CHAPTER

Using STADIA to quantify dynamic instability in microtubules

7

Riya J. Patel[a,b,c], Kristopher S. Murray[a], Peter O. Martin[a,d], Michael Sinclair[a,e], Jared P. Scripture[a], Holly V. Goodson[a,*], Shant M. Mahserejian[f,*]

[a]*Department of Chemistry and Biochemistry, University of Notre Dame, Notre Dame, IN, United States*

[b]*Penn High School, Mishawaka, IN, United States*

[c]*Indiana University, Bloomington, IN, United States*

[d]*Trinity School at Greenlawn, South Bend, IN, United States*

[e]*Kalamazoo Area Mathematics & Science Center, Kalamazoo, MI, United States*

[f]*Pacific Northwest National Laboratory, Richland, WA, United States*

**Corresponding authors: e-mail address: hgoodson@nd.edu*

Chapter outline

Methods in Cell Biology, Volume 158, ISSN 0091-679X, https://doi.org/10.1016/bs.mcb.2020.03.002

Abstract

Quantification of microtubule (MT) dynamic instability (DI) is essential to mechanistic dissection of MT assembly and the activities of MT binding proteins. Typical methods for quantifying MT dynamics assume that MT behavior consists of growth and shortening phases, with instantaneous transitions (rescues and catastrophes) in between. However, examination of DI data at high temporal and spatial resolution reveals the presence of ambiguous behaviors that cannot easily fit into these categories. Failure to objectively recognize and quantify these behaviors could reduce the reproducibility of DI data and impact attempts to dissect mechanisms. To address these problems, we recently developed STADIA (Statistical Tool for Automated Dynamic Instability Analysis), a MT analysis software package that uses length-history data as input and is (presently) implemented in MATLAB. STADIA uses machine learning methods to objectively analyze and quantify macro-level DI behaviors exhibited by MTs, including variable rates of growth and shortening and a newly quantified DI phase: stutter. Here we overview the process of using STADIA to quantify MT dynamics and provide a set of concrete protocols for using STADIA to process and analyze MT length history data.

1 Introduction

Microtubules (MTs) are cytoskeletal biopolymers that are characterized by a behavior known as dynamic instability (DI) (Desai & Mitchison, 1997; Mitchison & Kirschner, 1984). In typical methods for measuring MT DI, researchers quantify four "DI parameters": the rates of growth and shortening (V_g and V_s) and the frequencies of catastrophe and rescue (F_{cat} and F_{res}) (e.g., Walker et al., 1988). An often-employed approach is to capture MT dynamics via kymographs (see, e.g., Lawrence, Arpag, Norris, & Zanic, 2018); the rates of growth and depolymerization and the frequencies of transitions are then determined by fitting lines (perhaps by hand) to the kymographs. Implicit in this approach is the assumption that MT dynamics are biphasic (i.e., MTs are either growing or shortening), with instantaneous transitions between phases. However, there are two problems with this view. First, recent experiments and simulations have indicated that these assumptions may not be valid; periods of sustained behaviors that do not resemble classically recognized growth or shortening can be observed in data acquired at high temporal resolution (Fig. 1) (Duellberg, Cade, Holmes, & Surrey, 2016; Duellberg, Cade, & Surrey, 2016; Maurer et al., 2014; Rickman et al. 2017). In addition, the subjective nature of typical MT dynamics quantification can cause problems with comparability between

experiments and reproducibility across labs. These considerations led us to develop an analysis software STADIA (Statistical Tool for Automated Dynamic Instability Analysis), which was designed to quantify and characterize this DI behavior in a more objective, precise, and reproducible way.

The goal of this article is to provide an overview of the STADIA program and how to use it. The software code, example input files, and an accompanying tutorial ("Modeling microtubule mechanisms of dynamic instability", available as either a PowerPoint™ program or a PDF) are available in a GitHub repository (GoodsonLab GitHub STADIA Repository 2019). The tutorial provides examples of how to run STADIA, including initial conditions and parameters that need to be input for the software and the corresponding results as graphs and figures. The example data analyzed in this paper and tutorial were obtained from our detailed model of MT dynamics (Margolin et al. 2012). For an illustration of how STADIA can be used to analyze *in vitro* MT dynamics, please see (Mahserejian et al., 2019); this paper and its supplementary materials also contain additional information regarding the theory behind the development of STADIA. At the time of this paper's release, STADIA was developed in MATLAB; thus a STADIA user should be reasonably familiar with how MATLAB operates, though it is not necessary to know how to write code.

2 Methods

STADIA employs statistical and machine learning methods to impartially characterize behavior and calculate quantities associated with DI. STADIA accomplishes unbiased characterization and quantification of MT dynamics by creating an iteratively improved approximation of MT length history data that detects moments of significant changes in MT behavior. This improved approximation enables STADIA to perform unbiased classification of macro-level dynamic changes, because behaviors intermediate to growth and shortening can be considered. Previous approaches limit analysis by assuming that only two phases exist in MT dynamics, whereas STADIA allows for more complex behaviors, including additional phases and phase transitions.

STADIA works in three main stages: *segmentation*, *classification*, and *phase and transition analysis* (Mahserejian et al., 2019). The tasks for these stages are carried out after an initialization process, during which user-defined parameters are specified and the MT length history data are loaded from an input file.

- In segmentation, a piecewise linear approximation of the data is created, with the length history data segmented into regions separated by endpoints that mark significant changes in microtubule dynamics (Fig. 1A).
- In the classification stage, STADIA separates the segments into clusters based on their segment characteristics (time duration, length change, and slope) using unsupervised machine learning; cluster metrics are then used to further define DI phase classes.

- Finally, in the phase and transition analysis stage, STADIA analyzes segments in each DI phase, and uses them to calculate classical DI parameters (e.g., frequency of rescue, velocity of growth, etc.), as well as new parameters for transitions involving a DI phase intermediate to growth and shortening called "stutter" (Mahserejian et al., 2019).

STADIA builds upon traditional DI analysis methods since it not only calculates parameters classically used to characterize DI behavior, but now also considers additional behaviors previously ignored when restricting DI behavior to only growth and shortening.

The remainder of this section details the methodology associated with each of the three STADIA stages and discusses the user-defined parameters that control STADIA.

2.1 STADIA Setup

Users need a copy of MATLAB and the files that compose the STADIA program. The user is responsible for providing the user input parameters in the Input_and_Run.m script (suggestions for determining appropriate values can be found in Section 2.3).

Seven files should be in the current folder tab:

1. Length history data input file (either .dat, .txt, or .csv)
2. Input_and_Run.m
3. Loop_Thru_Inputs.m
4. PieceWiseLinearApproximation.m
5. DIphaseClassification.m
6. ExtractDIparameters.m
7. Findjobj.m

While the length history file can be located elsewhere (with the path name specified in the Input_and_Run.m file), it is simpler if the length history file is in the same directory as the other six STADIA .m files listed above. In brief, here is a description of each file:

- *The length history data input* file contains the microtubule length history data; one column of data should correspond to time step values, and the remaining column(s) to MT lengths (the unit for length must be in subunits). Input data can represent length history as a single long time series format in a single file, multiple columns for multiple MTs in a single file, or multiple files of one of these formats. STADIA can analyze data that has been generated from any source, including *in vitro* or *in silico* MTs, provided the input data collectively belongs to the same experimental conditions. Acceptable file formats include .dat, .txt, and .csv.
- *Input_and_Run.m* is the only STADIA script that a user will need to interact with when using the program. User-defined terms can be edited in this file, including the filename corresponding to the length history data to

be analyzed, the threshold and tolerance levels to conduct the desired analysis, and options for results that are to be displayed and saved. Details for user-defined terms and how they relate to corresponding STADIA stages are described in Sections 2.2 and 2.3.

