# Experimental data and program files

for the paper

**"A possible universal intermediate identified during the folding of a metamorphic protein" by C Correa et al. (in preparation)**

## Overview

The program files cover three main tasks:

1. Postprocessing of experiment files: Extract information about rips and zips to Matlab tables.
2. Fit mathematical models to the extracted data, resulting in estimated model parameters and their confidence intervals.
3. Create figures and tables shown in the paper and supplementary documents.

Matlab code for the three tasks is organised in three folders:

Task 1: Top7paper/Experiment\_postprocessing

Task2: Top7paper

Task 3: Top7paper/Figures\_and\_tables

All programs are written in Matlab. In addition to basic Matlab the following toolboxes are used:

|  |  |
| --- | --- |
| Signal Processing toolbox | Postprocessing |
| Optimization toolbox  Statistics and Machine Learning toolbox | Fitting models |

## Experiment files

The Optical Tweezers instrument records a set of data at regular time intervals. The recording frequency may vary between 10 and 4000 records per second but is normally 200 or 400 records per second. The record files have one column per variable recorded. The relevant variables for this study are

CycleCount: Number of controller cycles since the experiment started. The cycle frequency is 4000 cycles per second, so the recording time is CycleCount/4000

Y\_force: Pulling force (pN)

A\_dist-Y, B\_dist-Y: Position of the trapped pulling bead (trap position, nm). We use the mean of the two values.

Status: Used to code the heating power in the chamber. Can be translated to temperature difference between chamber and bath.

The uncompressed experiment files take up about 3GB of storage. We include only a few, in folder Experiment\_postprocessing/SampleExperiments, to enable testing of the postprocessing functions. The full set is available from the authors.

## Postprocessing

### Main functions

function [Trip,Tzip] = analyse\_experiment(file)

Input is a file generated by the optical tweezers instrument. Output is two Matlab tables with 16 columns and one row for each identified rip or zip.

|  |  |
| --- | --- |
| **Column** | **Content** |
| Filename | Name of experiment file |
| Time | Time of rip (seconds since experiment start) |
| Deltax | Δx = Change in trap position at force = Force (nm) |
| Force | Pulling force at start of rip/zip (pN) |
| Temperature | At protein (may be higher than in bath) °C |
| Forceshift | Shift in pulling force at rip/zip (pN) |
| Trapx | Position of trapped bead (nm) |
| Fdot | Rate of change of pulling force (pN/s) |
| Slope\_b | d(pulling force=/d(Trapx) before rip/zip |
| Slope\_a | d(pulling force=/d(Trapx) after rip/zip |
| Pullingspeed | Speed of trappd bead (nm/s) |
| Topforce | Maximum force in trace pN) |
| Noise | Measure of rapid change in pulling force (pN/s) |
| Cycleno | Index of pull/relax trace |
| Work | Crooks work |
| Timestep | Length of recording time step (s) |

[Tp,Tr] = analyse\_many(files);

This function runs analyse\_experiment for all files in the string array files and collects the output Trip tables into TP (P for pulling trace)and the Tzip tables into TR (R for relaxing trace). This somewhat confusing naming practice has historical reasons and is likely to change in the future!

For the paper, we have quantified nearly 22000 rips and zips in 239 experiment files. This takes about three minutes on a standard laptop computertic;Tout. The data for the rips and zips are stored tables TP and TR in tables.mat.

## Parameters

One challenge is to strike the best balance between maximising the rip/zip candidates found whine minimising the number of spurious rips/zips. The parameter file params.m contains a number of tuning parameters that were found by trial and error. Here the user can also specify if he wants to look for more than one rip per trace, and whether to include rips in the relaxing trace (late rips). For this paper, the search is limited to a single rip in the pulling trace.

