Stepfinder Manual

Jacob Kerssemakers

March 2018

Contents

Introduction

This manual describes content and workings of a 'Step finder' code. This code is an improved version of the original algorithm described in: '**Assembly dynamics of microtubules at molecular resolution',** Jacob W. J. Kerssemakers, E. Laura Munteanu, Liedewij Laan, Tim L. Noetzel, Marcel E. Janson and Marileen Dogterom, Nature. 2006 Aug 10;442(7103):709-12)

Notes on Development version

This stepfinder code pack was shared as early version. While the essential code works, some ''advanced buttons'' may still not function. The authors cannot guarantee bug-free code but code was tested on many datasets. Please refer to these publications when using the code, pending publication of the finalized code:

1. "Microbial Warfare: Illuminating CRISPR adaptive immunity using single-molecule fluorescence Chapter 6; Author Luuk Loeff, (TU Delft BN/Chirlmin Joo Lab) ; Thesis; Date 2017-10-06 doi:10.4233/uuid:08c08aec-53f0-4419-ba97-11fbb5a3dd49

2. Assembly dynamics of microtubules at molecular resolution', Jacob W. J. Kerssemakers, E. Laura Munteanu, Liedewij Laan, Tim L. Noetzel, Marcel E. Janson and Marileen Dogterom, Nature. 2006 Aug 10;442(7103):709-12)

March 2018
Jacob Kerssemakers
Luuk Loeff

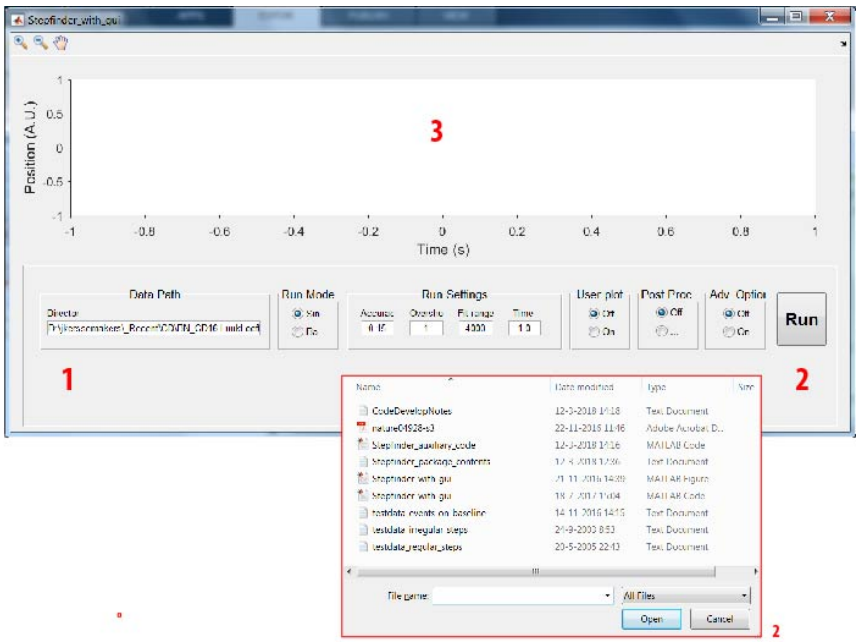
Package Contents

- Stepfinder_with_gui.m/ .fig: MATLAB code and gui of the stepfinder program
- testdata_events_on_baseline.txt; testdata_regular_steps.txt; testdata_irregular_steps.txt": various types of testdata showing the workings and input format of the code
- Stepfinder_AuxiliaryCode: zip with code containing extra functions
- nature04928-s3.pdf: pdf file with theory as described in the original paper:
- Kerssemakers et al, Nature. 2006 Aug 10;442(7103):709-12)

