CADETMatch Manual

William Heymann

Contents

[Introduction 4](#_Toc509928155)

[Scores 4](#_Toc509928156)

[Decay 4](#_Toc509928157)

[Breakthrough 5](#_Toc509928158)

[Breakthrough 5](#_Toc509928159)

[Breakthrough Cross 5](#_Toc509928160)

[Breakthrough Hybrid 5](#_Toc509928161)

[Breakthrough Hybrid 2 [Recommended Method] 5](#_Toc509928162)

[Curve Similarity 5](#_Toc509928163)

[Similarity [Decay] 6](#_Toc509928164)

[Similarity Cross [Decay] 6](#_Toc509928165)

[Similarity Hybrid [Decay] 6](#_Toc509928166)

[Similarity Hybrid 2 [Decay] [Recommended Method] 6](#_Toc509928167)

[Curve [Filter out additional erroneous peaks] 6](#_Toc509928168)

[Derivative Curve Similarity 6](#_Toc509928169)

[Derivative Similarity 7](#_Toc509928170)

[Derivative Similarity Cross 7](#_Toc509928171)

[Derivative Similarity Hybrid 7](#_Toc509928172)

[Derivative Similarity Hybrid 2 [Recommended Method] 7](#_Toc509928173)

[Fractionation 8](#_Toc509928174)

[Fractionation [Do not use] 8](#_Toc509928175)

[Fractionation Combine [Do not use] 8](#_Toc509928176)

[Fractionation Mean Variance [Do not use] 8](#_Toc509928177)

[Fractionation Moment [Do not use] 8](#_Toc509928178)

[Fractionation Slide [Recommended Method] 8](#_Toc509928179)

[Dextran 9](#_Toc509928180)

[Dextran [score not currently working correctly] 10](#_Toc509928181)

[Dextran Hybrid 11](#_Toc509928182)

[Dextran Hybrid 2 11](#_Toc509928183)

[Sum Squared Error 11](#_Toc509928184)

[Sum Squared Error 11](#_Toc509928185)

[Log Sum Squared Error 11](#_Toc509928186)

[Algorithms 11](#_Toc509928187)

[Mutation 11](#_Toc509928188)

[SPEA2 12](#_Toc509928189)

[NSGA2 [Recommended Method] 12](#_Toc509928190)

[ScoreTest 12](#_Toc509928191)

[MultiStart 12](#_Toc509928192)

[Examples 12](#_Toc509928193)

[Example 1 12](#_Toc509928194)

[Fit column properties 12](#_Toc509928195)

[Fit isotherm 19](#_Toc509928196)

[Example 2 20](#_Toc509928197)

[Fit Isotherm 20](#_Toc509928198)

[Example 4 21](#_Toc509928199)

[Example 5 21](#_Toc509928200)

# Introduction

CADETMatch is an evolutionary algorithm many-objective optimization software designed to solve parameter estimation problems using CADET via the HDF5 interface. The system can combine multiple experiments with multiple components and different scores for each experiment and component to estimate input parameters. Effectively any variable that exists in CADET can be used as an optimization target.

An essential aspect of many-objective optimization is that there is no unique best result (usually). Instead the result is a pareto front. Effectively you have many results which are considered equally good, but one set of parameters may fit the shape a bit better while another may fit the height or time a bit better. The more scores you fit to and the more parameters you estimate the larger the pareto front becomes.

Figure 1 Pareto Front

From Figure 1 the orange line represents the pareto front. Any point on this front is considered equally good and which ones works best for you is based on what your requirements are. The blue points are considered dominated by the points on the front and are inferior to the points on the front.

# Scores

Depending on the type of data being fit to different scores are available. Most of the scores run from 0 to 1 with 1 being the best value and the scores can be freely mixed. The exception to this are the sum squared error scores. Those scores cannot be mixed with the other scores.

## Decay

Most scores also have a decay version, and this is denoted with they [Decay] tag. The decay version of an algorithm gives an immediate penalty when the simulated signals and experimental signals have a small-time difference. The normal version of scores apply a very small penalty if the time difference between the experimental and simulated data is small but increases the penalty rapidly as the difference grows. This grace period is to cover uncertainty in pumps where small offsets occur naturally due to physical processes.

## Breakthrough

Breakthrough scores are special scores designed to handle breakthrough curves. The code expects to see a rise, flat area and then a fall. If your data fits into this form, then the breakthrough scores are a good fit.

### Breakthrough

Breakthrough uses the Pearson correlation for curve similarity. The score is then made up of when the maximum value starts, when the maximum value ends and what the maximum value is compared to the experimental values.

### Breakthrough Cross

Breakthrough cross uses cross-correlation to assess signal similarity between the simulated and experimental data along with a single time shift based on the cross correlation. The maximum value is also compared between the simulated and experimental data.

### Breakthrough Hybrid

Breakthrough hybrid uses a combination of Pearson correlation for curve similarity and cross-correlation for temporal offset. The Pearson correlation seems to work better for assessing curve similarity while the cross-correlation works better to find out the time difference between two signals. The system also uses the maximum value of the simulated and experimental data to compare them.

### Breakthrough Hybrid 2 [Recommended Method]

Breakthrough hybrid 2 uses a combination of Pearson correlation for curve similarity and cross-correlation for temporal offset. The Pearson correlation seems to work better for assessing curve similarity while the cross-correlation works better to find out the time difference between two signals. The system also uses the maximum value of the simulated and experimental data to compare them. The difference between this method and the normal hybrid method is that first the cross-correlation is taken and then the signals and then the simulated signal is offset based on the cross-correlation to the time of maximum overlap and the Pearson correlation is taken at that point.

## Curve Similarity

Curve similarity scores are general purpose scores. They are designed to handle arbitrary curves and match that against your data. These curves are built with 3 major pieces. The similarity, phase and amplitude are compared between simulated and experimental data.

### Similarity [Decay]

Similarity uses the Pearson correlation for curve similarity. The peak height and peak time is also compared between simulated and experimental data.

### Similarity Cross [Decay]

Similarity cross uses the cross-correlation for curve similarity and time offset between simulated and experimental data. The peak height is also compared between simulated and experimental data. Cross-correlation is not as shape sensitive as the Pearson correlation and will give higher scores to peaks with worse shapes.

### Similarity Hybrid [Decay]

Similarity hybrid attempts to fix the problem in Similarity cross by using the Pearson correlation for curve similarity but the cross-correlation for the time offset.

### Similarity Hybrid 2 [Decay] [Recommended Method]

Similarity hybrid 2 uses the cross-correlation to find the time offset and then offsets the signals for maximum overlap before applying the Pearson correlation for similarity.

### Curve [Filter out additional erroneous peaks]

Curve is a very simple score that only uses the Pearson correlation and does not look at the peak height or time. The purpose of this score is to filter out erroneous results. In solving some problems additional peaks may appear in incorrect places and the Curve score can be used to filter out these results.

## Derivative Curve Similarity

Derivative similarity scores are designed to fit with the curve similarity scores. If your chromatograms have slope changes in the chromatogram a general shape fit has a challenging time catching the subtle shape change. By looking at the derivative these changes are much easier to fit. In most cases a curve similarity and derivative similarity are used.