- *Loop_Thru_Inputs.m* is called by Input_and_Run.m and initializes analysis by reading in the MT length history data and calling the remaining modularized scripts to conduct the procedures in STADIA.
- *PieceWiseLinearApproximation.m* creates a continuous piecewise linear approximation of the microtubule length history data as part of the segmentation stage.
- *DIphaseClassification.m* conducts the classification stage procedures using the segments identified in the previous stage. This script relies on *k*-means clustering, an unsupervised machine learning approach that requires a priori knowledge of the *k*-values (number of clusters to separate the data). Additionally, this script handles the options involved with running STADIA in Diagnostic Mode, which provides useful information to the user for selecting the optimal *k*-values to be used in the automated mode of the classification stage.
- *ExtractDIparameters.m* is responsible for analysis of the data within each phase class associated with the phase and transition analysis stage of STADIA.
- *findjobj.m* is required for running STADIA to optimize the aesthetics of the display of the figures and tables. The functions in this script are not provided by MATLAB by default.

Once the script Input_and_Run.m is opened and the desired parameters are entered into the script, simply running Input_and_Run.m will begin analysis of the input file with the chosen parameters. For a quick overview of how to run STADIA and interpret its output, see the tutorial provided in the corresponding GitHub repository. More in-depth discussion is provided below.

2.2 Initialization

In the initialization process, STADIA loads the input data as well as the user-defined parameters necessary for analysis. In this section, we describe the terms and options available to the user in the Input_and_Run.m file that contributes to initialization steps associated with reading in the input file.

The parameters that must be set by the user to conduct initialization include:

- SKIP_FILE_READ: in the case where a user is conducting several run-throughs of the same data, this parameter gives the option to skip reading the data file, thus speeding up the time it takes to complete one run of STADIA. The possible choices for this term include:
  - SKIP_FILE_READ = 1; this will skip reading the input file and will keep any previously created variables in the workspace.
  - SKIP_FILE_READ = 0; will clear all previously created variables, close figures, and read in the data from the specified input file. This is the required choice if STADIA is being run on a data file for the first time.

- FILE_NAME_INPUT: defines the input filename(s) containing the length history data to be analyzed by STADIA. The length history data can come from multiple files, as long as the data are organized in a consistent manner for all files used (i.e., the time column and MT length data column numbers should be the same for all files). The allowable data file types include those with a .dat, .txt, or .csv extension. The file names should be listed using quotes (' ') and brackets ({ }). Here are some examples of indicating the input file names:
  - FILE_NAME_INPUT = {'MyLengthHistory.dat'}; indicates that a single input file should be used.
  - FILE_NAME_INPUT = {'length_history_1.txt', 'length_history_2.txt', 'length_history_3.txt', 'length_history_4.txt'}; indicates that multiple (four) input files should be used.
- FIRST_DATA_ROW: indicates the first row of the input file with MT length history data that will be analyzed. In cases where the input file has column headers with strings, this can ensure STADIA does not try to read them as MT data. Common choices for this parameter include 1 (if there is no header) or 2 (if there is a header).
- MT_LENGTH_COLUMN_INDICES: indicates which column(s) contain the MT length values. Note that it is possible for multiple columns to hold this data, provided that for each row of data, MT lengths correspond to the same time value. For MT lengths contained in a single column, define this parameter using a single integer value. For MT lengths in multiple columns, define this parameter using an array of integers. Also note that for instances where multiple data files are being used, the length column numbers must be the same for all data files. Some examples include the following:
  - MT_LENGTH_COLUMN_INDICES = 2; where the second column contains the MT length data.
  - MT_LENGTH_COLUMN_INDICES = [2:5, 8, 11:18]; where columns 2–5, 8, and 11–18 contain the MT length data to be analyzed.
- TIME_COLUMN: indicates which column in the length history data file contains the time values for the data. Only a single integer value is allowed. Note that for instances where multiple data files are being used, the time column number must be the same for all data files.
- TIME_CONVERSION_FACTOR: indicates how to convert time values from the data file into seconds. STADIA assumes the data are in seconds unless this parameter is changed. If the user is using step numbers instead of time, then they will need to convert the steps to seconds by inputting the frame rate of the input data. Examples of potential values of time conversion factors include:
  - TIME_CONVERSION_FACTOR = 1; if the time unit in the data are in seconds.
  - TIME_CONVERSION_FACTOR = 60; if the time unit in the data are in minutes.
  - TIME_CONVERSION_FACTOR = 3600; if the time unit in the data are in hours.
  - TIME_CONVERSION_FACTOR = 0.5; if the data acquisition rate was 2 frames per second.

- INPUT_FILE_DELIMITER: indicates the delimiter used to separate terms in the input file. This parameter is needed for STADIA to know what to look for in order to correctly identify each data point when it is reading the data file. Common choices for a delimiter include spaces, tabs, commas, or semicolons. Files with type .csv (comma separated values) will by default have a comma as the delimiter. Here are some examples for defining this term:
  - INPUT_FILE_DELIMITER = ' '; for files using a space to separate terms.
  - INPUT_FILE_DELIMITER = '\t'; for files using a tab to separate terms.
  - INPUT_FILE_DELIMITER = ','; for files using a comma to separate terms.
  - INPUT_FILE_DELIMITER = ';'; for files using a semicolon to separate terms.

2.3 STADIA stages

After initialization, STADIA progresses through the analysis of the input file in multiple stages. Depending on user-defined parameters chosen before running STADIA, the program will fully analyze the data, or it will run in diagnostic mode, which provides suggestions for how to perform the analysis (see the diagnostic mode documentation for more information (GoodsonLab GitHub STADIA Repository, 2019)).

2.3.1 Segmentation

During this stage, STADIA generates a continuous piecewise linear approximation of MT length history data. The segmentation stage approximates microtubule dynamics by first identifying and connecting major peaks and valleys in the length history data, and makes improvements by iteratively adding new vertices where significant changes in behavior occur. Thus, the final approximation accurately captures DI behaviors that can be characterized with linear rates of change. This stage provides a macro-level representation of microtubule behavior, rather than the finer level details observed from subunit-level fluctuations. During segmentation, the linear approximation is controlled by two user-defined error thresholds: the minimum time step and the maximum error tolerance:

- MIN_TIME_STEP_INPUT: indicates the minimum amount of time that STADIA will consider when choosing segment vertices to generate the piecewise linear approximation. In other words, linear segments cannot have a duration less than the MIN_TIME_STEP_INPUT.
- ERROR_TOLERANCE_LEVEL: this parameter controls the level of error (in units of dimer lengths) that will be tolerated when approximating a line segment to a period of MT behavior. Errors are measured pointwise (i.e., between measured points provided from the input data and the point on the corresponding line segment approximation).

Users can make changes to these parameters to find values that they believe are most appropriate for their data. It is important to find the right balance between the minimum time step input and the margin of error, so that moments representing significant changes in MT behavior are identified as vertices in the approximation without capturing noise (Fig. 4).

Note that trying to use a strict (small) value for the maximum error tolerance with a large value for the minimum time step can lead to an irreconcilable error scenario, where the algorithm cannot improve the line segment approximation to the user's desired error tolerance level. If errors are not resolved after several attempts to find an alternative vertex to improve the approximation, STADIA is forced to accept an approximation "as-is," and reports to the user where the approximation superseded the user's desired level of error.

2.3.2 Classification

The classification stage performs the fundamental task of applying DI phase labels to segments of MT length history data. At this point, the segmentation stage has produced individual segments representing periods of sustained behavior in the MT length history data. Each line segment has three features to define points in 3D: the time duration (change in the horizontal axis, or the run of the segment), overall height change (change in the vertical axis, or the rise of the segment), and the slope of the MT length change, which is calculated from the ratio of the segment height change to the time interval (i.e., rise over run). The classification stage utilizes these three segment features to distinguish between behaviors that correspond to different DI phase classes (see (Mahserejian et al., 2019) for more discussion).

STADIA has two modes that can be used together to achieve optimal classification of MT length history data: Automated Mode (Section 2.3.2.3) is used if you already know how many behaviors you expect to detect, and Diagnostic Mode is used for situations where the number of behaviors is unclear. The Automated Mode as well as initial classification steps are discussed in detail below. For more information regarding the Diagnostic Mode, please refer to (GoodsonLab GitHub STADIA Repository, 2019).

