Contents

Assignment 1 – Random Walk and Colored Noise

Shannon M. Gross

CIE4604 Simulation and Visualization

This assignment explores the simulation of random processes, which are crucial for analysing many types of signal systems. There are a number of different methods for generating "pseudorandom" numbers, which use a specified equation and a starting (seed) value. Because we, as the programmers, must provide the algorithm and seed, such a series cannot be considered truly random. Nevertheless, these pseudorandom simulations fulfill properties that allow us to accomplish tasks for a particular purpose (i.e. simulate some natural process)

In the next section, a random walk process is simulated using a recursive formula. Results of each sequence are plotted over time and then the values of their mean and standard deviation are added. Part II simulates a colored noise process and establishes functions to automate the procedure.

```
clc; clear all; close all;
```

PART I. Random Walk

The object of the random walk starts at some origin and takes a series of n (independent) steps. A random walk variable (x) is computed from recursive formula: x(k) = x(k-1)+w(k). In this case, w(k) is normal distributed with expectation 0 and variance qk. Variance is equal to qk=q*dt, with q the power spectral density.

Part I Notation:

- x: variable
- w: Gaussian noise

- k: index
- n: length of series
- m: number of simulations
- q: power spectral density
- dt: time interval (assume dt = 1)

```
% Set Random Walk Input Parameters
qk=5;           % Variance of the power spectral density (q)
n=2000;         % Length of series (i.e. steps per random walk)
m=100;          % Number of realizations/sequences
```

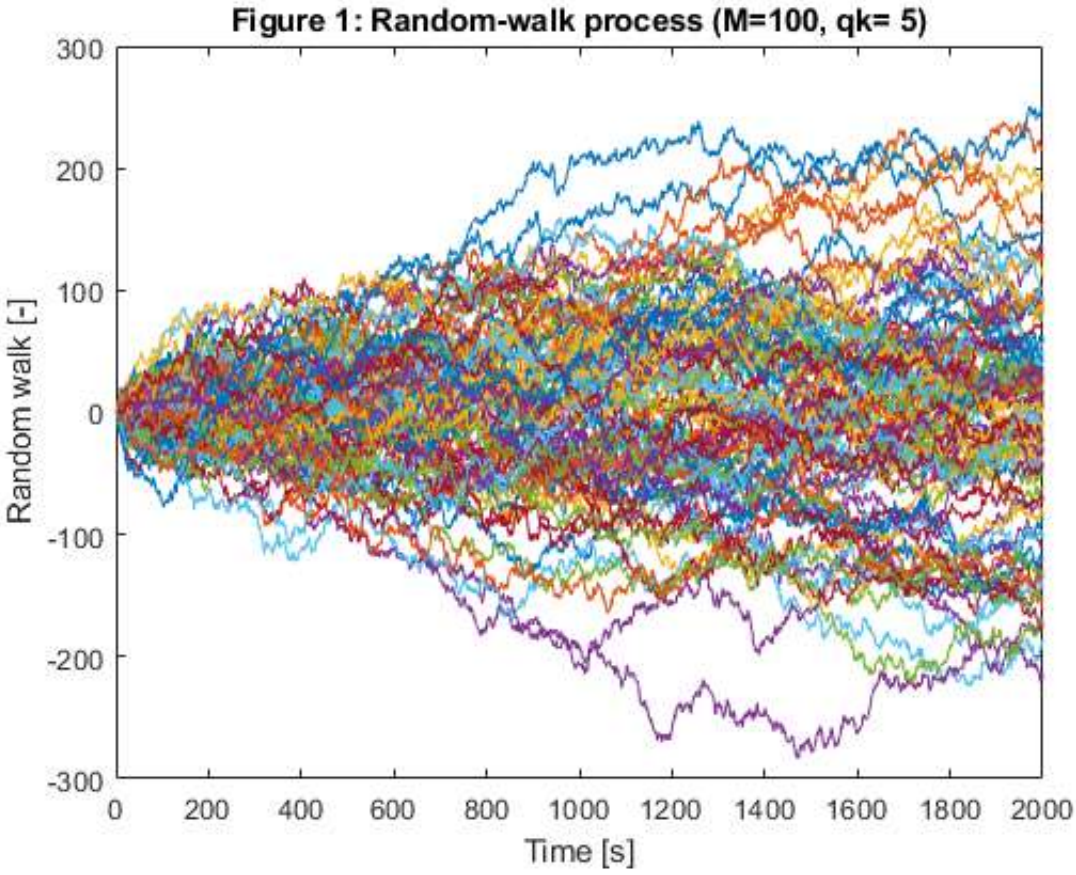
Question I.a. Generate Random Walk Simulation

