A theoretical analysis of the integral fluctuation theorem for accelerated colloidal systems in the long-time limit

Supplementary Information

Simulation codes:

1) *Mean-square displacement of a Brownian particle confined in a quartic potential well*: (for **Fig. 1**)

```
%% Langevin simulation (without interia) of a particle trapped within a
quartic potential well
% Note: Here, an underdamped motion of the particle has been considered (for
a relatively small gamma).

%% Problem setup
% Theoretical background (Ref.:
%
https://www.researchgate.net/publication/329413894_Langevin_equation_for_a_pa
rticle_in_magnetic_field_is_inconsistent_with_equilibrium)
% Section 1
% The Langevin equation for a particle trapped within a potential V (an
unidimensional potential of some form) at a finite temperature T is given by:
% dx/dt = (-1/gamma)*dV/dx + sqrt(2*D)*w(t)......(1)
% Note that: Here, D is the diffusion constant given by: D=k*T/gamma
% (gamma being the friction constant (depends on the radius of the
% particle, its velocity v relative to the fluid and the coefficient of
% viscosity of the fluid)
% Taking this into account, eq.(1) assumes the following form:
% dx/dt = (-1/gamma)*dV/dx + sqrt((2*k*T)/gamma)*w(t)
% Here, w(t) is the Gaussian random noise (characterised by the random
% forces acting on the system)
% Using eq.(1), one can obtain the Langevin equation for a particle trapped
% within a quartic potential well (V(x)=alpha*x^4) as follows:
% dx/dt = -(4*alpha*x^3)/gamma + sqrt((2*k*T)/gamma)*w(t)....(2)
% Finite-difference equation approach for solving the above ODE:
% Note that, dx/dt can be recasted (by first principles) into a more
% conventional form as: dx/dt = (x(i)-x(i-1))/del(t) (where del(t) happens
% to be the time step (initially set) over which the simulation is expected
% to run
% Also, the Gaussian random noise w(t) can be expressed as:
% w(t)=w(i)/sqrt(Dt)....(3), where w(i) happens to be a sequence of Gaussian
% random
% numbers lying between 0 and 1 (zero mean and unit variance)

%% Defining parameters for the problem
N_p= 1;  % Number of particles in the system
kT= 0.772;  % The value of kT is such that it is equal to 0.998
alpha = 0.967;  % Stiffness constant
gamma= 0.1; % Friction constant
D= kT/gamma;  % Einstein diffusion constant
Nt= 0.2*1.0e+5;  % Number of samples picked/# of iterations
```

```
Dt= 1.0e-3;  % Time step for the problem

%% Initialization
x= zeros(N_p, Nt);  % Vector containing the positions of the particle at all
times
x(1)= 0; % Particle starts at the origin

%% Numerical computations (using eq(2) and eq(3))

for i=2:Nt
x(i) = x(i-1) - 4*alpha*(x(i-1))^3*Dt/gamma;

% Adding a random Gaussian white noise to the system
x(i) = x(i) + sqrt(2*D*Dt)*randn(N_p, 1);  % Note: randn() generates a
sequence of random Gaussian numbers ranging from 0 to 1

end

t= (0:Dt:(Nt-1)*Dt);  % Values of t over which the simulation is expected to
run

%% Visualization
figure(1);
hold on
plot(t, x, '-bo');
title('Position plot of the particle vs. time', 'FontSize', 15);
xlabel('$Time [s]$', 'Interpreter', 'latex', 'FontSize', 16);
ylabel('$x(t)$', 'Interpreter', 'latex', 'FontSize', 16);
hold off

%% Computation of the mean-squared displacements of the particle
[MSD, t_vec] = LangevinFunction(x, Dt);

figure(2);
plot(t_vec, MSD, 'b', 'linewidth', 2);
grid on
grid minor
xlabel('Time [s] (sampled)', 'FontSize', 20);
ylabel('$\langle(x(t))^2 \rangle$', 'Interpreter', 'latex', 'FontSize', 20,
'Color', 'k');

function [MSD,t_array] = LangevinFunction(x, Dt)

% Numerical computations

for n = 0:1:round(sqrt(length(x))) + 100
MSD(n+1) = mean((x(n+1:end)-x(1:end-n)).^2);
end