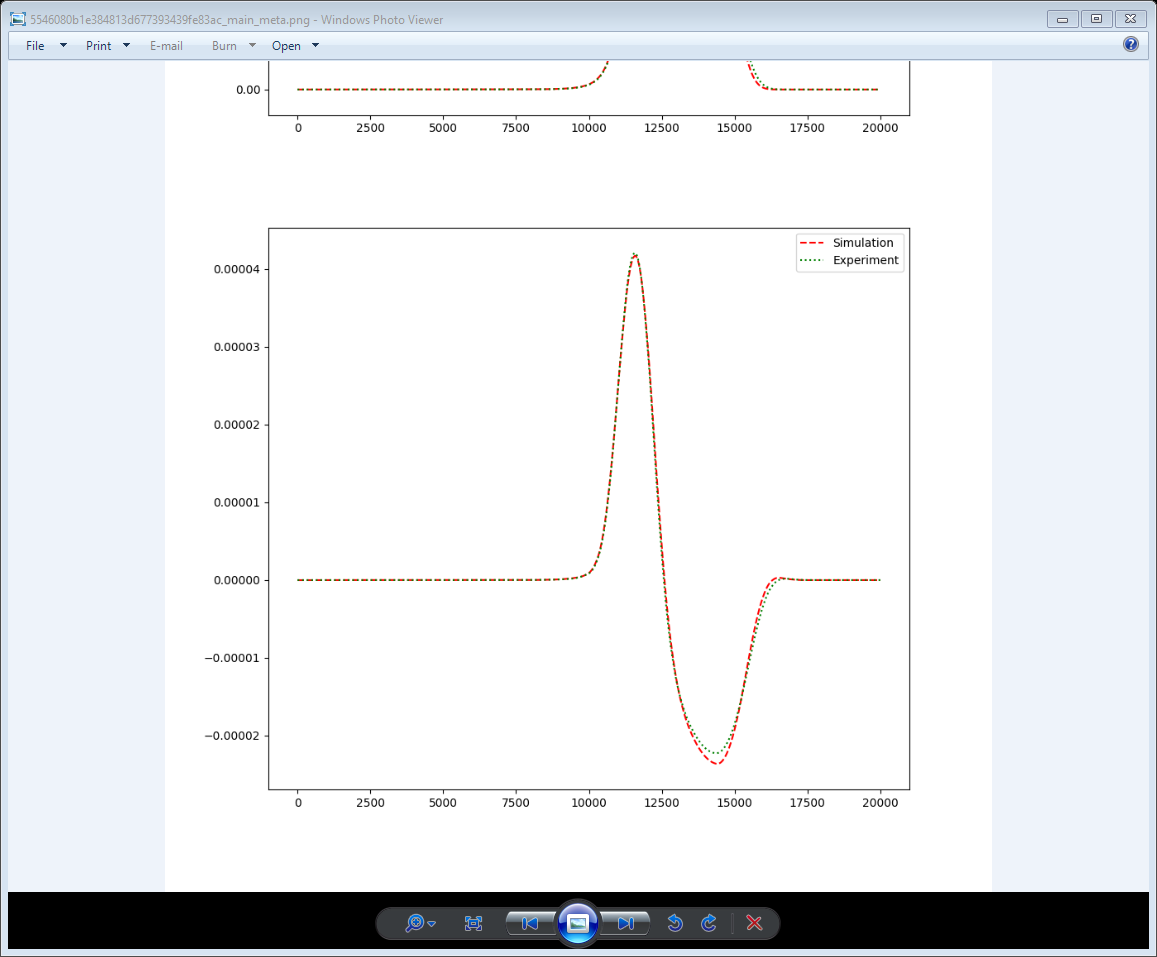


Figure 2 Chromatogram derivative fit

### Derivative Similarity

Derivative similarity uses a Pearson correlation for shape similarity and uses the time and value for the peak max and peak min for the score as seen in Figure 2.

### Derivative Similarity Cross

Derivative similarity cross uses the cross-correlation for shape similarity and a single time-offset instead of two times as seen in the previous score along with the peak max and peak min values. This score is less sensitive to the shape.

### Derivative Similarity Hybrid

Derivative similarity hybrid uses the Pearson correlation for shape similarity while using the cross-correlation for time-offset along with the peak max and peak min values.

### Derivative Similarity Hybrid 2 [Recommended Method]

Derivative similarity hybrid 2 uses the cross-correlation to find the time-offset between simulated and experimental data and then offsets the data to maximum overlap before calculating the Pearson correlation at the point of maximum overlap. The peak max and peak min values are used in the score also.

## Fractionation

Fractionation scores are designed to handle fractionation data where you have discrete averaged points of data. This is normally done by collecting the column outlet in set increments (25 ml for example) and then analyzing each fraction. The data is usually extremely course with only 5-10 points for the entire chromatogram. This type of data is incredibly challenging to work with.

|  |  |  |
| --- | --- | --- |
| animate_fraction.mp4 - VLC media player  A | animate_fraction.mp4 - VLC media player  B | animate_fraction.mp4 - VLC media player  C |

Figure 3 Fractionation at different offsets

### Fractionation [Do not use]

Fractionation is the simplest and most obvious fractionation score. For each fraction and each component, the value is compared between the simulated and experimental data. In practice this score does not work at all. Far too many scores are created which causes the dimensionality of the problem to explode. The system can also not determine how close the underlying system is. Based on Figure 3 while B has the exact same shape as A and the offset is small the point-based comparison gives a very bad score. This score suffers from the same problem as sum squared error in that minor changes in offset lead to substantial changes in score.

### Fractionation Combine [Do not use]

Fractionation combine is a simple and obvious extension to the fractionation score that also happens to be wrong. Fractionation combine tries to deal with the dimensionality problem by using the previous point-based score and creating one score per component by averaging the scores for the component. This score has the same problem where it does very little to guide the optimizer to the desired result.

### Fractionation Mean Variance [Do not use]

Fractionation mean variance calculates the mean and variance of the fractionated data for amplitude and phase. The scores are then compared with the experimental fractionation. This method seems like it should work but in practice but due to how much the shape changes for fractionation this score does not work.

### Fractionation Moment [Do not use]

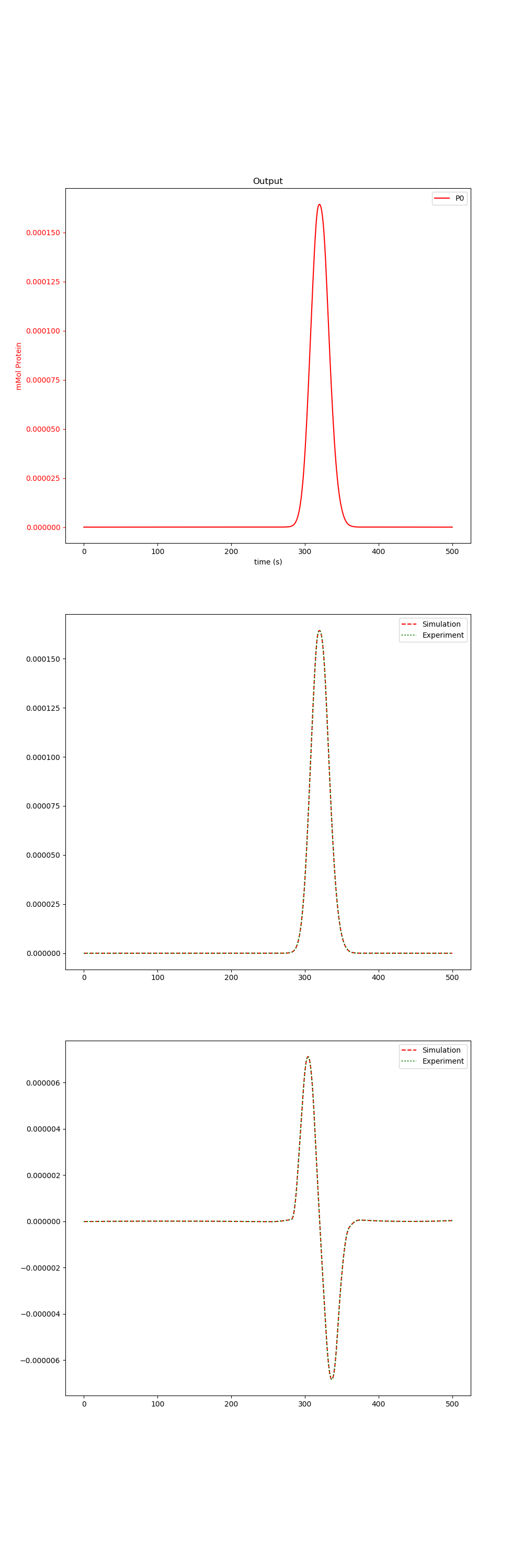
Fractionation moment is the same as “Fractionation Mean Variance” but it also adds the 3rd moment (skew) to the calculations. The purpose of adding skewness to the metric is to capture the asymmetry of columns. In practice the shape of the fractionation makes such large changes due to small changes in the chromatogram that this score does not work.

### Fractionation Slide [Recommended Method]

Fractionation slide is mathematically more complex and has the most complex implementation of any of the methods but conceptually it is one of the simplest methods. The simulated peak is shifted left and right and re-fractionated many times. Sum squared error is used to find the point of maximum overlap between experimental and simulated data. The time-offset is found based on the point of maximum overlap and then the Pearson correlation and peak maximum is taken for score similarity. Based on simulated studies this is the only score for fractionation that works. None of the others have the property that as the solution gets better the score always increases and then decreases as the solution gets worse.

## Dextran

Dextran scores are specifically designed to fit a non-binding and non-pore entering dextran tracer. The problem is that some Dextran does enter the pores, and this causes problems with normal fitting methods. The Dextran scores are designed to look at only the front of the peak, the curvature of the peak and the elbow where the elution starts. This critical region covers the larger dextran molecules moving through the column without the problem of the smaller pieces that enter the pores contaminating the results.



### Dextran [score not currently working correctly]

Dextran scores tries to find where the slope stops increasing and cuts off the matching there. However, this is currently not working very well. The system using the cross-correlation to find the time-offset and curve similarity along with the derivative similarity. This is a problem that currently takes about 5 minutes to do manually.

### Dextran Hybrid

Dextran hybrid works the same way as the Dextran score. The only difference is the Pearson correlation is used instead of the cross-correlation for curve similarity.

Dextran Hybrid 2 [Recommended Method]

Dextran Hybrid 2 is a more complex Dextran score, but it works better than even manual fitting does. What the system does is search for where the Dextran curve slope stops increasing and creates a cutoff point. It then cuts that part out of the data and inserts it into an array of the same length that is filled with zeros so the only feature in the data is the Dextran part we need. The same thing is done for the simulated data and the two curves are compared using the cross-correlation for the offset and Pearson at the point of maximum overlap.

## Sum Squared Error

Sum squared error is a more traditional scoring system. It has been implemented as a reference to compared with other score systems. Unlike the other scores the sum squared error score cannot be combined with other scores due to weighting problems.

