Usage Guide for MEDYAN **v4.0**

Papoian Lab, University of Maryland

Contents

1 Overview of new features and bug fixes in v3.1

We have addressed the following issues in MEDYAN v3.0:

- Mechanochemical side-effects producing heap mis-sorting in NRM reaction-diffusion algorithm
- Mechanical equilibration failures due to complexity of filament excluded volume
- Various parsing errors for input files and fixed output
- Detailed output in the case of a fatal run error

We have also added the following new features in MEDYAN v3.1:

- System restart ability from last output trajectory in the case of a stopped run
- Dynamic filament mechanochemistry between mechanical equilibration steps
- Ability to make simulation elements static at any point during runtime
- Visualization capability via VMD

2 Introduction

The cell cytoskeleton plays a key role in human biology and disease, contributing ubiquitously to such important processes as embryonic development, wound repair and cancer metastasis. The Papoian laboratory is interested in gaining deeper understanding of the physical chemistry behind these complex, far-from-equilibrium mechanochemical processes. His approach and model, named the *Mechanochemical Dynamics of Active Networks* (MEDYAN), is based on combining stochastic reaction-diffusion treatment of cellular biochemical processes with polymer physics of cytoskeletal filament network growth, while explicitly coupling chemistry and mechanics. For a more detailed description of MEDYAN, see [1].

The Papoian laboratory has developed a third-generation software package based on the MEDYAN model, to simulate growth dynamics of actin based filamentous networks *in vitro* and *in vivo*. Recent papers where MEDYAN or its predecessor, StochTools, were used can be found on the publication section of the Papoian group's main web page, or on the MEDYAN website. The MEDYAN package can also be extended to simulate the dynamics of any active matter network.

3 Overview of Features

MEDYAN is a package that can simultaneously simulate complex chemical and mechanical dynamics of an active matter network. For more information on the MEDYAN model, which MEDYAN implements, see [1].

3.1 Chemical capabilities

The chemical capabilities of MEDYAN include:

- Stochastic reaction-diffusion on a three dimensional grid using stochastic simulation algorithms, including the *direct* Gillespie algorithm and the *Next Reaction Method*.
- Complex chemical representation of filaments, allowing for heterogeneous chemical monomers in a single filament segment.
- A wide range of filament reactions, including:
  - Polymerization of either end of filament
  - Depolymerization of either end of filament
  - Filament nucleation, either branching or spontaneous
  - Severing of filament at chosen sections
  - Cross-linker binding and unbinding to filament
  - Motor walking, binding, and unbinding to filament
  - Monomeric reactions within a filament

3.2 Mechanical capabilities

MEDYAN allows for a wide range of mechanical interactions, including:

- Force fields for filament interactions, including
  - Filament stretching and bending
  - Branching point stretching, bending, and dihedral
  - Excluded volume
- Stretching force fields for cross-linkers and motors
- Boundary steric repulsion
- Bubble repulsion

These force fields can be minimized by a choice of conjugate gradient algorithm.

3.3 Mechanochemical coupling

MEDYAN couples chemistry and mechanics by altering reaction rates based on mechanical stresses in a given network. This allows for a full treatment of the complex mechanochemical responses in active matter networks. See [1] for a more detailed description.

4 Running MEDYAN

To run the MEDYAN executable, execute the following command in the terminal shell:

```
> ./MEDYAN -s <SystemFile> -i <InputDirectory> -o <OutputDirectory>
```

The `SystemFile` will be described in the later sections.

The `InputDirectory` specifies where all input files are contained, with their names being specified in the `SystemFile`. This must be an absolute directory path. The `OutputDirectory` specifies where the produced output will be placed. This also must be an absolute directory path. See the later sections for details on input and output files.

5 Input

5.1 System file

The system file is a simple text file that defines all parameters of the simulation. The MEDYAN executable must take in a system file as a command line argument.

Each parameter must be defined in the following syntax:

```
<PARAMETER>:  <PARAMETERVALUE>
```

where the parameter name is followed by a semicolon, and the value of the parameter is placed after the semicolon. Outlined below are the parameters that can be included.

Unless otherwise noted, all distance parameters are in units of nanometers.

All filament properties must be listed in the order of filament definition, and must be consistent across all definitions. To list various parameter values, provide a space between those values after the parameter qualifier. In cases where a filament type is needed, filament types range from 0 to `NUMFILAMENTTYPES`, as defined accordingly.

Parameter options will be listed as `<PARAMETER>` – value type – description.

5.1.1 Geometry

The following geometric parameters can be set. All geometry parameters must be set in the system file, or a startup error will result.

`NDIM` – 1, 2, 3 – Number of dimensions in system.

`NX` – int – Number of compartments in X direction.

`NY` – int – Number of compartments in Y direction.

`NZ` – int – Number of compartments in Z direction.

`COMPARTMENTSIZEX` – double – Size of compartment in X direction.

`COMPARTMENTSIZEY` – double – Size of compartment in Y direction.

`COMPARTMENTSIZEZ` – double – Size of compartment in Z direction.

`MONOMERSIZE` – double – Size of monomer for filament growth .

`CYLINDERSIZE` – double – Size of cylinder in filament.

`BOUNDARYSHAPE` – SPHERICAL, CUBIC, CAPSULE – Boundary shape.

`BOUNDARYDIAMETER` – double – Diameter for applicable shapes, including SPHERICAL and CAPSULE geometries.

If movement of boundaries is desired, the following parameters can also be set. If these parameters are not set, the system will assume non-moving boundaries. Currently, moving boundaries are only implemented for the CUBIC boundary shape.

`BOUNDARYMOVE` – NONE, ALL, TOP – Movement of a boundary. ALL specifies that all boundaries will move in the given direction, and top specifies that the top of the boundary in the z direction will move.

`BMOVESPEED` – double – Speed of boundary movement in nm/s. If a negative value is given, the boundary will move towards the center of the grid. If positive, the boundary will move away from the center of the grid.

`BMOVESTARTTIME` – double – Time at which the boundary will begin to move. If not specified, the boundary will start moving at the beginning of the simulation.

