Braid Groups Research

By Reese Madsen

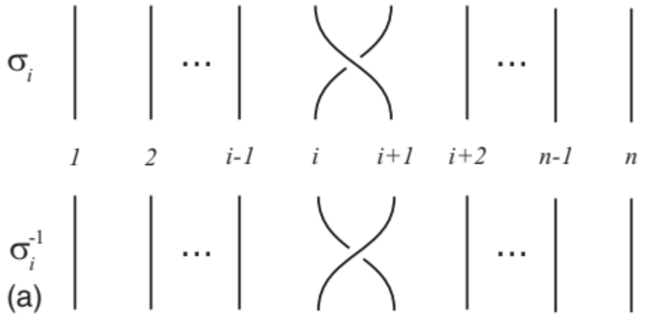
Braid Groups

$$B_n = \{\sigma_1, \ldots, \sigma_{n-1} : \sigma_i\sigma_j\sigma_i = \sigma_j\sigma_i\sigma_j$$
$$\text{if } |i-j| = 1 \text{ and } \sigma_i\sigma_j = \sigma_j\sigma_i \text{ if } |i-j| > 1\}$$

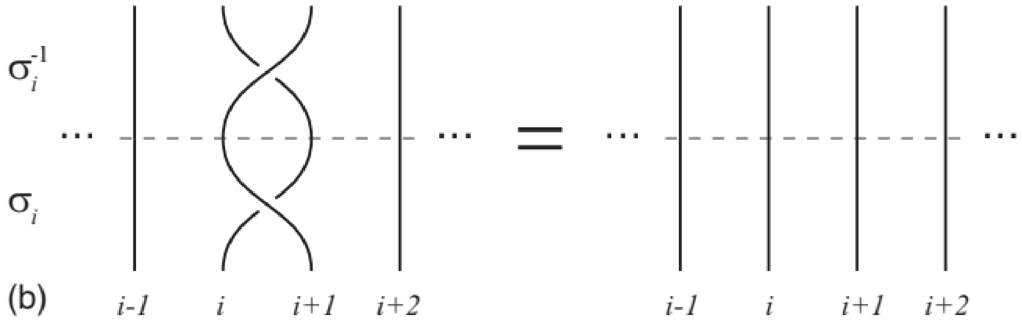
$< B_n, \| >$ is a group, where $\|$ is concatenation of generators

σ_i^{+1}

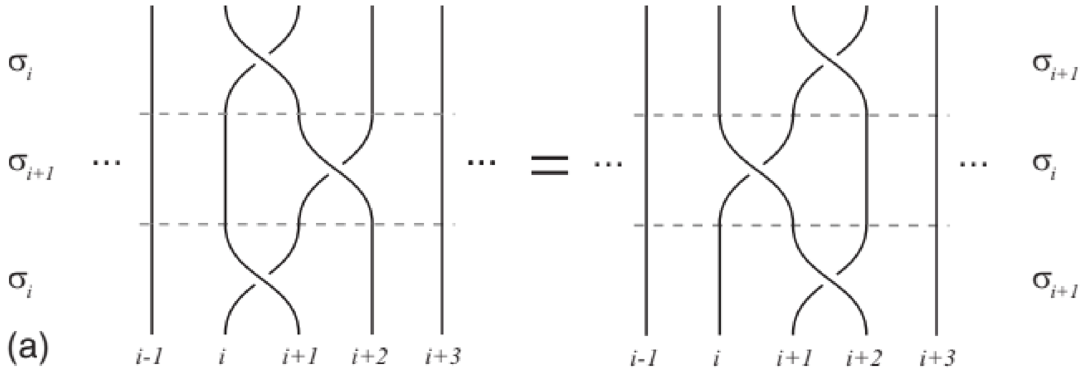
σ_i^{-1}



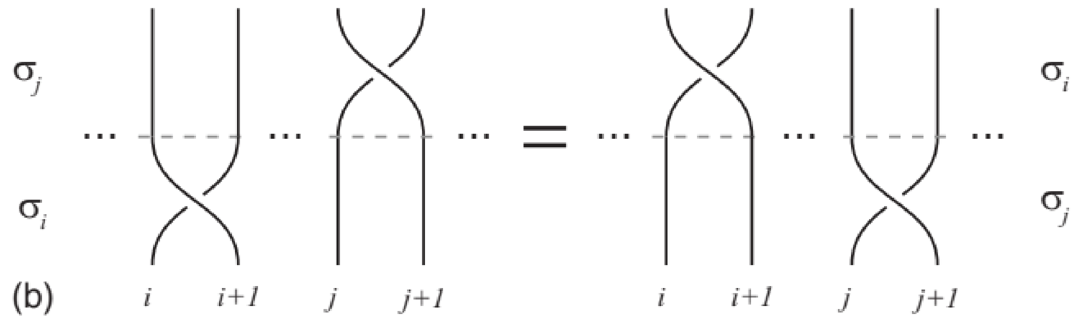
$$\sigma_i \sigma_i^{-1} = 1$$



$$\sigma_i \sigma_j \sigma_i = \sigma_j \sigma_i \sigma_j, |i - j| = 1 \tag{1}$$



$$\sigma_i \sigma_j = \sigma_j \sigma_i , |i - j| > 1$$



(2)

Braidlab

Algebraically braids do not have a time associated with them. However databraids (from braidlab package in MATLAB) have a time associated with them and can then be used to calculate decimal values for braids and functions of braids.

Creating braid in MATLAB:

T=1:NSTEPS.

XY=1:NSTEPS,1:2,1:N=interpolate(stochastic differential equation,**T**).

B=**databraid**(**XY**,**T**)- constructs a databraid object uses trajectory dataset XY and specified times of datapoints T.

Functions of a braid:

compact(**B**)- reduces databraid by cancelling generators using (1),(2) and the properties of the a group.

entropy(**B**)- returns the topological entropy of a braid from the loop method.

loop- class for representing topological loops in Dynnikov coordinates.

