eSPC, an Online Data Analysis Platform for Molecular Biophysics

MoltenProt 1.1 User Documentation

January 2026

Table of Contents

Overview
MoltenProt has seven panels (Figure 1). Panels 1-3 contain the necessary steps to analyse the data. Panel 4 can be used to export the results of the analysis.

Figure 1. Screenshot of MoltenProt sidebar.

1. New features v1.1

1.1. Barycentric mean

When importing whole spectrum data generated by *Applied Photophysics* or *Unchained Labs* DSF instruments, the barycentric mean is also computed. This value is defined as:

$$BCM(I,\lambda) = \sum_{i}^{n} \lambda_i I_i \,/\, \sum_{i}^{n} I_i \qquad (1)$$

where I_i is the intensity at wavelength λ_i. We recommend using the barycenter for qualitative studies (see Section 'Signal to analyse').

1.2. Singular value decomposition (SVD)

When importing whole spectrum data, a new menu can be opened by pressing the *'Show full spectrum menu*' button. Inside that menu, we can apply, separately for each sample, singular value decomposition. Briefly, the spectra at the different temperatures are decomposed as a linear combination of basis spectra as follows:

$$Spectrum(t) = c_1(t)\phi_1 + c_2(t)\phi_2 + c_3(t)\phi_3 + \ldots + c_n(t)\phi_n \qquad (2)$$

where t is the temperature, each ϕ_i is a basis spectrum and c_i are the associated coefficients. The basis spectra are orthogonal to each other and have unit norm. Typically, in differential scanning fluorimetry datasets, a few basis spectra are relevant and explain most of the variance in the data. In MoltenProt, only the coefficients of the first and second basis spectra are returned for analysis.

1.3. Native and unfolded baselines

The baseline of the native and unfolded states are centered at 25°C, instead of 0 Kelvin. For example, the signal produced by the folded state changed from

$$Signal(T) = f_n(k_n T + b_n) \qquad (3)$$

to

$$Signal(T) = f_n(k_n \Delta T + b_n) \qquad (4)$$

where *T* is the temperature, ΔT is the temperature minus the reference temperature 293.15K (20°C), f_n is the fraction of native protein, b_n is the intercept term, and k_n is the slope term.

Moreover, the baselines for the folded and unfolded state can be modelled with a constant, linear, or quadratic equation:

$$Signal(T) = f_n b_n \qquad (5)$$

or

$$Signal(T) = f_n(b_n + k_n \Delta T) \qquad (6)$$

or

$$Signal(T) = f_n(b_n + k_n \Delta T + q_n \Delta T^2) \qquad (7)$$

where b_n is the intercept, k_n is the linear term and q_n is the quadratic term.

1.4. Empirical models

There are two new implementations for the empirical models (See Appendix for a mathematical derivation).

1.5. Filters

After the fitting is done, the results can be filtered based on

a) A selected threshold for the relative errors of the fitted parameters
b) A selected threshold for the standard error of the fitting
c) Minimum and maximum values for certain parameters (T_m, T_{onset}, ΔH, *etc.*)

2. Data import and processing

2.1. Input file (raw data)

MoltenProt can parse several types of files:

A) The xlsx file (processed) generated by the Nanotemper Prometheus instrument that has one sheet called 'Overview' with a column called 'Sample ID' with the names of the samples (Figure 2), and four sheets called 'Ratio', '330nm', '350nm' and 'Scattering'. The first column of the signal sheet ('Ratio', '330nm', '350nm', 'Scattering') should be called 'Time [s]'. The second column should have the temperature data and all subsequent columns store the fluorescence data (Figure 3). The order of the fluorescence columns should match the order of the 'Sample ID' column in the 'Overview' sheet.

B
Sample ID
A1 GuHCl 0.05 M
A2 GuHCl 0.52 M
A3 GuHCl 1 M
A4 GuHCl 1.47 M
A5 GuHCl 1.95 M
A6 GuHCl 2.38 M
A7 GuHCl 2.5 M
A8 GuHCl 2.61 M
A9 GuHCl 2.73 M
A10 GuHCl 2.85 M
A11 GuHCl 2.97 M
A12 GuHCl 3.09 M
B1 GuHCl 3.21 M
B2 GuHCl 3.33 M
B3 GuHCl 3.45 M
B4 GuHCl 3.56 M
B5 GuHCl 3.68 M
B6 GuHCl 4.25 M
B7 GuHCl 4.82 M
B8 GuHCl 5.39 M

Figure 2. Example of the ‘SampleID’ column in the ‘Overview’ sheet required by MoltenProt to load the Nanotemper spreadsheet input file.

	A	B	C	D
1		Capillary	1	2
2		Sample ID	A1	
3	Time [s]	Temperature [°C]	Fluorescence [counts]	Fluorescence [counts]
4	7.0	25.000	0.940	0.934
5	24.3	25.054	0.939	0.935
6	33.3	25.108	0.943	0.933
7	40.6	25.162	0.939	0.936
8	46.8	25.215	0.942	0.933
9	52.5	25.269	0.941	0.933
10	57.4	25.323	0.942	0.934
11	62.1	25.377	0.941	0.934
12	66.7	25.431	0.942	0.934
13	71.0	25.485	0.940	0.933

Figure 3. Example of the ‘Ratio’ sheet required by MoltenProt to load the Nanotemper spreadsheet input file.

B) The xls file generated by the ThermoFluor assay in a qPCR instrument. This file has one sheet called 'RFU' where the first row has the sample positions (header), the first column has the temperature data and all subsequent columns store the fluorescence data (Figure 4).