`BMOVEENDTIME` – double –Time at which the boundary will stop movement.

5.1.2 Mechanics

The following mechanical parameters can be set. It is noted that the number of parameters for each force field must match the number of species of that type, specified in the

`SystemFile`. This must be consistent for all simulation elements, including filaments, cross-linkers, motors, branchers, and bubbles. To set multiple parameters corresponding to multiple species, list the parameter values with space in between after the parameter qualifier.

Force field constant units are dependent on the potential used, but in general will be in pN and nm scaling. For more information on force fields used in the MEDYAN model, see [1]. If a force field type is left blank, that force field will not be included in the simulation.

`CONJUGATEGRADIENT` – POLAKRIBIERE, FLETCHERRIEVES, STEEPESTDESCENT – Type of conjugate gradient minimization.

`GRADIENTTOLERANCE` – double – Gradient tolerance in conjugate gradient (in pN).

`MAXDISTANCE` – double – Maximum distance beads can be moved in minimization.

`LAMBDAMAX` – double – Maximum lambda that can be returned in line search.

`FSTRETCHINGTYPE` – HARMONIC – Filament stretching force field.

`FSTRETCHINGK` – double – Filament stretching force constant.

`FBENDINGTYPE` – HARMONIC, COSINE – Filament bending force field.

`FBENDINGK` – double – Filament bending force constant.

`FBENDINGTHETA` – double – Filament bending angle (radians).

`LSTRETCHINGTYPE` – HARMONIC – Cross-linker stretching force field.

`LSTRETCHINGK` – double – Cross-linker stretching force constant.

`MSTRETCHINGTYPE` – HARMONIC – Motor stretching force field.

`MSTRETCHINGK` – double – Motor stretching force constant.

`BRSTRETCHINGTYPE` – HARMONIC – Branching point stretching force field.

`BRSTRETCHINGK` – double – Branching point stretching force constant.

`BRBENDINGTYPE` – COSINE – Branching point bending force field.

`BRBENDINGK` – double – Branching point bending force constant.

`BRBENDINGTHETA` – double – Branching point bending angle (radians).

`BRDIHEDRALFFTYPE` – COSINE – Branching point dihedral force field.

`BRDIHEDRALK` – double – Branching point stretching force constant.

`BRPOSITIONTYPE` – HARMONIC – Branching point position force field.

`BRPOSITIONK` – double – Branching point position force constant.

`VOLUMEFFTYPE` – REPULSION – Volume force type.

`VOLUMECUTOFF` – double – Volume interaction cutoff distance.

`VOLUMEK` – double – Volume force constant.

`BOUNDARYFFTYPE` – REPULSIONEXP – Boundary force type.

`BOUNDARYCUTOFF` – double – Boundary interaction cutoff distance.

`BOUNDARYINTERACTIONK` – double – Boundary force constant.

`BOUNDARYSCREENLENGTH` – double – Boundary screening length constant.

`BUBBLEFFTYPE` – REPULSIONEXP – Bubble force type.

`BUBBLECUTOFF` – double – Boundary interaction cutoff distance.

`BUBBLEINTERACTIONK` – double – Bubble force constant.

`BUBBLESCREENLENGTH` – double – Bubble screening length constant.

`BUBBLERADIUS` – double – Bubble radius.

`NUMBUBBLETYPES` – int – Number of different bubble types.

`MTOCFFTYPE` – ATTACHMENTHARMONIC – MTOC force type.

5.1.3 Chemistry

The following chemical parameters can be set. It should be noted that the number of parameters listed for each chemical species type that resides on a filament must match the number of filament types, specified in the `SystemFile`. This must be consistent for all filament types. To set multiple parameters corresponding to multiple filaments, list the

parameters with space in between after the parameter qualifier.

All chemical parameters must be set unless otherwise noted in the description. For the motor parameters, the number of parameters must match the number of motor species in the system. For more information on chemical algorithms, see [1].

An alternate set of parameters can be specified in replacement of `RUNTIME` for smaller systems in which simulation time is based on explicit reaction steps; if `RUNTIME` is not initialized or set to zero, the parameter `RUNSTEPS` and its associated chemical step-based parameter set will be used if provided.

`CHEMISTRYFILE` – string – Input chemistry file. Should be in the `InputDirectory`.

`CALGORITHM` – GILLESPIE, NRM – Chemistry algorithm used.

`RUNSTEPS` – int – Number of total chemical steps. If RUNTIME is set, will not be used.

`RUNTIME` – double – Total runtime of simulation (s).

`SNAPSHOTSTEPS` – int – Number of steps per snapshot. If SNAPSHOTTIME is set, will not be used.

`SNAPSHOTTIME` – double – Time of each snapshot (s).

`MINIMIZATIONSTEPS` – int – Number of chemical steps per mechanical equilibration. If MINIMIZATIONTIME is set, will not be used.

`MINIMIZATIONTIME` – double – Time between each mechanical equilibration (s)

`NEIGHBORLISTSTEPS` – int – Number of chemical steps per neighbor list update. This includes updating chemical reactions as well as force fields which rely on neighbor lists. If NEIGHBORLISTTIME is set, will not be used.

`NEIGHBORLISTTIME` – int – Time between each neighbor list update (s)

`NUMDIFFUSINGSPECIES` – int – Diffusing species in system.

`NUMBULKSPECIES` – int – Bulk species in system.

`NUMFILAMENTTYPES` – int – Number of different filament types.

`NUMFILAMENTSPECIES` – int – Filament species in system for each filament type defined.

`NUMPLUSENDSPECIES` – int – Plus end species in system for each filament type defined.

`NUMMINUSENDSPECIES` – int – Minus end species in system for each filament type defined.

`NUMBOUNDSPECIES` – int – Bound species in system for each filament type defined.

`NUMLINKERSPECIES` – int – Cross-linker species in system for each filament type defined.

`NUMMOTORSPECIES` – int – Motor species in system for each filament type defined.

`NUMBRANCHERSPECIES` – int – Brancher species in system for each filament type defined.

`NUMBINDINGSITES` – int – Number of binding sites per cylinder for each filament type defined. This will set binding sites for cross-linkers, motors, and other binding molecules.