Trajectories

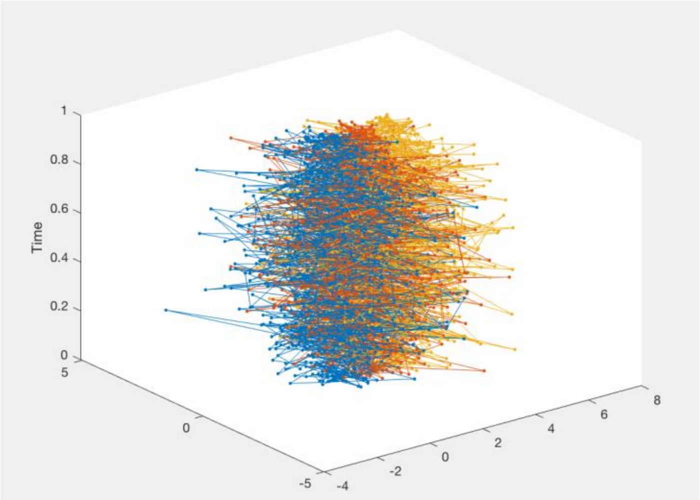
Types of Strands/Trajectories

1. Periodic/Unbounded Deterministic Trajectories- ex. double gyre flow.

1. Random Trajectories- ex. brownian bridge.

- The trajectories never cross at the same point in 3D space.
- We need to simulate the trajectories numerically and have the trajectories cross a reasonable number of times or else we will not get a meaningful braid.

Example of Random Trajectories:



Random Variables

$S = \{s_1, s_2, ..., s_k\}$ is the sample space which is the set of all possible outcomes of the random variable.

For a random experiment $\omega \in S$ you can obtain the following:
$\omega = \{s_i :$ where i is the index for the individual outcomes of the random experiment$\}$

$0 \leq t \leq P$ where P is the final time as well as the number of points between $t_0 = 0$ and $t_f = P$

A stochastic process is a function of two variables ω and t.
If we fix t, we obtain a family of random variables that depends on t :

$X(\omega, t) = X_t(\omega) = X_t(\{s_1, s_2, ..., s_k\}) = \{x_{t1}, x_{t2}, ..., x_{tk}\}$ in any given order.

$X_t(s_i) = x_{ti}$ for $1 \leq i \leq k$

Note: “In order to specify a stochastic process we have to provide the probability (or probability density) of occurrence of the various realizations.” (pg. 31) (3)

Random Walks

- can be represented as a stochastic differential equation

- A general random walk can be represented in the form:

$$X_{t+1} = X_t + v$$

$$v \in (\pm e_1, \dots, \pm e_n)$$

where

e_i is the ith basis element

Note: v is a random variable chosen from a distribution to obtain a random basis element

- The dimension of the space determines how many e's are available to choose from. For example if the space is 2 dimensional then the following basis elements are available: $\{e_1, -e_1, e_2, -e_2\}$

Brownian Bridge

- random trajectories are brownian bridges

- stochastic process based on brownian motion and having the same start and ending point

- brownian motion is the random movement of particles in a fluid, ex. movement pollen particles on a water surface or turbulent wind

- the word bridge is due to the fact that the starting point and ending point are the same

Brownian Motion

Simulated with the following SDE:

$$dX_t = F(t, X_t)dt + G(t, X_t)dW_t$$

Where:

t is the time used as an input for all other variables

X is the state vector of process variables

F is the vector of drift rates

G is the matrix of volatility rates

dW is a vector of possible correlated drift and volatility rates

- Drift is defined as the change in the average value.
- Volatility represents the random variance of the trajectory.
- MATLAB only needs the drift vector and volatility matrix to create a brownian motion model.

Brownian Motion in MATLAB

Settings needed to produce trajectories:

1. Initial conditions of strands. Note: the initial condition is the same as the terminal point because we are using a Brownian bridge.
2. Drift rates.
3. Volatility rates.
4. Duration of time.

1. $x_{i_n} = x_{f_n} = \dfrac{n}{N-1}$ where N is the total number of strands and $n \in \{0, 1, ..., N-1\}$ is the index of each strand.
$y_{i_n} = y_{f_n} = 0$ where (x_{i_n}, y_{i_n}) is an ordered pair for the initial conditions.
The same method applies for the final conditions.

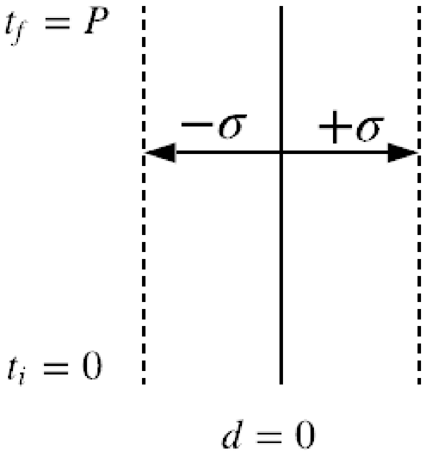
1. $F(t, X_t) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$

3. $G(t, X_t) = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$ where σ is the standard deviation.

4. $0 \leq t \leq P$ where P is the final time as well as the number of points between $t_0 = 0$ and $t_f = P$

Drift and Volatility Visualization

This is a trajectory diagram for one strand. It represents the random walks that each trajectory is taking.