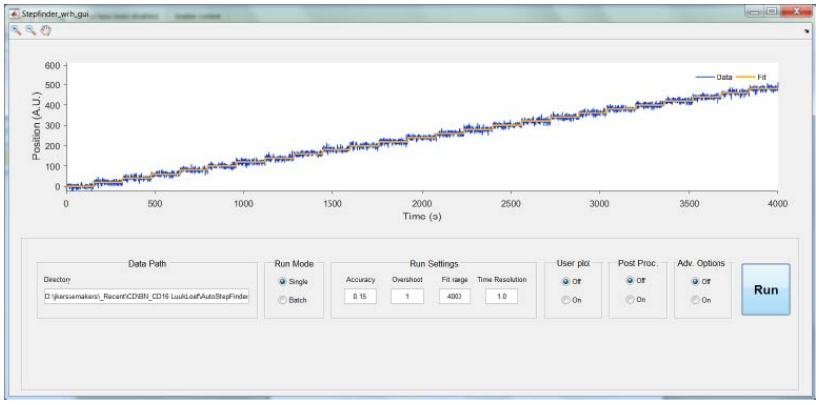
Step-by-Step-Use of the Stepfinder

Basic Use

Open the 'Stepfinder_with_gui' in MATLAB and run it. A User interface will appear with the main run settings set to their default values.



1. Here, set the path towards input data (default is the location of the code and testfiles)
2. Press 'Run' ; select a data file. For the demo, choose ''testdata_regular_steps'' . A representation of the data and the stepfit will appear here.



Data In/Out

Input Data should be 2-column (Time, Signal). Note: the time units will NOT be used for processing or output; These are user set by the button 'time resolution'.

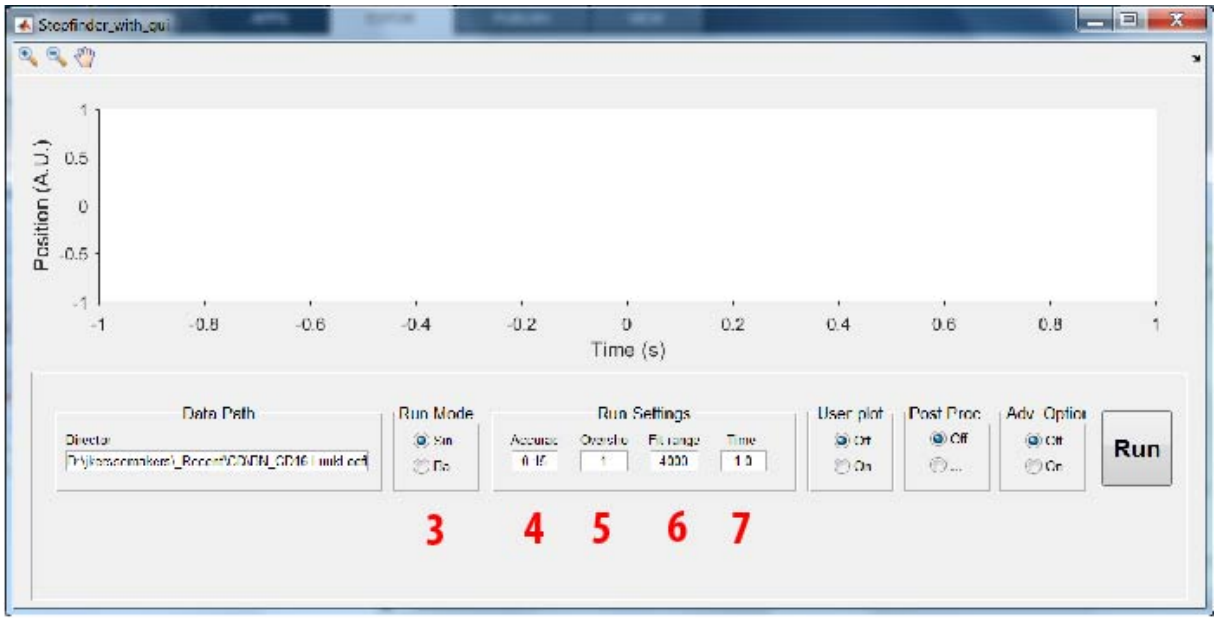
Otput data is auto-saved in a local directory; here "testdata_regular_steps_Fitting_Result". It contains:

1) testdata_regular_steps_fits: columns original time, original data, step fit as shown in the GUI screen

2) testdata_regular_steps_properties: columns with properties of individual steps.
   a. IndexStep: the data index of the step (defined as the rightmost point of the preceding plateau)
   b. TimeStep: the time associated with this point, taken as (number of datapoints times time-resolution)
   c. LevelBefore: the average of the plateau preceding the step
   d. LevelAfter: the average of the plateau following the step
   e. StepSize: the stepsize, defined as the value difference between the adjacent plateaus
   f. DwellTimeStepBefore: time interval covered by the preceding plateau
   g. DwellTimeStepAfter: time interval covered by the following plateau
   h. StepError: the error associated with the step size