`NUMMOTORHEADSMIN` – int – Minimum number of motor heads per motor species defined.

`NUMMOTORHEADSMAX` – int – Maximum number of motor heads per motor species defined.

`MOTORSTEPSIZE` – double – Single motor head step size.

`DISSIPATIONTRACKING` – ON – Switches on the dissipation tracking feature

`LINKERBINDINGSKIP` – int – Switches on the different binding tracking feature to allow motors to have more binding spots per cylinder than linkers. The specified integer is the number of binding sites that the cross-linkers will skip before accepting a possible binding site.

`EVENTTRACKING` – ON – Switches on the event tracking feature

5.1.4 Dynamic rates

The following dynamic rate forms and parameters can be set. These parameters are characteristic lengths and amplitudes of the rate changing equations outlined in [1]. These can be tuned to mimic the stall and unbinding mechanochemical coupling of cross-linkers and myosin II motors. Note that if dynamic rates are enabled, the number of dynamic rate forms for each type of reaction must match the number of species of that type specified in the `SystemFile`, i.e. the number of forms for cross-linker unbinding must match the number of cross-linker species, etc.

The number of parameters specified for each type of dynamic rate form must match the number of parameters required for those forms. See below for details, and see [1] for more information on the explicit forms. Parameters must be listed in order of the form that they correspond to, also corresponding to the species that they represent.

DFPOLYMERIZATIONTYPE – BROWRATCHET – Filament polymerization dynamic rate form.

DFPOLYMERIZATIONLEN – double – Characteristic length for filament polymerization dynamic rate form.

DLUNBINDINGTYPE – CATCHSLIP, SLIP – Cross-linker unbinding dynamic rate form. If CATCHSLIP, two parameters for DLUNBINDINGLEN and DLUNBINDINGAMP are needed to define the functional form. If SLIP, one DLUNBIDINGLEN is needed to define the functional form.

DLUNBINDINGLEN – double – Characteristic length of cross-linker unbinding dynamic rate form.

DLUNBINDINGAMP – double – Amplitude of cross-linker unbinding dynamic rate form.

DMUNBINDINGTYPE – LOWDUTYCATCHSLIP, LOWDUTYSLIP– Myosin II unbinding dynamic rate form. If LOWDUTYCATCHSLIP, two parameters for DMUNBINDINGFORCE are needed to define the functional form. If LOWDUTYSLIP, one DMUNBIDINGFORCE is needed to define the functional form.

DMUNBINDINGFORCE – double – Characteristic force of myosin II unbinding dynamic rate form.

DMWALKINGTYPE – LOWDUTYSTALL – Myosin II walking dynamic rate form.

DMWALKINGLEN – double – Characteristic force of myosin II walking dynamic rate form.

5.1.5 Starting filament configuration

The following filament initialization parameters can be set. These parameters define the initial configuration and length of filaments in the system. It is noted that at least one filament, plus end, and minus end chemical species must be initialized in the chemistry input file, or a startup error will result.

FILAMENTFILE – string – Name of filament initialization file. This is not required.

NUMFILAMENTS – int – Number of random filaments to initialize. These filaments will be randomly distributed in the system volume.

FILAMENTLENGTH – int – Number of cylinders per filament to initialize, defining the initial length of the filaments.

FILAMENTTYTPE – int – Filament type to initialize.

5.1.6 Starting bubble configuration

The following bubble initialization parameters can be set. These parameters define the initial configuration of bubbles in the system, similar to the filament configuration parameters. It is noted that at least one type of bubble must be set, or a startup error will result.

`BUBBLEFILE` – string – Name of bubble initialization file. This is not required.

`NUMBUBBLES` – int – Number of random filaments to initialize. These filaments will be randomly distributed in the system volume.

`BUBBLETYTPE` – int – Bubble type to initialize.

5.1.7 Special setup

The following special setups can be initialized with corresponding parameters. These setups must be set on different lines, so users should specify a new parameter, on separate lines, for each setup desired:

`SPECIALSETUP` – MTOC – Type of special setup.

For a microtubule organizing center (MTOC) setup, the following parameters can be defined:

`MTOCFILAMENTTYPE` – int – Type of filament to attach to the MTOC.

`MTOCNUMFILAMENTS` – int – Number of filaments to create and attach to the MTOC.

`MTOCFILAMENTLENGTH` – int – Length of filaments to attach to the MTOC, in number of cylinders.

`MTOCBUBBLETYPE` – int – Type of bubble to be the MTOC.

5.1.8 Special protocols

The following special protocols can be initialized. These protocols must be set on different lines, so users should specify a new parameter, on separate lines, for each setup desired:

`SPECIALPROTOCOL` – MAKEFILAMENTSSTATIC, MAKELINKERSSTATIC – Make either filament or cross-linker chemistry static after a certain amount of time. This parameter must be followed by a double specifying the simulation time that this protocol is activated.

SPECIALPROTOCOL TRANSFERSHAREAXIS – char– X OR Y OR Z the axis along which compartments should be activated/deactivated based on the span of actin network. Currently implemented only for cubic boundary framework.
SPECIALPROTOCOL PINBOUNDARYFILAMENTS – ⟨pinK⟩ ⟨pinDistance⟩ ⟨pinTime⟩ to tether minus and plus ends that are within pinDistance (nm) from the boundary after pinTime (s). This will result in additional forces on the actin network form the boundary.

5.1.9 Output formats

The output of MEDYAN will be directed to the OutputFile specified. The following output can be set. These outputs must be set on different lines, so users should specify a new parameter, on separate lines, for each output value desired. Output files will be explained in more detail in a later section.

OUTPUTTYPE – SNAPSHOT, FORCES, TENSIONS, BIRTHTIMES, CHEMISTRY – Output type.

5.2 Chemistry input file

The chemistry input file, whose name is specified in the SystemFile, contains the chemical configuration of the system, including species and reactions. It is noted that the order in which cross-linker, motor, and branches species are defined in the chemistry input file should match the relevant mechanical parameters, which are defined in the SystemFile. The number of species of each type should also match the SystemFile's species type numbers, or a startup error will result.

