(zip([3.0, 3.5, 4.0, 4.5, 5.0, 5.5], ph_colors))
pH_tuples = [pH_dict[x] for x in df["pH"].values]

temperature_map = matplotlib.cm.get_cmap('coolwarm')
colors = [temperature_map(x) for x in np.linspace(0,1,5)]
color_dict = dict(zip([25, 30, 35, 40 ,45], colors))
color_tuples = [color_dict[x] for x in df["temperature"]]

cools = ["b", "orange", "b", "orange","b", "orange","b", "orange","b",
"orange","b", "orange"]

ax = plt.gca()
for i, (kcatkm, kcatkm_error, temp, color, label) in enumerate(zip(df["kcat
[1/s]"], df["kcat stderr"], df["pH"] , cools, df["calibration model"].values)):

    ax.scatter(temp, kcatkm, color = color, label = label if i in [0,1] else "")
    ax.errorbar(temp, kcatkm, yerr=kcatkm_error, fmt=".", color= color,
markersize=0)
plt.legend(title = "calibration model")
plt.ylabel("kcat/Km [1/s * uM]")
plt.xlabel("pH")
plt.title("Datasets of various pHs and 45°C calibrated with different standard
curve fits")
```

```
enzmldoc = concentration_enzymemldocs[4]

kinetics = ParameterEstimator.from_EnzymeML(enzmldoc, "s0", "substrate")
kinetics.fit_models(only_irrev_MM=True, enzyme_inactivation=False)
kinetics.visualize()
```

Discussion

Discuss both viewpoints: biologist and RSE.

Jupyter notebooks for documentation of scientific analysis

Jupyter notebooks enable enable

Reproducibility in enzyme kinetics

- Need for documentation without limiting experimental possibilities

Data management / lab digitalization

- temperature in MTPs

Jupyter notebooks

- 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

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  fishy

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.

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.

By Max Häußler