2.3.2.1 Initial classifications: Identifying nucleation and flat segments

Before moving forward with classification, it is useful to recall that STADIA results can be used to compare data sets sourced from computational simulations and laboratory experiments. In the latter case, the limitations of light microscopy can make it impossible to observe short microtubules; such microtubules are described as being in a nucleation phase. For this reason, STADIA has a user-defined term to represent the nucleation threshold, such that MT lengths below this level will be removed from the classification process. Segments that have both endpoints below the nucleation threshold level are labeled as being in nucleation, and excluded from the remainder of the classification stage. Doing so allows STADIA to focus the analysis on periods of MT dynamics that are comparable across data sets from any source, where the more interesting DI behavior takes place.

Of the length history data that are observable, there may exist segments that display ambiguous behavior. In other words, there could exist near-flat segments that do not resemble either growth or shortening phases typically expected in MT behavior. Thus, STADIA performs the next step in the classification stage by identifying these flat segments, where substantial growth and shortening do not take place, according

to two user-defined thresholds: a slope threshold to indicate segments that are too flat, and a height threshold to find segments that do not change appreciably in length. A "flat stutter" phase label is applied to these segments, and they are set aside until later parts of the classification stage.

The following are the user-defined terms involved with the initial part of the classification stage:

- NUC_HEIGHT_THRESHOLD_INPUT: defines the nucleation level. Any segments below this threshold will not be included as part of the DI phase and transition analysis. This positive value has a unit of subunit-lengths (i.e., dimer-lengths, 80 Å or 8 nm units).
- FLATSTUT_MAX_SEGMENTSLOPE_THRESHOLD_INPUT: defines the maximum magnitude of segment slopes to be considered in the flat stutter phase class. The continuous positive values have a unit of subunit lengths per second.
- FLATSTUT_MAX_SEGMENTHEIGHT_THRESHOLD_INPUT: defines the maximum magnitude of segment height change to be considered in the flat stutter phase class. This continuous positive value has a unit of subunit lengths.

2.3.2.2 Classification using *k*-means clustering

The next task is to find how the remaining segments from the length history approximation are separated into different DI phase classes. The goal here is to conduct clustering in the 3D space where data points representing individual segments from the piecewise linear approximation reside. At the heart of the classification stage lies the *k*-means clustering algorithm, an unsupervised machine learning approach that separates a given data set into *k*-many clusters. The boundaries of these clusters are defined by a Voronoi diagram (Lloyd, 1982; Macqueen, 1967). The algorithm begins by randomly sampling *k* initial data points to represent the centroid, or mean, of a cluster defined by the nearest data points. Iteratively, the centroid location is recalculated, and the cluster is redefined until the clustering results cease to change appreciably. Since the final centroid locations are sensitive to the random selection at the first iteration (i.e., *k*-means clustering is not guaranteed to converge to a global optimum), this process is repeated 500 times, and the result with centroid locations that produce the best inter-cluster separation is selected to be used for the final clustering.

The *k*-means clustering algorithm is a distance-based method that assumes that clusters to be identified follow a Gaussian distribution. However, the raw segment dataset has an inconsistent distribution in each of its three dimensions, with the time duration values varying over a particularly large range. To aid the analysis, STADIA uses a pre-processing step to scale and standardize the data. More specifically, each point in 3D first has a natural logarithm applied to its three components. Then, each of the data points is standardized with respect to the mean and standard deviation of the data set. This scaled and standardized dataset is then passed into the clustering procedure.

More precisely, the pre-processing step applies the following transformation:

$$[X_{new}, Y_{new}, Z_{new}] = \left[\frac{log\,(X_{old}) - \mu_X}{S_X{}^2}, \frac{log\,(Y_{old}) - \mu_Y}{S_Y{}^2}, \frac{log\,(Z_{old}) - \mu_Z}{S_Z{}^2}\right]$$

where,

- $[X_{old}, Y_{old}, Z_{old}]$= the raw data provided from the linear segment features identified in the segmentation stage (time duration, height change, and slope for X, Y, and Z, respectively)
- μ_i= mean of the i-th dimension for the collective data passed into clustering
- $S_i{}^2$ = standard deviation of the i-th dimension for the collective data passed into clustering
- $[X_{new}, Y_{new}, Z_{new}]$= the scaled and standardized version of the segment data now centered around the origin in 3D space

The clustering procedures then use the scaled and standardized points to identify how the data should be separated. Once labels are determined for each point, the cluster labels are applied to data points in the original space so that the phase and transition analysis is conducted on observed segment features.

The intention of developing STADIA was to take an automated approach that did not make assumptions on how the data should be separated. However, the k-means algorithm requires prior knowledge of the appropriate number of clusters to conduct the method. To this end, we rely on the gap statistic to aid in selecting the optimal k-value (Tibshirani, Walther, & Hastie, 2001). In cases where the k-value is not yet determined, users can run STADIA in the *diagnostic mode*, where a gap statistic plot is generated to help compare clustering results using different k-values. Once the user determines the appropriate k-values to cluster positive and negative slopes, k-values can be entered into the Input_and_Run.m file, and STADIA can then operate in *automated mode* to complete the classification stage.

Below, we assume that users have already chosen a k-value. Users may use $k = 3$ for both positive and negative slope segments, as was found to be appropriate for both our simulation data and some related experiments (Mahserejian et al., 2019). Alternatively, the user may run STADIA in diagnostic mode to determine the best k-values for their data. Information on running STADIA in diagnostic mode can be found at (GoodsonLab GitHub STADIA Repository, 2019).

2.3.2.3 Running STADIA in automated mode

Once the user has determined the optimal number of clusters to classify the positive and negative slope segments (e.g., by using external information or by running STADIA in Diagnostic Mode), STADIA analysis can move forward using the automated mode. For practical purposes, the user's choices for k-value are restricted to the positive integers 1, 2, or 3. With the selected k-values, the classification stage progresses by conducting 500 iterations of k-means clustering (repeats are done to increase the chances of encountering the best clustering result). The next step is to apply relevant labels to the clusters that are identified.

Selecting $k = 1$ or $k = 3$ for k-means clustering yields scenarios that are straightforward for STADIA to automate phase class labeling. For illustration purposes, consider just the positive sloped segments. If $k = 1$ is selected, then there is only a single cluster for the positive-sloped segment subset, effectively casting all of those segments into a single growth phase. If $k = 3$ is selected, all of the observed variations of cluster labels will be needed to describe these segments: the cluster corresponding to the centroid that has the lowest slope component is labeled as the *up stutter* cluster, and the other two clusters are labeled as *brief* and *sustained growth* depending on the time component of their cluster centroids.

However, if $k = 2$ is selected, further information is needed to determine how to label the clusters. Having two stutter phases is unlikely, leaving two possible options: one up stutter phase and one growth phase, or two growth phases. A similar situation exists for the negative slope segments, but with *down stutters* corresponding to the cluster with the highest (lowest absolute value) slope component and with *brief* and *sustained shortening* for clusters with the lower (higher absolute value) slope components, in the $k = 3$ case. If $k = 2$ is selected for either the positive or negative sloped segments, users must also select an additional option for applying phase labels.

The following user-defined terms control the automated mode of STADIA's classification stage:

- KMEANS_NumClust_PosSlope: the k-value used to conduct k-means clustering on the positive sloped segments. Choices are limited to the integer choices 1, 2, or 3.
- KMEANS_NumClust_NegSlope: the k-value used to conduct k-means clustering on the negative sloped segments. Choices are limited to the integer choices 1, 2, or 3.
- KMEANS_Pos2_Option: option for phase labeling if the user sets KMEANS_NumClust_PosSlope = 2. The two possible choices for cluster labels are the following:
  - KMEANS_Pos2_Option = 'A'; brief growth and sustained growth.
  - KMEANS_Pos2_Option = 'B'; up-stutter and growth.
- KMEANS_Neg2_Option: option for phase labeling if the user sets KMEANS_NumClust_NegSlope = 2. The two possible choices for cluster labels are the following:
  - KMEANS_Neg2_Option = 'A'; brief shortening and sustained shortening.
  - KMEANS_Neg2_Option = 'B'; down-stutter and shortening.

Before finishing the classification step, the flat stutter segments identified in the earlier parts of the classification are assembled together with the results of the clustering algorithm. Following the statistical similarities between the clusters, they are bundled into three phase classes: growth, stutter, and shortening (Mahserejian et al., 2019). Doing so concludes the classification stage, and the phase segments and their chronological order can now be analyzed.