All species names given must be unique strings, or a startup error will result.

In all species and reaction definitions, FILAMENTTYPE is an integer that specifies the type of filament that this filament species belongs to. For example, if there are two filament types defined, the FILAMENTTYPE parameter could be 0 or 1. An invalid value of this parameter will result in an error.

5.2.1 Species

Different types of species can be defined as follows:

- A **diffusing species** is defined in the following form:

  ```
  SPECIESDIFFUSING: <NAME> <COPYNUMBER> <DIFFRATE> <RELEASETIME> <REMOVALTIME>
  <QUALIFIER> (<NUMEVENTS>)
  ```

  where NAME is any string defining the name of the species, COPYNUMBER is the number

of molecules of that species in the system, and DIFFUSIONRATE is a float value that determines the diffusion rate of this molecule between compartments. RELEASETIME specifies when this molecule populates the system in simulation (in seconds). REMOVALTIME specifies whether the species should be removed from the simulation. If no removal is desired, this can be set to 0.

The QUALIFIER field is used to define the type of reacting species. The options are the following:

- REG : A regular reacting species. Copy numbers are updated typically.
- AVG : An averaging reacting species. The species will use a copy number averaged over a set number of copy number changes (NUMEVENTS) for efficiency.

The NUMEVENTS field, denoted in parentheses as optional, only used in the case of defining an averaging reacting species. If using a regular, this should not be included in the file or an error will result.

- A **bulk species**, which is assumed to be spatially homogeneous, is defined in the following form:

```
SPECIESBULK: <NAME> <COPYNUMBER> <RELEASETIME> <REMOVALTIME> <QUALIFIER>
```

where NAME is any string defining the name of the species, COPYNUMBER is the number of molecules of that species in the system, and RELEASETIME specifies when this molecule populates the system in simulation (in seconds). REMOVALTIME specifies whether the species should be removed from the simulation. If no removal is desired, this can be set to 0.

The QUALIFIER field is used to define the type of reacting species. The options are the following:

- REG : A regular reacting species. Copy numbers are updated typically.
- CONST : An constant reacting species. The species will never change copy number upon reacting.

- Any **filament-related species** can be defined in the following form:

```
SPECIES<SPECIESTYPE>:  <NAME> <FILAMENTTYPE>
```

where SPECIESTYPE can be:

- FILAMENT : A filamentous species. At least one filament species must be defined if using filaments in simulation.

- `PLUSEND` : A plus end species on a filament, which is defined as the front of the filament. There must be at least one plus end species for every filament species defined in the system.
- `MINUSEND` : A minus end species on a filament, which is defined as the back of the filament. There must be at least one minus end species for every filament species defined in the system.
- `BOUND` : A bound species on a filament. There must be at least one bound species defined for each filament type.
- `LINKER` : A cross-linker species. The ordering of cross-linker initializations should match their mechanical parameters, as stated above.
- `MOTOR` : A myosin II motor species. The ordering of motor initializations should match their mechanical parameters, as stated above.
- `BRANCHER` : A branching species. The ordering of branches initializations should match their mechanical parameters, as stated above.

5.2.2 Binding sites

For every species that binds to filaments (linkers, motors, branchers), a binding site species must be set. This binding site must be a bound species on any filament type. It is declared in the following form:

```
LINKERBINDINGSITE: <NAME> <FILAMENTTYPE>
MOTORBINDINGSITE: <NAME> <FILAMENTTYPE>
BRANCHERBINDINGSITE: <NAME> <FILAMENTTYPE>
```

where `NAME` is the name of a pre-defined bound species on a filament of type `FILAMENTTYPE`.

5.2.3 Reactions

Reaction definitions must follow these common rules:

- Species that are defined in reactions must be previously defined in the chemistry file.
- For filament-related reactions, most species type and ordering parameters are fixed; if they are fixed, they will be pre-defined in the reaction definition below. If the ordering is not properly followed, a startup error will result.
- All species declarations in a reaction must be separated by white space, with `+` markers between reactants and products. A `->` must be placed between reactants and products, separated by whitespace. If this syntax is not followed, a startup error will result.

- The optional string <HRCDID> and accompanying float <DELGZERO> specify the identity and the ΔG^0 value for the reaction, respectively, for use in dissipation tracking. If this feature is turned on, then this field must be supplied, and if it is turned off then this field must be omitted. Currently only some reactions support dissipation tracking, as specified below, and it is only supported when the Next Reaction Method algorithm choice is used. For those reactions which do not support dissipation tracking, the <HRCDID> and <DELGZERO> fields should not be set. This allows those reactions to be included in the chemical system while allowing dissipation tracking for supported reactions.

Different types of reactions can be defined as follows:

- A **general reaction** between any bulk or diffusing species can be defined in the following form:

  ```
  GENREACTION: (<DELGZERO>:<HRCDID>)
  <NAME>:BULK/DIFFUSING + <NAME>:BULK/DIFFUSING + ...  ->
  <NAME>:BULK/DIFFUSING + <NAME>:BULK/DIFFUSING + ...  <RATE>
  ```

  where any bulk or diffusing species can be included, and <RATE> is a float value that determines the rate constant of the reaction.

  This reaction supports dissipation tracking, so if the feature is turned on, then <HRCDID> should be set to a user specified short unique string, and <DELGZERO> should be set to a float value based on the user's parameterization of the chemical energetics.

- A **bulk reaction** between bulk species only can be defined in the following form:

  ```
  BULKREACTION:
  <NAME>:BULK + <NAME>:BULK + ...  ->
  <NAME>:BULK + <NAME>:BULK + ...  <RATE>
  ```

  where any bulk species can be included. If the reaction only contains bulk species, it must be specified as a bulk reaction. <RATE> is a float value that determines the rate constant of the reaction.

  This reaction does not support dissipation tracking, so <HRCDID> and <DELGZERO> should not be provided.