In the random walk visualisation, the person's path is represented along the vertical axis, starting at a specified origin (e.g. zero, the mean). The horizontal axis represents time and each of the colored trajectories represents one realization of the random walk simulation. In each sequence, the person takes n=2000 steps. The person stays relatively close to the origin with high probability when the sequence is chosen at random. However, the more steps away from the origin, the wider the ensemble envelope appears (i.e more uncertainity regarding future position).

```
x=nan(n,m);               % Initialize x to NaN and preallocate memory
w=sqrt(qk)*randn(n,m);    % N(0,qk) is variance

% Short loop version
x(1,:)=zeros(1,m);
for k=2:n   % (Note: Matlab counter starts at 1)
    x(k,:)=x(k-1,:)+w(k,:);
end

% Plot the 100 random walk sequences
figure(1);
plot(x);  % Store results in a matrix, then plot
xlabel('Time [s]'); ylabel('Random walk [-]');
title(['Figure 1: Random-walk process (M=', num2str(m), ', qk= ', num2str(qk), ')'] )
```

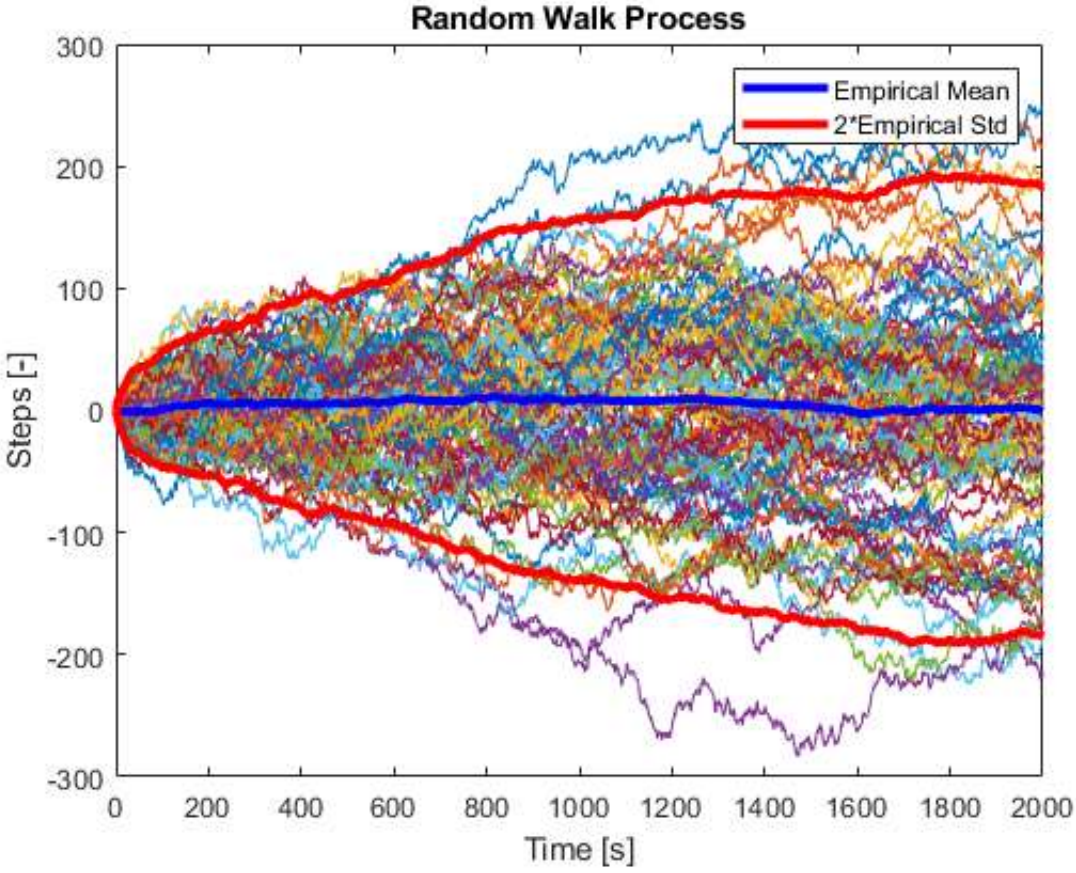


Question I.a. Determine the Empirical Mean and Standard Deviation

Empirical values are based on observation rather than a priori deduction. Thus, depending on the simulations (i.e. the measurements of the particular random walk sequences), the empirical mean and standard deviation will change.

```
xmean=mean(x,2);
xstd=std(x,0,2);

hold on