P is the number of points between t_i and t_f

d is the change in the average value

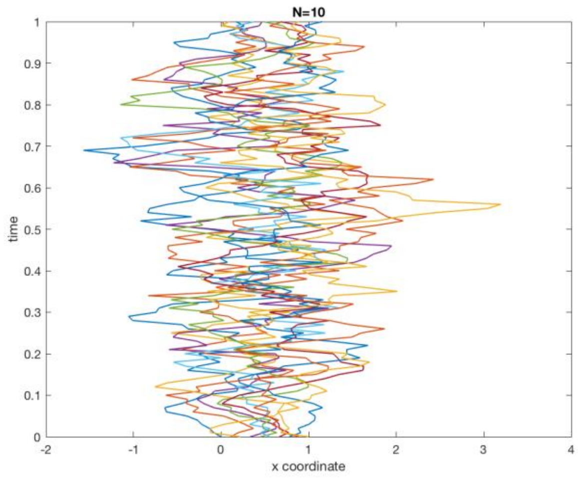
Anchor Parameter Set

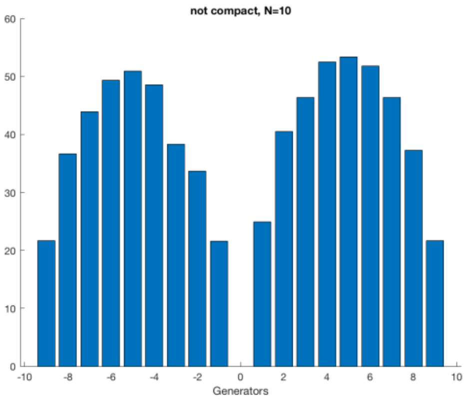
$$N = 10, \sigma = 3$$
$$dt = 0.01, T_0 = 0,$$
$$T_f = 1, t = T_0 : dt : T_f$$

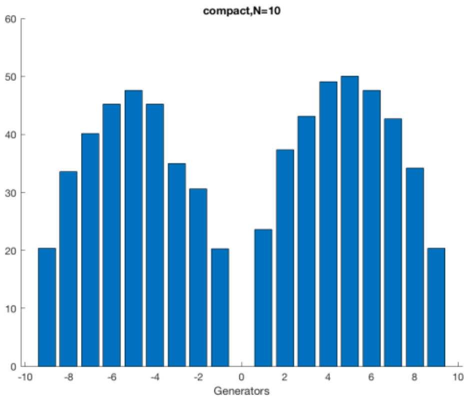
Initial Conditions

$$x_{i_n} = x_{f_n} = \frac{n}{9}$$
$$n \in \{0, 1, ..., 9\}$$
$$y_{i_n} = y_{f_n} = 0$$

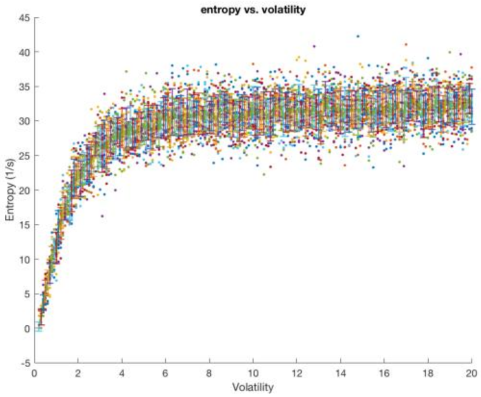
entropy=24.035
FTBE=24.786
number of generators before compact=690
number of generators after compact=658