A	B	C	D	E	F	G	H
	A01	A02	A03	A04	A05	A06	A07
5	64.79	501.82	398.53	61.91	73.26	129.38	38.53
6	63.14	513.32	416.32	63.13	72.41	130.21	40.43
7	61.52	522.98	437.17	64.34	72.21	131.14	42.45
8	59.75	529.89	459.98	64.97	72.52	131.29	42.69
9	57.78	535.95	483.14	65.89	72.30	131.90	43.18
10	55.73	540.72	504.85	67.40	71.83	131.75	42.82
11	54.00	545.15	527.02	68.86	71.27	131.86	42.98
12	52.82	549.80	549.27	70.20	71.55	131.55	42.65
13	52.14	554.45	570.59	70.37	72.86	131.79	42.15
14	51.42	558.53	589.62	70.41	74.39	132.50	41.38

Figure 4. Example of the ‘RFU’ sheet required by MoltenProt to load ThermoFluor data.

C) The xlsx file generated by a Prometheus Panta instrument. This file has one sheet called ‘Overview’ with a column called 'Sample ID' with the names of the samples (Figure 2), and one sheet called ‘Data Export’ where all the data is stored (Figure 5). The ‘Data Export’ sheet columns should have the following order:

Temperature capillary 1 ; Ratio capillary 1 ; … ; Temperature capillary 1 ; 350 nm capillary 1 ; … ; Temperature capillary 1 ; 330 nm capillary 1 ; … ; Temperature capillary 1 ; scattering capillary 1 ; … ; Temperature capillary 2 ; Ratio capillary 2; … ; Temperature capillary *n* ; Ratio capillary *n*.

Columns whose names include "Derivative" are not read.

Temperature for Cap.1 (°C)	Ratio 350 nm / 330 nm for Cap.1
24.9993991851807	0.668331980705261
25.0000820159912	0.668272197246552
25.0013084411621	0.668575644493103
25.001501083374	0.66842645406723
25.0040893554687	0.667966544628143
25.0060691833496	0.668425559997559
25.0104427337646	0.668148577213287
25.0115489959717	0.668168842792511
25.016487121582	0.668507099151611

Figure 5. Example of the 'Data Export' sheet required by MoltenProt to load the Prometheus Panta spreadsheet input file.

D) The text file (.txt) generated by a QuantStudio™ 3 System instrument (Figure 6). Comments should be at the beginning of the file and start with '*'. The header is the first line with more than 6 words, i.e. 'Well', 'Well Position', … , 'Target Name'. The column number 2 has the sample IDs. The columns number 4 and 5 have the signal and temperature data.

```
* Pre-read Stage/Step =
* Quantification Cycle Method = Ct
* Signal Smoothing On = true
* Stage where Melt Analysis is performed = Stage2
* Stage/ Cycle where Ct Analysis is performed = Stage0, Step0
* User Name = user

[Melt Curve Raw Data]
Well    Well Position   Reading Temperature     Fluorescence    Derivative      Target Name
10      A10     1       24.913  4,828.486       -1,144.751      Target 1
10      A10     2       25.028  4,951.091       -905.159        Target 1
10      A10     3       25.142  4,722.069       -659.505        Target 1
10      A10     4       25.257  4,786.491       -469.428        Target 1
10      A10     5       25.371  5,124.026       -375.954        Target 1
10      A10     6       25.486  4,948.212       -373.484        Target 1
10      A10     7       25.601  4,832.502       -411.521        Target 1
10      A10     8       25.715  4,961.322       -422.667        Target 1
10      A10     9       25.830  4,779.467       -357.793        Target 1
10      A10     10      25.944  5,039.901       -208.366        Target 1
```

Figure 6. Example of the input file required by MoltenProt to load the QuantStudio™ 3 System instrument data.

E) The xlsx file generated by the Nanotemper Prometheus Tycho instrument. This file has one sheet called 'Results' (with 6 columns named '#', 'Capillary label',..., 'Sample Brightness') (Figure 7), and one sheet called 'Profiles_raw' where the fluorescence data is stored.

#	Capillary label	Ti#1	Ti#2	Ti#3	Initial Ratio	Δ Ratio	Sample Brightness
1	Sample1	54.2			0.6700	0.2637	763.4
2	Sample2	54.7			0.5777	0.1825	688.3
3	Sample3	89.7			0.6071	0.0950	382.1
4	Sample4				0.6230	0.0480	73.6
5	Sample5						
6	Sample6						

Figure 7. Example of the 'Results' sheet required by MoltenProt to retrieve the sample names. This example file was generated by the Nanotemper Prometheus Tycho instrument.

The 'Profiles_raw' sheet columns should have the following structure (Figure 8):

One row with information about the recorded signal, e.g., 'Ratio 350 nm / 330 nm', 'Brightness @ 330 nm', 'Brightness @ 350 nm'.

One row with the capillary numbers.

One row with the time, temperature and sample names.

The remaining rows store the temperature and signal data.

	Signal:	Ratio 350 nm / 330 nm			
	Capillary:	1	2	3	4
Time [s]	Temperature [°C]	Sample1	Sample2	Sample3	Sample4
81.5	35.1	0.6699	0.5774	0.6065	0.6233
81.7	35.2	0.6701	0.5781	0.6063	0.6160
81.9	35.3	0.6705	0.5774	0.6079	0.6229
82.1	35.4	0.6704	0.5772	0.6056	0.6272

Figure 8. Example of the 'Profiles_raw' sheet required by MoltenProt to load the temperature and signal data. This example file was generated by the Nanotemper Prometheus Tycho instrument.

F) The text file (.txt) generated by Agilent's **MX3005P qPCR instrument**. The data format is the following:

Line 1: Header
Line 2: Segment 2 Plateau 1 Well 1
Line 3: ROX
Line 4: 1 1706 25.0
Line 5: 2 2581 25.8
Line n: 70 4845 93.7
Line n+1: Segment 2 Plateau 1 Well 2
Line n+2: ROX
Line n+3: 1 1707 25.0

The data of each well is separated by rows containing the sentence ‘Segment No Plateau No Well No’. The rows after the line with ‘ROX’ contain the fluorescence and temperature data (second and third columns respectively).

G) The JSON file (.supr) exported by the SUPR-DSF instrument software from AppliedPhotophysics (https://www.photophysics.com/product-pages/supr-dsf/).