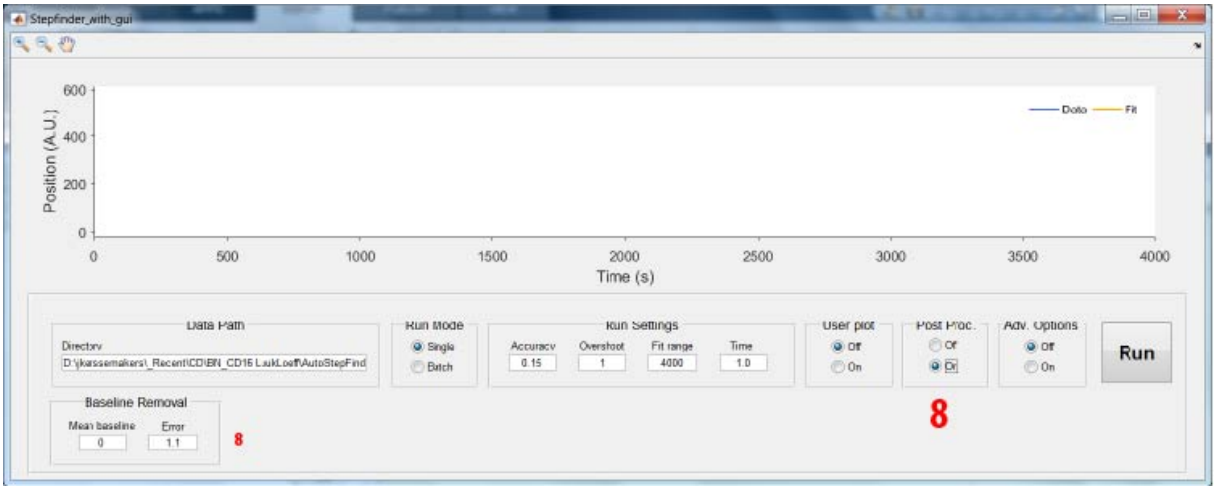
3) testdata_regular_steps_SCurve: columns with the 'S-curves '' produced during the fitting procedure (see reference text).