- A **polymerization reaction** can be defined in the following form:

  ```
  POLYMERIZATIONREACTION: (<DELGZERO>:<HRCDID>) <FILAMENTTYPE>
  <NAME>:BULK/DIFFUSING + <NAME>:PLUSEND/MINUSEND ->
  <NAME>:FILAMENT + <NAME>:PLUSEND/MINUSEND <RATE>
  ```

where <NAME> is the string name of the species, and <RATE> is a float value that determines the rate constant of the reaction. It is noted that the first species listed can be either DIFFUSING or BULK, and the reaction can contain a PLUSEND or MINUSEND.

This reaction will polymerize the filament, producing a new chemical species on the end of the filament and increasing the length of the filament by a single monomer.

This reaction supports dissipation tracking, so if the feature is turned on, then <HRCDID> should be set to a user specified short unique string, and <DELGZERO> should be set to a float value based on the user's parameterization of the chemical energetics.

- A **depolymerization reaction** can be defined in the following form:

```
DEPOLYMERIZATIONREACTION: (<DELGZERO>:<HRCDID>) <FILAMENTTYPE>
<NAME>:FILAMENT + <NAME>:PLUSEND/MINUSEND ->
<NAME>:BULK/DIFFUSING + <NAME>:PLUSEND/MINUSEND <RATE>
```

where <NAME> is the string name of the species, and <RATE> is a float value that determines the rate constant of the reaction. It is noted that the third species listed can be either DIFFUSING or BULK, and the reaction can contain a PLUSEND or MINUSEND.

This reaction will depolymerize the filament, removing a chemical species from the end of the filament and decreasing the length of the filament by a single monomer.

This reaction supports dissipation tracking, so if the feature is turned on, then <HRCDID> should be set to a user specified short unique string, and <DELGZERO> should be set to a float value based on the user's parameterization of the chemical energetics.

- A **cross-linker reaction** between two filaments can be defined in the following form:

```
LINKERREACTION: (<DELGZERO>:<HRCDID>) <FILAMENTTYPE>
<NAME>:BOUND:1 + <NAME>:BOUND:2 + <NAME>:BULK/DIFFUSING <->
<NAME>:LINKER:1 + <NAME>:LINKER:2 <ONRATE> <OFFRATE> <RMIN> <RMAX>
```

where <NAME> is the string name of the species, and <ONRATE> and <OFFRATE> are float values that determines the rate constant of the binding and unbinding reactions. <RMIN> and <RMAX> are the range of the chemical reaction, and this can be set depending on the structure of the simulated cross-linker. It is noted that the third species listed can be either DIFFUSING or BULK. The bound species listed must be the corresponding cross-linker binding site.

This reaction produces cross-linker species at two separate positions on each respective filament which are chemically and mechanically connected. If mechanical force fields are defined for the cross-linkers, a potential will be created between the filaments. The unbinding reaction will remove these species from the filaments, as well as remove any linker potentials that have been created between the filaments.

This reaction supports dissipation tracking, so if the feature is turned on, then `<HRCDID>` should be set to a user specified short unique string, and `<DELGZERO>` should be set to a float value based on the user's parameterization of the chemical energetics.

- A **motor reaction** between two filaments can be defined in the following form:

```
MOTORREACTION: (<DELGZERO>:<HRCDID>) <FILAMENTTYPE>
<NAME>:BOUND:1 + <NAME>:BOUND:2 + <NAME>:BULK/DIFFUSING <->
<NAME>:MOTOR:1 + <NAME>:MOTOR:2 <ONRATE> <OFFRATE> <RMIN> <RMAX>
```

  where `<NAME>` is the string name of the species, and `<ONRATE>` and `<OFFRATE>` are float values that determines the rate constant of the binding and unbinding reactions. `<RMIN>` and `<RMAX>` are the range of the chemical reaction, and this can be set depending on the structure of the simulated motor. It is noted that the third species listed can be either `DIFFUSING` or `BULK`. The bound species listed must be the corresponding motor binding site.

  This binding reaction produces motor species at two separate positions on each respective filament which are chemically and mechanically connected. If mechanical force fields are defined for the motor, a potential will be created between the filaments. The unbinding reaction will remove these species from the filaments, as well as remove any motor potentials that have been created between the filaments.

  This reaction supports dissipation tracking, so if the feature is turned on, then `<HRCDID>` should be set to a user specified short unique string, and `<DELGZERO>` should be set to a float value based on the user's parameterization of the chemical energetics.

- A **motor walking reaction** can be defined in the following form:

```
MOTORWALKINGREACTION: (<DELGZERO>:<HRCDID>) <FILAMENTTYPE>
<NAME>:MOTOR:N/N+1 + <NAME>:BOUND:N/N+1 ->
<NAME>:MOTOR:N/N+1 + <NAME>:BOUND:N/N+1 <RATE>
```

  where `<NAME>` is the string name of the species, and `<RATE>` is a float value that determines the rate constant of the reaction. The choice of `N/N+1` will determine

whether the motor is stepping forward or backward. A motor movement from `N` to `N+1` is defined as forward movement (towards the plus end of the filament), and the opposite is backward (towards the minus end). These choices for the reactants and products must be self-consistent as well as consistent with the bound species positions chosen in the reaction, or a startup error will result. The bound species listed must be the corresponding motor binding site.

This reaction will move a motor head in the given direction.

This reaction supports dissipation tracking, so if the feature is turned on, then `<HRCDID>` should be set to a user specified short unique string, and `<DELGZERO>` should be set to a float value based on the user's parameterization of the chemical energetics.

- A **branching reaction** can be defined in the following form:

```
BRANCHINGREACTION: <FILAMENTTYPE>
<NAME>:BULK/DIFFUSING + <NAME>:BULK/DIFFUSING + <NAME>:BOUND <->
<NAME>:BRANCHER + <NAME>:PLUSEND <ONRATE> <OFFRATE> <NUCLEATIONZONE> <NUCLEATIONDIST>
```

where `<NAME>` is the string name of the species, and `<ONRATE>, <OFFRATE>` are float values that determine the rate constants of the reaction. It is noted that the first and second species listed can be either `DIFFUSING` or `BULK`. The bound species listed must be the corresponding branching binding site.