### Sum Squared Error

Sum squared error is simple the sum squared error between the simulated and experimental data. Without a way to normalize sum squared error between 0 and 1 like the other scores

### Log Sum Squared Error

Log sum squared error takes the log of sum squared error in order to magnify the differences near the objective.

# Algorithms

There are several search algorithms supported by CADETMatch. Some of them are designed for testing the system and some are designed for solving parameter estimation problems.

## Mutation

Both evolutionary algorithms support different adaptive mutation strategies. For now, it is recommended to use the XXXX\_mut\_adapt version.

## SPEA2

Strength Pareto Evolutionary Algorithm 2 (SPEA2) is an evolutionary algorithm built to explore a pareto front. This is a general-purpose algorithm and should be able to solve most parameter estimation problems.

## NSGA2

Non-dominated sorting genetic algorithm (NSGA2) is another general purpose evolutionary algorithm. It is designed based on mutating and breeding non-dominated solutions while maintaining population diversity. This is a general-purpose algorithm and should be able to solve more parameter estimation problems.

## NSGA3 [Recommended Method]

NSGA3 is a newer version of NSGA2. CADETMatch has a modified version of NSGA3 that uses a Sobol sequence for reference points. The method is typically a bit more robust than NSGA2 or SPEA2 with more complex systems.

## ScoreTest

ScoreTest is designed to test if scores are working correctly. One or more points can be given to this search method and the points will be evaluated and the algorithm will terminate. This is extremely useful for synthetic data to verify that the optimal point generates an optimal score.

## MultiStart

MultiStart uses a multi-start gradient descent to solve problems. The gradient is calculated with finite difference. In some circumstances a gradient can exist, and this method can explore it. For most parameter estimation problems this method should not be used as no clear gradient normally exists. This method is normally only used after a solution has already been found and a gradient has been verified to exist locally.

## MCMC

MCMC is based on the emcee project (<http://dfm.io/emcee/current/>) which is an Affine Invariant Markov chain Monte Carlo Ensemble sample. This method is not designed for searching and is instead used to find parameter distributions once the optimal value has been found. The system has automatic-burn in, automatic termination and automatic acceptance fraction tuning. Once complete percentiles on each parameter are available along with corner plots and confidence plots. The MCMC code can be automatically run on the result of one of the above search algorithms with the search space automatically narrowed to improve the MCMC algorithm.

# Parameter Transforms

Parameter transformations are an important part of any optimization software. Often parameters are not independent or span wildly different changes which degrades optimization progress and the quality of the results. There are many transformations available that come in a regular version and a norm version. The norm version maps the parameter transformation to a range of 0 to 1. The norm versions should pretty much always be used.

## Norm

**Condition: upper\_bound/lower\_bound < 100**

This is a simple score that maps a variable to a range of 0 to 1. This is often used for variables that already have a very narrow range and so a log transform is not needed. If you have a variable where the upper limit divided by the lower limit is less than 100 then use this transform.

## kEQ [Norm]

**Condition: ka and kd estimated**

This is a transform that changed ka and kd into ka and keq = ka/kd. In most systems ka and kd are not independent and this build the dependence into the optimization process. If you are estimation ka and kd you should always use this transform. This score does a log transform of ka and kd. This score has a norm variant which should be used.

## Log [Norm]

**Condition: upper\_bound/lower\_bound > 100**

This transform is when you have a single variable that spans a large range but is not coupled to another parameter.

## Nu + Sigma [Norm]

## Volume + Area [Norm]

## Volume + Length [Norm]

## Null

# Examples

Examples cover how to use the matching software and not how to create simulations. All examples need a basic hdf5 file to work with and a csv file with experimental data to match against. For each fit that is done there is a JSON section that covers how the matching software is setup. Each one builds on the previous example and only introduces new features.

## Example 1

Example 1 is a simple artificial example with a 1-component Linear isotherm using the General Rate Model. It covers what experiments are needed to fit the data along with how to do the fitting. All the data and scripts needed are in the examples folder. The example starts with fitting the column properties and ends with fitting the isotherm. More complex examples will cover fitting with multiple experiments, isotherms, and fractionation.

### Fit column properties

There are 5 column properties that need to be fit for the General Rate Model

* Column Dispersion
* Column Porosity
* Particle Porosity
* Particle Diffusion
* Film Diffusion

It is sometimes not possible to tell Film Diffusion and Particle diffusion apart in real data since one often ends up much faster than the other. It is only when both are very similar that both can be identified. However, if one is much faster than the other such that the other is rate limiting it does not matter to the model if the both can be identified since only the rate limiting one would make an impact.

Fitting column properties can be done in a laptop in a few minutes.

#### Dextran

A Dextran pulse is normally used to find the Column Porosity and Column Diffusion. Ideally none of the Dextran would enter the pore but from the peak width and peak height this is clearly not true. Only the part of the front of the peak can be used since it contains the larger Dextran molecules. From the peak start position and slope the Column Porosity and Column Diffusion can be fitted. It is important that the pump settings are very accurate since a pump delay is mathematically the same as a different Column Porosity (since both cause the peak to shift in time).

In the Examples folder under Example1/Dextran there is a basic simulation along with the csv to fit against and the CADETMatch JSON file to use. After CADETMatch has been run check the folder fit/meta for the fit quality. The fit should look like Figure 4. The resulting parameters are in fit/meta/results.csv or fit/meta/results.xlsx. If there are multiple results you can look at them and see which result looks best to you and take the corresponding entry in the results file. From Figure 5 the values needed are in “COL\_DISPERSION” and “COL\_POROSITY”. Your results should be approximately the same values as Table 1.