h(1)=plot(xmean, 'b', 'linewidth',3, 'DisplayName','Empirical Mean');
h(2)=plot(xmean+xstd*2, 'r', 'linewidth',3, 'DisplayName','2*Empirical Std');
plot(xmean-xstd*2, 'r', 'linewidth',3, 'DisplayName','2*Empirical Std')
xlabel('Time [s]'); ylabel('Steps [-]'); title('Random Walk Process');
legend([h(1),h(2)], 'Empirical Mean', '2*Empirical Std')
```



Question I.a. Determine the Formal Standard Deviation and add to plot

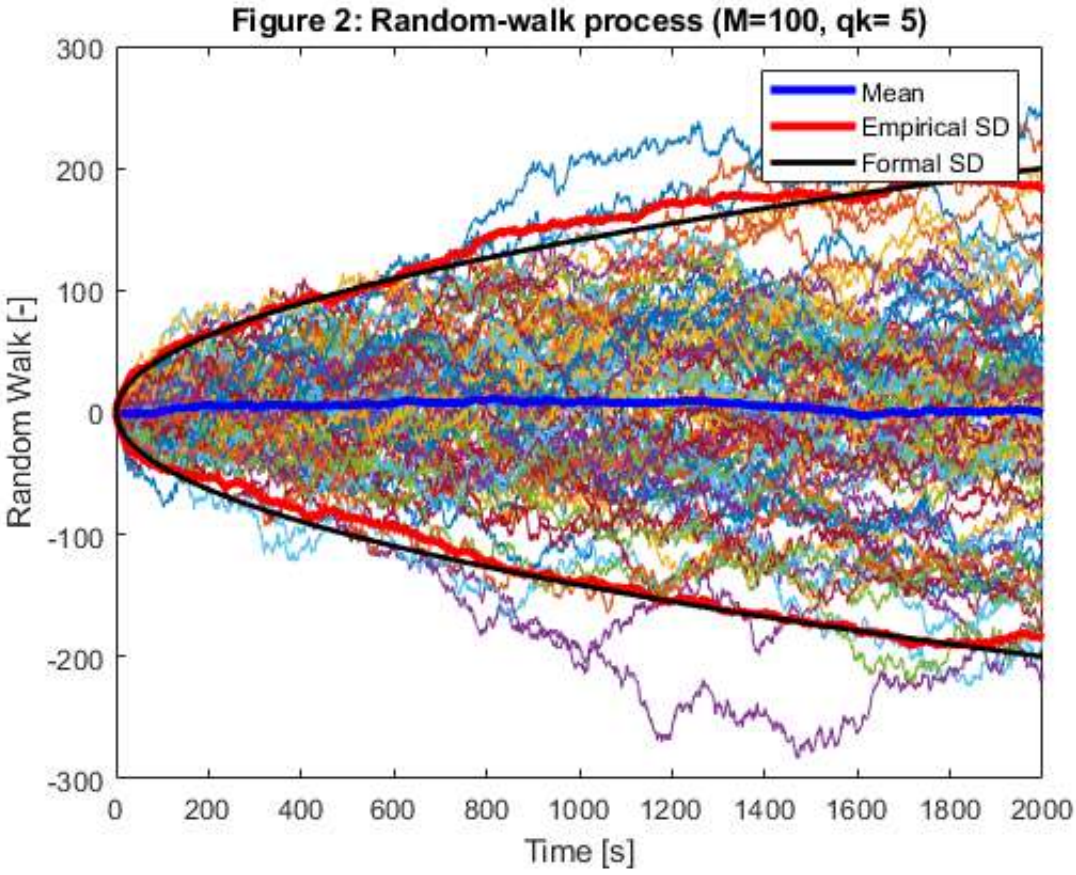
The variance of a random variable measures the spread of its distribution. In calculting the variance using propagation of errors we assumed that the measured variable (x) is described by Gaussian distribution. The standard deviation is equal to the square root of the variance. The expectancy of the mean value in this case is zero, since we expect the center of gravity of the distribution function will tend towards for larger numbers of measurements.

```
% Compute formal variance using error propigation
sx =zeros(n,1);
for k=2:n
    sx(k)=sx(k-1)+qk;
end
sx=sqrt(sx);

% Create Figure 2
hold on
h(3)=plot(2*sx, 'k', 'linewidth',2);
plot(-2*sx, 'k', 'linewidth',2);

xlabel('Time [s]'); ylabel('Random Walk [-]');
title(['Figure 2: Random-walk process (M=', num2str(m), ', qk= ', num2str(qk), ')'])
legend(h, {'Mean', 'Empirical SD', 'Formal SD'})
hold off;

% Note: for visual appearance, 2*standard deviation is plotted (both for
% empirical and formal SD)
```