The JSON file should have the following structure:

- An item called 'Samples' that contains:
    - A sub-item called 'SampleName'
    - A sub-item called 'WellLocations'
- An item called 'Wells' that contains:
    - A sub-item called '_scans'
    - A sub-item called 'PhysicalLocation'
- An item called 'Wavelengths'

Additionally, within the '_scans' sub-item, there are further sub-items called:
- 'Temperature'
- 'Signal'

Below you’ll find a minimal example.

```
{
 "Samples": [
  {
   "SampleName": "SampleA1",
   "WellLocations": "A1"
  }
 ],
 "Wells": [
  {
   "_scans": [
    {
     "Temperature": 20,
     "Signal": [
      7993,
      9579,
      10742
     ]
    },
    {
     "Temperature": 30,
     "Signal": [
      8993,
      10579,
      11742
     ]
    }
```

```
    ],
    "PhysicalLocation": "A1"
   }
  ],
  "Wavelengths": [
   310,
   311,
   312
  ]
}
```


After loading the file, the temperature data will be interpolated using steps of 0.5 degrees. Additionally, if there is wavelength data near 330 nm and 350 nm, the 'Ratio 350nm / 330nm' will be automatically calculated and available for further analysis.

H) A csv file with the temperature data in the first column and the signal data in the subsequent columns. The condition labels are read from the header.

	A	B	C	D
1	Temperature	Condition 1	Condition 2	Condition 3
2	10	20	20	20
3	11	30	30	30
4	12	40	40	40
5	13	50	50	50
6	14	60	60	60
7	15	70	70	70
8	16	80	80	80
9	17	90	90	90
10	18	100	100	100
11	19	110	110	110
12	20	120	120	120
13	21	130	130	130
14	22	140	140	140

Figure 9. Example of a csv file that can be imported into MoltenProt.

I) The spreadsheet file (.xlsx) exported by the UNCLE instrument. In this file, there is one sheet per measured condition. The name of the condition is extracted from the cell located in the first row and fifth column. The second row has the temperature data in the format ‘Temp: xx, Time: xx’. The signal matrix starts at the fifth row, second column. The wavelength data is in the first column.

	A	B	C	D	
1				Sample Name	1
2		Temp :25, Time:134.9	Temp :25.49, Time:18	Temp :25.99, Time:23	T
3	Wavelength	Intensity	Intensity	Intensity	In
4					
5	311.911193847656	1157.984	1136.362	1118.812	
6	312.387084960938	1194.752	1185.197	1162.002	
7	312.862884521484	1244.214	1232.758	1214.172	
8	313.338684082031	1299.843	1286.473	1264.072	
9	313.814392089844	1358.016	1330.473	1312.704	
10	314.290100097656	1410.235	1378.81	1368.539	
11	314.765716552734	1483.058	1447.716	1421.273	
12	315.241333007813	1525.251	1492.625	1492.752	

Figure 10. Example of one sheet from the UNCLE xlsx file.

J) The spreadsheet file (.xlsx) exported by the AUNTY instrument. In this file, there is one sheet per measured condition. The name of the condition is extracted from the sheet name. The first column has the temperature data, while the second row contains the wavelength data. The signal matrix starts at the third row, second column.

	A	B	C	D
1		wavelength		
2	temperature	250	250.49	250.98
3	14.99	100.00	100.00	100.00
4	15.48	100.00	100.00	100.00
5	16.03	100.00	100.00	100.00
6	16.54	100.00	100.00	100.00
7	17.05	100.00	100.00	100.00
8	17.55	100.00	100.00	100.00

Figure 11. Example of one sheet from the AUNTY xlsx file.

2.2. Normalisation

There are 3 available options to normalise each fluorescence-based melting curve.

a. Divide by initial value: Divide by the median value of the signal corresponding to the first two degrees of temperature.

b. Max-min normalisation: Transform the signal by applying

$$y_{norm}(y) = \frac{y - min(y)}{max(y) - min(y)} \quad (8)$$

c. Area normalisation: Divide the signal by the area under the curve (calculated using the trapezoidal rule).

2.3. Median filter (smoothing)

The median filter consists of calculating the median value of a temperature rolling window.

2.4. Savitzky-Golay (SG) window size

This parameter, in degrees Celsius, is used to calculate the number of data points to apply the Savitzky-Golay filter corresponding to a polynomial of degree 4 before computing the first or second derivative as implemented in Scipy (scipy.signal.savgol_filter — SciPy v1.6.1 Reference Guide). For the second derivative, we add 5 degrees to the selected SG temperature window size.

The number of data points $n(w)$ is obtained by computing

$$n(w) = ceil(\frac{w}{\Delta T}) \,//\, 2 * 2 + 1 \qquad (9)$$

where w is the SG parameter fixed by the user, $ceil(x)$ returns the smallest integer i such that $i >= x$, and ΔT corresponds to the average number of data points in one degree of temperature.

2.5. Melting temperature (T_m) estimation using the first derivative

A non-model approach to estimate the melting temperature involves estimating the maximum or the minimum of the first derivative, depending on the way the signal changes with the temperature. In MoltenProt, the Tm values are estimated as follows. First, the median value of the first derivative in the interval $[min(temperature) + 6; min(temperature) + 11]$ and $[max(temperature) - 11; max(temperature) - 6]$ is calculated. Then, we obtain the mean of those two median values and add it (if it positive), or subtract it (if it is negative), to the first derivative in the interval $[min(temperature) + 6; max(temperature) - 6]$. This is done to shift the derivative baseline. Last, if the absolute value of the minimum (of the derivative) is higher than the absolute value of the maximum, we use the minimum to estimate the Tm. Otherwise, we use the maximum. If many curves are present, we always use the same option.