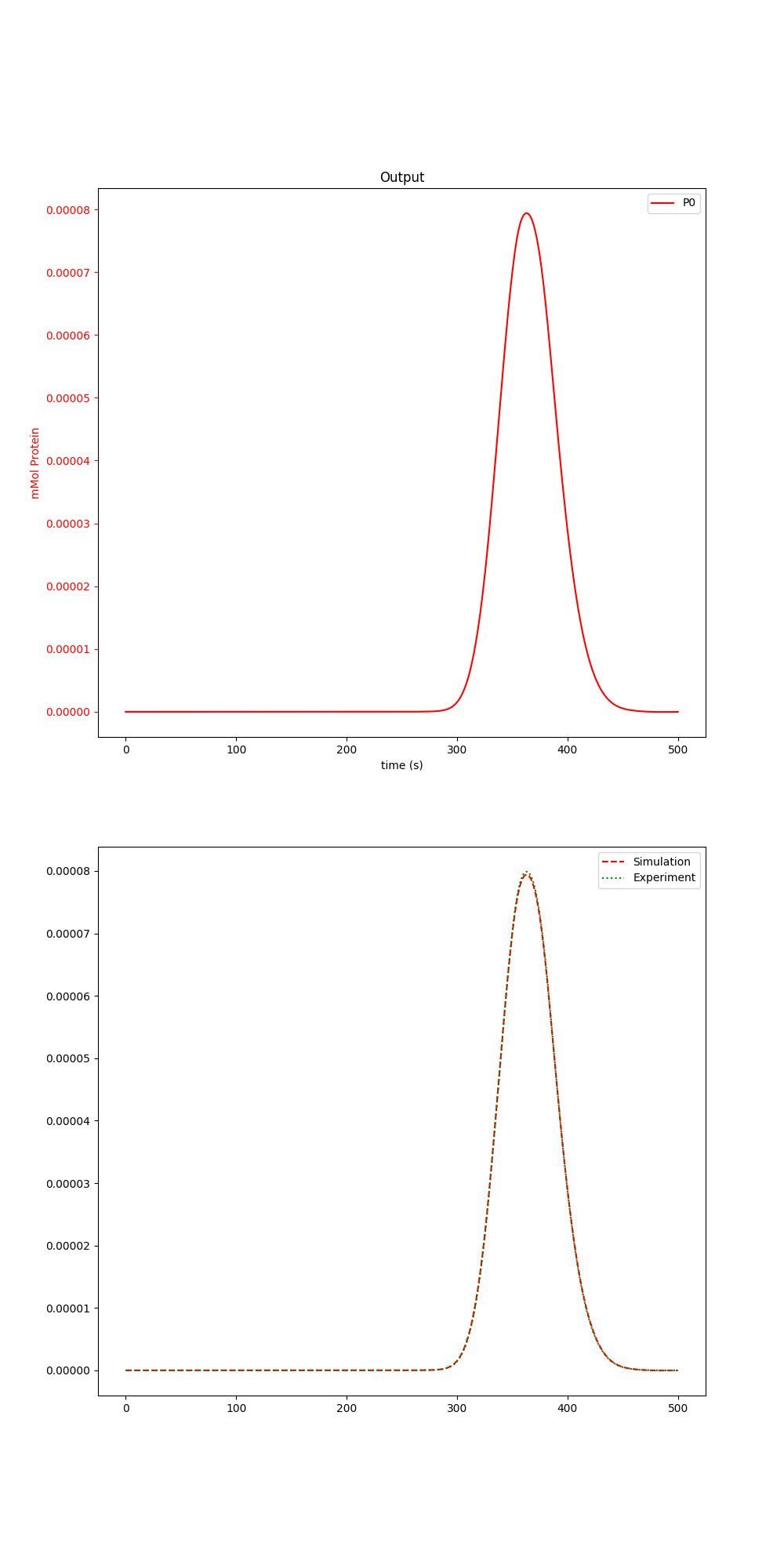


Figure 4 Example 1 Dextran fit

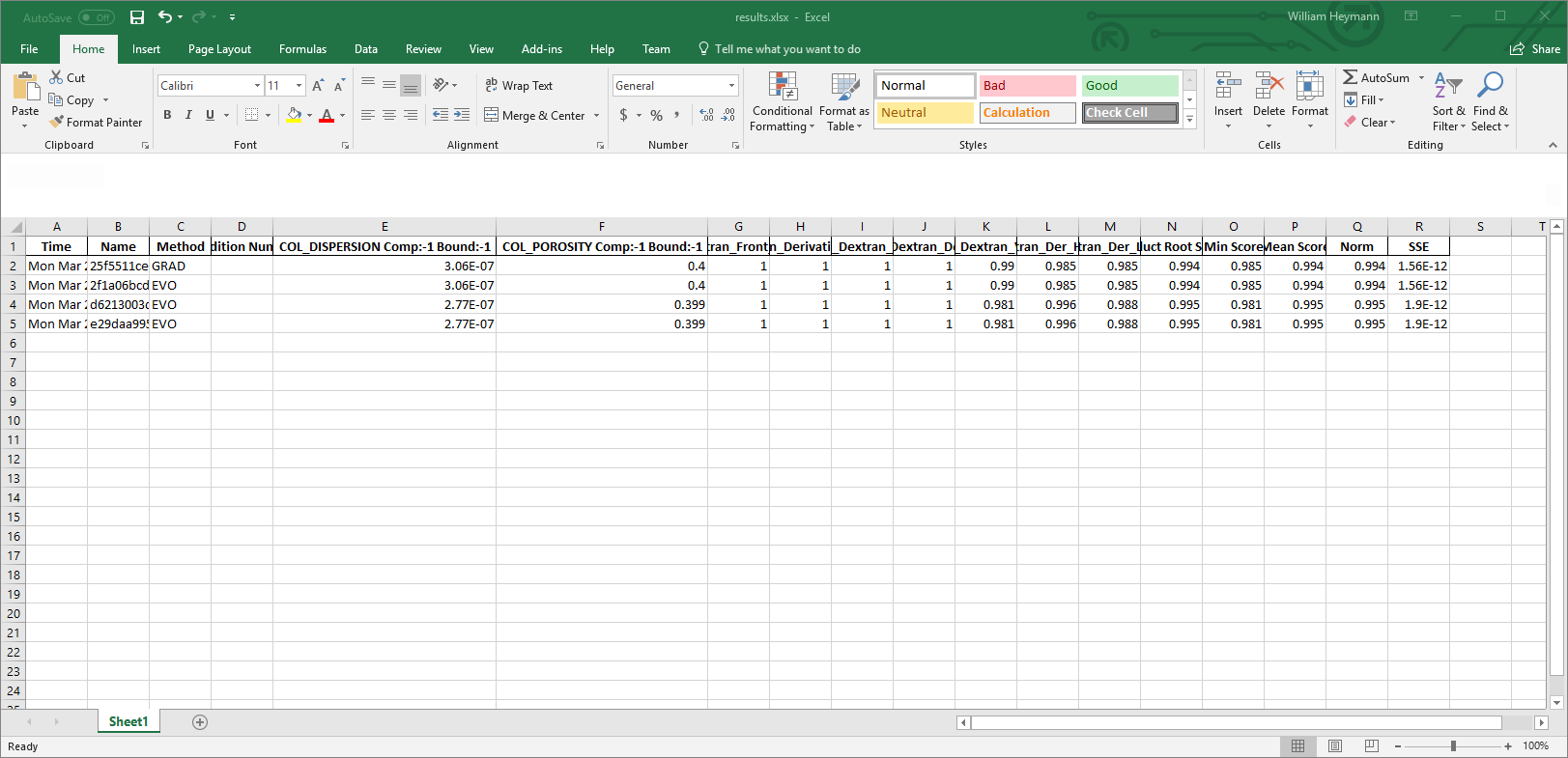


Figure 5 Example 1 Dextran results

Table 1 Example 1 Dextran fit correct answers

|  |  |
| --- | --- |
| Name | Value |
| Column Dispersion | 3.0e-7 |
| Column Porosity | 0.4 |

##### Identifiability

Parameter identifiability is critically important when designing experiments. CADETMatch has some features built into it to assess if an experiment provides identifiable parameters. Once the matching process is complete look in Examples/Example1/Dextran/fit/space and look through the graphs. The important part is that for each parameter there must exist at least 1 graph that has identifiability for the parameter.

A drawing of a face

Description generated with low confidence

Figure 6 Dextran parameter identifiability

##### JSON file

The following JSON file is read by CADETMatch for parameter estimation. I will go through it piece by piece and introduce new features. Future JSON sections will contain only the new parts instead of the whole file. JSON is a standardized configuration file format that is easy to generate and read in many programming languages and tools. It does not matter what order the entries appear in so long as all of them appear. Capitalization matters. All paths can be relative or absolute with absolute paths preferred due to being less confusing. If “baseDir” is set, then all paths are evaluated relative to “baseDir” and otherwise they are evaluated relative to the directory the matching is started from. If “baseDir” is not set, then absolute paths should be used.

1. {
2. "CADETPath": "C:/Users/kosh\_000/cadet\_build/CADET-dev/MS\_SMKL\_RELEASE/bin/cadet-cli.exe",
3. "baseDir": "F:/Examples/Example1/Dextran",
4. "resultsDir": "fit/",
5. "CSV": "dextran\_NSGA2.csv",
6. "checkpointFile": "check",
7. "stopAverage": 0.999,
8. "stopBest": 0.999,
9. "gradCheck": 0.9,
10. "searchMethod": "NSGA2",
11. "crossoverRate": 0.8,
12. "generations": 100,
13. "population": 100,
14. "roundParameters": 3,
15. "roundScores": 3,
16. "metaResultsOnly": 1,
17. "stallGenerations": 10,
18. "parameters": [
19. {
20. "transform": "log",
21. "component": -1,
22. "bound": [
23. -1
24. ],
25. "location": "/input/model/unit\_001/COL\_DISPERSION",
26. "min": [
27. 1e-12
28. ],
29. "max": [
30. 1.0
31. ]
32. },
33. {
34. "transform": "log",
35. "component": -1,
36. "bound": [
37. -1
38. ],
39. "location": "/input/model/unit\_001/COL\_POROSITY",
40. "min": [
41. 0.1
42. ],
43. "max": [
44. 0.9
45. ]
46. }
47. ],
48. "experiments": [
49. {
50. "CSV": "dextran\_pulse.csv",
51. "isotherm": "/output/solution/unit\_001/SOLUTION\_OUTLET\_COMP\_000",
52. "HDF5": "dextran\_pulse.h5",
53. "name": "main",
54. "timeout": 3.4842967987060547,
55. "features": [
56. {
57. "name": "Dextran",
58. "start": 0,
59. "stop": 500.0,
60. "type": "dextranHybrid2"
61. }
62. ]
63. }
64. ]
65. }