t_array = Dt*(0:1:length(MSD)-1);

end
```

2) *Harmonic potential well*: (for **Fig. 2**, **Fig. 3** and **Fig. 4**)

```
%% Brownian motion of a particle trapped within a moving optical trap
(translating linearly with a constant acceleration)

%% Problem setup
% Theoretical background (Ref. 1:
%
https://www.researchgate.net/publication/329413894_Langevin_equation_for_a_pa
rticle_in_magnetic_field_is_inconsistent_with_equilibrium)
% Section 1
% The Langevin equation for a particle trapped within a potential V (an
unidimensional potential of some form) at a finite temperature T is given by:
% dx/dt = (-1/gamma)*dV/dx + sqrt(2*D)*w(t)......(1)
% Note that: Here, D is the diffusion constant given by: D=k*T/gamma
% (gamma being the friction constant (depends on the radius of the
% particle, its velocity v relative to the fluid and the coefficient of
% viscosity of the fluid)
% Taking this into account, eq.(1) assumes the following form:
% dx/dt = (-1/gamma)*dV/dx + sqrt((2*k*T)/gamma)*w(t)
% Here, w(t) is the Gaussian random noise (characterised by the random
% forces acting on the system)
% Using eq.(1), one can obtain the Langevin equation for a particle trapped
% within a parabolic potential well (V(x)=0.5*alpha*x^2) as follows:
% dx/dt = -(alpha*x)/gamma + sqrt((2*k*T)/gamma)*w(t)....(2)
% Note: The same holds for the other 2 dimensions
% Finite-difference equation approach for solving the above ODE:
% Note that, dx/dt can be recasted (by first principles) into a more
% conventional form as: dx/dt = (x(i)-x(i-1))/del(t) (where del(t) happens
% to be the time step (initially set) over which the simulation is expected
% to run
% Also, the Gaussian random noise w(t) can be expressed as:
% w(t)=w(i)/sqrt(Dt)....(3), where w(i) happens to be a sequence of Gaussian
% random
% numbers lying between 0 and 1 (zero mean and unit variance)
% Optical trap (Ref. 2:
%
https://www.researchgate.net/publication/11238596_Experimental_Demonstration_
of_Violations_of_the_Second_Law_of_Thermodynamics_for_Small_Systems_and_Short
_Time_Scales?enrichId=rgreq-635562a6c8ab7784ee764346ab3ecd50-
XXX&enrichSource=Y292ZXJQYWdlOzExMjM4NTk2O0FTOjEwNDU1NTA3NzQzOTUwNEAxNDAxOTM5
MjgyMjE0&el=1_x_3&_esc=publicationCoverPdf)
% For an optical trap, the particle trapped within it will experience a
% linear restoring force (to a good approximation) in the neighborhood of the
trap center, which tranalates to
% saying that the particle is trapped within a harmonic potential. This is
% similar to the case of a Brownian particle trapped within a parabolic
% potential well

%% Defining parameters
N_p = 1;  % The trajectory of one Brownian particle is being simulated
alpha = 3.87*1.0e-7;   % The trapping constant of the optical trap (in N/m)
eta = 2.5; % Viscosity of the fluid (in poise)
k = 1.38*1.0e-23;  % Boltzmann's constant
```

```
R = 9.23*1.0e-6;  % Radius of the particle (in m)
gamma = 6*pi*eta*R;  % Friction constant for the medium
T = 400;  % Temperature of the fluid (in K)
D = (k*T)/gamma;  % Diffusion constant for the medium
Nt = 0.2*1.0e+5;  % Number of samples picked/# of iterations considered
a = 4*1.0e-7;  % The trap is translating at 2.5 micrometers/sec

%% Initialization
Dt = 1.0e-3;   % Time step for the problem
x = zeros(N_p, Nt); % Creating a storage vector for the positions of the
particle along the particle's trajectory
x(1) = 8.5*1.0e-7;  % The particle starts at 8.5 micrometers relative to the
position of the trap center in each simulated trajectory
qt = zeros(N_p, Nt);  % Implementing a vector that contains the positions of
the trap center relative to the bottom the sample cell
qt(1) = 3.5*1.0e-7;  % The trap center's initial position relative to the