Question I.b Repeat Simulations

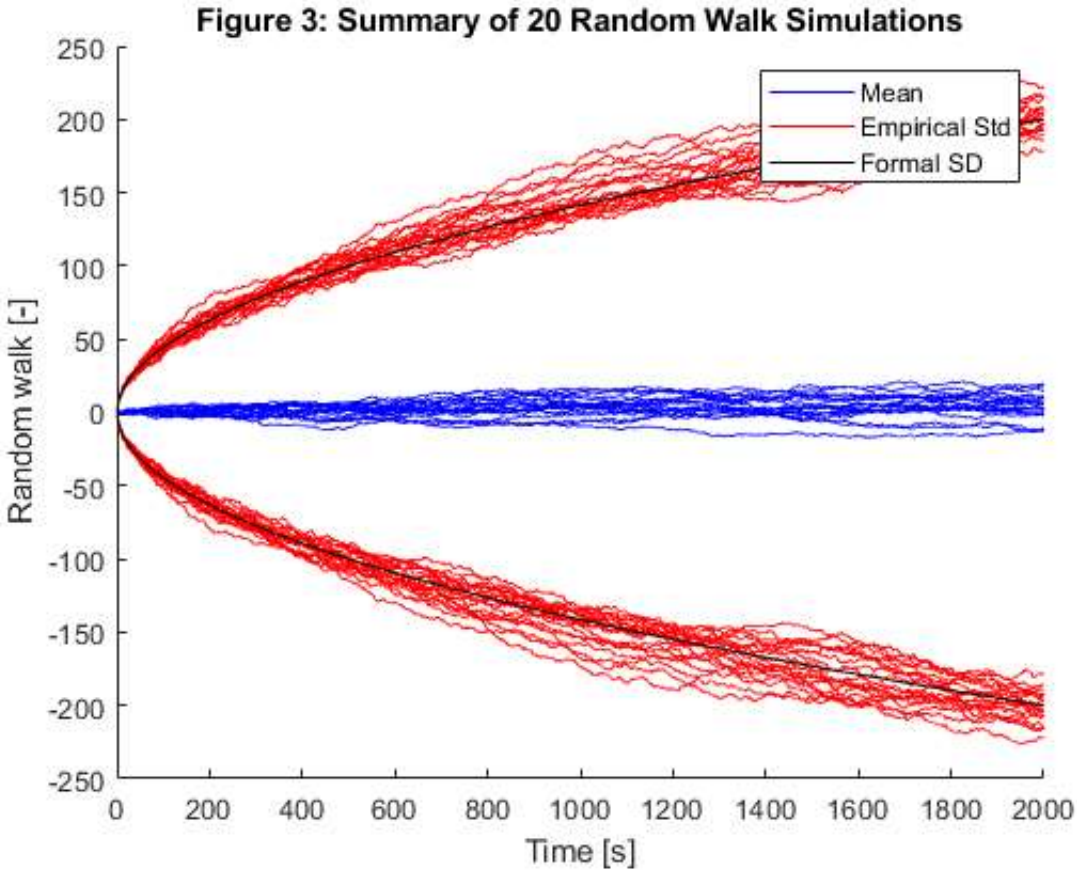
Below, a Matlab for loop is created in order to test the random walk process 20 times. To preserve figure clarity, only the mean and std deviations are incuded in the figure (i.e. individual walks are not shown).

```
for i=1:20
    w=sqrt(qk)*randn(n,m);
    x=randn(1,m);
    for k=2:n
        x(k,:)=x(k-1,:)+w(k,:);
        sx(k)=sx(k-1)+qk;
    end
    empmean=mean(x,2);
    empstd=std(x,0,2);
    sx=sqrt(sx);

    figure(2); hold on;
    j(1)=plot(empmean, 'b');
    j(2)=plot(2* empstd, 'r');
    plot(-2*empstd, 'r');
    j(3)=plot(2*sx, 'k');
    plot(-2*sx, 'k');
end

xlabel('Time [s]'); ylabel('Random walk [-]')
title(['Figure 3: Summary of 20 Random Walk Simulations'])
legend(j, {'Mean', 'Empirical Std', 'Formal SD'})
hold off
```

```
% Brief explanation of Figure 3
% The empirical mean and standard deviation change per simulation
% run (see Figure 3). As anticipated, the formal standard deviation
% remains the same no matter how many simulation repititions are run.
```



Question I.c. Stationarity

A random process is referred to as stationary if the statistical properties of that process do not change over time (e.g. constant mean, variance, autocorrelation).

The random walk process simulated here is not wide-sense stationary becuase its empirical variance differs from its formal variance (it is not time invariant).

PART II - Colored Noise

A first order random process is referred to as (first order) stationary if its density function does not change over time. When creating a colored noise processes, the distribution can be Uniform, Bernoulli, or any other distribution as long as there is no correlation between its values at different times. In the following examples, we will use a Gaussian distribution to simulate the colored noise process.

Notation

- x_g: series
- w_g: Gaussian noise
- qk_g: variance
- q: power spectral density of underlying white noise process

- std_xk: standard deviaion of xk (std_xk=1)
- T: correlation period (T=100)
- dt: time interval (assume dt=1)
- n_g: Number of steps
- m_g: Number of realizations/sequences (m_g=30)

```
% Set Input Parameters
T=100; std_kx=1; dt=1; n_g=2000; m_g=30;

% Describe Noise Process
B = 1/T;
q = 2*B*std_kx^2;    % power spectral density
```