* Line 2: CADETPath: The path to the cadet binary that will be used for matching
* Line 3: baseDir: Path to the base directory for the simulation
* Line 4: resultsDir: Path to where results are stored
* Line 5: CSV: Filename for results
* Line 6: checkpointFile: Name of the file used for checkpointing (resuming parameter estimation if interrupted). Normally this is set to “check”
* Line 7: stopAverage: Stop if the average score on the pareto front is greater than stop average
* Line 8: stopBest: Stop if the best score on the pareto fronts worst individual score is better than stopBest
* Line 9: gradCheck: Enable gradient descent and refinement if the score is greater than or equal to gradCheck. If gradCheck is higher than stopBest or stopAverage then gradient descent will not be used.
* Line 10: searchMethod: This line defines what search method to use.
* Line 11: crossoverRate: This is the crossover rate for the genetic algorithm. Unless 0.8 is not working for your problem it is best to leave it at 0.8.
* Line 12: generations: This is a scaling parameter that sets how many generations the matching software can run before forced termination. The actual number of generations is # of parameters \* generations. 100 is a good default.
* Line 13: population: This is another scaling parameter and it sets how large the population is at each generation. The actual number is # of parameters \* population. 100 is a good default. Smaller values can decrease total simulation time but at the risk of the algorithm becoming stuck while larger values rarely help.
* Line 14: roundParameters: Round estimated parameters to roundParameters number of significant digits. If this value is not set the pareto front will tend to have many results that are nearly identical. In general, only 1 to 2 significant figures can be identified from experiments and a value of 3 works well.
* Line 15: roundScores: Round scores rounds the score system to the specified number of significant digits. This reduces the size of the pareto front and prevents many values with no effective difference between them.
* Line 16: metaResultsOnly: This only saves results and generates graphs for members of the meta front instead of all the normal pareto and gradient front members. Most of the time this should be set to 1.
* Line 17: stallGenerations: This is the maximum number of generations that can pass without a new member being added to the meta front before the search is ended.
* Line 18: parameters: This starts the section for the parameters to be estimated. This is a list of dictionaries.
* Line 10: transform: Parameter transform to use. In this case the natural log is taken of the parameter. Other parameter transforms will be covered later.
* Line 21: component: This is the component to estimate starting with 0 as the first component. If the parameter is independent of components, then the value is set to -1. In this case COL\_DISPERSION is component independent and so component is set to -1.
* Line 22: bound: This is the bound state to estimate starting with 0 for the first bound state. If the parameter is independent of bound states, the values is set to -1.
* Line 25: location: This is the path to the parameter inside the HDF5 file starting with / as the root. This path must be absolute to the root of the HDF5 file. Any floating-point parameter can be estimated in the HDF5 file although some do not make much sense to change. In general, only parameters for unit operations and isotherms should be estimated. While something like the absolute tolerance could be estimated it would make little sense to do so.
* Line 26: min: This is the smallest value allowed for the estimated parameter.
* Line 29: max: This is the largest value allowed for the estimated parameter.
* Line 48: experiments: This starts the section for the experiments to be use. This is a list of dictionaries.
* Line 50: CSV: This is the path to the experimental data to match against. The file must have 2 columns with the first being time and the second being the chromatogram concentration. The file columns have no headers.
* Line 51: isotherm: This is the path within the HDF5 where the simulation data to match against the experimental data is.
* Line 52: HDF5: This is the path to the HDF5 file used for this experiment.
* Line 53: name: This is the name of the experiment. Name is used in the CSV file as part of the column headings to separate experiments. Two experiments cannot have the same name.
* Line 54: timeout: This sets how long an experiment can run, in seconds, before being terminated and the point marked as a failure. Some combinations of parameters cause extremely slow convergence, and this helps catch those cases. Normally this is set to about 10x what it takes for the basic simulation to run.
* Line 55: features: This is a list of dictionaries that defines what scores to use.
* Line 57: name: Name of the score. Two features cannot share the same name.
* Line 58: start: This is the time in seconds that the score will start at.
* Line 59: stop: This is the time in seconds that the score will end at.
* Line 60: type: This is the score to use. It will only be applied between start and stop. It is important to define a narrow area around the feature of interest. Large areas around a small feature ends up mostly looking at the similarity of the surrounding area.

#### Non-binding protein

Non-binding protein allows us to fit the rest of the column parameters. Given the column porosity and column dispersion a protein that enters the beads but does not bind allows particle porosity, film diffusion and pore diffusion to be estimated. It is important that the pump settings are very accurate since a pump delay is mathematically the same as a different Particle Porosity (since both cause the peak to shift in time).

In the Examples folder under Example1/NonBindingProtein there is a basic simulation along with the csv to fit against and the CADETMatch JSON file to use. After CADETMatch has been run check the folder fit/meta for the fit quality. The fit should look like Figure 6. The resulting parameters are in fit/meta/results.csv or fit/meta/results.xlsx. If there are multiple results you can look at them and see which result looks best to you and take the corresponding entry in the results file. From Figure 7 the values needed are in “FILM\_DIFFUSION”, “PAR\_POROSITY”, and “PAR\_DIFFUSION”. Your results should be approximately the same values as Table 2.

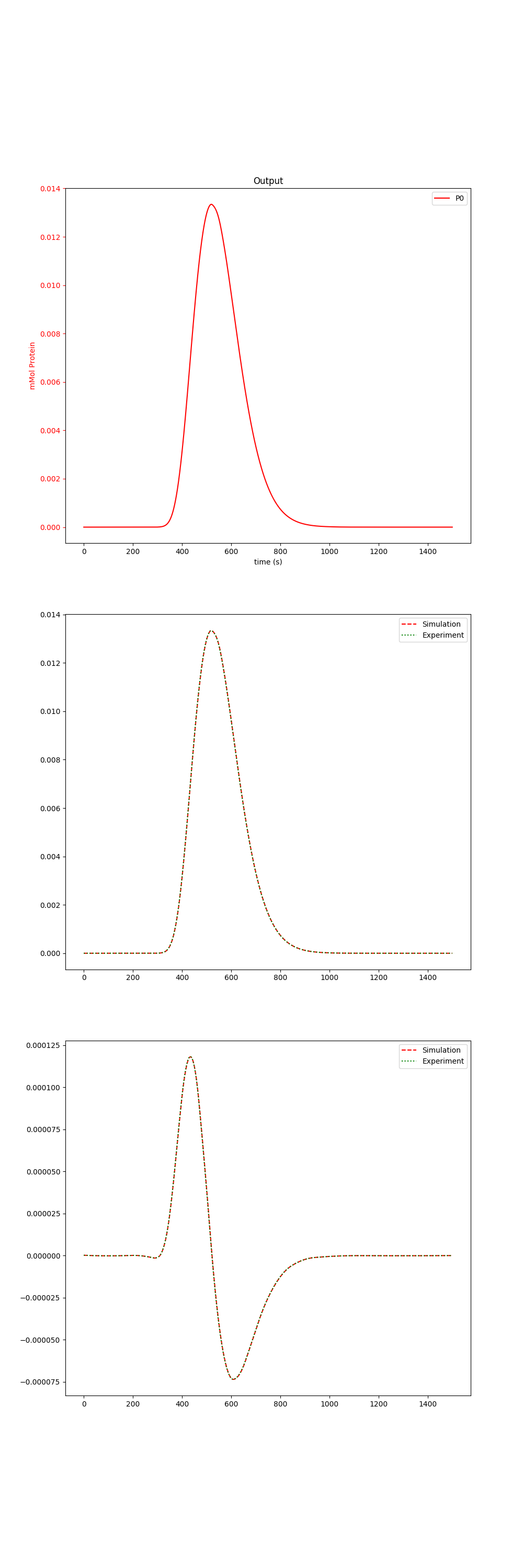


Figure 7 Example 1 Protein fit

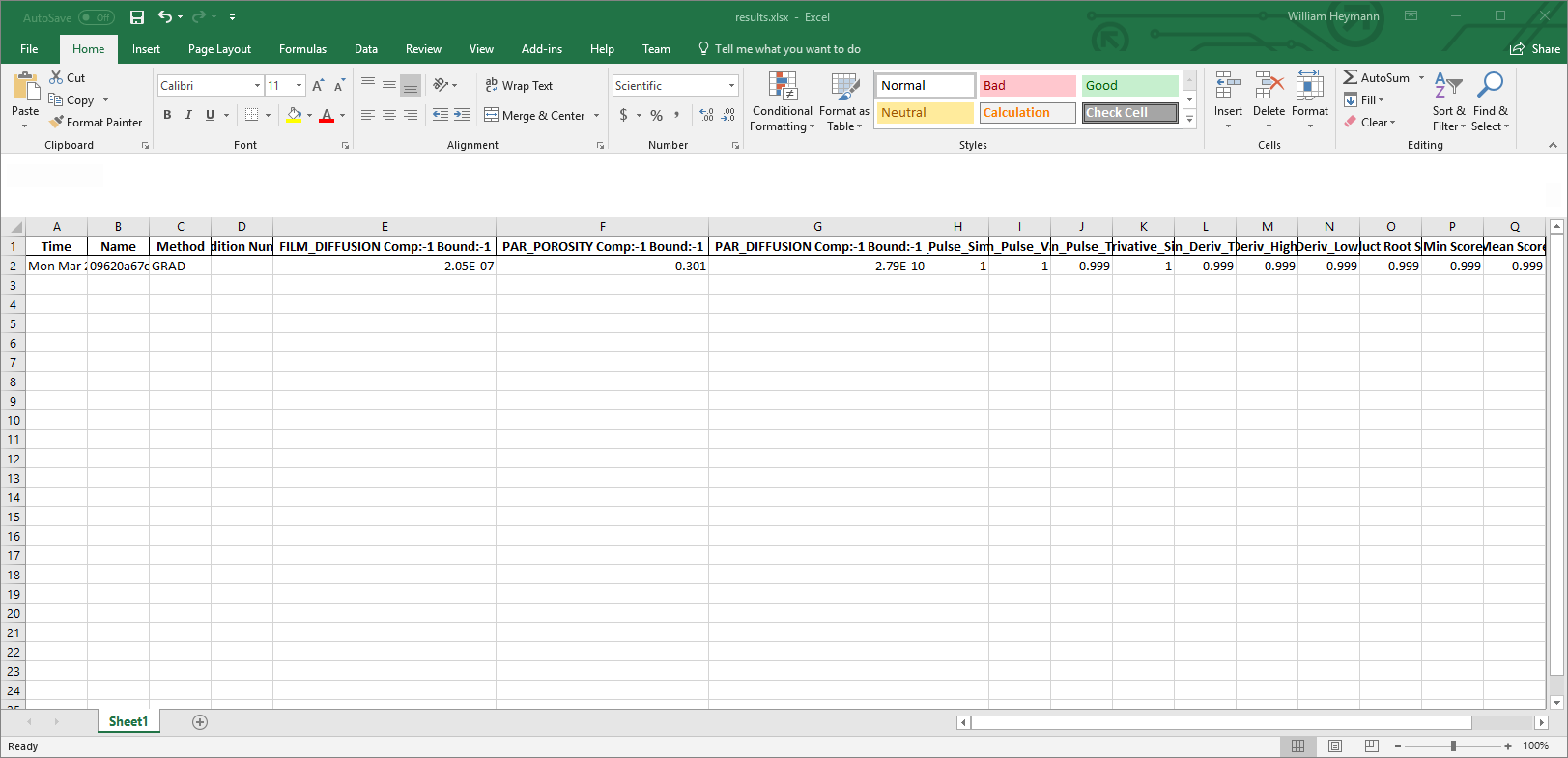


Figure 8 Example 1 Protein results

Table 2 Example 1 protein fit correct answers

|  |  |
| --- | --- |
| Name | Value |
| Film Diffusion | 2.0e-7 |
| Particle Porosity | 0.3 |
| Particle Diffusion | 1.0e-9 |