2.3.3 Phase and transition analysis

In the final stage, STADIA uses the segment phase classes to quantify characteristics of each phase and how phases transition into one another. More specifically, the values of interest include the rates of change for each phase, and frequencies of different types of phase transitions.

The rates of MT length changes are obtained simply from the average slope values for each corresponding phase, which are easily transferable from the features created during the segmentation stage, after applying the labels from the classification stage. The growth and shortening rates are typically close to those determined by classical methods of characterizing MT DI, whereas the rates of length change during stutter rates are typically much closer to zero. It is important to note that segments are now binned into more appropriate phase classes, and so the rate calculations for each phase are more precise (e.g., after the STADIA classification stage, flatter stutter segments are identified and excluded when calculating the average growth rate).

To obtain the frequencies of phase transitions, one must consider the chronological order of how growth, stutter, and shortening phase segments are observed. Note that segments within nucleation thresholds are not considered when calculating phase transitions. Consistent with classical DI analysis, we are interested in phase transitions that begin and end only with either growth or shortening phases. Thus we omit the permutations of phase orderings that start or terminate in a stutter phase. The first two phase transitions are familiar from standard DI analysis: catastrophe, where a MT that begins in growth transitions into shortening; and rescue, where a MT that begins in shortening transitions into growth. However, now that stutter phase segments are identifiable, we broaden these two classical transitions to include different variations, abrupt and transitional, depending on whether or not the phase switch involves a stutter. Additionally, a new type of phase transition should also be considered, where a stutter phase interrupts an on-going growth or shortening phase. The complete list of phase transitions and the corresponding order of phases considered in STADIA are as follows:

- Abrupt catastrophe: growth to shortening
- Transitional catastrophe: growth to stutter to shortening
- Abrupt rescue: shortening to growth
- Transitional rescue: shortening to stutter to growth
- Interrupted growth: growth to stutter to growth
- Interrupted shortening: shortening to stutter to shortening

For each of these types of phase transitions, frequencies are calculated by counting the number of occurrences, and dividing by the total time spent in the initial phase. For example, the formula for the frequency of transitional catastrophes is as follows:

$$F_{TransCat} = \frac{\#\ of\ transitional\ catastrophe\ events}{total\ time\ spent\ in\ growth\ phase}$$

Note that the classically recognized DI phase transitions are analogous to the abrupt catastrophe and abrupt rescue transitions considered here. However, past transition quantification methods would typically have lumped the transitional catastrophe and transitional rescue events in with the catastrophe and rescue measurements. In order to make direct comparisons of past results to those produced by STADIA, a user can simply take the sum of the frequencies of the abrupt and transitional variants. For instance, the following formula can be used to compare frequency of catastrophe from past approaches to frequencies produced from STADIA results:

$$F_{Cat} = F_{AbruptCat} + F_{TransCat}$$

STADIA does not automatically compute these types of frequencies, and so the simple additional step is left to the user for the cases where users are interested in determining the overall F_{cat}.

Rates of MT length change, along with other phase segment attributes (e.g., total number of segments per phase), and the six types of phase transition frequencies are all automatically computed for the phase classes identified during the classification stage. There are no user-defined parameters associated with this stage.

2.4 Visualizing STADIA results

The final options that remain for users to control are associated with visualizing the results from each stage. Simply put, the user decides which figures should be created, and of these, which should be saved automatically to file. It is important to note that any figures selected to be saved will automatically close after being written to the appropriate file. Otherwise, the MATLAB figure window will remain open for the user to investigate as desired. Detailed descriptions of figures and how to generate them can be found in Sections 3 and 4.

3 STADIA outputs

Once all stages of analysis have been completed, the resulting output for STADIA is produced in two formats: text files organizing the results, and figures illustrating the results. Both text files and figures are generated when using STADIA in both diagnostic and automated modes, but they consist of different content placed in different output directories. This section covers the different files and content that are generated, and where to find them once STADIA procedures are completed. Note that multiple formats of figures are saved: .fig files allow users to open the original figures in MATLAB in the event that additional exploration or plot manipulation is desired, while .png files allow quick access to figures that are compatible with other word processing and presentation software without requiring MATLAB.

All outputs discussed here are relevant only for the *Automated Mode*. For information regarding *Diagnostic Mode* outputs, users should refer to the diagnostic mode documentation that can be found at (GoodsonLab GitHub STADIA Repository, 2019), where users can find guidance on how to select appropriate k-values for their data.

3.1 STADIA output from automated mode

The output files generated from the automated mode are produced using the user-defined k-values in the clustering procedures. Since the results will be sensitive to STADIA's user-defined parameters, a dedicated directory is created to store the output files using the following naming convention:

"STADIA_Output_for_" + FILE_NAME_INPUT + DATETIMESTAMP

where FILE_NAME_INPUT is the filename associated with the input file for the length history data that STADIA analyzed, and DATETIMESTAMP is the date and time stamp associated with the time that STADIA is executed to prevent overwriting results from different runs on the same input file. Note that if multiple input files are provided, only the first one is used to create the directory name.

Recall that the automated mode runs through all the stages of STADIA, which involve different types of analysis within each stage that produce different types of results. These results are organized as a combination of figures and text files in the output directory using the following filenames that correspond to each stage. The outputs listed under "General STADIA Relevant Values," "Segmentation Results," "Classification Results," and "Phase and Transition Analysis Results" are useful for more in depth analysis, but most users will be primarily interested in the summary figures generated from these outputs, listed below as Figs. 1–7.

- General STADIA relevant values:
  - DI_parameters_output.txt: a text file listing user-defined STADIA parameters, and the resulting DI parameters output from the phase and transition analysis stage, which include the statistics for the slopes/rates corresponding to DI phase classes, and frequency of all types of phase transitions.
- Segmentation and classification results:
  - DIsegmentPhaseData.txt: a tab delimited text file indicating the following information for each segment identified during the segmentation stage: starting time, starting MT length, time duration, height change, and slope. Additionally, the DI phase class labels from the classification stage are also included in this file (Phase Label key: -12 = Long Shortening, -11 = Brief Shortening, -1 = Down Stutter, 0 = Flat Stutter, 1 = Up Stutter, 11 = Brief Growth, 12 = Long Growth).
- Phase and transition analysis results:
  - Abrupt_Catastrophe_output.txt: text file with information regarding the time at which each abrupt catastrophe takes place as well as the length of the MT at the time of each catastrophe.

- Abrupt_Rescue_output.txt: text file with information regarding the time at which each abrupt rescue takes place as well as the length of the MT at the time of each rescue.
- Transitional_Catastrophe_output.txt: text file with information regarding the start and end times of the stutter phase within each transitional catastrophe as well as the length of the MT at the start and end of each stutter phase.
- Transitional_Rescue_output.txt: text file with information regarding the start and end times of the stutter phase within each transitional rescue as well as the length of the MT at the start and end of each stutter phase.
- Interrupted_Growth_output.txt: text file with information regarding the start and end times of the stutter phase within each interrupted growth as well as the length of the MT at the start and end of each stutter phase.
- Interrupted_Shortening_output.txt: text file with information regarding the start and end times of the stutter phase within each interrupted shortening as well as the length of the MT at the start and end of each stutter phase.

Figures are generated to visualize the results from running STADIA in the fully automated mode. The following seven figures have full descriptions that can be found in Section 4:

(1) Microtubule length history approximation:

FILE_NAME_INPUT + "_PWLinearFit_Tol" + ERROR_TOLERANCE_LEVEL + .fig (.png). An example output filename would be "filename_PWLinearFit_Tol20.fig."

(2) *k*-Means clustering results:

FILE_NAME_INPUT + "_PosNegSlopClusterResults" + .fig (.png)

(3) DI segment phase classification:

FILE_NAME_INPUT + "_DIPhaseClassification" + .fig (.png)

(4) DI phase classes based on MT length history:

FILE_NAME_INPUT + "_DIPhasesOnLengthHistory" + .fig (.png)

(5) Segment statistics for each DI phase:

FILE_NAME_INPUT + "_AvgDIMeasurements" + .fig (.png)

(6) Total measurements for DI phase segments:

FILE_NAME_INPUT + "_TotDIMeasurements" + .fig (.png)

(7) Resulting measurements for changes in DI phase:

FILE_NAME_INPUT + "_DIPhaseChangeResults" + .fig (.png).