The `NUCLEATIONZONE` and `NUCLEATIONDIST` specify the volume in which a branching reaction can occur. The choices for the zone parameter are the following

  - `ALL` : A new filament can nucleate anywhere in the simulation volume due to branching.
  - `BOUNDARY` : A new filament can nucleate a given distance away from a boundary due to branching, specified by the `NUCLEATIONDIST` from the system boundary.
  - `TOPBOUNDARY` : Similar to `BOUNDARY` except only in the top half of the volume (in the z direction).

It is noted that `NUCLEATIONDIST` needs to be specified for all nucleation zones, but it is unused for `ALL`.

This reaction will create a new branching point, as well as a filament with the desired chemical plus end. If mechanical force fields are defined for the branching point, a potential will be created between the parent and child filament. The unbinding reaction will remove the branching point from the filaments, thus freeing the child filament from the parent. It will also remove any branching point potentials that have been created between the filaments.

This reaction does not support dissipation tracking, so <HRCDID> and <DELGZERO> should not be provided.

- A **nucleation reaction** can be defined in the following form:

```
NUCLEATIONREACTION: <FILAMENTTYPE>
<NAME>:BULK/DIFFUSING + <NAME>:BULK/DIFFUSING ->
<NAME>:PLUSEND + <NAME>:FILAMENT + <NAME>:MINUSEND <RATE>
```

  where <NAME> is the string name of the species, and <RATE> is a float value that determines the rate constant of the reaction. It is noted that the first and second species listed can be either DIFFUSING or BULK.

  This reaction will create a new filament with the given chemical plus end, minus end, and filament species. PLEASE REFER TO THE EXAMPLE FILES FOR A COMPLETE NUCLEATION CYCLE.

  This reaction does not support dissipation tracking, so <HRCDID> and <DELGZERO> should not be provided.

- A **destruction reaction** can be defined in the following form:

```
DESTRUCTIONREACTION: <FILAMENTTYPE>
<NAME>:PLUSEND + <NAME>:MINUSEND ->
<NAME>:BULK/DIFFUSING + <NAME>:BULK/DIFFUSING <RATE>
```

  where <NAME> is the string name of the species, and <RATE> is a float value that determines the rate constant of the reaction. It is noted that the third and fourth species listed can be either DIFFUSING or BULK.

  This reaction will destroy a filament, removing it from the system.

  This reaction does not support dissipation tracking, so <HRCDID> and <DELGZERO> should not be provided.

- An **filament aging reaction** can be defined in the following form:

```
AGINGREACTION: (<DELGZERO>:<HRCDID>) <FILAMENTTYPE>
<NAME>:FILAMENT/PLUSEND/MINUSEND ->
<NAME>:FILAMENT/PLUSEND/MINUSEND <RATE>
```

  where <NAME> is the string name of the species, and <RATE> is a float value that determines the rate constant of the reaction. Either of the reactant or product species can be FILAMENT, PLUSEND, or MINUEND, but the product and reactant species must be the same type, or a startup error will result.

This reaction will change the chemical species that resides in a filament.

This reaction supports dissipation tracking, so if the feature is turned on, then `<HRCDID>` should be set to a user specified short unique string, and `<DELGZERO>` should be set to a float value based on the user's parameterization of the chemical energetics.

- A **filament severing reaction** can be defined in the following form:

```
SEVERINGREACTION: <FILAMENTTYPE>
AT <NAME>:FILAMENT <RATE>
```

where `<NAME>` is the string name of the species, and `<RATE>` is a float value that determines the rate constant of the reaction.

This reaction will sever the filament at the closest cylinder connection to a given chemical position, producing two child filaments.

This reaction does not support dissipation tracking, so `<HRCDID>` and `<DELGZERO>` should not be provided.

5.3 Filament input file

The filament input file, whose name is specified in the `SystemFile`, contains the type and coordinates of filaments to initialize in the system at startup. The format of an initial filament declaration is as follows:

```
FILAMENT <FILAMENTTYPE> coord1x coord1y coord1z coord2x coord2y coord2z
```

where {`coord1x, coord1y, coord1z`} and {`coord2x, coord2y, coord2z`} specify the starting and ending coordinates of the filament.

5.4 Bubble input file

The bubble input file, whose name is specified in the `SystemFile`, contains a coordinate of bubbles to initialize in the system at startup. The format of an initial filament declaration is as follows:

```
BUBBLE: <BUBBLETYPE> coordx coordy coordz
```

where {`coordx, coordy, coordz`} is the coordinate of the bubble.

6 Output

MEDYAN can produce a number of output types, set in the `SystemFile`, produced at a snapshot frequency also defined in the `SystemFile`. These output files will be placed in the `OutputDirectory` specified at runtime. The output types and visualization of this output are described below.

6.1 Types of output files

6.1.1 snapshot.traj

The snapshot file gives the basic trajectory information of the system. It includes a brief description for all filaments, cross-linkers, motors, and branching points in the system, as well as information on the current chemical step. It is produced with the following form:

```
chemstepnumber time numfilaments numlinkers nummotors numbranchers
F filamentid filamenttype filamentcyllength deltal deltar
beadcoord1x beadcoord1y beadcoord1z beadcoord2x beadcoord2y beadcoord2z ...
...
L linkerid linkertype
startcoordx startcoordy startcoordz endcoordx endcoordy endcoordz
...
M motorid motortype
startcoordx startcoordy startcoordz endcoordx endcoordy endcoordz
...
B brancherid branchertype
coordx coordy coordz
```

6.1.2 plusend.traj

The plusend file gives the plus end coordinates and types information. The plus end coordinates should be the same as the last bead coordinates in the snapshot.traj for each filament. The plus end type is recorded as 0, 1, 2, etc. that follows the same order as the SPECIESPLUSEND in chemistry input files. It is produced with the following form:

```
chemstepnumber time numfilaments numlinkers nummotors numbranchers
F filamentid filamenttype filamentcyllength deltal deltar
plusendcoordx plusendcoordx plusendcoordx
PLUSEND: type
```

6.1.3 forces.traj, stresses.traj, and birthtimes.traj

The forces file gives the forces on each element in the system, in similar form to the snapshot file. It is produced with the following format:

```
chemstepnumber time numfilaments numlinkers nummotors numbranchers
F filamentid filamenttype filamentcyllength deltal deltar
bead1property bead2property ...
...
L linkerid linkertype
linkerproperty
...
M motorid motortype
motorproperty
...
B brancherid branchertype
*no property printed for branching points*
...
```

where the properties are as follows:

- forces.traj: the magnitude forces on each cylinder, as well as the magnitude of stretching force on each cross-linker and motor are printed.
- stresses.traj: the stretching stress on cylinders, cross-linkers, and motors are printed.
- birthtimes.traj: the birth time of on cylinders, cross-linkers, and motors are printed.