##### Identifiability

A picture containing indoor

Description generated with high confidence

### Fit isotherm

Fitting the isotherm is the longest and most complex part of parameter estimation. For this example, a trivial Linear isotherm example is used with 1 component. The linear isotherm is hard to get back the original parameters with kinetic binding due to very small changes moving the system between non-binding behavior and rapid equilibrium. As this is an example it is only important to see that the chromatogram is reproduced correctly instead of the exactly parameter values.

In the Examples folder under Example1/Isotherm there is a basic simulation along with the csv to fit against and the CADETMatch JSON file to use. After CADETMatch has been run check the folder fit/meta for the fit quality. The fit should look like Figure 8. The resulting parameters are in fit/meta/results.csv or fit/meta/results.xlsx. If there are multiple results you can look at them and see which result looks best to you and take the corresponding entry in the results file. From Figure 9 the values needed are in “LIN\_KA” and “LIN\_KD”. Your results should be approximately the same values as Table 3.

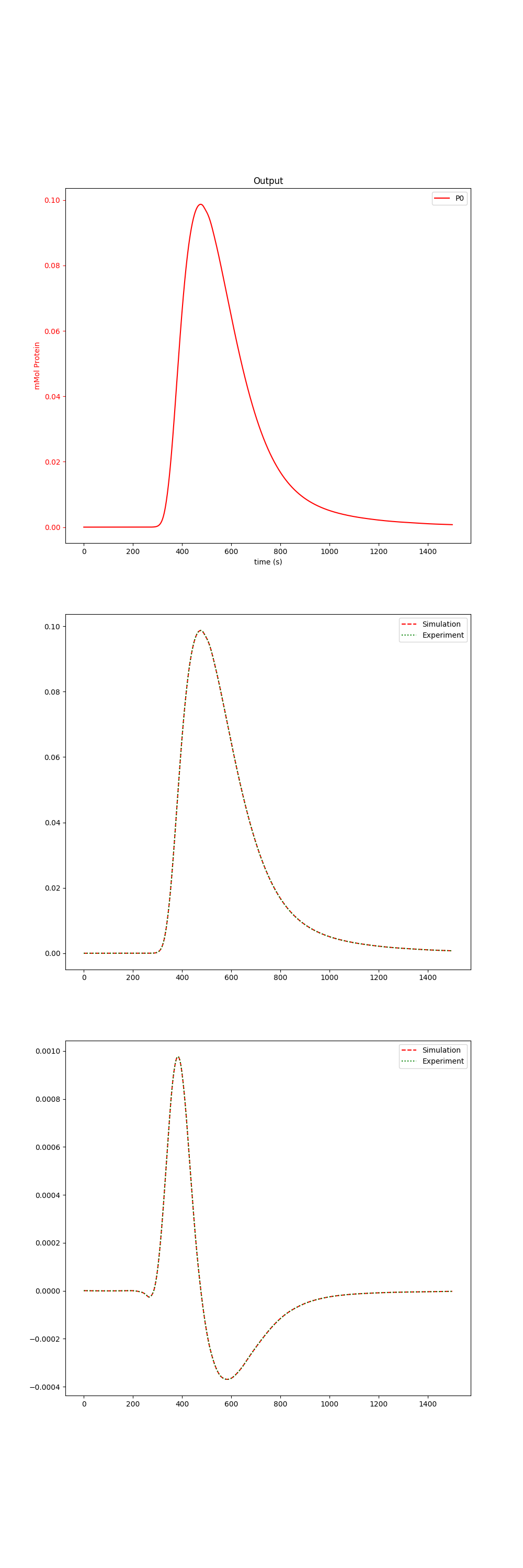


Figure 9 Example 1 Linear isotherm fit

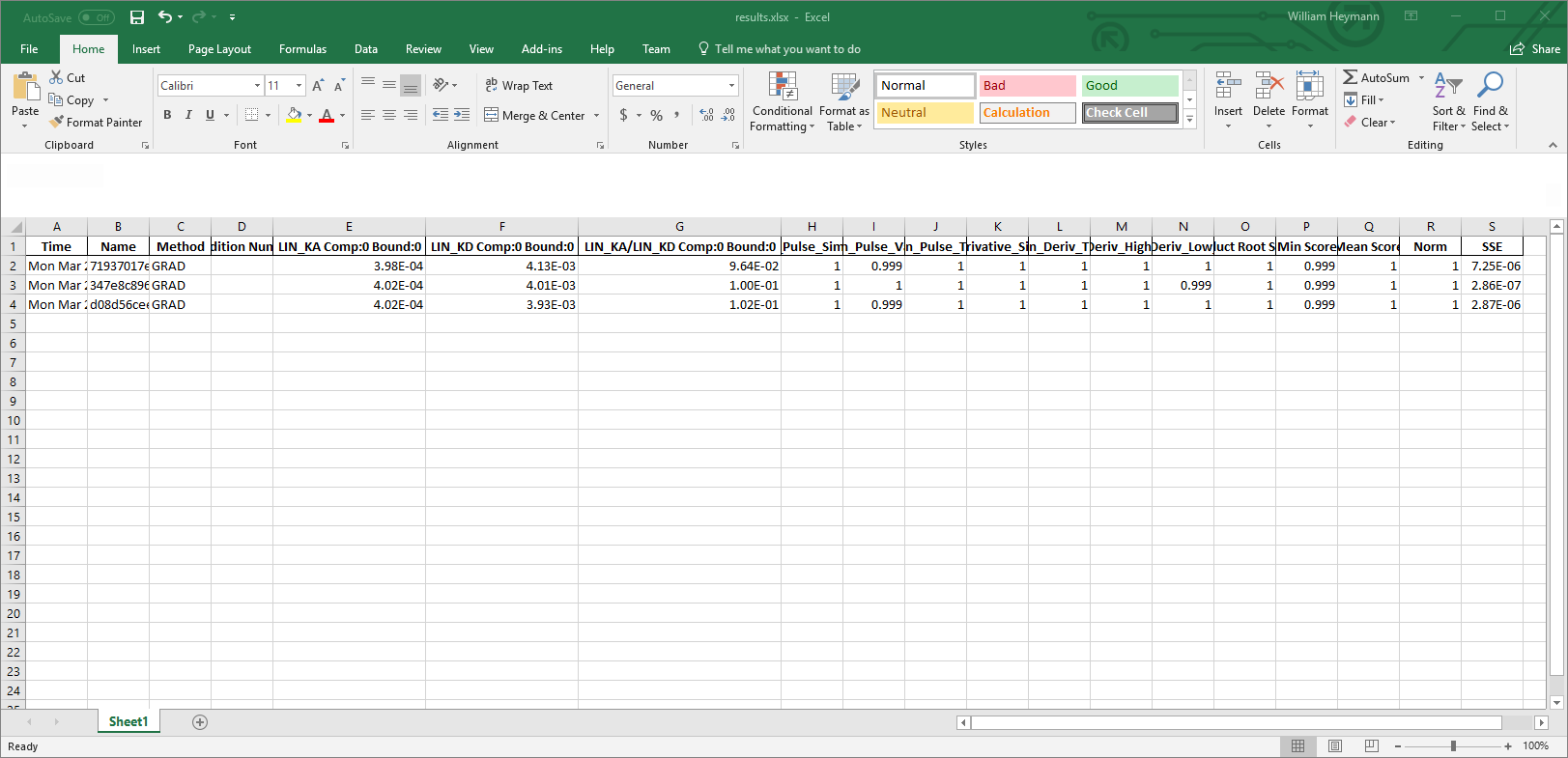
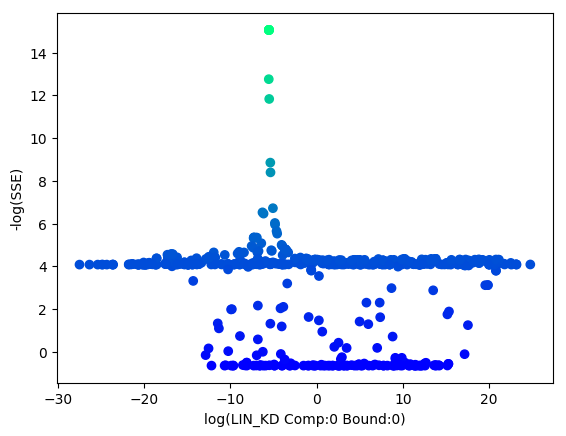
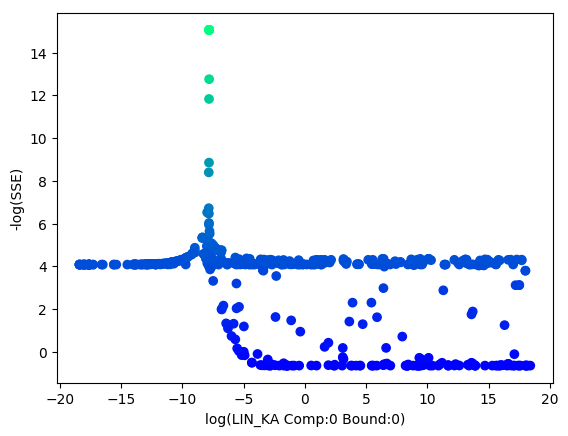


