Required information for the implementation of active filament network simulations with mass turnover and pinning field channel confinement

Benjamin A. Dalton,[1,2,3,4,5] David Oriola,[1,2,3,4,6] Franziska Decker,[1,2,3,4,7] Frank Jülicher,[2,3,4] and Jan Brugués[1,2,3,4]

[1] *Max Planck Institute of Molecular Cell Biology and Genetics, 01307, Dresden, Germany*
[2] *Max Planck Institute for the Physics of Complex Systems, 01187, Dresden, Germany*
[3] *Center for Systems Biology Dresden, 01307 Dresden, Germany*
[4] *Cluster of Excellence Physics of Life, TU Dresden, 01307 Dresden, Germany*
[5] *Department of Physics, Freie Universität Berlin, 14195, Berlin, Germany*
[6] *EMBL, Carrer del Dr. Aiguader 88, 08003, Barcelona, Spain*
[7] *ETH Zürich, D-BSSE, Mattenstraße 26, 4058, Basel, Switzerland*

(Dated: March 15, 2021)

Contents

Introduction

These directories contain code versions that are intended to generate simulations similar to those published by Dalton *et. al* (2021). Parameters, such as those pertaining to system size and numbers of components, differ from those used in the publication. Each directory contains a distinct code version that is specified for a particular task. The three code versions are:

- 01_Active_Flow: generates pinning-channel active-flow as described in Fig. 4 of the corresponding paper.
- 02_AMR: generates Active Microrheology (AMR) system with harmonically trapped probe filament and oscillating pinning field, as described in Fig. 5 of the corresponding paper.
- 03_Branching_Nucleation: simulation of two branching MT networks of opposite polarity that grow and overlap, and interact via active motors. These simulations are described in Fig. 6 of the corresponding paper.

Typical run times required to reach a steady-state and subsequently accumulate a minimal amount of data suitable for analysis are of the order of 1-3 hours on a standard desktop computer. The code is serial, without any parallelization features. Shorter simulations can be used for the sake of visualizing the early transient developmental phase of each system.

Each directory contains a subdirectory labelled *analysis_scripts*. Here one finds some MATLAB scripts to generate movies and plots, and perform some elementary analysis. Note that for appropriate statistics, one typically needs long simulations (at least 10 hours on a standard computer) or some form of ensemble averaging. The included analysis and visualization scripts are discussed below.

Included files and analysis scripts

The simulation software is written in C++. Each directory contains a *main.cpp* file, a collection of *.cpp* and *.h* files, a *Makefile* for compiling, and a subdirectory with analysis scripts. The current version does not include a parameter input file. The header of the *main.cpp* file indicates the *.cpp* and *.h* files that should be contained in a given directory.

The corresponding analysis scripts for each code version are:

01_Active_Flow

- network_movie.m: generate an mpg-4 movie (*network_flow.mp4*) showing filaments and cross-links in a pinning field confinement channel.
- speckle_movie.m: generate an mpg-4 movie (*speckle_flow.mp4*) showing just filaments minus-end located speckles driven along pinning channel.
- velocity_profile.m: generates *Velocity_Profile.jpeg*, displaying actively driven velocity profiles generated along pinning channel. Only filaments minus-end are used, so long simulations are required for reasonable statistics.
- Number_Reader.m: generates *Numbers_vs_time.jpeg* to monitor the number of filaments and cross-linkers as a function of time.

02_AMR

- AMR_network_movie.m: generate an AMR mpg-4 movie (*AMR_network.mp4*) showing full filament and cross-linker network, linking the pinning field to a permanent harmonically trapped probe filament.
- AMR_speckle_movie.m: generate an mpg-4 movie (*AMR_speckle.mp4*) showing just filaments minus-end located speckles driven along pinning channel. Also showing the trapped filament and the pinning field.
- AMR_Signal_Reader: generates *AMR_Signal.jpeg* to compare the trajectory of the oscillating pinning field with the trajectory of the trapped probe filament. The signal propagation depends on the cross-linker concentration and the filament density.

03_Branching_Nucleation

- branching_networks.m: generate an mpg-4 movie (*branching_networks.mp4*) to visualize the interaction of two branching filament networks in a confinement channel.

Compiling and running software

The code can be compiled with most standard C++ compilers. The code was developed by compiling with the GCC g++ compiler, complete with the C++14 standard compiler flag. Ensure that the *Makefile* is modified to contain the relevant information for your compiler specifics. Assuming one has an appropriate compiler, the simplest way to compile and run the code is from the command line. From within the version directory, run the following commands:

```
make clean
make
./execute_dynamics
```

Since the current version does not contain an independent parameter input file, the program must be recompiled after each change to *.cpp* or *.h* files. This is done by executing the *make* command, without the need to run *make clean*.

