The random walker's toolbox for analyzing single-particle tracking data – Supplement

Florian Rehfeldt and Matthias Weiss

Experimental Physics I, University of Bayreuth, Universitätsstr. 30, D-95447 Bayreuth, Germany

I. LIST OF ROUTINES IN THE TOOLBOX

Matlab routines and data files are provided on GitHub (*https://github.com/mweisslab/sptanalysis*). For consistency, all Matlab routines should be moved into a subfolder `routines`, data files should be moved into a subfolder `routines/data`). Routines have been tested extensively with Matlab R2018b on MacOSX, R2020b on Linux, and with the comparable open-source clone Octave 6.2.0 on Linux. Full functionality was seen for both Matlab versions. For Octave, the function `wfbm` is not available, i.e. the creation of fractional Brownian motion tracks will require an auxiliary function. Also, in Octave the subroutines `plotter` and `oplot` in the program → `driver.m` (with which all figures have been prepared) will have to be placed before the rest of the code (instead at the end), and graphics handles for changing fonts may have to be amended. Generating histograms will require a replacement of the command `histcounts` to the more basic command `histc`. Besides these points, all analysis routines seemed to work also in Octave.
For working with the evaluation routines, the program → `driver.m` may serve as an initial template for designing the data-analysis workflow. This master program not only provides examples of how to call the individual evaluation routines but also how to create and store simulated trajectories with different properties by distinct data production routines (these parts are commented by a % sign and will have to be uncommented before use). Altogether, the following routines define the toolbox:

make_rndwalk.m `pos = make_rndwalk(N,alpha,dx)`

`N`	length of trajectories N
`alpha`	twofold Hurst coefficient
`dx`	mean step length (per dimension)
`pos`	$N \times 2$-array of positions

This routine creates a two-dimensional FBM trajectory of length N with mean step size dx and Hurst coefficient $H = \alpha/2$.

make_switchwalk.m `[sx,sy] = make_switchwalk(dt,N,M,xx,yy,k1,k2,fact)`

`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx,yy`	array of x-,y-coordinates of FBM trajectories
`k1`	switch rate to low mobility
`k2`	switch rate to high mobility
`fact`	ratio of diffusion coefficients at high & low mobility ($f_K = K_{\mathrm{high}}/K_{\mathrm{low}} \geq 1$)
`sx,sy`	array of x-,y-coordinates of intermittent FBM trajectories

Based on an ensemble of FBM trajectories, this routine creates intermittent FBM trajectories that switch between a high- and a low-mobility state (ratio $f_K = K_{\mathrm{high}}/K_{\mathrm{low}} \geq 1$) with rates k_1 and k_2, while maintaining the Hurst coefficient $H = \alpha/2$.

make_blurwalk.m `pos = make_blurwalk(N,alpha,dx,np)`

`N`	length of trajectories N
`alpha`	twofold Hurst coefficient
`dx`	mean step length (per dimension)
`np`	number of photons per substep (resonable range: $n_p \in [10, 10^3]$)
`pos`	$N \times 2$-array of positions

This routine creates a two-dimensional FBM trajectory of length N with mean step size dx and Hurst coefficient $H = \alpha/2$ including static and dynamic localization errors (tuned by the specified number of photons n_p).

ta_msd.m `[tau,msdt] = ta_msd(dt,xx,yy,dim,dis)`

`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for TA-MSD calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`msdt`	array of TA-MSD values

This routine calculates the TA-MSD of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

ta_msd_nonoverlap.m `[tau,msdt] = ta_msd_nonoverlap(dt,xx,yy,dim,dis)`

`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for TA-MSD calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`msdt`	array of TA-MSD values

Similar to ta_msd.m but without sliding-window average.

ta_mom.m `[tau,mom] = ta_mom(dt,xx,yy,q,dim,dis)`

`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`q`	order of the moment to be calculated
`dim`	dimension(s) for TA k-th moment calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`mom`	array of TA q-th moment values

This routine calculates the TA q-th moment of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

ea_msd.m	`[tau,msde] = ea_msd(dt,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for EA-MSD calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`msde`	array of EA-MSD values

This routine calculates the EA-MSD of an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

eata_msd.m	`[tau,msdte] = eata_msd(dt,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for EA-TA-MSD calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`msdte`	array of EA-TA-MSD values

This routine calculates the EA-TA-MSD of an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

passage_time.m	`[tau] = passage_time(dt,xx,yy,r,dim)`
`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`r`	escape radius
`dim`	dimension(s) for TA k-th moment calculation (0=xy, 1=x, 2=y)
`tau`	time at which the trajectory overcomes the escape radius

This routine calculates the time at which the trajectory overcomes the escape radius in two dimensions (xy) or along an individual coordinate (x or y).

eb.m	`[tau,E] = eb(dt,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for EB calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispacing on linear/logarithmic scale
`tau`	array of lag times
`E`	array of ergodicity breaking parameters