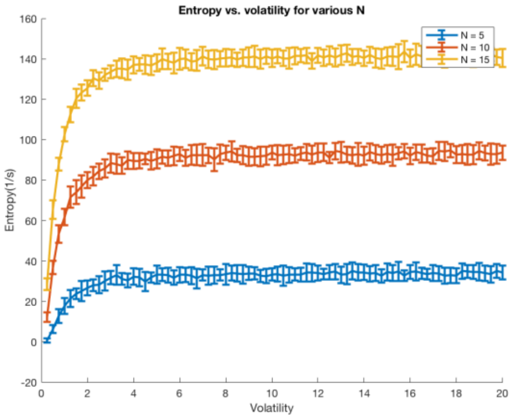


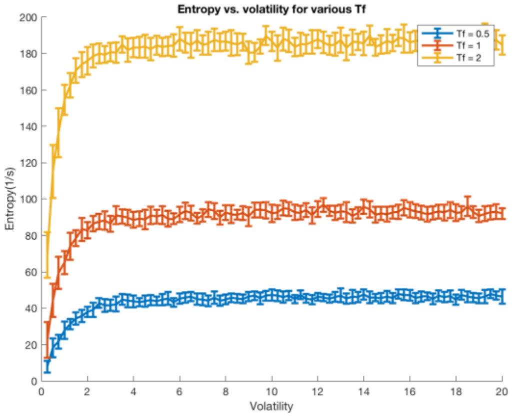




Entropy vs. Volatility/Sigma

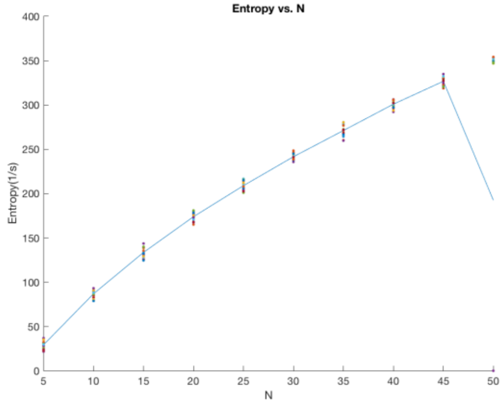


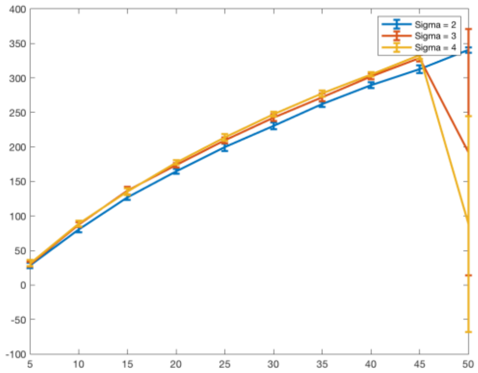


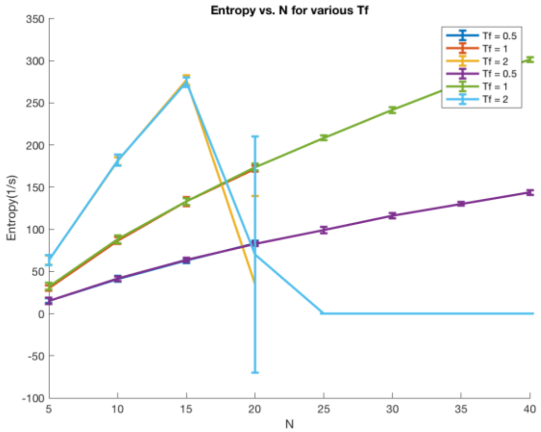


Entropy vs. Number of Strands

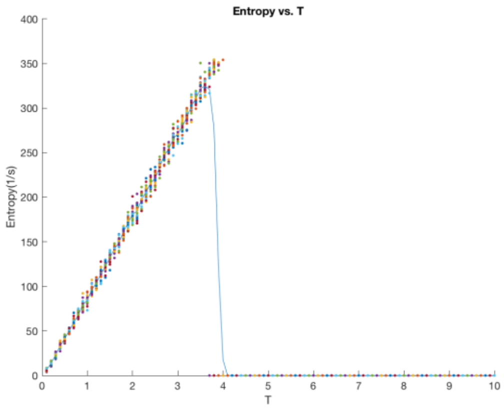
Entropy vs. N various Sigma



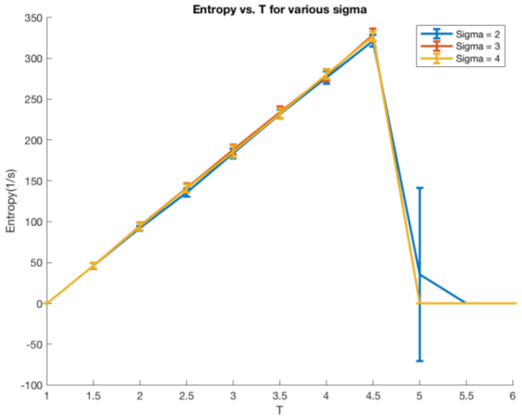




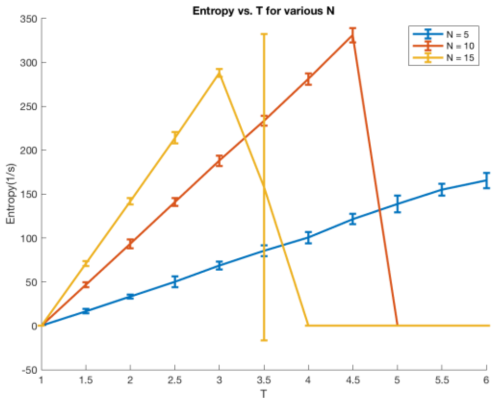
Entropy vs. Time



m=92.07

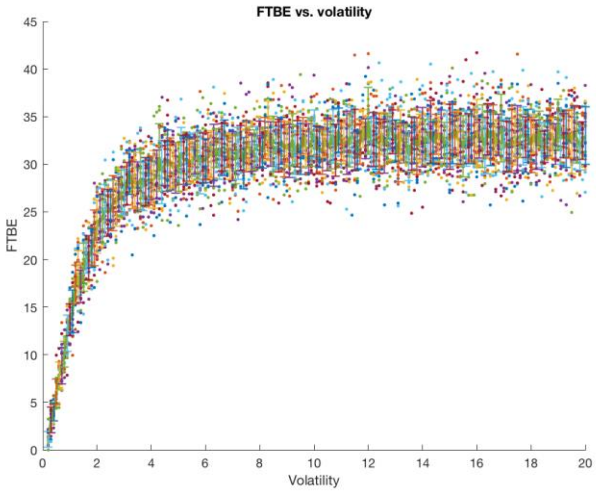


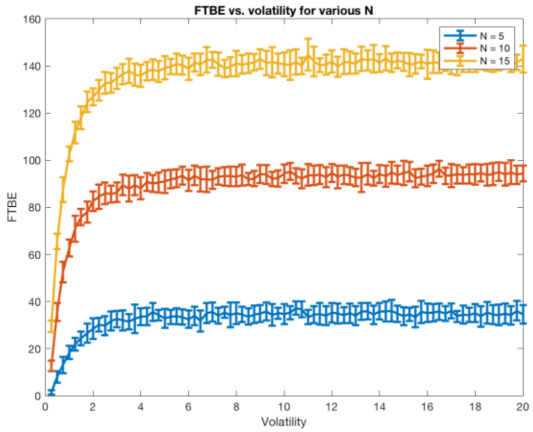
sigma=2: m=90.74
sigma=3: m=94.46
sigma=4: m=94.11