6.1.4 chemistry.traj

The chemistry trajectory file gives the copy numbers of all species in the system, along with the current chemical step and time. It is produced with the following form:

```
chemstepnumber time
SPECIESNAME COPYNUMBER
```

where `SPECIESNAME` represents the name of the system species and `COPYNUMBER` is the current copy number of that species at the given timestep.

6.1.5 concentration.traj

The concentration trajectory file gives the center point coordinates of compartments and the copy numbers of all diffusing species in the compartment. It is produced with the

following form:

```
chemstepnumber time
COMPARTMENT: coordx coordy coordz
SPECIESNAME COPYNUMBER
```

6.1.6 monomers.traj

The monomers trajectory files gives the number of reactions occurred since last snapshot. DeltaMinusEnd and DeltaPlusEnd shows the number of cylinder is created and destructed. (Dd)Poly refers to (de)polymerization. IfNucleation = 1 suggests that this filament is nucleated by nucleation reaction during this period. It is produced with the following form:

```
chemstepnumber time numfilaments numlinkers nummotors numbranchers
F filamentid filamenttype filamentcyllength DeltaMinusEnd DeltaPlusEnd
DeltaMinusEnd DeltaPlusEnd PolyMinusEnd PolyPlusEnd DepolyMinusEnd DepolyPlusEnd
IfNucleation TotalNumMonomers
```

6.1.7 dissipation.traj

The dissipation file gives the cumulative changes in the Gibbs free energy of the system resulting from chemical reactions and mechanical rearrangements. The difference between the values at consecutive times will approximate the dissipation rates. The distinction between `chemdiss` and `chemenergy` and between `mechdiss` and `mechenergy` is explained in accompanying material. The format of the output file is:

```
chemstepnumber time
total chemdiss mechdiss chemenergy mechenergy
```

6.1.8 HRCD.traj

The high resolution chemical dissipation file gives the cumulative changes in the Gibbs free energy of the system resulting from chemical reactions, with the contributions from each reaction specified separately. The order of the reactions is preserved from time step to time step and is set by the order of their first occurrence in the trajectory. For the change in Gibbs free energy resulting from diffusion of a diffusing species, the `HRCDDID` is given as `DIF_SPECIESNAME`. For the change in Gibbs free energy resulting from the unbinding of linkers or motors, the `HRCDDID` is given as `HRCDIDoff`, where `HRCDID` is that of the corre-

sponding binding reaction. The format of the output file is:

```
chemstepnumber time
HRCDID1 HRCDID2 ...
chemenergy1 chemenergy2 ...
```

Here the `chemenergy` fields are cumulative changes in Gibbs free energy owing to this reaction.

6.1.9 HRMD.traj

The high resolution mechanical dissipation file gives the cumulative values of ΔG_{mech} and $\Delta G_{mech,\ diss}$, with contributions from each force field specified separately. The format of the output file is:

```
chemstepnumber time
ForceField1 ForceField2 ...
mechenergy1 mechenergy2 ...
mechdiss1 mechdiss2 ...
```

Here the `mechenergy1` fields are cumulative net changes in mechanical energy owing to force field 1, and the mechdiss1 are cumulative changes in mechanical energy during energy minimization owing to force field 1. The difference between these two quantities is explained in accompanying material. Summing across the rows of the `mechenergy` and `mechdiss` lines will give the corresponding values found at that time point in the `dissipation.traj` file. When parsing this file, note that the `Excluded Volume` force field consists of two words whereas the other force fields (e.g. `Bubble`) consist of only one word.

6.1.10 CMGraph.traj

The connectivity-based mapping output gives information which can be used to construct a weighted graph, in which filaments are nodes and the weighted edges between them represent the number of cross-linkers connecting them. The format of the output file is:

```
chemstepnumber time
filid1a filid1b numlinks1 filid2a fildid2b numlinks2 ...
```

Here the `filida` and `filidb` fields are the filament identification numbers, indicating that this pair of filaments is connected by `numlinks` cross-linkers.

6.1.11 motorwalkingevents.traj, linkerbindingevents.traj, linkerunbindingevents.traj

These output files will be generated if the event tracking feature is turned on. They give detailed information on the spatiotemporal occurrences of motor walking, and cross-linker binding and unbinding. The format for these output files is:

```
event_t event_x event_y event_z
```

6.2 Visualization of output

6.2.1 VMD based visualization

Visual Molecular Dynamics (VMD) can be downloaded from `http://www.ks.uiuc.edu/Research/vmd/`. To convert snapshot.traj into VMD compatible pdb format, use the MATLAB function `Medyan2VMD.m` in `InstallDirectory/visual`. Function call in MATLAB is given as

```
> Medyan2VMD(path,outputfile)
```

- path - path to snapshot.traj.
- outputfile - desired outputfile name.

Once the PDB file is obtained, load it in VMD using `File` → `New Molecule` to choose the .pdb file and click `Load`. The trajectories are best visualized when `Graphics` → `Representations` window is used to set `Coloring Method` to `Chain` and `Drawing Method` to `Trace`. Linkers will have chain IDs l, m, n, ... etc. while Motor chain IDs will be L, M, N, ... Actin filaments have chain ID F. Trajectory can be looped through using the slider on the bottom of `VMD Main` window.

Note - Although very powerful, this method can currently handle just one kind of linker and motor.