This routine calculates the ergodicity breaking parameter E of an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

get_increm.m	`[dx,dy] = get_increm(dn,xx,yy,chi)`
`dn`	lag in units of frame time, i.e. $\delta t/\Delta t$
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`chi`	true/false: normalized step increments (y/n)
`dx`	steps taken within `dn` frames in x
`dy`	steps taken within `dn` frames in y

This routine calculates the steps δx and δy taken within a period $\delta t = n\Delta t$ in a single trajectory, normalization is optional.

ta_quad.m	`[tau,quat] = ta_quad(dt,xx,yy,dim,dis)`
`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for TA 4th-moment calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`quat`	array of TA 4th-moment values

This routine calculates the TA 4th-moment of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

ta_gaussianity.m	`[tau,g] = ta_gaussianity(dt,xx,yy,dim,dis)`
`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for TA-Gaussianity calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`g`	array of TA-Gaussianity values

This routine calculates the TA-Gaussianity of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

eata_gaussianity.m	`[tau,g] = eata_gaussianity(dt,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for EA-TA-Gaussianity calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`g`	array of EA-TA-Gaussianity values

This routine calculates the EA-TA-Gaussianity of an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

acf_sqinc.m	`[tau,acf] = acf_sqinc(dt,dn,xx,yy,dim,dis)`
`dt`	frame time Δt
`dn`	frame lag for steps, given by $\delta t/\Delta t$
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for correlator calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`acf`	autocorrelation of squared increments

This routine calculates the autocorrelation of fluctuations of squared increments of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

ea_acf_sqinc.m	`[tau,acfe] = ea_acf_sqinc(dt,dn,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`dn`	frame lag for steps, given by $\delta t/\Delta t$
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for correlator calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`tau`	array of lag times
`acfe`	ensemble-averaged autocorrelation of squared increments

This routine calculates the ensemble-averaged autocorrelation of fluctuations of squared increments in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

lch.m	`[tau,Sd] = lch(dt,dn,xx,yy)`
`dt`	frame time Δt
`dn`	number of positions to be used for local convex hull, typically 3-10
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`tau`	array of lag times
`Sd`	normalized maximum diameter of local convex hull, S_d

This routine calculates the maximum diameter of the LCH (based on dn points) of a single trajectory as a function of time. Values are normalized to the mean within the trajectory.

vacf.m	`[xi,vacf] = vacf(dt,dn,xx,yy,dim,dis)`
`dt`	frame time Δt
`dn`	period for velocity, given by $\delta t/\Delta t$
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for VACF calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`xi`	array of rescaled time $\xi = \tau/\delta t$
`vacf`	array of VACF values

This routine calculates the VACF of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

ea_vacf.m	`[xi,vacfte] = ea_vacf(dt,dn,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`dn`	period for velocity, given by $\delta t/\Delta t$
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for EA-VACF calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`xi`	array of rescaled time $\xi = \tau/\delta t$
`vacfte`	array of EA-VACF values

This routine calculates the EA-VACF of an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of lag times on a linear or logarithmic scale.

vacf_fbm_theo.m	`[xi,vacf] = vacf_fbm_theo(alpha)`
`alpha`	scaling exponent (twofold Hurst coefficient)
`xi`	array of rescaled time $\xi = \tau/\delta t$
`vacf`	array of VACF values for FBM

This routine calculates the FBM prediction for the VACF.

psd.m	`[f,psdt] = psd(dt,xx,yy,dim,dis)`
`dt`	frame time Δt
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`dim`	dimension(s) for PSD calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`f`	array of frequencies
`psdt`	array of PSD values

This routine calculates the PSD of a single trajectory in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of frequencies on a linear or logarithmic scale.

ea_vacf.m	`[f,psde] = ea_psd(dt,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for EA-PSD calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`f`	array of frequencies
`psde`	array of EA-PSD values

This routine calculates the EA-PSD of an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of frequencies on a linear or logarithmic scale.

cov_gamma.m	`[fT,gam] = cov_gamma(dt,N,M,xx,yy,dim,dis)`
`dt`	frame time Δt
`N`	length of trajectories N
`M`	trajectory ensemble size M
`xx`	array of x-coordinates of trajectories
`yy`	array of y-coordinates of trajectories
`dim`	dimension(s) for calculation (0=xy, 1=x, 2=y)
`dis`	'lin'/'log': lag times equispaced on linear/logarithmic scale
`fT`	array of frequencies times total time, fT
`gam`	array of coeff. of variation values γ

This routine calculates the coefficient of variation γ of PSDs from an ensemble of M trajectories in two dimensions (xy) or along an individual coordinate (x or y), with an equidistant spacing of dimensionless frequencies fT on a linear or logarithmic scale.

straightness.m	`S = straightness(xx,yy,lb,ub)`
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`lb`	position at which straightness calculation starts
`ub`	position at which straightness calculation ends
`S`	straightness