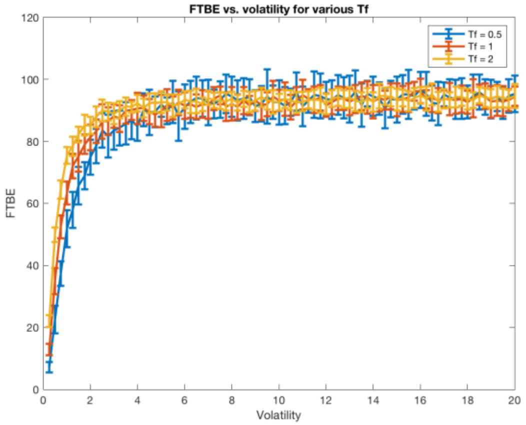


N=5: m= 33.5596
N=10: m= 93.8200
N=15: m= 142.5552

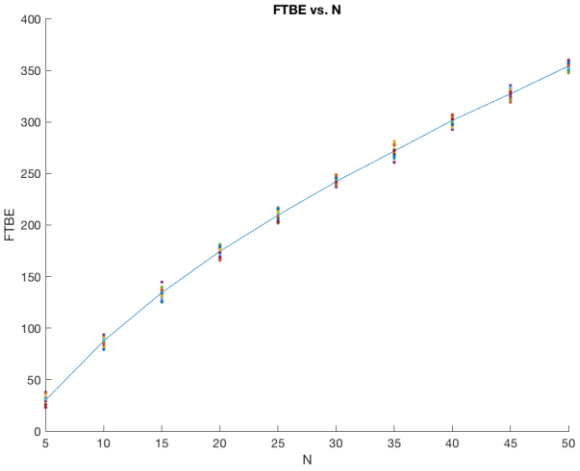
FTBE vs. Volatility/Sigma

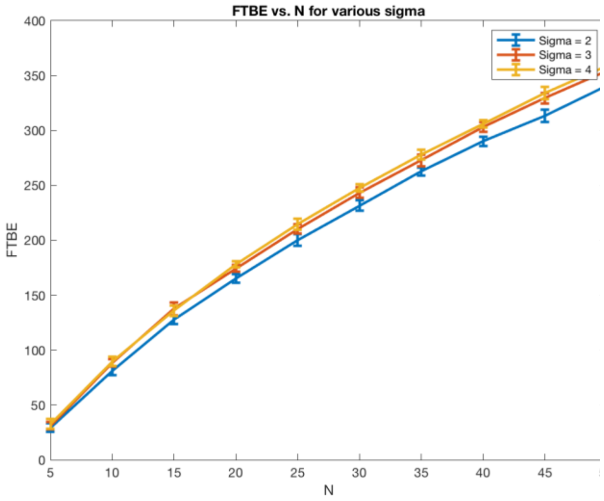


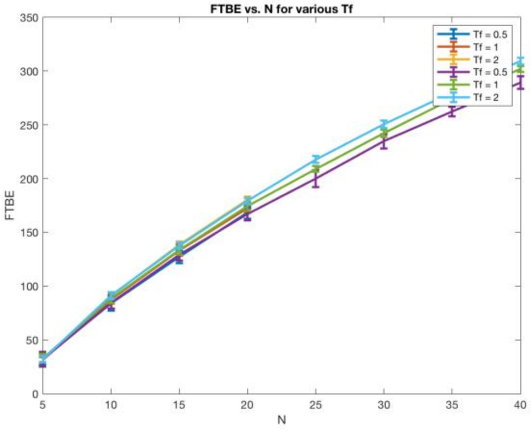




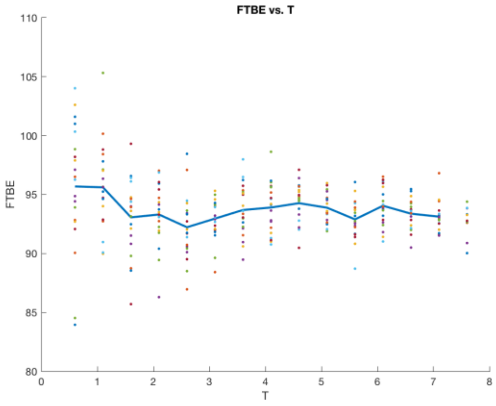
FTBE vs. Number of Strands, N=100 will taper off?