Figure 10 Example 1 Linear isotherm results

Table 3 Example 1 linear isotherm fit correct answers

|  |  |
| --- | --- |
| Name | Value |
| LIN\_KA | 4e-4 |
| LIN\_KD | 4e-3 |

#### Identifiability



## Example 2

Example 1 is an artificial example with a 1-component Steric Mass Action (SMA) isotherm using the General Rate Model. It covers what experiments are needed to fit the data along with how to do the fitting. All the data and scripts needed are in the examples folder. The example assumes that fitting the column properties has already been done from Example 1 and directly starts with fitting SMA properties. This example introduces multiple experiments.

### Fit Isotherm

SMA is a more complex isotherm to fit due to the additional parameters and the salt-interaction. Normally two experiments are needed when working with a single component: a load-wash-elution experiment and a breakthrough experiment. Neither experiment is unique on its own but together the result tends to be unique. If your system is effectively in rapid-equilibrium, then the kinetic version cannot return a unique answer and instead you will only be able to determine k­eq instead of ka and kd. When running the breakthrough experiment it is important that after the breakthrough is reached that the system is flushed with buffer until it returns to baseline as in Figure 10. This provides roughly double the information of just the front of the peak since the front and back and not symmetrical and provide additional information to use when fitting.

This type of fitting needs to be done on something with more computing power than a laptop or desktop computer. Many simulations need to be run and on a normal laptop or desktop this could take days to run.

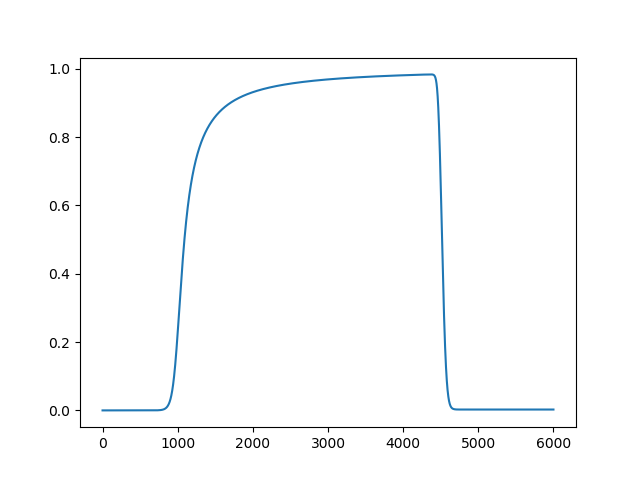
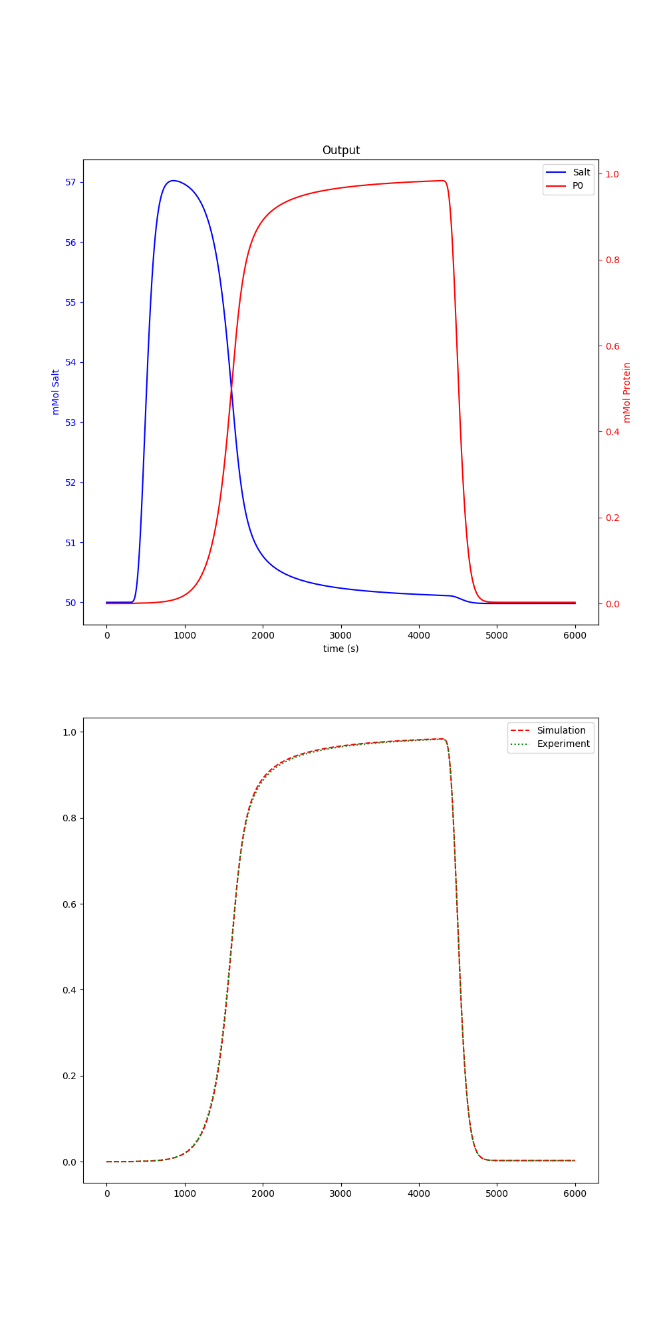
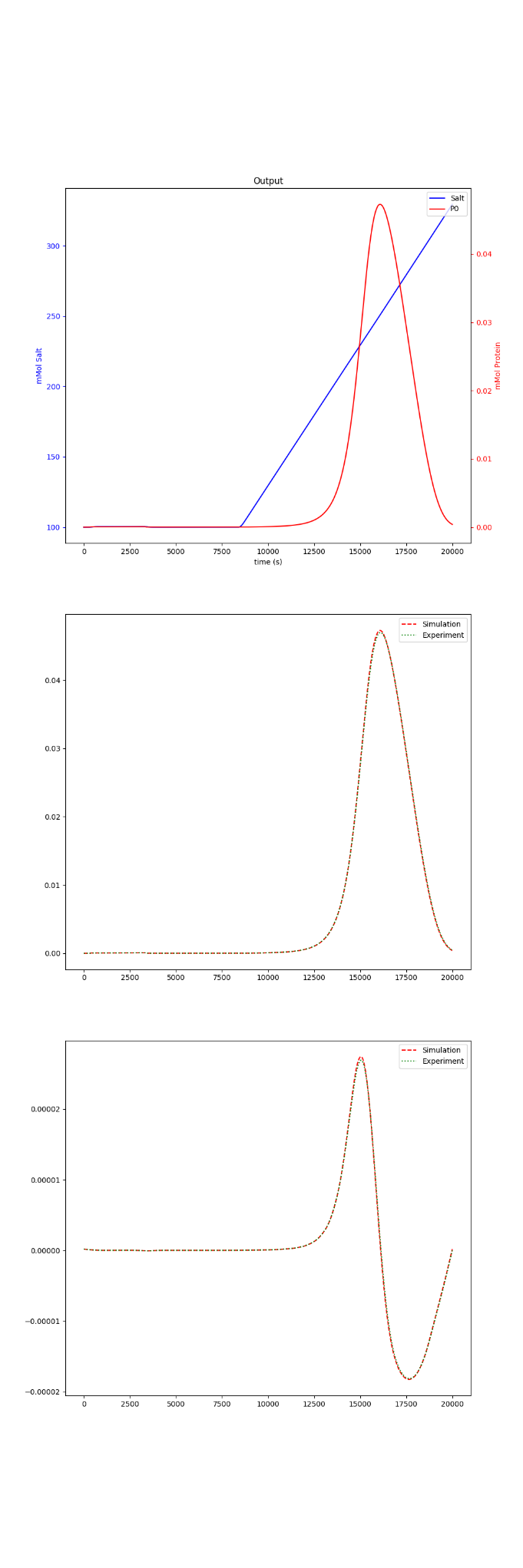


Figure 11 Example SMA breakthrough curve

In the Examples folder under Example2/Isotherm there is a CADETMatch JSON file to use. The Isotherm setup this time uses two experiments. There is a breakthrough experiment and a normal elution experiment to match against. This example uses a narrower than normal search range to speed up the search time. After CADETMatch has been run check the folder fit/meta for the fit quality. The fit should look like Figure 8. The resulting parameters are in fit/meta/results.csv or fit/meta/results.xlsx. If there are multiple results you can look at them and see which result looks best to you and take the corresponding entry in the results file. From Figure 9 the values needed are in “SMA\_KA”, “SMA\_KD”, “SMA\_NU”, and “SMA\_SIGMA”. Your results should be approximately the same values as Table 4.