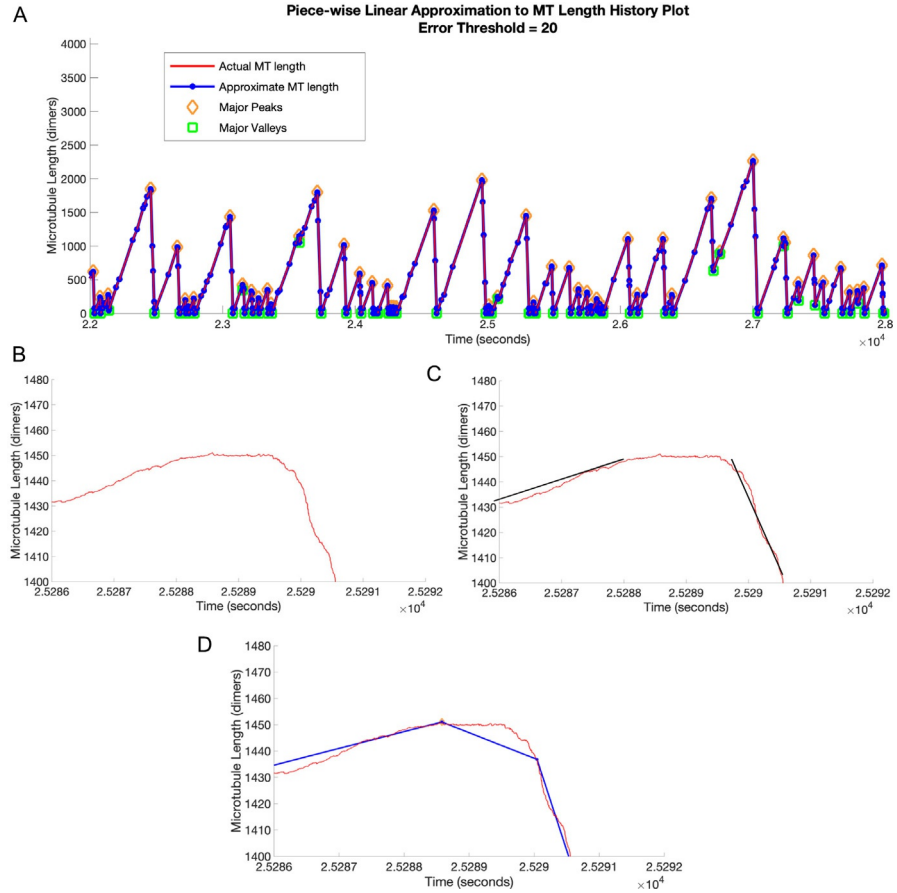


FIG. 1

STADIA output of MT length history data. (A) Length history plot as traditionally viewed for DI analysis, where STADIA's piecewise linear approximation (blue) of the length history is plotted on top of the actual data (red). STADIA identifies points of rescue and nucleation (major valleys, green boxes) and points of catastrophe (major peaks, orange diamonds). This length-history plot corresponds to a time period between 22,000 and 28,000 s of a 10 h simulation. Note that each line segment in this panel corresponds to a point in the clustering analysis shown in Fig. 2. (B) Close-up of a peak of simulation data (indicated by the black arrow in panel (A)) showing ambiguous MT behavior leading up to a catastrophe. (C) Linear approximation of peak close-up using classical two-state analysis, where the black lines indicate the growth and shortening phases identified as part of traditional DI analysis, with an ambiguous area between. (D) STADIA linear approximation of peak behavior (blue; same line segments as in (A)), which is able to capture the ambiguous behavior. Note that a more stringent error threshold would allow a closer linear approximation of the ambiguous behavior, but can also capture noise. We have found that an error threshold of 20 subunits as used here (this corresponds to 160 nm) is a reasonable choice for our simulations and experiments.

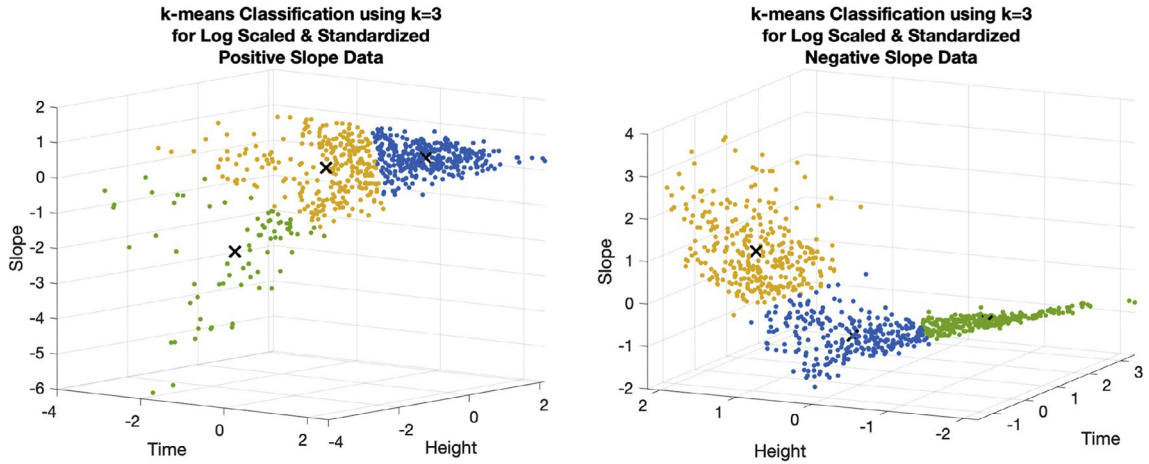


FIG. 2

STADIA classifies each line segment from the piecewise linear approximation into clusters using the iterative, *k*-means clustering algorithm based on the segment features (height, time duration, and slope). Positive slopes (left) and negative slopes (right) are clustered separately. Line segments are represented as individual data points and the centroid of each group is represented by an X. The third dimension, slope, is also present in the .fig version of this output and users may rotate these plots in order to better visualize these clusters in 3-D space. Finally, note that the line segment data have been log-transformed and standardized in these plots to better suit the *k*-means clustering algorithm.