3. Fitting

3.1. Signal to analyse

There are several fluorescence changes that can be monitored, such as the fluorescence intensity at a given excitation and emission wavelengths, the quantum yield, the emission maximum, the fluorescence lifetime, the ratio of fluorescence intensities at two different emission wavelengths, *etc.* (Maurice R. Eftink *et al.*, 1994). To be able to reliably estimate thermodynamic parameters, the signal must be related to the mole fractions of the different species. The most straightforward approach is to use extensive properties that are proportional to the amount of material in the system (Hugues Bedouelle, 2016). This way, the observed signal can be expressed as a linear combination of the signals of the different species weighted by their mole fractions.

The fluorescence intensity at a given excitation and emission wavelengths is an extensive property and can be used to monitor unfolding as described above. However, the emission maximum, or the ratio signal, defined as the ratio of the fluorescence intensities at two different emission wavelengths, are not extensive properties and using them may result in unreliable estimates (Hugues Bedouelle, 2016; Gabriel Žoldák, 2017).

3.2. Model selection

The models presented here assume that the fluorescence signal can be expressed as a linear combination of the signal produced by the different protein states. The difference in the models lies in establishing which are the possible states and how to calculate their concentration. The Python functions are available at Github.

$N \Leftrightarrow U$ Equilibrium (or Empirical) two-state model

$N \Leftrightarrow I \Leftrightarrow U$ Equilibrium (or Empirical) three-state model

$N \Rightarrow U$ Irreversible two-state model

Furthermore, there can be no fluorescence dependence on temperature, a linear dependence, or a quadratic dependence

Equilibrium two-state[1,2]

This thermodynamic-based model presupposes that the protein only exists in the native (folded) or unfolded state and that there is an equilibrium between these two states given by the unfolding reaction N ⇆ U.The fluorescence signal $F(T)$ is described by the equation

$$F(T) = f_n(q_n\Delta T^2 + k_n\ \Delta T + b_n) + f_u(q_u\Delta T^2 + k_u\ \Delta T + b_u) \quad (10)$$

where q_n, k_n, b_n are the quadratic, linear and constant terms of the pre-transition baseline (native), q_u, k_u and b_u are the quadratic, linear and constant terms of the post-transition (unfolded) baseline, f_n is the fraction of protein in the native state and f_u is the fraction of protein in the denatured state. f_n is calculated as

$$f_n(T) = (1 + K_u)^{-1} \quad (11)$$

where

$$K_u(T) = e^{-\Delta G\ /\ RT} \quad (12)$$

and

$$\Delta G(T) = \Delta H_m(1 - \frac{T}{T_m}) - \Delta C_p(T_m - T + T\ ln(\frac{T}{T_m}))) \quad (13)$$

where *R* is the universal gas constant, ΔH_m is the enthalpy of unfolding at the melting temperature T_m, and ΔC_p is the heat capacity of unfolding. ΔC_p is not fitted and assumed to be zero. If the User has knowledge about that value, it can be used to correct the calculated Gibbs energy of unfolding at the standard temperature.

Figure 12 shows a simulation of the model.

---

[1] Santoro, M. M., & Bolen, D. W. (1988). Unfolding free energy changes determined by the linear extrapolation method. 1. Unfolding of phenylmethanesulfonyl. alpha.-chymotrypsin using different denaturants. *Biochemistry*, *27*(21), 8063-8068.

[2] Bedouelle, H. (2016). Principles and equations for measuring and interpreting protein stability: From monomer to tetramer. *Biochimie*, *121*, 29-37.

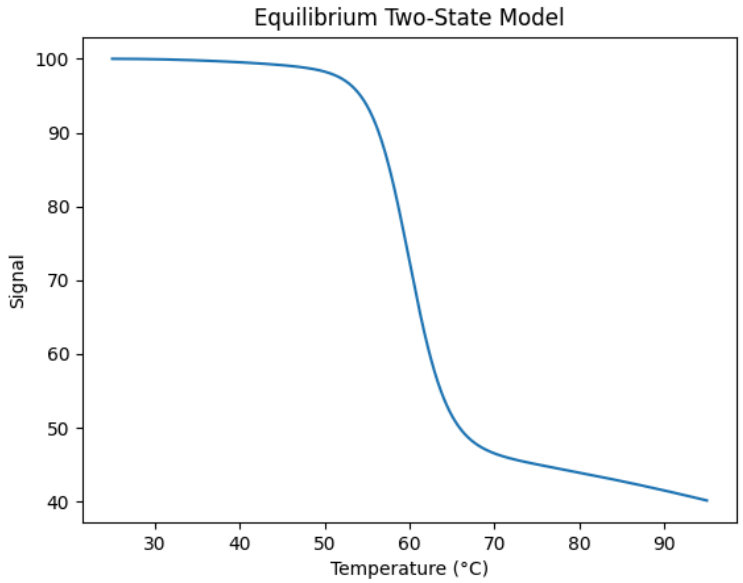

Figure 12. Simulated signal of the Equilibrium two-state unfolding model. The following parameters were used: $\Delta H = 100\, kcal/mol$, $T_m = 55°C$, $b_n = 100$, $b_u = 50$, $k_n = -0.001$, $k_u = -0.001$, $q_n = -0.002$, and $q_u = -0.002$.

Empirical two-state v1.1

This model is similar to the Equilibrium two-state, but instead of enthalpy of unfolding, it uses the parameter T_{onset} to model the steepness of the fluorescence curve. The signal is described by the same expression as Equation 10, with the difference that $\Delta G(T)$ is calculated as follows:

$$\Delta G(T) = \frac{(T_m - T)(RT_{onset} ln(0.01/0.99))}{T_{onset} - T_m} \quad (14)$$

Figure 13 shows a simulation of the model.