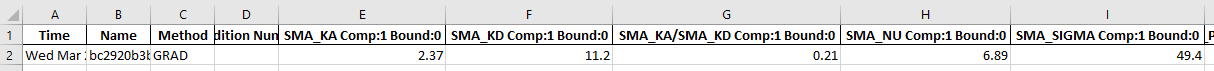
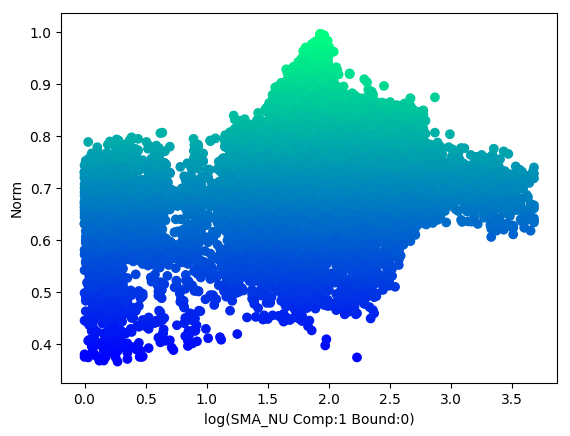
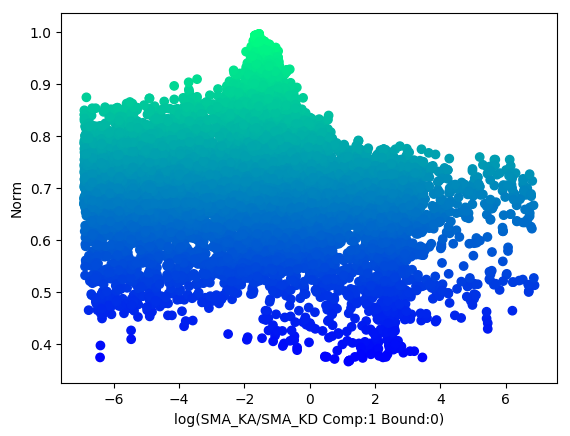


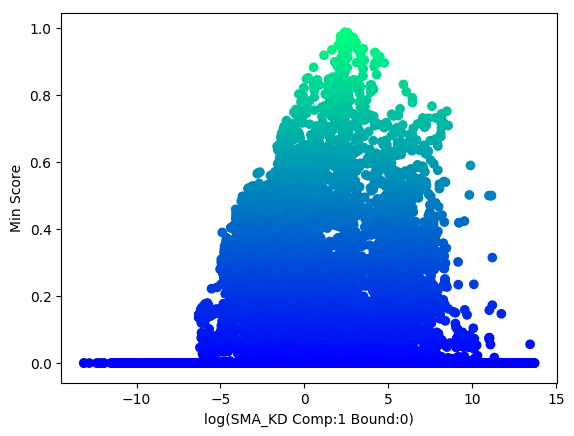
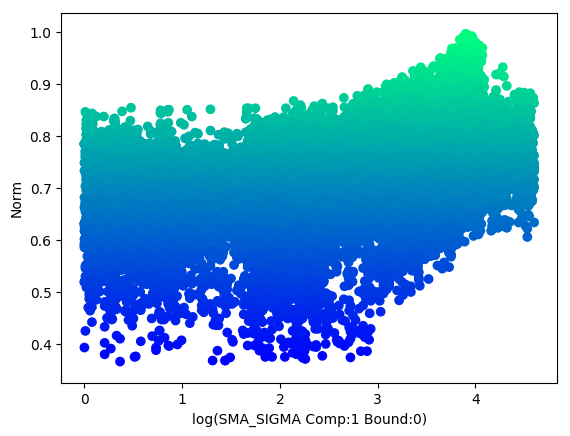
Table 4 Example 2 SMA isotherm fit correct answers

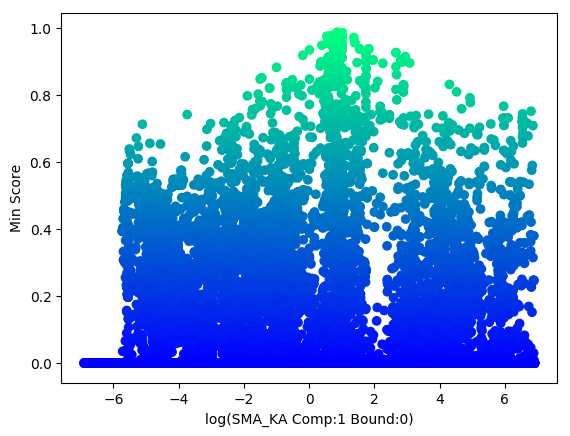
|  |  |
| --- | --- |
| Name | Value |
| SMA\_KA | 2.0 |
| SMA\_KD | 10.0 |
| KEQ | 0.2 |
| SMA\_NU | 7.0 |
| SMA\_SIGMA | 50.0 |

#### Identifiability

KA, KD, KEQ, NU, and SIGMA have good identifiability







#### JSON file

This example adds multiple experiments and multiple features into the JSON File. When adding multiple experiments each one of them is set apart with {…}, which you can see on line 86. The same thing is also true when using multiple features in lines 94-105 with the separation at 99. In Python terms experiments and features are lists of dictionaries. The important part is that two experiments cannot share the same name and two features within a single experiment cannot share the same name.

1. "experiments": [
2. {
3. "CSV": "break.csv",
4. "isotherm": "/output/solution/unit\_001/SOLUTION\_OUTLET\_COMP\_001",
5. "HDF5": "break.h5",
6. "name": "break",
7. "timeout": 13.23845386505127,
8. "features": [
9. {
10. "name": "Pulse",
11. "start": 0,
12. "stop": 6000.0,
13. "type": "breakthroughHybrid2"
14. }
15. ]
16. },
17. {
18. "CSV": "pulse.csv",
19. "isotherm": "/output/solution/unit\_001/SOLUTION\_OUTLET\_COMP\_001",
20. "HDF5": "pulse.h5",
21. "name": "pulse",
22. "timeout": 32.43798017501831,
23. "features": [
24. {
25. "name": "Pulse",
26. "start": 0,
27. "stop": 20000.0,
28. "type": "similarityHybrid2"
29. },
30. {
31. "name": "Deriv",
32. "start": 0,
33. "stop": 20000.0,
34. "type": "derivative\_similarity\_hybrid2"
35. }
36. ]
37. }
38. ]

## Example 3

Example 1 is an artificial example with a 2-component Steric Mass Action (SMA) isotherm using the Lumped Rate Model with Pores. It covers what experiments are needed to fit the data along with how to do the fitting. All the data and scripts needed are in the examples folder. The example assumes that fitting the column properties has already been done from Example 1 and directly starts with fitting SMA properties. This example introduces multiple components.

### Fit Isotherm

Fitting multiple components largely works the same way as fitting a single component. However, with SMA the components compete for binding sites. This makes fitting harder and thus take much longer. The more components you have the harder the system is to fit.

#### Identifiability

#### JSON file

This example adds multiple components and per-component scores. Lines 93 to 120 has an example of multiple components and lines 130 to 145 has an example of separate scores for each component.

1. {
2. "transform": "log",
3. "component": 1,
4. "bound": [
5. 0
6. ],
7. "location": "/input/model/unit\_001/adsorption/SMA\_SIGMA",
8. "min": [
9. 1.0
10. ],
11. "max": [
12. 100.0
13. ]
14. },
15. {
16. "transform": "log",
17. "component": 2,
18. "bound": [
19. 0
20. ],
21. "location": "/input/model/unit\_001/adsorption/SMA\_SIGMA",
22. "min": [
23. 1.0
24. ],
25. "max": [
26. 100.0
27. ]
28. }

## 

1. {
2. "CSV": "pulse\_c1.csv",
3. "isotherm": "/output/solution/unit\_001/SOLUTION\_OUTLET\_COMP\_001",
4. "name": "Pulse",
5. "start": 0,
6. "stop": 25000.0,
7. "type": "similarityHybrid"
8. },
9. {
10. "CSV": "pulse\_c2.csv",
11. "isotherm": "/output/solution/unit\_001/SOLUTION\_OUTLET\_COMP\_002",
12. "name": "Pulse2",
13. "start": 0,
14. "stop": 25000.0,
15. "type": "similarityHybrid"
16. },

## Example 4

Multiple-Components (SMA) with Fractionation