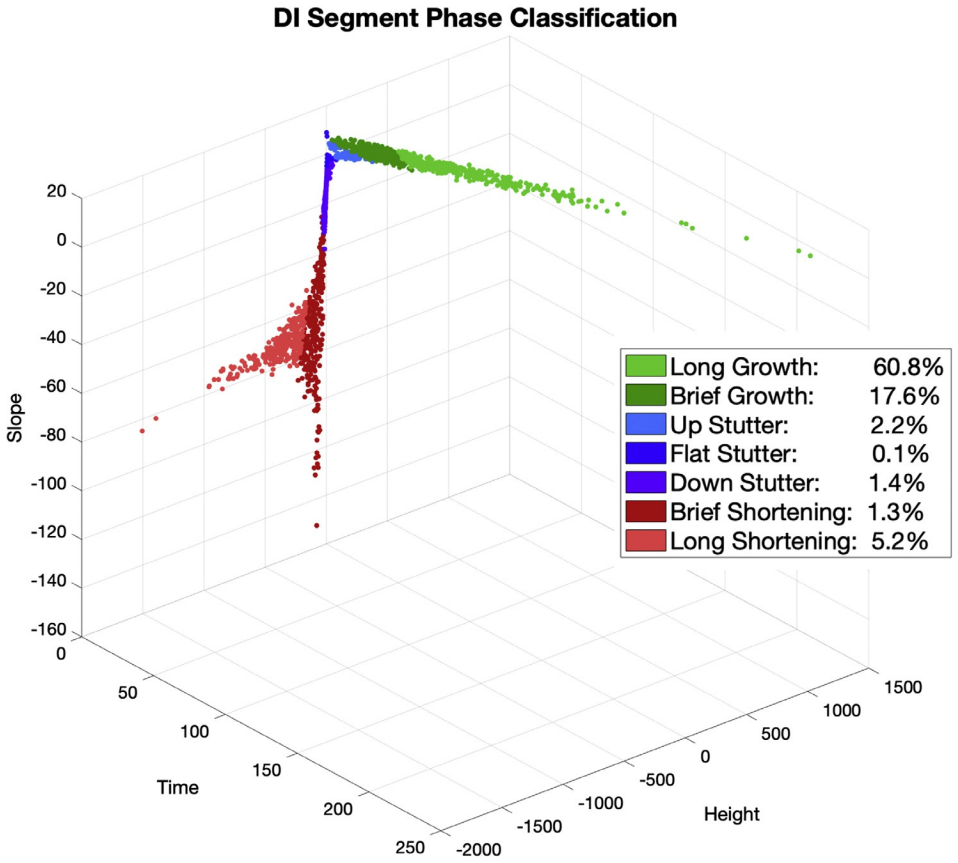


FIG. 3

STADIA classification of all line segment clusters in one three-dimensional plot, including near zero slope segments. Each data point is labeled with the color corresponding to the cluster to which it belongs. Note that in contrast to Fig. 2, data in this figure is no longer log transformed and standardized (i.e., after classification with *k*-means is complete, line segment data points are converted back to raw data in all three dimensions while still retaining the cluster label assigned to them during *k*-means clustering).

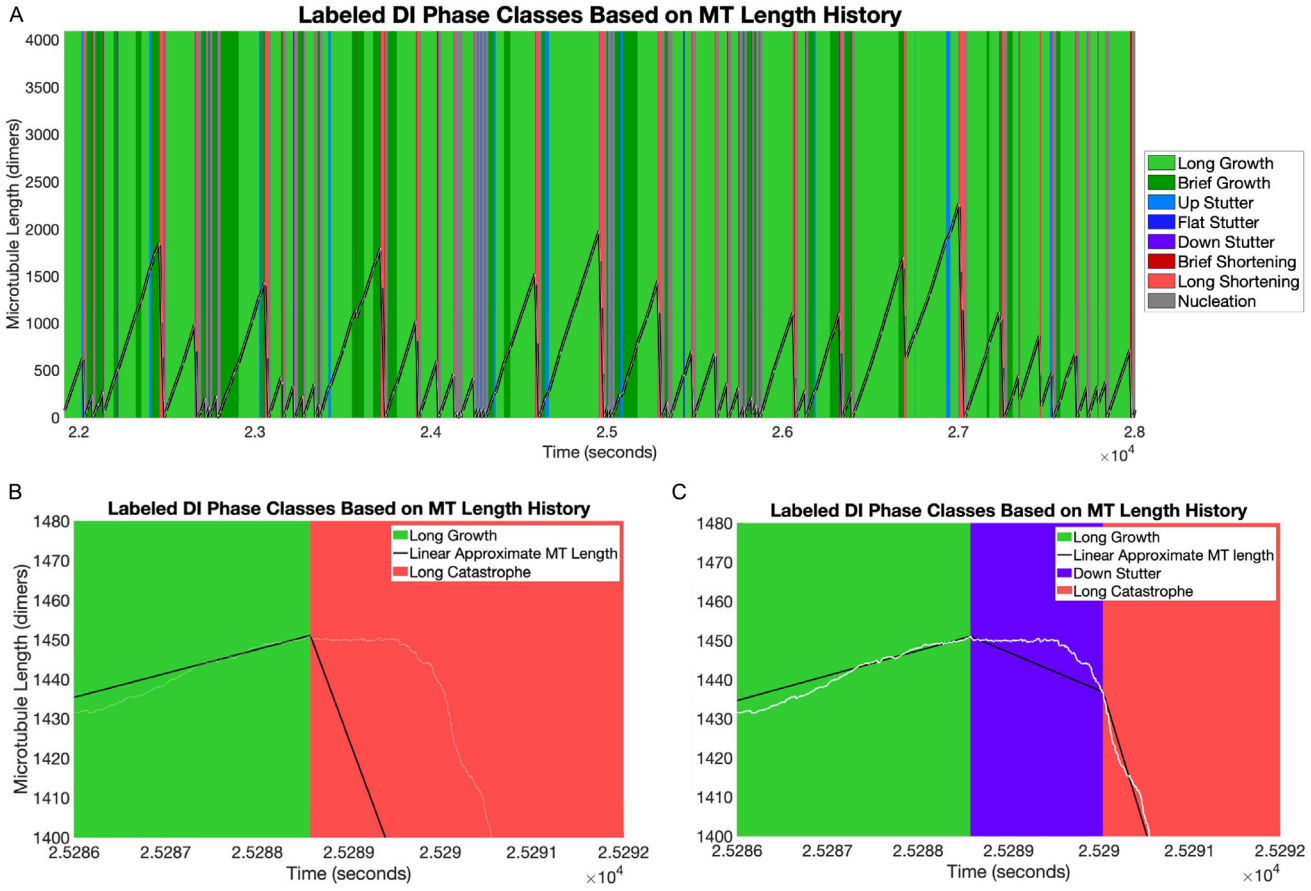


FIG. 4

Length history data labeled according to the STADIA phase output. (A) STADIA-annotated version of the length-history plot from Fig. 1. (B) Annotated peak from Fig. 1D, indicating that a stutter (purple) occurs in between a growth phase (green) and a depolymerization phase (red) (i.e., this is a transitional catastrophe).

Segment Statistics for each DI Phase

	# of Segments	Time Mean	Time StDev	Height Mean	Height StDev	Slope Mean	Slope StDev
LngGr	333	65.7	29.0	386.4	168.1	5.9	0.5
BrfGr	278	22.7	8.5	124.4	46.1	5.6	1.0
UpSt	74	10.9	5.1	33.7	19.6	3.1	1.0
FltSt	21	1.8	1.8	-0.8	1.3	-0.1	3.1
DnSt	320	1.6	0.9	-20.2	10.8	-14.5	7.4
BrfSh	276	1.7	0.8	-116.1	56.7	-68.2	21.3
LngSh	271	6.9	3.4	-468.5	234.1	-67.7	5.2

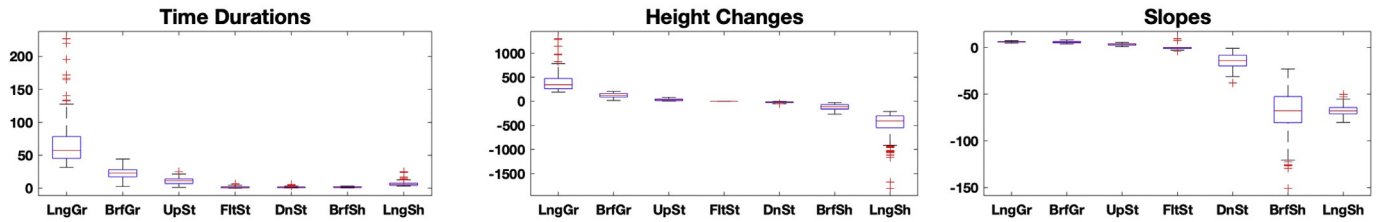


FIG. 5

STADIA table listing values and statistics of DI phase behavior with corresponding box and whisker plots to visualize the data for each cluster.