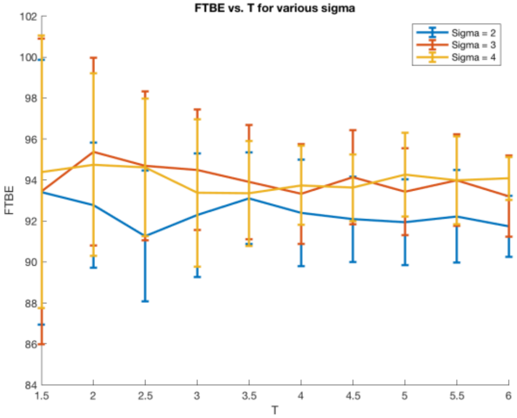


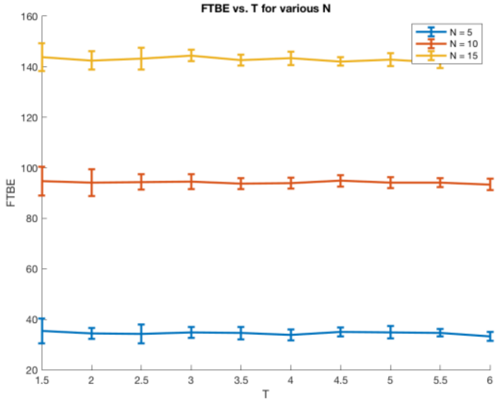




FTBE vs. Time

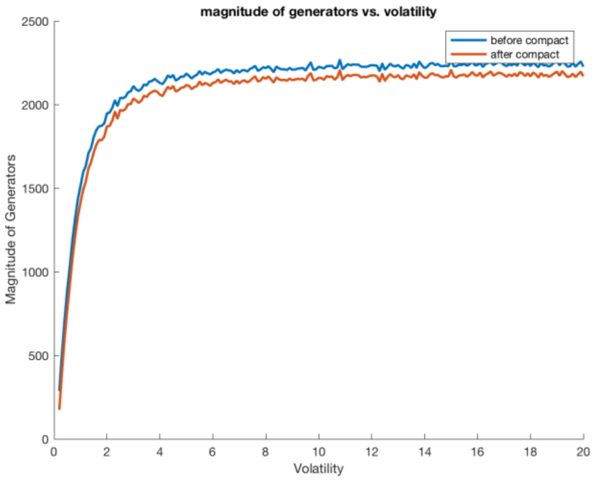


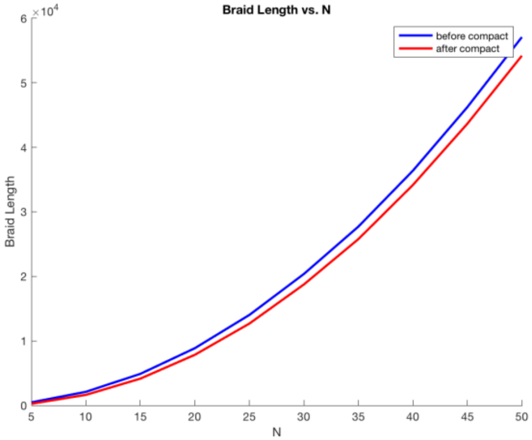


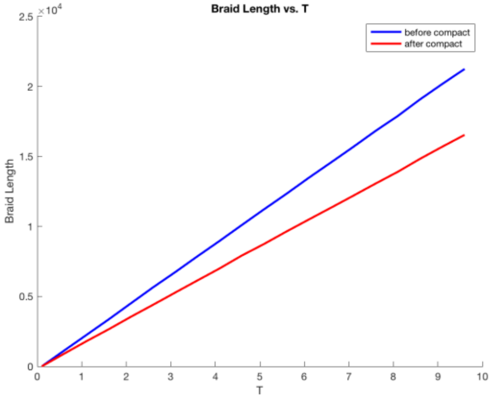


fix warning by looking at command line.mat 11/14 file

Braid Length (before and after compact) vs. Inputs

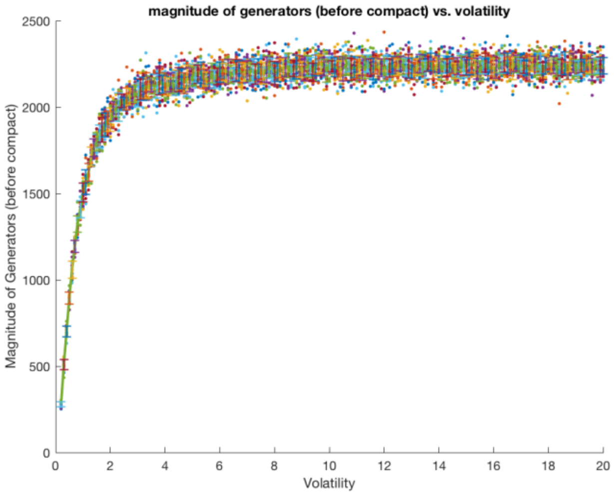


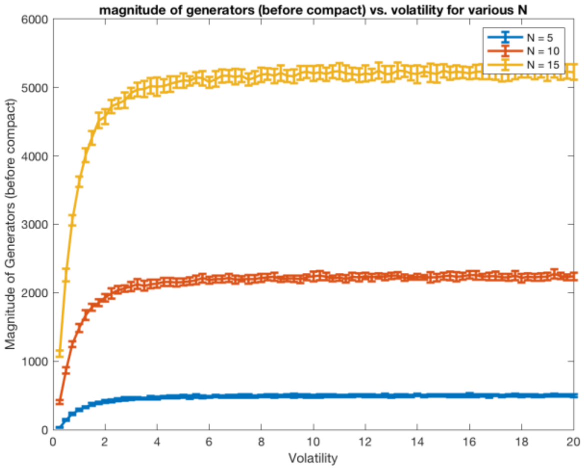


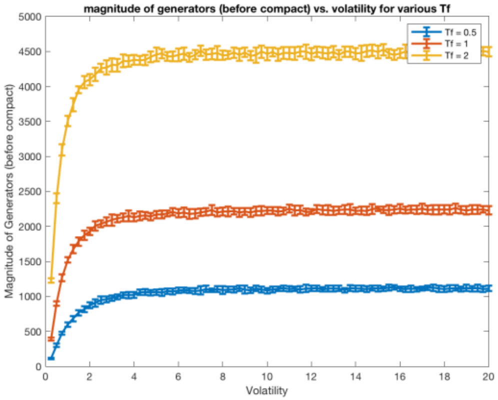


be_compact: m=2218.2
af_compact: m=2152.9

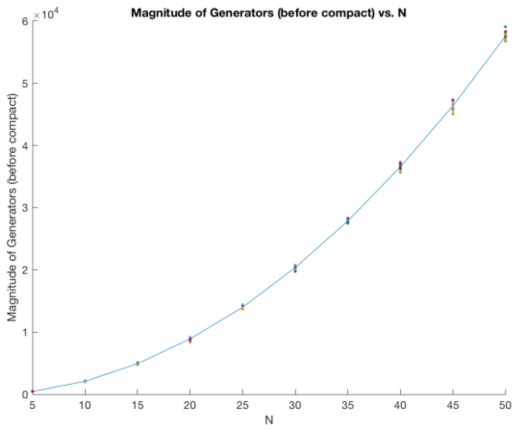
Braid Length (before compact) vs. Volatility/Sigma