Run settings

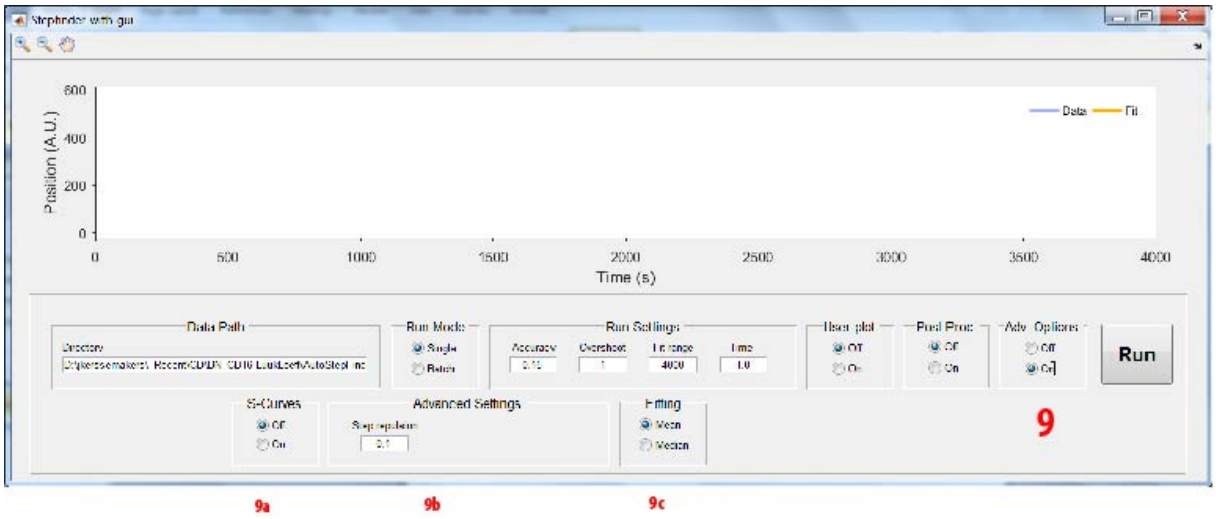


3. When setting this button to 'batch mode' , the user can select a directory containing multiple text files, which the code will analyze in batch style.
4. ''Accuracy'' determines how far the code will fit steps into the noise by setting the ''S-value' limits; see reference text for a more detailed explanation
5. ''Overshoot'' : forcing a factor …more steps than the fitting algorithm itself found as the optimal number. This setting normally be left at 1, unless the user has other criteria for determining the optimal step number (such a known, fixed number of steps in the data).
6. "Fit range" : this sets the maximum number of steps to be fit; use it for large datasets with limited step numbers, to save computing time.
7. "Time'' this sets the time resolution, i.e. the time interval between each data point. This will be used for time data in the output, the original time axis is ignored.

Advanced options



8. Pressing “Post Proc.” Shows a tab for removing a baseline from the fit in post-processing



9. Pressing ‘Advanced ‘ options shwos tabs for advanced use. It is assumed the user has an understanding of the workings of the algorithm.
    a. Show the ‘S-curves’ after fitting.
    b. ‘Step repulsion’ puts a limit on very short plateaus during fitting. Normally, it should not be changed.
    c. Optionally, one can determine plateaus for the final fit using the median of plateaus, not the average. The fit procedure itself always uses averages.

Stepfinder resolution test

The function ‘’Step_Extractor_InjectionTest’’ processes the results of a 'step injection'run. This is a repeated stepfinding run where artificial steps of varying size are injected
in the original data at regular distances. The degree to which these steps
are found back is a measure for the detection limite of the stepfinder,
specifically for the used dataset. Prior to step analysis, a 'square wave' with varying amplitude per step was added to the original data trace. The amplitudes per block

are chosen at random, in a range in accordance to the steps sizes of interest in the original data. The locations are chosen at fixed intervals for easy retrieval.
a small info file is saved containing the step information for later use.
since we use a square wave, the net growth or shrinkage of the original signal is preserved.Further, the number of injected steps is kept low compared to the typical step number returned on analyzing the original data. Thus, injected steps only rarely occur per trace. To build statistics, one original trace is combined with many different square waves. Next, step analysis is done as usual.
Here, the injected steps are collected separately if, and only if, the exact locations match those originally placed. The stepfinder never updates step locations during iterations; Further, accidental non-injected steps on these expected locations turn out to be very rare, so post-selection is easy

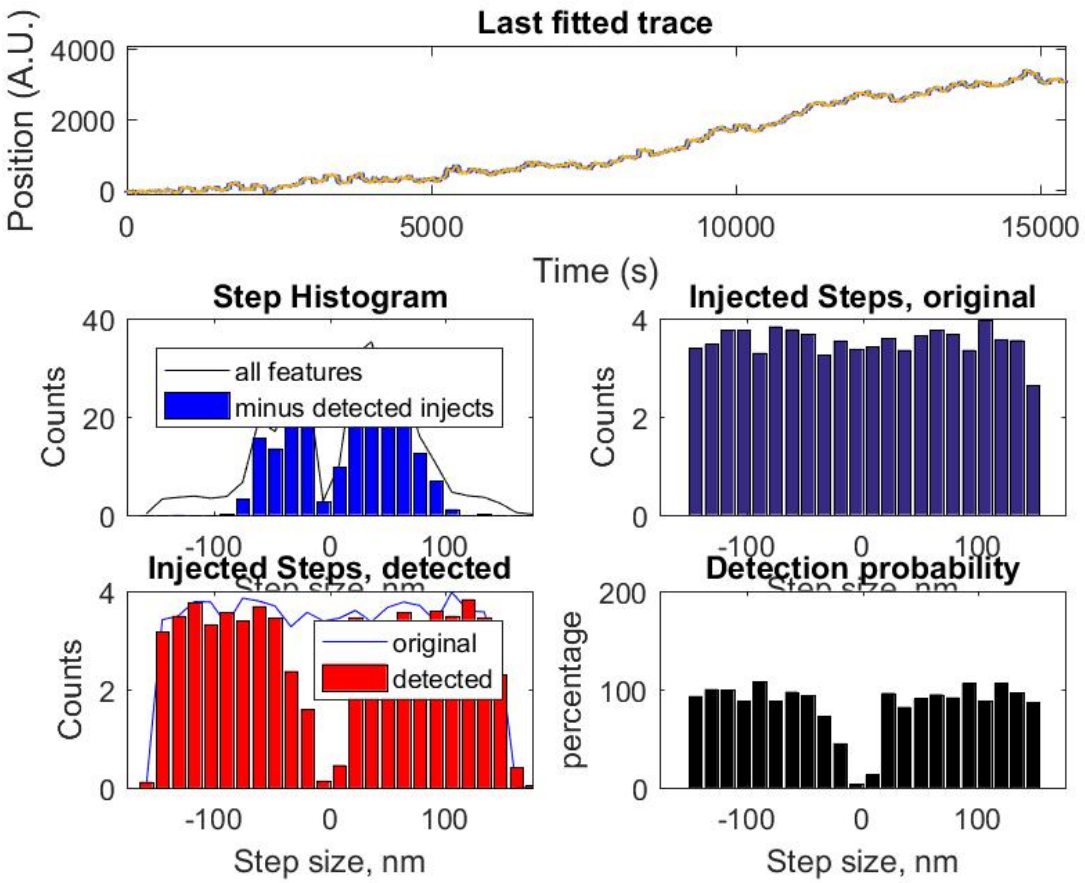
Approach:

1) run the stepfinder to get an idea of the step size range of the original data. For demo purposes, run it on 'testdata_simulatedGaussian250_steps_from-70to100'
2) go to the section starting at line 116 in the stepfinder code: and set these values:

initval.InjectionRepeatMode=100;
(do not forget to set it back to 1 after testing!)
injectprops.minstepsize=0; adapt to range original steps (demo:0)
injectprops.maxstepsize=50; adapt to range original steps (demo:50)
injectprops.L_quarterblock=299; keep larger than the original dwell times, as not to change the original data too much (demo:299)

3 run the stepfinder again on the demo data. It will now repeat 100 times a step fit run, now with injected steps included. Results are saved as separate 'repeats'.
properties of these injected steps are saved in a directory 'StepInjectionInfo'.
4 Run this program: Step_Extractor_InjectionTest (you may check that the paths are set right)

Input: step finder output files rom ''injection run''
Output: histogram summary; excel files (optional)



References:

[2] Assembly dynamics of microtubules at molecular resolution.
Nature. 2006 Aug 10;442(7103):709-12. Epub 2006 Jun 25.
Kerssemakers JW1, Munteanu EL, Laan L, Noetzel TL, Janson ME, Dogterom M.
[3] Loeff, Kerssemakers et al 2017

Auxiliary code

Stepfinder_auxiliary_code.m contains extra functions for custom Matlab use. It is assumed the user can edit in Matalab and has a good understanding of the workings of the algorithm.

StepfinderCoreCopy

This function contains the ‘’core part’’ of the stepfinder code. It can be run separately in single-run mode and in demo mode using the included ‘BottomUpTraceBuilder’ function, that will produce a trace containing nested steps at widely different sizes.

Simulated step trace code

For getting used to the stepfinder, it is useful to run it on artificial traces. Below a small set of handy code to do so

StepMaker_GeneralDistribution

This function allows easy making of simulated step traces, for testing the stepfinder. Traces are made by sampling from a step distribution

StepMaker_NestedStepsBuilder

StepMaker_VinexSkyline

Step_Extractor

Step_Extractor

This function provides a general stub to process data from stepfinder output, for single files or batch runs.

Step_Extractor_InjectionTest

This function specifically processes the results of a stepfinder resolution test. See the chapter 'Stepfinder resolution test' for details

References

1. "Microbial Warfare: Illuminating CRISPR adaptive immunity using single-molecule fluorescence Chapter 6; Author Luuk Loeff, (TU Delft BN/Chirlmin Joo Lab) ; Thesis; Date 2017-10-06 doi:10.4233/uuid:08c08aec-53f0-4419-ba97-11fbb5a3dd49

2. Assembly dynamics of microtubules at molecular resolution', Jacob W. J. Kerssemakers, E. Laura Munteanu, Liedewij Laan, Tim L. Noetzel, Marcel E. Janson and Marileen Dogterom, Nature. 2006 Aug 10;442(7103):709-12)

3. Real-time detection of condensin-driven DNA compaction reveals a multistep binding mechanism Authors, Jorine Mirjam Eeftens, Shveta Bisht, Jacob Kerssemakers, Christian Haering, Cees Dekker Publication date 2017/1/1, Journal bioRxiv