bottom of the sample cell
t_vec = zeros(N_p, Nt); % Values of t over which the trap center's position
will be varied (corresponding to multiple simulated trajectories)
t_vec(1) = 0; % Observation time starts at t=0 (after the stage starts
translating at a constant velocity)

%% Numerical computations

for i= 2:Nt

    t_vec(i) = t_vec(i-1) + Dt;  % Updating the time instant at each
iteration
    qt(i) = qt(1) + 0.5*a*(t_vec(i-1))^2;  % Updating the position of the
trap center
    % From eq. (2) and eq. (3)
    x(i) = x(i-1) - alpha*Dt*((x(i-1) - qt(i-1)))/gamma;  % Gets appended
with the position of the trap center at each iteration
    % Note: Here, the position of the trap center as a function of time is:
    % q(t) = q0 + v*t (such that t lies within the time vector:
    % 0<t<max(t) and q0 is the initial position of the trap center relative
to the bottom of the sample cell)

    % Adding a Gaussian white noise to the system
    x(i) = x(i) + sqrt(2*D*Dt)*randn(N_p, 1);
end

t= (0:Dt:(Nt-1)*Dt);  % Generating the time vector for the problem

%% Visualization
figure(1);
hold on
plot(t, x, '-bo');
title('Position of the particle vs. time', 'FontSize', 20);
xlabel('$t$', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('$x(t)$', 'Interpreter', 'latex', 'FontSize', 20);
hold off

%% Computing the particle displacements along the trajectory
x_disp = zeros(N_p, Nt);
```

```
for i= 2:Nt

    x_disp(N_p, i) = x(i) - x(i-1);
end

% Storing the values of the optical force acting on the particle along its
trajectory
F = zeros(N_p, Nt);

for i= 2:Nt

    F(N_p, i) = -alpha*x_disp(N_p, i);  % The particle is trapped within a
harmonic potential near the focal point of the optical trap
end

t_int = reshape(t_vec, [4, 5000]);
b = reshape(F.*t_vec, [4, 5000]);
y = reshape(x_disp, [4, 5000]);  % Reshaping the F and x_disp row vectors
into matrices for further computations

% Numerical computation of Eq.(2)
Q = zeros(4, 5000);   % Creating a storage vector to store the results of the
numerical integrations....
% performed over the discrete F and x_disp datasets

for i= 1:5000

    Q(:, i) = cumtrapz(y(:, i), b(:, i));  % cumtrapz performs numerical
integrations over discrete datasets
end

sigma = a/(k*T*4*1.0e-3).*Q(4, :);  % Solving for sigma_t

sigma_scaled = sigma./1.0e-11;

sigma_array = sigma(1:1:5000); % Values of sigma used for generating the
Gaussian fit

entro_pos1 = sort(sigma(sigma>=0));

entro_pos = sort(sigma_scaled(sigma_scaled>=0));  % Generating a vector that
contains only the positive values of sigma_scaled....
% i.e., entropy-production values along the particle's transient trajectories

sigma_pos = find(sigma_scaled>=0);  % Returns a vector that contains the
index values of sigma, where sigma>=0
t_pos = t_vec(sigma_pos);
sigma_exp = exp(-entro_pos);

%% Visualization
figure(2);
grid on
```

```matlab
histogram(sigma_scaled, 40, 'facecolor', 'k', 'BinWidth', 4.5);   % Bin size
= 0.101
xlabel('$\sigma_t$ (scaled)', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('Number of trajectories', 'Interpreter', 'latex', 'FontSize', 20);

figure(3);
plot(t_pos, sigma_exp, 'b', 'linewidth', 2);
grid on
grid minor
xlabel('Time [s]', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('$P(\sigma_{t} < 0)/P(\sigma_{t} > 0)$', 'Interpreter', 'latex',
'FontSize', 20);

%% Generating a smooth plot using the obtained histogram distribution
[N, edges] = histcounts(sigma_scaled, 5000);  % Generating vectors that
contain the values of the bin edges and the number of trajectories....
% in the histogram distribution (Note: Vector 'N' contains the values of the
number of transient trajectories and vector 'edges'....