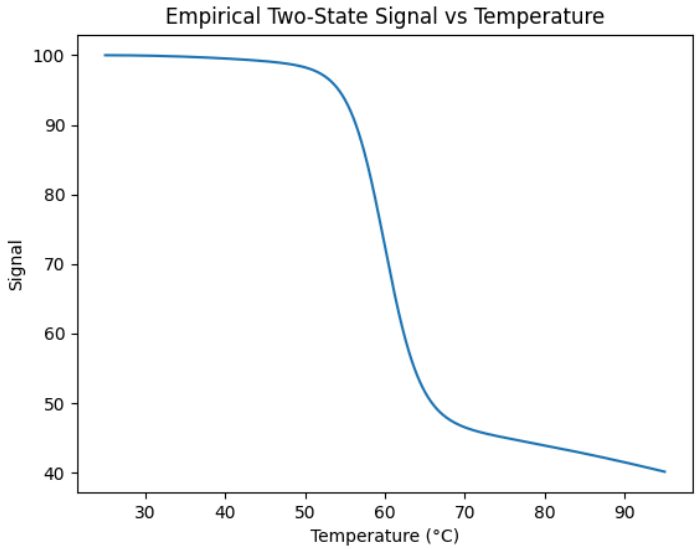

Figure 13. Simulated signal of the Empirical two-state unfolding model. The following parameters were used: $T_{onset} = 50.15°C$, $T_m = 55°C$, $b_n = 100$, $b_u = 50$, $k_n = -0.001$, $k_u = -0.001$, $q_n = -0.002$, and $q_u = -0.002$. The signal values are the same as in Figure 12.

Equilibrium three-state[3]

This model adds the presence of one short-lived protein state: native (N), intermediate (I) and unfolded (U). The fluorescence signal is described by the equation:

$$F(T) = f_n(q_n\Delta T^2 + k_n\ \Delta T + b_n) + f_u(q_u\Delta T^2 + k_u\ \Delta T + b_u) + f_i b_i \quad (15)$$

where

f_i is the fraction of protein in the intermediate state, and b_i is the signal produced by the intermediate state (no dependence on temperature is included).

There are two unfolding equilibria, N ⇆ I and I ⇆ U. The respective equilibrium constants K_1 and K_2 are calculated using Equation 12. ΔC_p is set to zero in both cases. As a result, the model contains four thermodynamic parameters. T_1 and ΔH_1 for the first reaction, and T_2 and ΔH_2 for the second one. f_n, f_i and f_u are given by:

[3] Mazurenko, S., Kunka, A., Beerens, K., Johnson, C. M., Damborsky, J., & Prokop, Z. (2017). Exploration of protein unfolding by modelling calorimetry data from reheating. *Scientific reports*, *7*(1), 1-14.

$$f_n = \frac{1}{1+K_1+K_1K_2} \qquad (16)$$

$$f_i = \frac{K_1}{1+K_1+K_1K_2} \qquad (17)$$

$$f_u = \frac{K_1K_2}{1+K_1+K_1K_2} \qquad (18)$$

Figure 14 shows a simulation of the model.

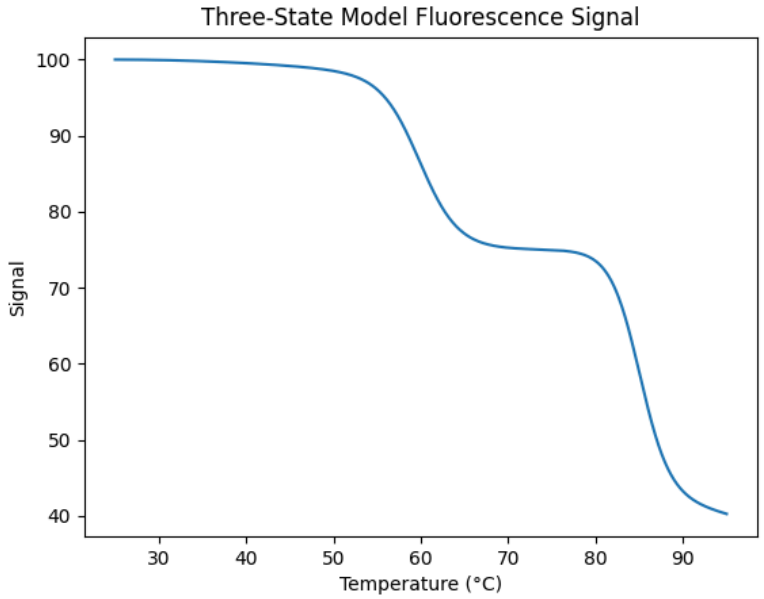

Figure 14. Simulated signal of the Equilibrium three-state unfolding model. The following parameters were used: $\Delta H_1 = 100\, kcal/mol$, $T_1 = 55°C$, $b_n = 100$, $b_i = 75$, $b_u = 50$, $k_n = -0.001$, $k_u = -0.001$, $q_n = -0.002$, and $q_u = -0.002$, $\Delta H_2 = 150\, kcal/mol$, $T_2 = 85°C$.

Empirical three-state v1.1

This model is similar to the Equilibrium three-state, but instead of enthalpy of unfolding, it uses the parameters $T_{onset,1}$ and $T_{onset,2}$ to model the steepness of the two transitions:

$$\Delta G_{N \leftrightarrows I}(T) = (T_1 - T)\,(R\,T_{onset,1} ln(\frac{0.01}{0.99}))\,/\,(T_{onset,1} - T_1) \qquad (19)$$

$$\Delta G_{I \leftrightarrows U}(T) = (T_2 - T)\,(R\,T_{onset,2} ln(\frac{0.01}{0.99}))\,/\,(T_{onset,2} - T_2) \qquad (20)$$

Figure 15 shows a simulation of the model.