6.2.2 MayaVi based visualization

The output described in the previous section can be visualized using a python script found in `InstallDirectory/visual`, named `AnimateTrajectory.py`. This script uses MayaVi (http://mayavi.sourceforge.net) to produce a visualization of trajectory frames, as well as an animation of an entire simulation. The following python-related dependencies for MayaVi should be installed:

- vtk5 5.10.1

- qt4 4.8.6
- ipython 2.2.0
- matplotlib 1.4.0
- pyside 1.2.2

Most of these packages are available through MacPorts or Homebrew, if using an Apple computer. The following environment variables may need to be declared, depending on your system configuration:

```
export QT_API = pyside (or qt4)
```

A helpful alias to run ipython is:

```
alias ip = "ipython --gui = qt --pylab = qt"
```

We will assume for the next section that this alias is configured.

6.2.2.1 Running the visualization script, AnimateTrajectory.py The visualization script must be edited by the user on lines 6-8 to include the desired snapshot file. A color file can also be specified, which will color the network based on either a force, stress, or birth time file. If no color file is included, the elements of the network will be colored to a default value. The script can also be edited on lines 311-353 to include titles, boundaries, scales, default colors, and choice of color map.

Run the script using the following commands:

```
> ip
> run -i AnimateTrajectory.py
```

This will load all snapshots of the trajectory files specified. To show a snapshot, execute the following commands:

```
> show_snapshot(snapshot number)
```

To show the entire simulation frame by frame, execute the following:

```
> anim()
```

7 Restart trajectories

If the simulation does not reach completion and you would like to restart it from the last snapshot, you can use MATLAB scripts found in `InstallDirectory/restart`. The following steps are needed to prepare the last snapshot for restart.

7.1 Parse the last snapshot from snapshot.traj

The following set of commands can be used in terminal to parse through snapshot.traj and extract the snapshot from which you would like to restart the simulation. Windows users are encouraged to use Notepad++ or similar text processors. In order to get the last but one snapshot, type the following in terminal.

```
> tail -10 chemistry.traj
```

Shown below each command is a sample output from a particular trajectory.
121189600 1502.71
AD:DIFFUSING 0
MD:DIFFUSING 0
LD:DIFFUSING 797
FA:FILAMENT 23940
PA:PLUSEND 30
MA:MINUSEND 30
LA:LINKER 403
MOA:MOTOR 48

```
> grep -n -E " ∧ [0 − 9] + 1502 ∗ " snapshot.traj
```

259043:121189600 1502.71252 30 403 48 0 0

```
> wc -l snapshot.traj
```

260006 snapshot.traj

```
> sed -n ‘‘259044,260006p" snapshot.traj > restartinput.txt
```

Thus the last snapshot is parsed into `restartinput.txt`.

7.2 Convert to Medyan readable format

To convert the parsed snapshot to Medyan compatible format, you will need the list of linker, motors and branchers names along with a list of filament IDs that they bind with.

The MATLAB script `InstallDirectory/restart/createinputfile.m` can be used for this purpose.

```
> createinputfile(Ifile,Ofile,L,Lf,M,Mf,B,Bf)
```

- Ifile - input file eg. 'restartinput.txt'
- Ofile - output file eg. 'restartoutput.txt'
- L - cell list of SPECIESLINKER names. eg. {'LA1','LA2'}
- Lf -list of filament IDs corresponding to L. eg. [0,1]
- M - cell list of SPECIESMOTOR names. eg. 'MOA1','MOA2'
- Mf -list of filament IDs corresponding to M. eg. [0,0]
- B - cell list of SPECIESBRANCHER names. eg. 'BA1,'BA2'
- Bf -list of filament IDs corresponding to B eg. [1,1]

7.3 Modify system input file

Edit the following lines in the system input file.

.....
FILAMENTFILE: restartoutput.txt
PROJECTIONTYPE: PREDEFINED
RESTARTPHASE
.....

You can now run the simulation to desired time. Make sure to save trajectories in a different folder.
Note - If your restart files were initialized properly, the stack of reactions listed after "Reactions fired! Displaying heap" should only consist of zeros. Any non zero number indicates an initialization failure.

7.4 Merge snapshot, chemistry output files

Once the restarted simulation completes successfully, you can concatenate the two trajectories through the following MATLAB function call. The relevant files are in `InstallDirectory/restart`.

```
> concatenatefiles(frame1,time_limit,file1dir,file2dir,outputdir)
```

- frame1 - is the last time step to consider in file1. Any frame with time stamp lesser or equal will be written. eg. 1502.37
- time_limit - desired time duration of simulation eg. 2000.
- file1dir - full path leading to directory of original simulation.
- file2dir - full path leading to directory of restarted simulation
- outputdir - output directory for concatenated files (add full path). eg. 'restartdir'

8 Developer guide

8.1 Analysis of output

The output described in the previous section can be analyzed using a python script found in `InstallDirectory/analysis`, named `AnalyzeTrajectory.py`. This script reads trajectory files in a similar manner to AnimateTrajectory.py and has a number of functions for various analysis tools related to cytoskeletal networks. For more information and examples of using these functions, please see the ExampleGuide found in `InstallDirectory/docs`.

8.2 Repository requests

Requests for pulling the MEDYAN repository should be directed to James Komianos (jkomianos@gmail.com). From there, detailed instructions on cloning the repository, and general guidelines on adding features will be given.

8.3 Coding style

MEDYAN follows standard style guides and object-oriented design principles. For details on general C++ coding style, see the Google C++ style guide at http://google-styleguide.googlecode.com/svn/trunk/cppguide.html. In general, the following best practices should be followed:

- Use camelCase for all declarations, with classes beginning with a capital letter
- Use proper tabs for all declarations, loops, control flow, etc.
- Keep code under 80 characters per line

For documentation, please use Doxygen-style commenting. See the Doxygen user guide at http://www.stack.nl/ dimitri/doxygen/ for more information.

8.4 Other information

Please direct all code inquiries to James Komianos (jkomianos@gmail.com).

9 References

[1] Popov K, Komianos J, and GA Papoian. "MEDYAN: Mechanochemical Simulations of Contraction and Polarity Alignment in Actomyosin Networks." PLoS Computational Biology 2016; 12(4): e1004877. doi:10.1371/journal.pcbi.1004877