This routine calculates the straightness of a trajectory between the specified time points.

asphericity.m	`[An,Ad] = asphericity(xx,yy)`
`xx`	array of x-coordinates of single trajectory
`yy`	array of y-coordinates of single trajectory
`An`	nominator of asphericity A_s
`Ad`	denominator of asphericity A_s

This routine calculates nominator and denominator of the asphericity A_s of a single two-dimensional trajectory.

vacf_fbm_err.m	`[xi,vacf] = vacf_fbm_err(dn,theta,alpha)`
`dn`	period for velocity, given by $\delta t/\Delta t$
`theta`	constant for static localization offset
`alpha`	scaling exponent (twofold Hurst coefficient)
`xi`	array of rescaled time $\xi = \tau/\delta t$
`vacf`	array of VACF values for FBM with localization errors

This routine calculates the FBM prediction for the VACF with localization errors.

II. LIST OF DATA FILES

For test purposes, several ensembles of two-dimensional fractional Brownian motion (FBM) based trajectories and an experimental data set are included in subfolder `routines/data`. All numerically obtained data sets, produced with the master program → `driver.m`, consist of $M = 100$ trajectories, each with a length of $N = 500$ positions, using a time increment $\Delta t = 0.1$ s, and a (basic) average step size $\Delta x = 0.01$ μm.

data set 1

FBM tracks (created with `make_rndwalk.m`).

1. `FBM_xx_a_0.6.dat` & `FBM_yy_a_0.6.dat` ($\alpha = 2H = 0.6$, subdiffusion)
2. `FBM_xx_a_1.0.dat` & `FBM_yy_a_1.0.dat` ($\alpha = 2H = 1.0$, normal diffusion)
3. `FBM_xx_a_1.4.dat` & `FBM_yy_a_1.4.dat` ($\alpha = 2H = 1.4$, superdiffusion)

data set 2

Intermittent FBM tracks (created with `make_switchwalk.m`).

1. `FBM_switch_xx_a_0.6.dat` & `FBM_switch_yy_a_0.6.dat`
   ($\alpha = 2H = 0.6$, subdiffusion)
   created from data set (1a) with $k_1 = 0.4$/s, $k_2 = 0.1$/s, and $f_K = 2$
2. `FBM_switch_xx_a_1.0.dat` & `FBM_switch_yy_a_1.0.dat`
   ($\alpha = 2H = 1.0$, normal diffusion)
   created from data set (1b) with $k_1 = 0.2$/s, $k_2 = 0.3$/s, and $f_K = 3$

data set 3

FBM tracks with static localization error (created with `make_blurwalk.m`, $n_p = 50$).

1. `FBM_xx_a_0.6_np_50.dat` & `FBM_yy_a_0.6_np_50.dat` ($\alpha = 2H = 0.6$, subdiffusion)
2. `FBM_xx_a_1.0_np_50.dat` & `FBM_yy_a_1.0_np_50.dat` ($\alpha = 2H = 1.0$, normal diffusion)
3. `FBM_xx_a_1.4_np_50.dat` & `FBM_yy_a_1.4_np_50.dat` ($\alpha = 2H = 1.4$, superdiffusion)

data set 4

FBM tracks with dynamic localization error (created with `make_blurwalk.m`, $n_p = 900$).

1. `FBM_xx_a_0.6_np_900.dat` & `FBM_yy_a_0.6_np_900.dat` ($\alpha = 2H = 0.6$, subdiffusion)
2. `FBM_xx_a_1.0_np_900.dat` & `FBM_yy_a_1.0_np_900.dat` ($\alpha = 2H = 1.0$, normal diffusion)
3. `FBM_xx_a_1.4_np_900.dat` & `FBM_yy_a_1.4_np_900.dat` ($\alpha = 2H = 1.4$, superdiffusion)

data set 5

Ensemble of $M = 100$ experimentally obtained two-dimensional tracks of telomeres in the nucleus of untreated mammlian cells, each having a length $N = 2000$ and a frame time $\Delta t = 0.125$ s (part of the data set that has been analyzed and discussed in detail in (Krapf *et al.*, 2019; Stadler and Weiss, 2017)). Data have been shown to exhibit a transient FBM-like subdiffusion with $\alpha = 2H \approx 0.5$, followed by normal diffusion on larger time scales (Stadler and Weiss, 2017), in line with the motion of monomers in a Rouse polymer (for which such a subdiffusion is expected below the Rouse time). In addition, an excellent agreement of these data with key predictions for the power-spectral density of FBM trajectories has been found (Krapf *et al.*, 2019).

1. `telomers_xx.dat` & `telomers_yy.dat`

References

Krapf, D., N. Lukat, E. Marinari, R. Metzler, G. Oshanin, C. Selhuber-Unkel, A. Squarcini, L. Stadler, M. Weiss, and X. Xu, 2019, Phys Rev X **9**, 011019.
Stadler, L., and M. Weiss, 2017, New J. Phys. **19**, 113048.