## Fitting models

Programs for analysing the data files from the experiments are written in Matlab. Typically, a batch of experiment files are analysed by function analyse\_many, which returns Matlab tables TP (rips from the pulling trace) and TR (zips from the relaxing trace). These tables contain sixteen recorded values per trace, most notably the time (seconds from experiment start), rip or zip force (pN), the rip/zip length Δx (nm), the temperature (°C) and the pulling speed (nm/s).

The tables for all experiments are stored in the file Tables.mat and are used to generate most figures and tables in the paper and supplement.

Each of the tables and figures are generated by a separate m-file named after the figure or table.

## Data files

The paper is based on 239 time series of experimental recordings. The columns used are:

CycleCount: Time measure relative to experiment start. 400 are: counts per second

Y\_force: Pulling force (pN)

A\_dist-Y, B\_dist-Y: Position of pulling bead (nm) (trap position). We use the mean of the two.

Status: Used to code the temperature in the chamber.

The uncompressed data files take up about 3GB of storage. We include only a few, to be used for those who want to test analyse\_experiment.m or analyse\_many.m. Expand SampleExperiments.zip to try this.

## Automatic analysis of experimental raw data

The experiment text files are read by read.experiment\_file.m which returns t (time, seconds), f (force, pN). x (trap position, nm) and temperature T (°C).

Analyse\_file.m uses this information to find individual stretching and relaxing traces. Each trace is analysed by analyse\_trace.m which looks for changes in the slope of the force trace to identify candidates for unfolding (rip) or refolding (zip) events. The candidates must fulfil a set of requirements to qualify, and the best of the qualified candidates is chosen. analyse\_trace returns a Matlab struct with the rip/zip properties. analyse\_file uses this to create one row per event in the output tables Tu and Tr. Several parameters have been tweaked in order to maximise detection efficiency, with a higher priority given to avoiding spurious events. The same parameters were used for all experiments. The parameters are specified in params.m.

Specify plotting to see a graph of the force time series with rips and zips marked. Example:

analyse\_experiment("fA.txt",1);

This file is from an experiment with three different pulling speeds. Use the zoom and pan tools that appear at the upper right to see more detail.

[TP,TR] = analyse\_many(files) calls analyse\_experiment repeatedly for all file names in the string array “files” and returns Matlab tables with data for all rips and zips. Such tables are stored in Tables.mat.

## Clusters

The categorisation of events as belonging to Cluster 1, Cluster2 or as outliers was done by function no\_outliers.m, which uses the Matlab clustering algorithm dbscan to assign points in a scatter plot to groups based on distances between neighbouring points. The parameters were chosen by trial and error until the results seemed reasonable.

## Grouping experiments and fitting models to data

The experiments were grouped into 14 unfolding and 8 refolding groups based on temperature and pulling speed. The Bell and Dudko models calculate force probability density functions. The Crooks model uses histograms of unfolding and refolding work instead. For each group and cluster the models were fitted to the data using Matlab’s lsqcurvefit function. The grouping and fitting were done in the function run\_fit\_dual.m. The resulting parameters were reported in the output table Tout. This table was used to generate most tables and graphs.

## Matlab details

The following Matlab toolboxes are required:

|  |  |
| --- | --- |
| Toolbox | Used by |
| Optimization | Fit\_Bell\_unfold  Fit\_Bell\_refold  Fit\_Dudko\_unfold |
| Signal Processing | Anayse\_experiment  Analyse\_trace |
| Statistics and Machine Learning | Fit\_Crooks  Fig\_S7  Fit\_Bell\_unfold  Fit\_Bell\_refold  Fit\_dudko\_unfold  Run\_fit |

All files were tested in Matlab R2024a. Most were also run in Matlab R2023a.

## Acknowledgement

movingslope.m was created by John D’Errico and was copied from the Matlab Central File Exchange.

Ref.: John D'Errico (2024). Movingslope (https://www.mathworks.com/matlabcentral/fileexchange/16997-movingslope), MATLAB Central File Exchange. Retrieved April 19, 2021.