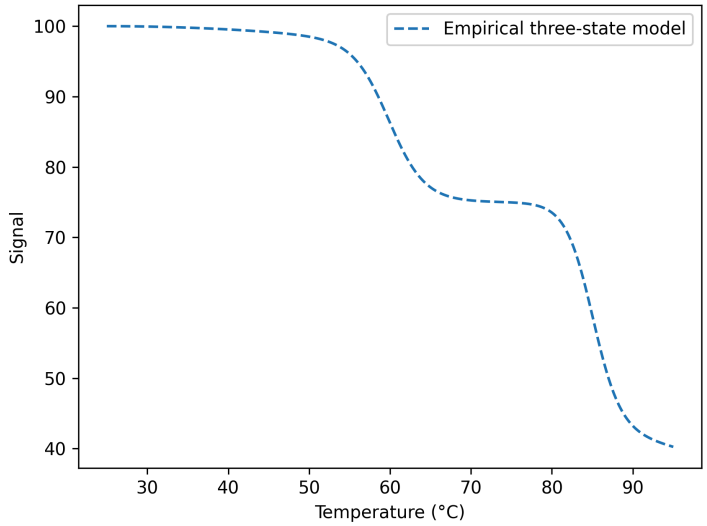

Figure 15. Simulated signal of the Empirical three-state unfolding model. The following parameters were used: $T_{onset,1} = 50.16°C$, $T_1 = 55°C$, $b_n = 100$, $b_i = 75$, $b_u = 50$, $k_n = -0.001$, $k_u = -0.001$, $q_n = -0.002$, and $q_u = -0.002$, $T_{onset,2} = 77.37°C$, $T_2 = 85°C$. The signal values are the same as in Figure 14.

Irreversible two-state[4],[5]

This model assumes that the protein only exists in the native (N) and unfolded (U) state and that the unfolding reaction is irreversible. The signal is described by the equation:

$$F(T) = x_n(q_n\Delta T^2 + k_n\Delta T + b_n) + (1 - x_n)(q_u\Delta T^2 + k_u\Delta T + b_u) \qquad (21)$$

where $x_n(T)$is the fraction of natively folded molecules as a function of temperature and can be obtained via numerical integration:

$$x_n(T) = \int_{T_{min}}^{T_{max}} \frac{-1}{v} * exp(\frac{-E_a}{R}(\frac{1}{T} - \frac{1}{T_f}) * x_n) \qquad (22)$$

[4] Mazurenko, S., Kunka, A., Beerens, K., Johnson, C. M., Damborsky, J., & Prokop, Z. (2017). Exploration of protein unfolding by modelling calorimetry data from reheating. *Scientific reports*, *7*(1), 1-14.

[5] Sanchez-Ruiz, J. M. (1992). Theoretical analysis of Lumry-Eyring models in differential scanning calorimetry. *Biophysical journal*, *61*(4), 921-935.

where T_{max} and T_{min} are the start and end temperatures of the measurement, v is the scan rate in degrees/minutes, E_a is the activation energy of unfolding, T_f is the temperature where the reaction rate constant of unfolding equals 1. For simplicity, x_n is assumed to be 1 at T_{min}.

Figure 16 shows a simulation of the model.

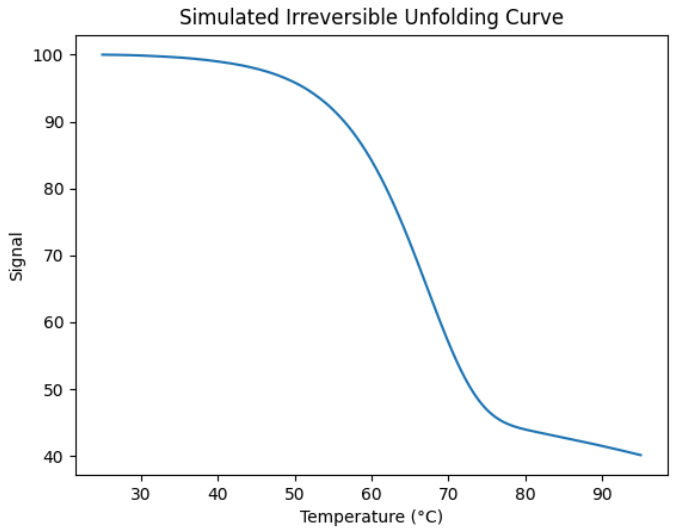

Figure 16. Simulated signal of the Irreversible three-state unfolding model. The following parameters were used: $T_f = 80°C$, $E_a = 35\ kcal/mol$, $b_n = 100$, $b_u = 50$, $k_n = -0.001$, $k_u = -0.001$, $q_n = -0.002$, and $q_u = -0.002$.

3.3. Temperature range for baseline estimation

The initial values of the parameters q_n, k_n, b_n, q_u, k_u, b_u are estimated by fitting the first or last n-degrees (selected by the user). If 'quadratic' baselines are used, a polynomial of second degree is used. If 'linear' baselines are used, the equation of a line is used. If no dependence on temperature is modelled, the average value is used.

3.4. Curve fitting

Each curve is fitted individually using the Levenberg Marquardt (damped least-squares) algorithm as implemented in the curve_fit function from the Scipy package.

3.5. Fitting errors

The standard deviation of all fitted parameters is computed using the square root of diagonal values from the fit parameter covariance matrix reported by scipy.curve_fit function. These values are an underestimation of the true errors.

3.6. Fitting residuals

The residuals of the fitting are normalized by dividing them by the standard error.

4. Analyse

4.1. Protein stability score

After fitting a model, a protein stability score is provided which can be used to sort the conditions.

Model	Score
Equilibrium two-state	$\Delta G\ of\ unfolding\ at\ 298.15\ K$
Empirical two-state	$distance((T_m; T_{Onset}), (0; 0))$
Equilibrium three-state	$\Delta G\ of\ unfolding\ at\ 298.15\ K$ $(\Delta G\ of\ reaction\ N \leftrightarrows I\ +\ \Delta G\ of\ reaction\ I \leftrightarrows U)$
Empirical two-state	$distance((T_1; T_{onset,1}), (0; 0))\ +$ $distance((T_2; T_{onset,2}), (0; 0))$
Irreversible two-state	$-\ ln(k_{irrev}(298.15\ K))$

Contact details

For further assistance, please contact us:

📧 spc@embl-hamburg.de
📌EMBL (c/o DESY), Notkestrasse 85, Build. 25a, 22607 Hamburg, Germany

References

Eftink, Maurice R. "The use of fluorescence methods to monitor unfolding transitions in proteins." Biophysical journal 66.2 (1994): 482-501.