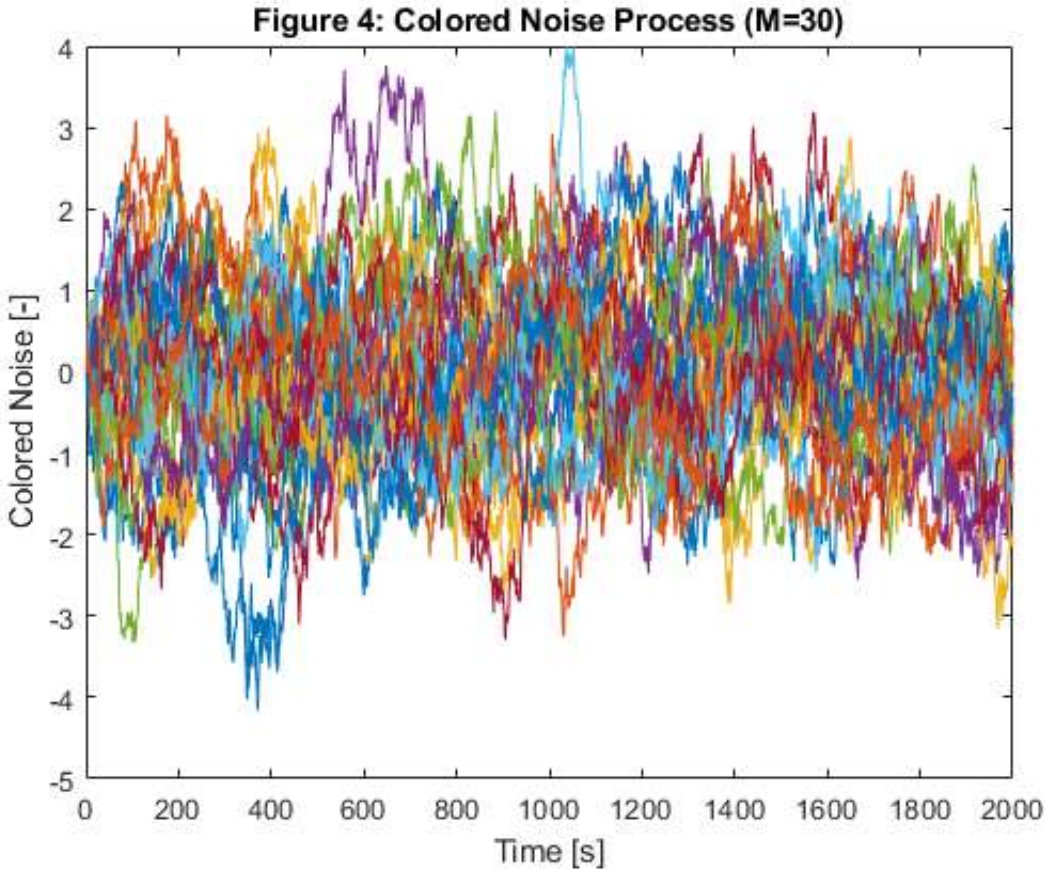
Question II.a First order Gauss-Markov sequence

A Gauss-Markov sequence has features of both Gaussian (in terms of probability distribution) and Markov (memoryless) processes. Simulating a first order Gauss-Markov process is an example of a colored noise process, which are frequently encountered in many physical applications.

```
qk_g = q/(2*B)*(1-exp(-2*B*dt)); % Variance
x_g = nan(n_g,m_g);              % Initialize x to NaN, preallocate memory
w_g = sqrt(qk_g)*randn(n_g,m_g); % Generate Gaussian noise

% Generate time series
x_g(1,:)=zeros(1,m_g);
for k_g=2:n_g
    x_g(k_g,:)=exp(-B*dt)*x_g(k_g-1,:)+w_g(k_g,:);
end

% Plot the realizations
figure;
plot(x_g);
xlabel('Time [s]');
ylabel('Colored Noise [-]');
title(['Figure 4: Colored Noise Process (M=', num2str(m_g),')']) ;
```



Note: the same colored noise simulation can be generated using the `randomatrix` function, which returns a pseudo-randomly generated Gauss- Markov process. See `help randomatrix` for more information.