Output files and output file formats

All output files are tab-delimited *.txt* files. Column headers are not included since typically the number of output elements varies with each time step. All output files are printed directly to the current working directory. Here we describe the output file formats for the essential output files. The necessary variables are labelled as:

- r_{ix}, r_{iy}, r_{iz}: the filament c.o.m position vector components of the i^{th} filament ($i = 1, 2, 3, ..., N_{\text{fil}}$, for a total of N_{fil} at a given time)

$r^k_{cl,ix}$, $r^k_{cl,iy}$, $r^k_{cl,iz}$: cross-linker position vector for binding domain $k = 1, 2$.

- u_{ix}, u_{iy}, u_{iz}: the orientation unit vector of the i^{th} filament

- L_i: the length of the i^{th} filament
- $N_{\text{fil}}(t)$: instantaneous number of filaments (also just N_{f})
- $N_{\text{tot\_all}}$: total number of all filaments generated through whole simulation
- fil_{id}: labels a filament in order of when it was created
- $N_{\text{cl}}(t)$: instantaneous number of cross-linkers (also just N_{cl})
- $N_{\text{cl}-1}$: instantaneous number of cross-linkers in state-1
- $N_{\text{cl}-2}$: instantaneous number of cross-linkers in state-2
- CL_{s}: cross-linker state (1 or 2)
- CL_{t}: type of cross-linker (can only be type 1 in the current version)
- j: simulation cycle count (distinct from the time of at each cycle)
- t_j: time of j^{th} cycle ($t_j = j * dt$)

Relevant output file formats are:

trajectory.txt: main filament trajectory file

j, N_{f}, L_1, r_{1x}, r_{1y}, r_{1z}, u_{1x}, u_{1y}, u_{1z}, ... , $L_{N_{\text{f}}}$, $r_{N_{\text{f}}x}$, $r_{N_{\text{f}}y}$, $r_{N_{\text{f}}z}$, $u_{N_{\text{f}}x}$, $u_{N_{\text{f}}y}$, $u_{N_{\text{f}}z}$

trajectory_id.txt: the filament trajectory with filament labels that tell the order in which a given MT was nucleated. The IDs are needed to link the text file entries through time, used to calculate the velocity profiles for example. The trajectory_id does not start at t=0. A delay time allows that the system can reach a steady state before collecting data. Thus trajectory_id is typically used for steady-state analysis. Pinning filaments are neglected. Note also that here the position vector is for the filament minus end, rather than the c.o.m.

t_j, N_{f}, $N_{\text{tot\_all}}$, $fil_{\text{id},1}$, r_{1x}, r_{1y}, r_{1z}, ... , $fil_{\text{id},\text{N}_\text{f}}$, $r_{N_{\text{f}}x}$, $r_{N_{\text{f}}y}$, $r_{N_{\text{f}}z}$

orientation_id.txt: filament orientation trajectory with filament labels. Otherwise, same format as trajectory_id

t_j, N_{f}, $N_{\text{tot\_all}}$, $fil_{\text{id},1}$, u_{1x}, u_{1y}, u_{1z}, ... , $fil_{\text{id},\text{N}_\text{f}}$, $u_{N_{\text{f}}x}$, $u_{N_{\text{f}}y}$, $u_{N_{\text{f}}z}$

trajectory_cl.txt: main cross-linker trajectory file. Note that when $CL_{1\text{s}} = 1$, then there is only one binding domain, and so there will only be a single position vector corresponding to that cross-linker.

j, $N_{\text{cl}-1}$, $N_{\text{cl}-2}$, $CL_{1\text{s}}$, $CL_{1\text{t}}$, $r^1_{cl,1x}$, $r^1_{cl,1y}$, $r^1_{cl,1z}$, $r^2_{cl,1x}$, $r^2_{cl,1y}$, $r^2_{cl,1z}$, ... , $CL_{\text{N}_\text{cl}\text{s}}$, $CL_{\text{N}_\text{cl}\text{t}}$, $r^1_{cl,N_{\text{cl}}x}$, $r^1_{cl,N_{\text{cl}}y}$, $r^1_{cl,N_{\text{cl}}z}$, $r^2_{cl,N_{\text{cl}}x}$, $r^2_{cl,N_{\text{cl}}y}$, $r^2_{cl,N_{\text{cl}}z}$

Glossary of parameters

All parameters are found in the *main.cpp* file. Since there is no independent input file, any change to parameters will require you to recompile the program, which is done by simply running the *make* command. Note that all simulation parameters are expressing standard units, so no conversion of reduced units is required.