Bedouelle, Hugues. "Principles and equations for measuring and interpreting protein stability: From monomer to tetramer." Biochimie 121 (2016): 29-37.

Žoldák, Gabriel, Daniel Jancura, and Erik Sedlák. "The fluorescence intensities ratio is not a reliable parameter for evaluation of protein unfolding transitions." Protein Science 26.6 (2017): 1236-1239.

Kotov, V., Mlynek, G., Vesper, O., Pletzer, M., Wald, J., Teixeira‐Duarte, C. M., ... & Marlovits, T. C. (2021). In‐depth interrogation of protein thermal unfolding data with MoltenProt. Protein Science, 30(1), 201-217.

Burastero, O., Niebling, S., Defelipe, L. A., Günther, C., Struve, A., & Garcia Alai, M. M. (2021). eSPC: an online data-analysis platform for molecular biophysics. *Biological Crystallography*, *77*(10), 1241-1250.

Appendix

Derivation of the Empirical Two-state v1.1 model

We start by assuming that ΔH and ΔS are constant over the temperature range of interest. ΔG at the temperature of melting T_m and onset temperature T_{onset} is defined as:

$$\Delta G_{T_m} = \Delta H - T_m \Delta S \qquad (23)$$

$$\Delta G_{T_{onset}} = \Delta H - T_{onset} \Delta S \qquad (24)$$

Using Equations (23) and (24), we have that

$$\Delta S = \frac{\Delta G_{T_m} - \Delta G_{T_{onset}}}{T_{onset} - T_m} \qquad (25)$$

ΔG at any given temperature T is calculated as:

$$\Delta G(T) = \Delta H - T\Delta S \qquad (26)$$

Substituting ΔH,

$$\Delta G(T) = \Delta G_{T_m} + T_m \Delta S - T\Delta S \quad (27)$$

Given that ΔG_{T_m} equals zero,

$$\Delta G(T) = (T_m - T)\Delta S \quad (28)$$

Substituting ΔS,

$$\Delta G(T) = (T_m - T)\frac{-\Delta G_{T_{onset}}}{T_{onset} - T_m} \quad (29)$$

Finally, $\Delta G_{T_{onset}}$ corresponds to the temperature where 1 % of the protein is unfolded and 99 % is folded:

$$\Delta G_{T_{onset}} = -RT_{onset} ln(\frac{0.01}{0.99}) \quad (30)$$

Derivation of the Empirical Three-state v1.1 model

We start by assuming that $\Delta H_{N \leftrightarrows I}$, $\Delta S_{N \leftrightarrows I}$, $\Delta H_{I \leftrightarrows U}$ and $\Delta S_{I \leftrightarrows U}$ are constant over the temperature range of interest. $\Delta G_{N \leftrightarrows I}$ and $\Delta G_{I \leftrightarrows U}$ at the temperature of transition (T_1 and T_2) and onset temperature ($T_{onset,1}$ and $T_{onset,2}$) are defined as:

$$\Delta G_{N \leftrightarrows I, T_1} = \Delta H_{N \leftrightarrows I} - T_1 \Delta S_{N \leftrightarrows I} \quad (31)$$

$$\Delta G_{N \leftrightarrows I, T_{onset,1}} = \Delta H_{N \leftrightarrows I} - T_{onset,1} \Delta S_{N \leftrightarrows I} \quad (32)$$

$$\Delta G_{I \leftrightarrows U, T_2} = \Delta H_{I \leftrightarrows U} - T_2 \Delta S_{I \leftrightarrows U} \quad (33)$$

$$\Delta G_{I \leftrightarrows U, T_{onset,2}} = \Delta H_{I \leftrightarrows U} - T_{onset,2} \Delta S_{I \leftrightarrows U} \quad (34)$$

If we combine Equations 31 and 32, and Equations 33 and 34,

$$\Delta S_{N\leftrightarrows I} = \frac{\Delta G_{N\leftrightarrows I,T_1} - \Delta G_{N\leftrightarrows I,T_{onset,1}}}{T_{onset,1} - T_1} \quad (35)$$

$$\Delta S_{I\leftrightarrows U} = \frac{\Delta G_{I\leftrightarrows U,T_2} - \Delta G_{I\leftrightarrows U,T_{onset,2}}}{T_{onset,2} - T_2} \quad (36)$$

Then $\Delta G_{N\leftrightarrows I}$ at any given temperature T can be written as,

$$\Delta G_{N\leftrightarrows I}(T) = \Delta G_{N\leftrightarrows I,T_1} + T_1\Delta S_{N\leftrightarrows I} - T\Delta S_{N\leftrightarrows I} \quad (37)$$

and $\Delta G_{I\leftrightarrows U}$ can be expressed as,

$$\Delta G_{I\leftrightarrows U}(T) = \Delta G_{I\leftrightarrows U,T_2} + T_2\Delta S_{I\leftrightarrows U} - T\Delta S_{I\leftrightarrows U} \quad (38)$$

By definition, $\Delta G_{N\leftrightarrows I,T_1}$ and $\Delta G_{I\leftrightarrows U,T_2}$ are zero. Therefore,

$$\Delta G_{N\leftrightarrows I}(T) = (T_1 - T)\frac{-\Delta G_{N\leftrightarrows I,T_{onset,1}}}{T_{onset,1} - T_1} \quad (39)$$

and

$$\Delta G_{I\leftrightarrows U}(T) = (T_2 - T)\frac{-\Delta G_{I\leftrightarrows U,T_{onset,2}}}{T_{onset,2} - T_2} \quad (40)$$