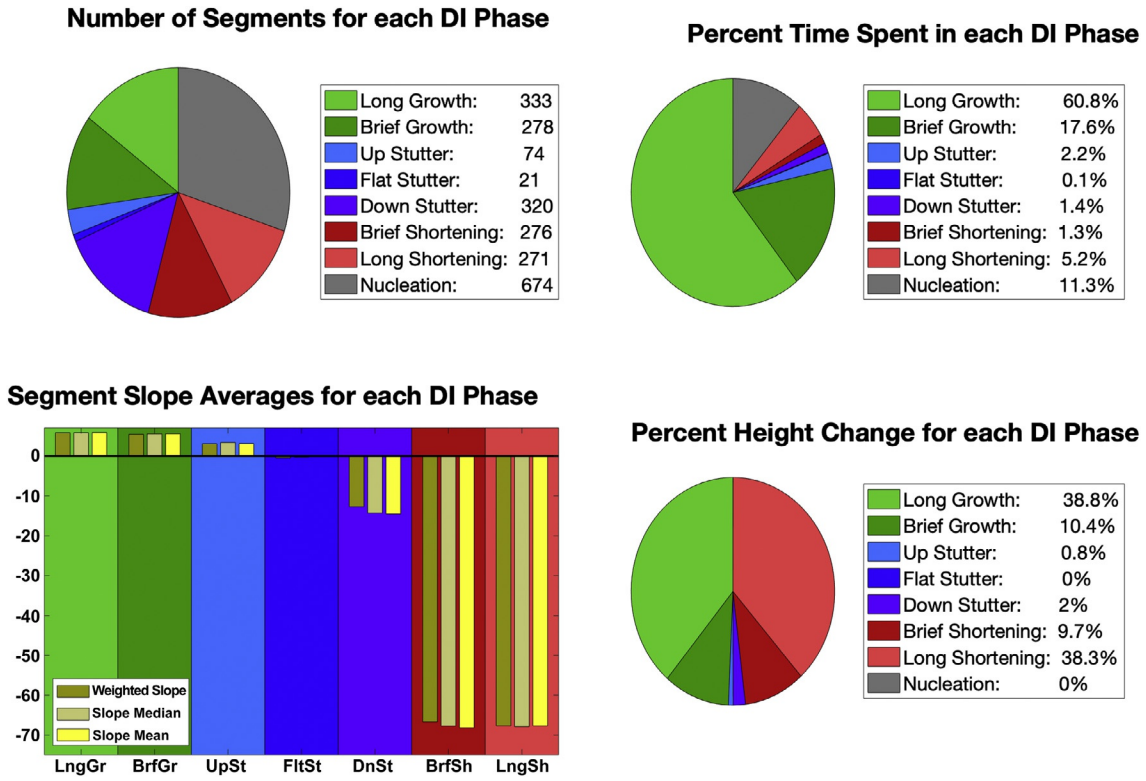
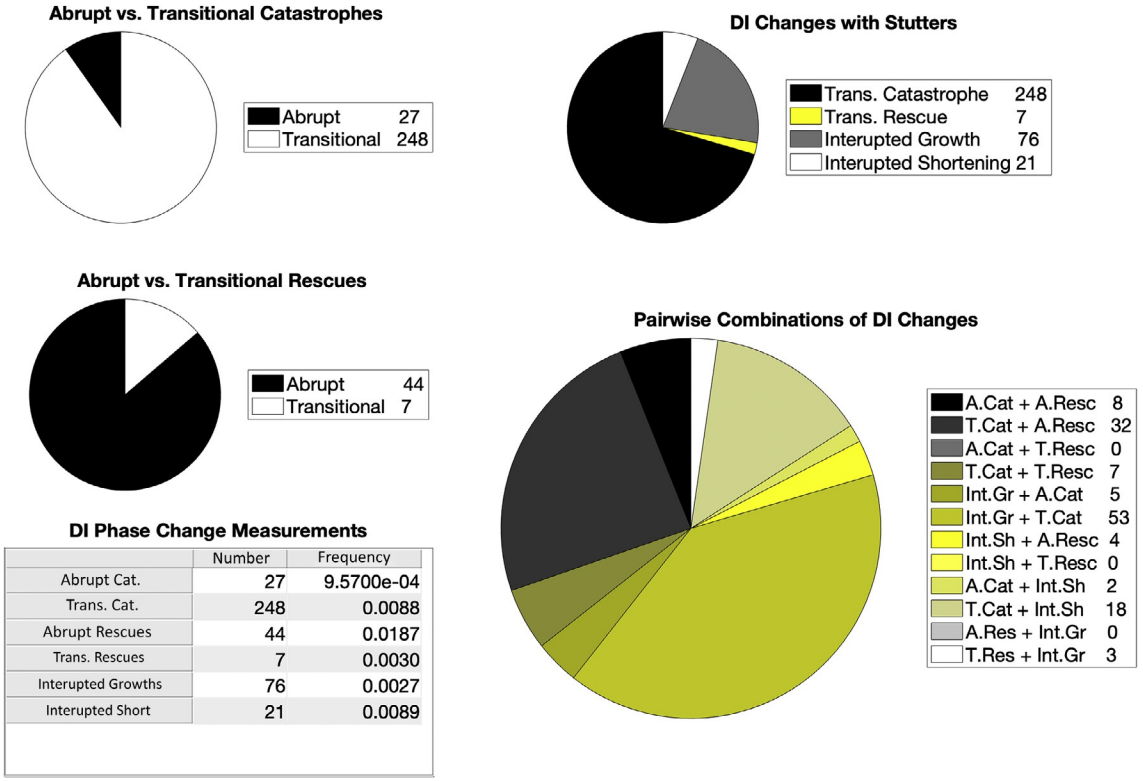


FIG. 6

STADIA output of statistics of the number and behavior of each DI phase. Different DI phase behaviors are quantified based on their prevalence (Number of segments and percent time spent in each phase) and the percent of total height change attributed to each phase. Statistics regarding the average slope values of each phase are provided as well.



	Number	Frequency
Abrupt Cat.	27	9.5700e-04
Trans. Cat.	248	0.0088
Abrupt Rescues	44	0.0187
Trans. Rescues	7	0.0030
Interupted Growths	76	0.0027
Interupted Short	21	0.0089

FIG. 7

STADIA output of statistics of the prevalence of specific types of DI transitions. DI transitions with (transitional) and without (abrupt) stutters are quantified and compared in the pie charts “Abrupt vs. Transitional Catastrophes” and “Abrupt vs. Transitional Rescues.” “DI Changes with Stutters” provides information about how often the different transitions involving stutters occur. “Pairwise Combinations of DI Changes” visualizes an analysis of the chronological ordering of consecutive DI transitions to provide information about the pattern of MT behavior on longer timescales. Finally, frequencies of all transitions considered in this analysis are reported in the table at the bottom left of this figure.

3.2 User-defined inputs for automated mode outputs

In order to activate plots, the command, PLOT_FIG_#, for whichever plot(s) the user desires to view, must be set equal to 1. If the user does not want to see a specific plot, they can set the parameter equal to 0. Note: The default mode of the program is set equal to 1, so STADIA will generate all plots unless the user commands otherwise. In order to save the plots that will be created, users should set the SAVE_FIG_# parameter equal to 1. STADIA will automatically delete the graphs once they are closed, so to save the files this parameter should be set to 1.

STADIA users can adjust the viewing windows for Figs. 1 and 4. The parameters:

FIG_WINDOW_START = #; and
FIG_WINDOW_END = #;

control the time ranges plotted in these figures. The MT height window is automatically set to the maximum, so there is no need for a parameter to change the y-axis.

Users will also be able to save the date and time in the name of each output file by setting the parameter:

INCLUDE_DATE_TIME = 1;

which is useful when running STADIA on the same data multiple times as a way to differentiate between each run. The default mode for this parameter is 0, where the date and time is not saved in each filename.

The variables in the MATLAB workspace can also be saved as a .mat file by setting the parameter:

SAVE_MATLAB_WORKSPACE = 1;

The default mode for this parameter is also 0, so users need to manually change this parameter to save the workspace variables.

Additionally, since Fig. 3 displays a 3-D scatter plot that has features that are typically difficult to display to audiences, an option to save an animated spinning version is offered by assigning a 1 value to SAVE_FIG_3_movie, which will generate an animated GIF in a .gif file format.

For examples of all possible figures generated by STADIA using an analysis of simulated MTs, readers are encouraged to refer to Section 4.