Compiler directives

- N_DIM: number of spatial dimensions (only option N_DIM=3 for this version)
- BC_TYPE: boundary condition type (0,1,2,3 explained above)
- CL: include cross-linkers/motors (0=do not include, 1= include)
- PIN_FIELD: include pinning field (0=do not include, 1= include)
- BRANCH_TYPE: include branching nucleation (0=do not include, 1= include)

Time related

- T_STEP: total number of time steps for simulation run/number of time increments
- dt: real-time increment in seconds (best not to exceed 5e^{-6} s)
- t_write: write frequency to trajectory.txt and trajectory_cl.txt
- t_print: write frequency to standard output
- t_write_2: write frequency to trajectory_id.txt and orientation_id.txt

- start_write: time step to begin writing delayed output files (_id.txt files)

Microtubule parameters

- N_FILS: number of filaments in initial condition (pinning filaments no included)
- init_length: length of initial condition filaments (pinning filaments no included)
- nuc_fil_min: length of newly nucleated filaments
- k_n: filament bulk nucleation rate
- k_c: filament catastrophe rate
- v_g: filament polymerization rate
- v_s: filament depolymerization rate
- d: filament diameter
- sig_shift: Lennard-Jones parameter σ
- beta: parameter for Lennard-Jones to soften JL core

Motors/cross-linkers

- N_CLS: total number of available cross-linkers/motors
- motor_vel1: unloaded walking velocity for motors (in ms^{-1})
- k_0_3D: CL binding rate from unbound to single bound state ($\mathrm{m}^{-1}\mathrm{s}^{-1}$)
- k_off: CL unbinding rate from single-bound state to unbound
- k_spr: CL spring constant
- F_stall: CL stall force
- sig_cl: pair-binding search radius cutoff
- cl_delta: related to the binding energy barrier
- eta_sites: concentration of binding site, used for numeric integration of rates over filament length

Pinning field

- N_PIN: number of pinning field filaments
- z_stack: number of filaments stacked in z to make periodic arrays (ordering x and z)
- y_cryst: location of pin c.o.m in y-dimension (along channel length)
- cryst_length: length of pin filaments
- osc_start: time to begin oscillating the pinning field for AMR simulations
- osc_period: the period of oscillation (expressed in per hour)
- osc_amp: amplitude of oscillation
- trap_eqm_y: equilibrium position of AMP harmonic trap
- k_trap: AMP harmonic trap stiffness

Branching nucleation

- N_BRANCH: total number of available branching nucleators
- k_b_pul: branching rate per unit length of filament
- k_b_detach: branching detachment rate (currently not in use)
- k_b_react: branching reactivation rate (currently not in use)
- force_stable: impose that the population of nucleators is split between the two networks
- br_nuc_upper_y: nucleation domain lower limit
- br_nuc_lower_y: nucleation domain upper limit
- delta_eps: the separation between two nucleator connection tethers
- k_spr_bra: spring stiffness of branching connection tethers

Simulation control parameters

- xz_nuc_frac: avoid walls when nucleating new filaments
- y_nuc_frac: modify bulk nucleation domain
- rand_or_switch: control new filament orientation
- rand_orient: control new filament orientation
- uvec_stab: impose depolymerization of poorly aligned filaments

- init_cutoff: avoid initial state filaments being too close

Neighbour-list related

- n_list_update: update frequency for inner n-list, using the outer list members
- n_list_update_outer: update frequency for outer n-list, using all filaments
- outer_nlist_cut: cut-off for larger radius n-list
- inner_nlist_cut: cut-off for smaller radius n-list
- wall_update: update wall n-lists
- cl_neigh_update: update CL n-lists for state-1 CLs

Kinetic sampling times

- sample_inc_kn: bulk nucleation event sample interval
- sample_inc_kcat: catastrophe event sample interval
- sample_inc_on1: unbound-to-state-1 binding event sample interval
- sample_inc_off1: state-1-to-unbound unbinding event sample interval
- sample_inc_pair_on: state-1-to-state-2 binding event sample interval
- sample_inc_pair_off: state-2-to-state-1 unbinding event sample interval
- sample_inc_branch: branching nucleation event sample interval
- t_depoly: interval to check filaments for depolymerization states

Wall/BC and LJ related

- L_x, L_y, L_z: simulation box dimensions - locations for walls (m)
- sig_shift_wall: LJ-type wall parameter
- beta_wall: wall JL softening parameter
- wall_cut: wall n-list cut-off distance

Physical parameters

- k_B: Boltzmann constant
- T: system temperature
- eta: solvent viscosity

Contact details

For more information, please contact:

- dalton@zedat.fu-berlin.de
- brugues@mpi-cbg.de