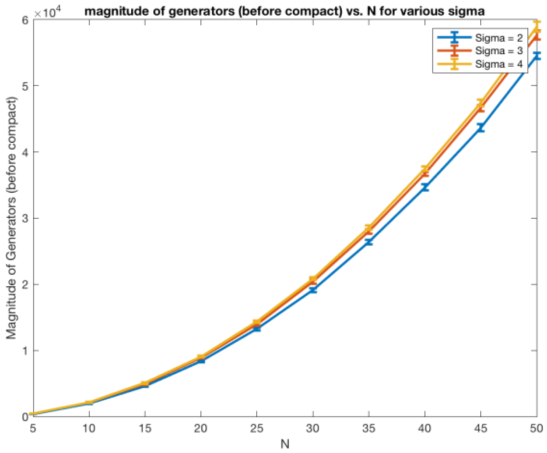


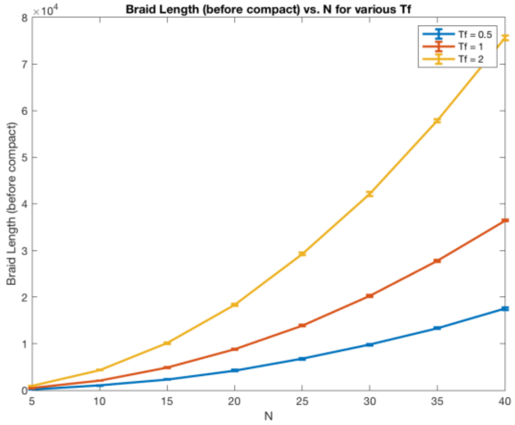




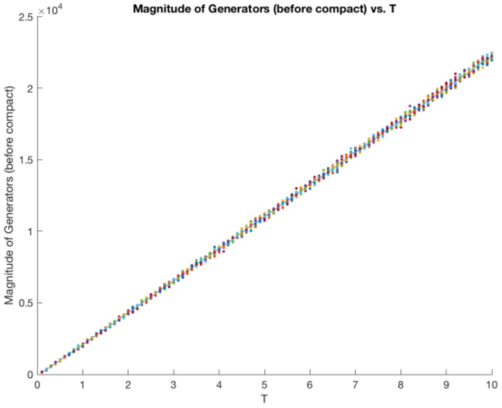
Braid Length (before compact) vs. Number of Strands



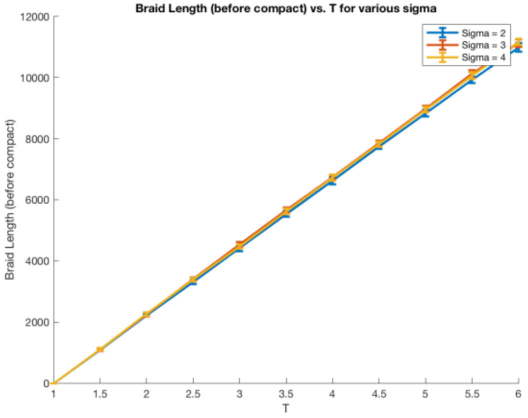




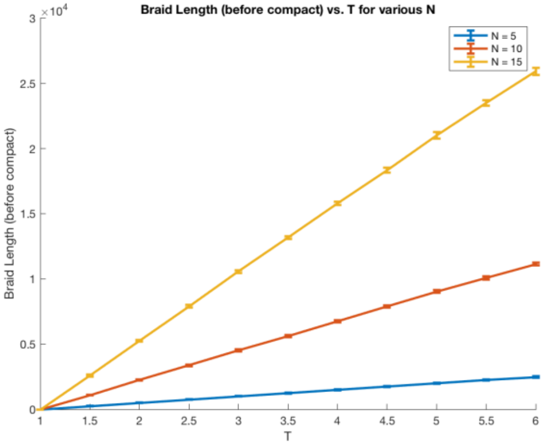
Braid Length (before compact) vs. Time



m=2218.2

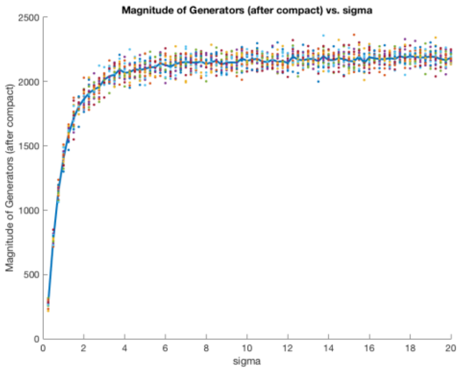


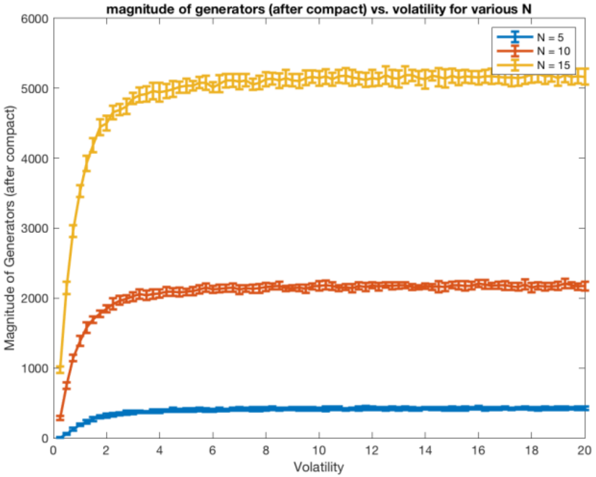
sigma=2: m=2205.4
sigma=3: m=2270.7
sigma=4: m=2263.8

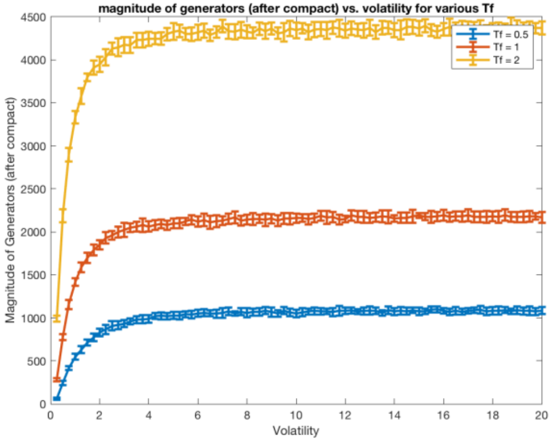


N=5: m= 503.03
N=10: m= 2258.2
N=15: m= 5276.7

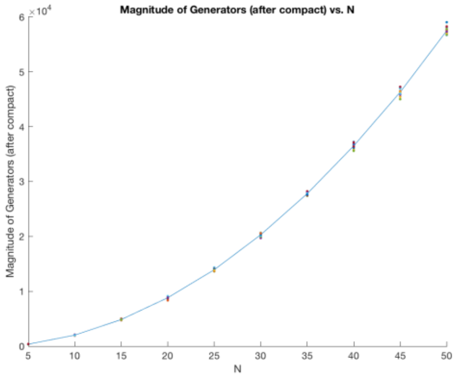
Braid Length (after compact) vs. Volatility/Sigma