Finally,

$$\Delta G_{N\leftrightarrows I,T_{onset,1}} = -RT_{onset,1}ln(K_{N\leftrightarrows I,T_{onset,1}}) \quad (41)$$

where

$$K_{N\leftrightarrows I,T_{onset,1}} = \frac{[N]_{T_{onset,1}}}{[I]_{T_{onset,1}}} \quad (42)$$

Assuming that $[U]$ is negligible at $T_{onset,1}$, then

$$K_{N\leftrightarrows I,T_{onset,1}} = \frac{[N]}{[I]} \quad (43)$$

$$K_{N \leftrightarrows I, T_{onset,1}} = \frac{0.01}{0.99} \quad (44)$$

$\Delta G_{I \leftrightarrows U, T_{onset,2}}$ can be derived in a similar way (assuming that $[N]$ is negligible at $T_{onset,2}$) and equals

$$\Delta G_{I \leftrightarrows U, T_{onset,2}} = -RT_{onset,2} ln(\frac{0.01}{0.99}) \quad (45)$$

To summarise, T_1 is the temperature where $\Delta G_{N \leftrightarrows I}$ equals zero, T_2 is the temperature where $\Delta G_{I \leftrightarrows U}$ equals zero, $T_{onset,1}$ is the temperature where 99 % of the protein is in the folded state, and $T_{onset,2}$ is the temperature where 99 % of the protein is in the intermediate state.

Packages

MoltenProt (online version) is possible thanks to:

R language: R Core Team (2020). R: A language and environment for statistical computing. R Foundation for Statistical Computing, Vienna, Austria. URL https://www.R-project.org/.

R package shiny: Winston Chang, Joe Cheng, JJ Allaire, Yihui Xie and Jonathan McPherson (2020). shiny: Web Application Framework for R. R package version 1.4.0.2. https://CRAN.R-project.org/package=shiny

R package viridis: Simon Garnier (2018). viridis: Default Color Maps from 'matplotlib'. R package version 0.5.1. https://CRAN.R-project.org/package=viridis

R package tidyverse: Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

R package pracma: Hans W. Borchers (2019). pracma: Practical Numerical Math Functions. R package version 2.2.9. https://CRAN.R-project.org/package=pracma

R package shinydashboard: Winston Chang and Barbara Borges Ribeiro (2018). shinydashboard: Create Dashboards with 'Shiny'. R package version 0.7.1. https://CRAN.R-project.org/package=shinydashboard

R package ggplot2: H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

R package xlsx: Adrian Dragulescu and Cole Arendt (2020). xlsx: Read, Write, Format Excel 2007 and Excel 97/2000/XP/2003 Files. R package version 0.6.3. https://CRAN.R-project.org/package=xlsx

R package reshape2: Hadley Wickham (2007). Reshaping Data with the reshape Package. Journal of Statistical Software, 21(12), 1-20. URL http://www.jstatsoft.org/v21/i12/.

R package tippy: John Coene (2018). tippy: Add Tooltips to 'R markdown' Documents or 'Shiny' Apps. R package version 0.0.1. https://CRAN.R-project.org/package=tippy

R package shinyalert: Pretty Popup Messages (Modals) in 'Shiny'. R package version 1.1. https://CRAN.R-project.org/package=shinyalert

R package plotly: C. Sievert. Interactive Web-Based Data Visualization with R, plotly, and shiny. Chapman and Hall/CRC Florida, 2020.

R package tableHTML: Theo Boutaris, Clemens Zauchner and Dana Jomar (2019). tableHTML: A Tool to Create HTML Tables. R package version 2.0.0. https://CRAN.R-project.org/package=tableHTML

R package rhandsontable: Jonathan Owen (2018). rhandsontable: Interface to the 'Handsontable.js' Library. R package version 0.3.7. https://CRAN.R-project.org/package=rhandsontable

R package remotes: Jim Hester, Gábor Csárdi, Hadley Wickham, Winston Chang, Martin Morgan and Dan Tenenbaum (2020). remotes: R Package Installation from Remote Repositories, Including 'GitHub'. R package version 2.1.1. https://CRAN.R-project.org/package=remotes

R package devtools: Hadley Wickham, Jim Hester and Winston Chang (2020). devtools: Tools to Make Developing R Packages Easier. R package version 2.3.0. https://CRAN.R-project.org/package=devtools

R package shinyjs: Dean Attali (2020). shinyjs: Easily Improve the User Experience of Your Shiny Apps in Seconds. R package version 1.1. https://CRAN.R-project.org/package=shinyjs

R package data.table: Matt Dowle and Arun Srinivasan (2019). data.table: Extension of data.frame. R package version 1.12.8. https://CRAN.R-project.org/package=data.table

R package reticulate: Kevin Ushey, JJ Allaire and Yuan Tang (2020). reticulate: Interface to 'Python'. R package version 1.16. https://CRAN.R-project.org/package=reticulate

R package shinycssloaders: Andras Sali and Dean Attali (2020). shinycssloaders: Add CSS Loading Animations to 'shiny' Outputs. R package version 0.3. https://CRAN.R-project.org/package=shinycssloaders

Baptiste Auguie (2019). egg: Extensions for 'ggplot2': Custom Geom, Custom Themes, Plot Alignment, Labelled Panels, Symmetric Scales, and Fixed Panel Size. R package version 0.4.5. https://CRAN.R-project.org/package=egg

Python3.7 language: Van Rossum, G., & Drake, F. L. (2009). Python 3 Reference Manual. Scotts Valley, CA: CreateSpace.

Python package numpy: Travis E, Oliphant. A guide to NumPy, USA: Trelgol Publishing, (2006). Stéfan van der Walt, S. Chris Colbert, and Gaël Varoquaux. The NumPy Array: A Structure for Efficient Numerical Computation, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

Python package pandas: Wes McKinney. Data Structures for Statistical Computing in Python, Proceedings of the 9th Python in Science Conference, 51-56 (2010)

Python package scipy: Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

Python package xlrd: https://xlrd.readthedocs.io/en/latest/index.html

Python package natsort: https://natsort.readthedocs.io/en/master/