4 Data analysis

In order to illustrate the process of performing an analysis of MT DI with STADIA, we first ran a simulation using our detailed model of MT DI (Margolin et al. 2012). The simulation parameters were identical to those used in (Mahserejian et al., 2019), except that the simulation was initiated from a different random number seed. One output file from the simulation contains the MT length values at each time interval throughout the duration of the simulation. This file can be obtained from GitHub along with the STADIA software if users would like to replicate the

process shown below. The file was copied into the same directory as the remaining STADIA files, and the filename was copied into the FILE_NAME_INPUT field. After STADIA finished running in automated mode with $k = 3$ for both positive and negative slopes, the following seven plots were generated:

4.1 Microtubule length history (Fig. 1)

This plot illustrates the MT length history by mapping the actual lengths in red and the linear approximations in blue. With this input file, the plot reflects the behavior of one MT over 10 h, but in cases where there are multiple input files, the plot will "stitch" together the length history of multiple MTs onto a single plot, representing the behavior as one long MT. Major peaks and valleys are also plotted to help the user more clearly identify events of dynamic behavior. The linear approximations of the MT length should look similar to the actual length. When examined in closer detail, subtle differences between the approximation and the actual data should be visible. Examination of these differences is important when choosing values for the MIN_-TIME_STEP_INPUT and ERROR_TOLERANCE_LEVEL parameters because these determine the level of accuracy in the approximation.

4.2 *k*-means classification (Fig. 2)

As discussed above, STADIA uses *k*-means clustering to classify the MT length history line segments from the segmentation stage (Fig. 1) into groups that share similar characteristics. Before clustering, the length segments identified during the segmentation phase are log-transformed and standardized, then plotted in 3-D space, with the dimensions corresponding to height change, slope, and time (each point corresponds to a single line segment). In the graphs created during this step, colors indicate the different clusters, and a set of black X's indicate the location of the mean of each cluster.

4.3 DI segment phase classification (Fig. 3)

This figure shows the results of line segment classification, mapped in 3-D space; note that data points in this plot have been converted back to raw data in all three dimensions (i.e., all features have been un-standardized and un-log-transformed). The user can rotate the graph to view it as desired.

4.4 MT length history plot labeled with colors corresponding to DI phases (Fig. 4)

In this figure, the length history is color-coded according to the assigned DI phase. Growth phases are variants of green, depolymerization phases are variants of red, and stutters are intermediate colors.

4.5 Segment statistics for each DI phase (Fig. 5)

This figure presents a summary of the statistics for each detected behavior phase, including the mean and standard deviation of the length change, time duration, and slope of each cluster. At the top of the figure is a table listing these values; box and whisker plots in the lower half of the figure illustrate how these data are distributed for each phase class.

4.6 Total measurements for DI phase segments (Fig. 6)

This figure compares the characteristics of the individual clusters, comparing the relative time, height changes, and number of individual segments for each phase. This figure also represents the average of the slope of each cluster as an arithmetic mean, median, and weighted average.

4.7 Resulting measurements for changes in DI phase (Fig. 7)

The last figure that STADIA produces in automated mode quantifies the transitions between phases. In addition to the classically recognized rescue and catastrophe transitions, STADIA has the capability of detecting newly identified DI phases called stutters, where a MT has little length change over a significant period of time. This figure presents the prevalence of stutters in various types of transitions by classifying DI transitions as abrupt (i.e., no stutter) catastrophes/rescues, transitional (i.e., occurring with a stutter) catastrophes/rescues, or interrupted growth/shortening (e.g., growth to stutter to growth, or short to stutter to short). Frequencies are calculated and reported for each type of transition.

5 Future developments

We will continue to develop STADIA and the associated tutorials (both found on GitHub, see (GoodsonLab GitHub STADIA Repository, 2019)) in response to user input and our own ideas for improvements. In addition, because STADIA is recently developed software, it is likely that bugs exist. Please submit comments about suggested improvements and/or bugs by creating an issue on the Github repository (GoodsonLab GitHub STADIA Repository, 2019).

6 Conclusion

The dynamic behavior of microtubules has been a focus of research in cell biology for three decades. This behavior has generally been described as consisting of growth and depolymerization phases, with instantaneous transitions (catastrophe and rescue) separating them. However, with the advent of technical improvements in acquiring DI data, it has become apparent that this traditional approach to

quantifying DI is not sufficient because ambiguous behaviors exist that do not fit neatly into this framework. This manuscript provides detailed instructions on how to use a new software tool called STADIA (Statistical Tool for Automated Dynamic Instability Analysis) to quantify microtubule DI in a more objective, precise, and reproducible way. STADIA does not employ arbitrary criteria or subjective input from the user, but rather engages machine learning and statistical methods to read and analyze MT length history data gathered *in vitro* or *in silico*. As long as researchers are careful to run the program correctly and are mindful of the possible discrepancies that can occur, STADIA can generate more detailed, objective, and precise quantitative analysis of MT dynamic instability behavior than is possible using classical approaches.

Acknowledgments

This work was supported by NSF MCB 1817966 (research grant to H.V.G.), NSF PHY 1806631 (Quarknet grant that supported or partially supported R.J.P., P.O.M., and M.S.) and an NSF GRFP DGE-1313583 award (to K.S.M.).

References

Desai, A., & Mitchison, T. J. (1997). Microtubule polymerization dynamics. *Annual Review of Cell and Developmental Biology*, *13*(1), 83–117. https://doi.org/10.1146/annurev.cellbio.13.1.83.

Duellberg, C., Cade, N. I., Holmes, D., & Surrey, T. (2016). The size of the EB cap determines instantaneous microtubule stability. *eLife, 5*. https://doi.org/10.7554/eLife.13470.

Duellberg, C., Cade, N. I., & Surrey, T. (2016). Microtubule aging probed by microfluidics-assisted tubulin washout. *Molecular Biology of the Cell*, *27*(22), 3563–3573. https://doi.org/10.1091/mbc.E16-07-0548.

GoodsonLab GitHub STADIA Repository (2019). https://doi.org/10.5281/zenodo.3575037.

Lawrence, E. J., Arpag, G., Norris, S. R., & Zanic, M. (2018). Human CLASP2 specifically regulates microtubule catastrophe and rescue. *Molecular Biology of the Cell*, *29*(10), 1168–1177. https://doi.org/10.1091/mbc.E18-01-0016.

Lloyd, S. P. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, *28*(2), 129–137. https://doi.org/10.1109/TIT.1982.1056489.

Macqueen, J. (1967). *Some methods for classification and analysis. In Proceedings of the fifth berkeley symposium on mathematical statistics and probability, volume 1: statistics* (pp. 281–297). *233*. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.308.8619.

Mahserejian, S. M., Scripture, J. P., Mauro, A. J., Lawrence, E. J., Jonasson, E. M., Murray, K. S., et al. (2019). Stutter: A transient dynamic instability phase that is strongly associated with catastrophe. *BioRxiv*. https://doi.org/10.1101/2019.12.16.878603.

Margolin, G., Gregoretti, I. V., Cickovski, T. M., Li, C., Shi, W., Alber, M. S., et al. (2012). The mechanisms of microtubule catastrophe and rescue: Implications from analysis of a dimer-scale computational model. *Molecular Biology of the Cell*, *23*(4), 642–656. https://doi.org/10.1091/mbc.E11-08-0688.

Maurer, S. P., Cade, N. I., Bohner, G., Gustafsson, N., Boutant, E., & Surrey, T. (2014). EB1 accelerates two conformational transitions important for microtubule maturation and dynamics. *Current Biology*, *24*(4), 372–384. https://doi.org/10.1016/j.cub.2013.12.042.

Mitchison, T., & Kirschner, M. (1984). Dynamic instability of microtubule growth. *Nature*, *312*(5991), 237–242. https://doi.org/10.1038/312237a0.

Rickman, J., Duellberg, C., Cade, N. I., Griffin, L. D., & Surrey, T. (2017). Steady-state EB cap size fluctuations are determined by stochastic microtubule growth and maturation. *Proceedings of the National Academy of Sciences of the United States of America*, *114*(13), 3427–3432. https://doi.org/10.1073/pnas.1620274114.

Tibshirani, R., Walther, G., & Hastie, T. (2001). Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *63*(2), 411–423. https://doi.org/10.1111/1467-9868.00293.

Walker, R. A., O'Brien, E. T., Pryer, N. K., Soboeiro, M. F., Voter, W. A., Erickson, H. P., et al. (1988). Dynamic instability of individual microtubules analyzed by video light microscopy: Rate constants and transition frequencies. *The Journal of Cell Biology*, *107*(4), 1437–1448. https://doi.org/10.1083/jcb.107.4.1437.