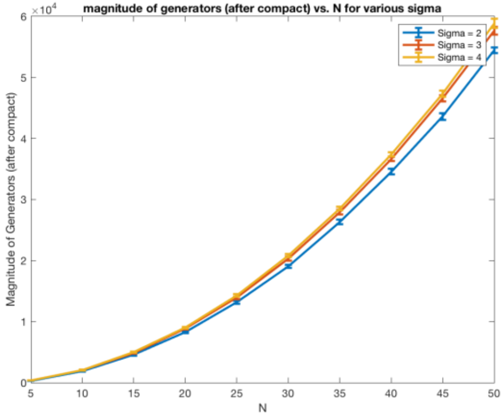


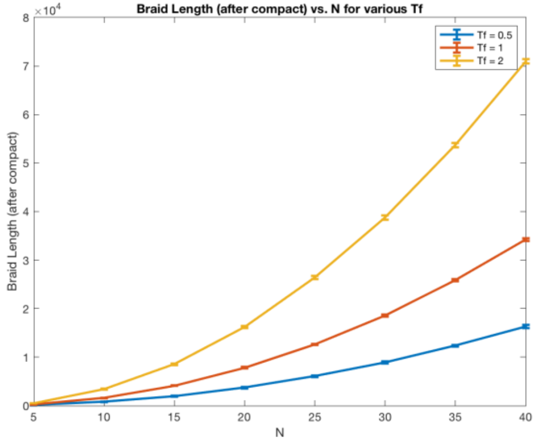




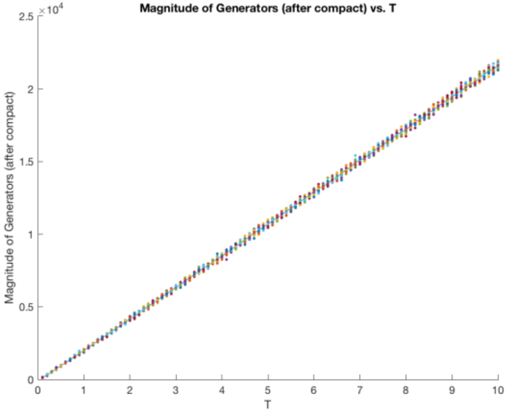
Braid Length (after compact) vs. Number of Strands

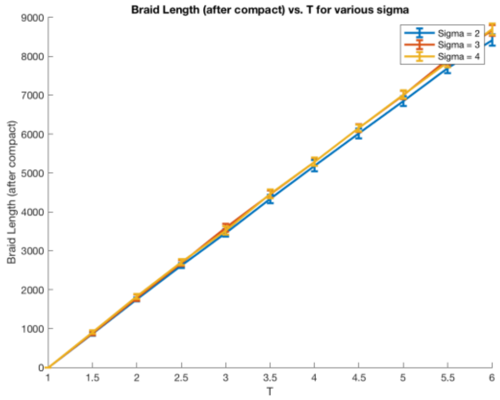


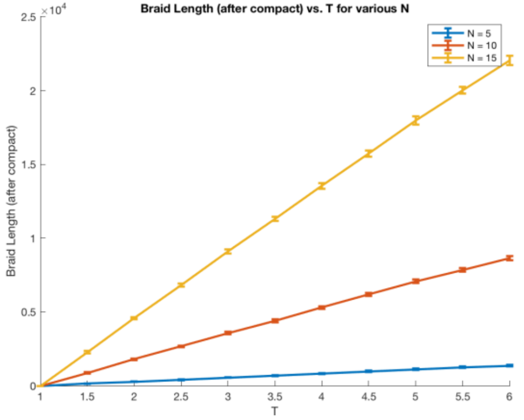




Braid Length (after compact) vs. Time







m=2152.9

sigma=2: m=1748.8
sigma=3: m=1793.7
sigma=4: m=1805.9

N=5: m= 274.69
N=10: m= 1788.6
N=15: m= 4553.7

```
% clear
% clc
% addpath('/Users/ReeseMadsen/Google Drive/matlab/toolboxes/braidlab_compiled');
% warning('off','BRAIDLAB:braid:entropy:noconv');
% M=20;                          %iterations of generating different random entropies
% sigma=3;
% N_initial=2;
% N_final=20;
% N_increment=1;
% N_slider=N_initial:N_increment:N_final;
%
% e_array=zeros(M,length(N_slider));
% average_line=zeros(1,length(N_slider));
%
% figure      %initializing plot
% xlabel('Number of Strands')
% ylabel('Entropy of Braid (1/s)')
% title('entropy vs. number of strands')
% hold on
%
% % ----------------------------------------------------------------------
% N_index=1;
% for N=N_slider
%     x0b= [0:1/(N-1):1;zeros(1,N)];                                   %initialize starting points
%     x0 = permute(x0b, [3,1,2]);                                  %rearrange dimensions, instead of spreadsheet boxes use a 3rd dimension to better represent the slices of which is starting points
%     constraints=[x0;x0];                                          %start and end at same point
%     Mu=zeros(2,1);                                                %drift vector
%     Sigma=sigma*eye(2);                                                 %volatility vector
%     brown=bm(Mu,Sigma);                                          %initialization of SDE, 3*Sigma because identity braid if just Sigma
%     T0=0; dt=0.01; Tf=1;  t=T0:dt:Tf;                     %generate 1000 points between 0 and 1
```

References

1. Braids of entangled particle trajectories. Jean-Luc Thiffeault.
2. braidlab: a software package for braids and loops. Jean-Luc Thiffeault and Marko Budisic.
3. Applied Stochastic Processes in science and engineering. M. Scott. 2013.