% contains the values of the bin edges for the generated histogram
distribution

Gauss_Fit = zeros(1, length(N)); % Creating a storage vector that contains
the values of the midpoints of the bin edges

for i= 1:length(N)
    Gauss_Fit(i) = (edges(i) + edges(i+1))/2;
end

%% Visualization
yi = smooth(N);  % Generating a smooth curve for the histogram distribution

figure(4);
hold on
plot(Gauss_Fit, yi, '-r', 'linewidth', 2);
title('Plot for the stochastic entropy distribution along transient
trajectories', 'FontSize', 20);
xlabel('$\sigma_t$', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('Number of trajectories', 'FontSize', 20);
hold off

%% Linear fit for the number ratio values

sigma_val = find(sigma_exp > 0.1);
t_val = t_vec(sigma_val); % Vector containing the times at which the linear
fit is being generated
sigma_linear = sigma_exp(sigma_val);
sigma_log = -log(sigma_linear); % Vector containing the values used for
generating the linear fit

figure(5);
plot(t_val, sigma_log, 'b');
grid on
grid minor
xlabel('Time [s]', 'Interpreter', 'latex', 'FontSize', 20);
```

```
ylabel('-ln($P(\sigma_{t} < 0)/P(\sigma_{t} > 0)$)', 'Interpreter', 'latex',
'FontSize', 20);
```

3) *Quartic potential well*: (for **Fig. 7**, **Fig. 8** and **Fig. 9**)

```
% The Langevin equation for a particle trapped within a potential V (an
unidimensional potential of some form) at a finite temperature T is given by:
% dx/dt = (-1/gamma)*dV/dx + sqrt(2*D)*w(t)......(1)
% Note that: Here, D is the diffusion constant given by: D=k*T/gamma
% (gamma being the friction constant (depends on the radius of the
% particle, its velocity v relative to the fluid and the coefficient of
% viscosity of the fluid)
% Taking this into account, eq.(1) assumes the following form:
% dx/dt = (-1/gamma)*dV/dx + sqrt((2*k*T)/gamma)*w(t)
% Here, w(t) is the Gaussian random noise (characterised by the random
% forces acting on the system)
% Using eq.(1), one can obtain the Langevin equation for a particle trapped
% within a parabolic potential well (V(x)=0.5*alpha*x^2) as follows:
% dx/dt = -(alpha*x)/gamma + sqrt((2*k*T)/gamma)*w(t)....(2)
% Note: The same holds for the other 2 dimensions
% Finite-difference equation approach for solving the above ODE:
% Note that, dx/dt can be recasted (by first principles) into a more
% conventional form as: dx/dt = (x(i)-x(i-1))/del(t) (where del(t) happens
% to be the time step (initially set) over which the simulation is expected
% to run
% Also, the Gaussian random noise w(t) can be expressed as:
% w(t)=w(i)/sqrt(Dt)....(3), where w(i) happens to be a sequence of Gaussian
% random
% numbers lying between 0 and 1 (zero mean and unit variance)
% Optical trap (Ref. 2:
%
https://www.researchgate.net/publication/11238596_Experimental_Demonstration_
of_Violations_of_the_Second_Law_of_Thermodynamics_for_Small_Systems_and_Short
_Time_Scales?enrichId=rgreq-635562a6c8ab7784ee764346ab3ecd50-
XXX&enrichSource=Y292ZXJQYWdlOzExMjM4NTk2OOFTOjEwNDU1NTA3NzQzOTUwNEAxNDAxOTM5
MjgyMjE0&el=1_x_3&_esc=publicationCoverPdf)
% For an optical trap, the particle trapped within it will experience a
% linear restoring force (to a good approximation) in the neighborhood of the
trap center, which tranalates to
% saying that the particle is trapped within a harmonic potential. This is
% similar to the case of a Brownian particle trapped within a parabolic
% potential well

%% Defining parameters
N_p = 1;  % The trajectory of one Brownian particle is being simulated
alpha = 3.87*1.0e-7;   % The trapping constant of the optical trap (in N/m)
eta = 2.5; % Viscosity of the fluid (in poise)
k = 1.38*1.0e-23;  % Boltzmann's constant
R = 9.23*1.0e-6;  % Radius of the particle (in m)
gamma = 6*pi*eta*R;  % Friction constant for the medium
T = 433;  % Temperature of the fluid (in K)
D = (k*T)/gamma;  % Diffusion constant for the medium
Nt = 0.2*1.0e+5;  % Number of samples picked/# of iterations considered
a = 4*1.0e-7;  % The trap is translating at 2.5 micrometers/sec
```

```
%% Initialization
Dt = 1.0e-3;   % Time step for the problem
x = zeros(N_p, Nt); % Creating a storage vector for the positions of the
particle along the particle's trajectory
x(1) = 8.5*1.0e-7;  % The particle starts at 3 micrometers relative to the
position of the trap center in each simulated trajectory
qt = zeros(N_p, Nt);  % Implementing a vector that contains the positions of
the trap center relative to the bottom the sample cell
qt(1) = 3.5*1.0e-7;  % The trap center's initial position relative to the
bottom of the sample cell
t_vec = zeros(N_p, Nt); % Values of t over which the trap center's position
will be varied (corresponding to multiple simulated trajectories)
t_vec(1) = 0; % Observation time starts at t=0 (after the stage starts
translating at a constant velocity)

%% Numerical computations

for i= 2:Nt

    t_vec(i) = t_vec(i-1) + Dt;  % Updating the time instant at each
iteration
    qt(i) = qt(1) + 0.5*a*(t_vec(i-1))^2;  % Updating the position of the
trap center
    % From eq. (2) and eq. (3)
    x(i) = x(i-1) - alpha*Dt*(x(i-1) - qt(i-1))^3/gamma;  % Gets appended
with the position of the trap center at each iteration
    % Note: Here, the position of the trap center as a function of time is:
    % q(t) = q0 + v*t (such that t lies within the time vector:
    % 0<t<max(t) and q0 is the initial position of the trap center relative
to the bottom of the sample cell)

    % Adding a Gaussian white noise to the system
    x(i) = x(i) + sqrt(2*D*Dt)*randn(N_p, 1);
end

t= (0:Dt:(Nt-1)*Dt);  % Generating the time vector for the problem

%% Visualization
figure(1);
hold on
plot(t, x, '-bo');
title('Position of the particle vs. time', 'FontSize', 15);
xlabel('$t$', 'Interpreter', 'latex', 'FontSize', 16);
ylabel('$x(t)$', 'Interpreter', 'latex', 'FontSize', 16);
hold off

%% Computing the particle displacements along the trajectory
x_disp = zeros(N_p, Nt);

for i= 2:Nt

    x_disp(N_p, i) = x(i) - x(i-1);
end
```

```
% Storing the values of the optical force acting on the particle along its
trajectory
F = zeros(N_p, Nt);

for i= 2:Nt

    F(N_p, i) = -alpha*(x_disp(N_p, i)).^3;  % The particle is trapped within
a harmonic potential near the focal point of the optical trap
end

b = reshape(F.*t_vec, [4, 5000]);
y = reshape(x_disp, [4, 5000]);  % Reshaping the F and x_disp row vectors
into matrices for further computations
t_int = reshape(t_vec, [4, 5000]);
% Numerical computation of Eq.(2)
Q = zeros(4, 500);   % Creating a storage vector to store the results of the
numerical integrations....
% performed over the discrete F and x_disp datasets

for i= 1:5000

    Q(:, i) = cumtrapz(y(:, i), b(:, i));  % cumtrapz performs numerical
integrations over discrete datasets
end

sigma = a/(k*T*4*1.0e-3).*Q(4, :);  % Solving for sigma_t

sigma_scaled = sigma./1.0e-30;

sigma_array = sigma(1:1:5000); % Values of sigma used for generating the
Gaussian fit

entro_pos = sort(sigma_scaled(sigma_scaled>=0));  % Generating a vector that
contains only the positive values of sigma_scaled....
% ie., entropy-production values along the particle's transient trajectories

sigma_pos = find(sigma_scaled>=0);  % Returns a vector that contains the
index values of sigma, where sigma>=0
t_pos = t_vec(sigma_pos);
sigma_exp = exp(-entro_pos);

%% Visualization
figure(2);
grid on
histogram(sigma_scaled, 40, 'facecolor', 'k', 'BinWidth', 35.5);
xlabel('$\sigma_t$ (scaled)', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('Number of trajectories', 'Interpreter', 'latex', 'FontSize', 20);

figure(3);
plot(t_pos, sigma_exp, 'b', 'linewidth', 2);
grid on
grid minor
xlabel('Time [s]', 'Interpreter', 'latex', 'FontSize', 20);
```

```
ylabel('$P(\sigma_{t} < 0)/P(\sigma_{t} > 0)$', 'Interpreter', 'latex',
'FontSize', 20);

%% Generating a smooth plot using the obtained histogram distribution
[N, edges] = histcounts(sigma_scaled, 5000);  % Generating vectors that
contain the values of the bin edges and the number of trajectories....
% in the histogram distribution (Note: Vector 'N' contains the values of the
number of transient trajectories and vector 'edges'....
% contains the values of the bin edges for the generated histogram
distribution

Gauss_Fit = zeros(1, length(N)); % Creating a storage vector that contains
the values of the midpoints of the bin edges

for i= 1:length(N)
    Gauss_Fit(i) = (edges(i) + edges(i+1))/2;
end

%% Visualization
yi = smooth(N);  % Generating a smooth curve for the histogram distribution

figure(4);
hold on
plot(Gauss_Fit, yi, '-r', 'linewidth', 2);
title('Plot for the stochastic entropy distribution along transient
trajectories', 'FontSize', 20);
xlabel('$\sigma_t$', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('Number of trajectories', 'FontSize', 20);
hold off

%% Linear fit for the number ratio values

sigma_val = find(sigma_exp > 0.6);
t_val = t_vec(sigma_val); % Vector containing the times at which the linear
fit is being generated
sigma_linear = sigma_exp(sigma_val);
sigma_log = -log(sigma_linear); % Vector containing the values used for
generating the linear fit

figure(5);
plot(t_val, sigma_log, 'b');
grid on
grid minor
xlabel('Time [s]', 'Interpreter', 'latex', 'FontSize', 20);
ylabel('-ln($P(\sigma_{t} < 0)/P(\sigma_{t} > 0)$)', 'Interpreter', 'latex',
'FontSize', 20);
```