Question II.a Formal Standard Deviation

Formal standard deviation gives you a reference for error propigation, particuarly when dealing with empirical measurements. Because the random generator uses Gaussian distribution (mean=0, std=1), the sequence should approach these parameters (especially if the ensemble runs are increased).

```
std_xg = sqrt(q/(2*B)); % Test/confirm that std_kx=1
fstd_xg = std_xg* ones(n_g, m_g);
```

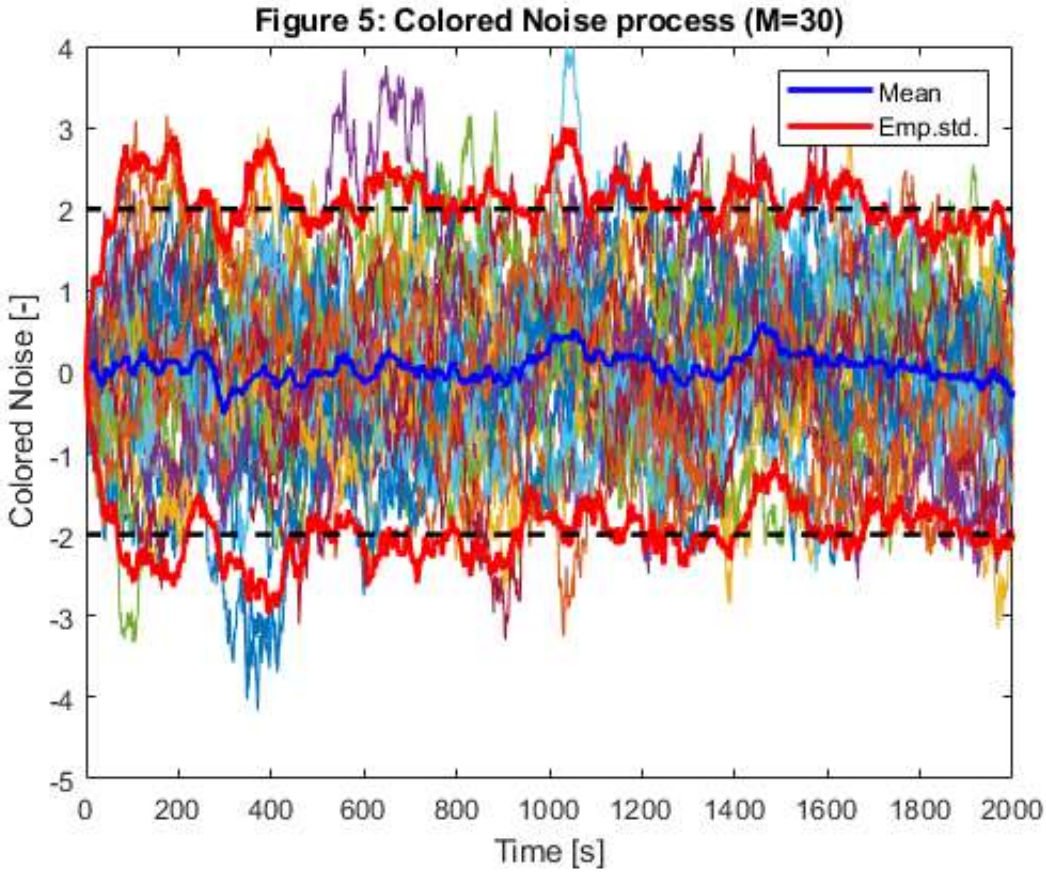
Question II.a Empirical Mean and Standard Deviation

```
emean_xg=mean(x_g,2);
estd_xg=std(x_g,0,2);

% Add Empirical SD and Mean to Figure 5
hold on
a1=plot(emean_xg, 'b', 'linewidth',2);
a2=plot(emean_xg+2*estd_xg, 'r', 'linewidth',2);
plot(emean_xg-2*estd_xg, 'r', 'linewidth',2);

% Add Formal SD to Figure 5
plot(2*fstd_xg, '--k', 'linewidth',2);
plot(-2*fstd_xg, '--k','linewidth',2);
```

```
% Figure Formatting
xlabel('Time [s]');
ylabel('Colored Noise [-]');
title(['Figure 5: Colored Noise process (M=', num2str(m_g), ')']);
legend([a1 a2],{'Mean','Emp.std.'})
```



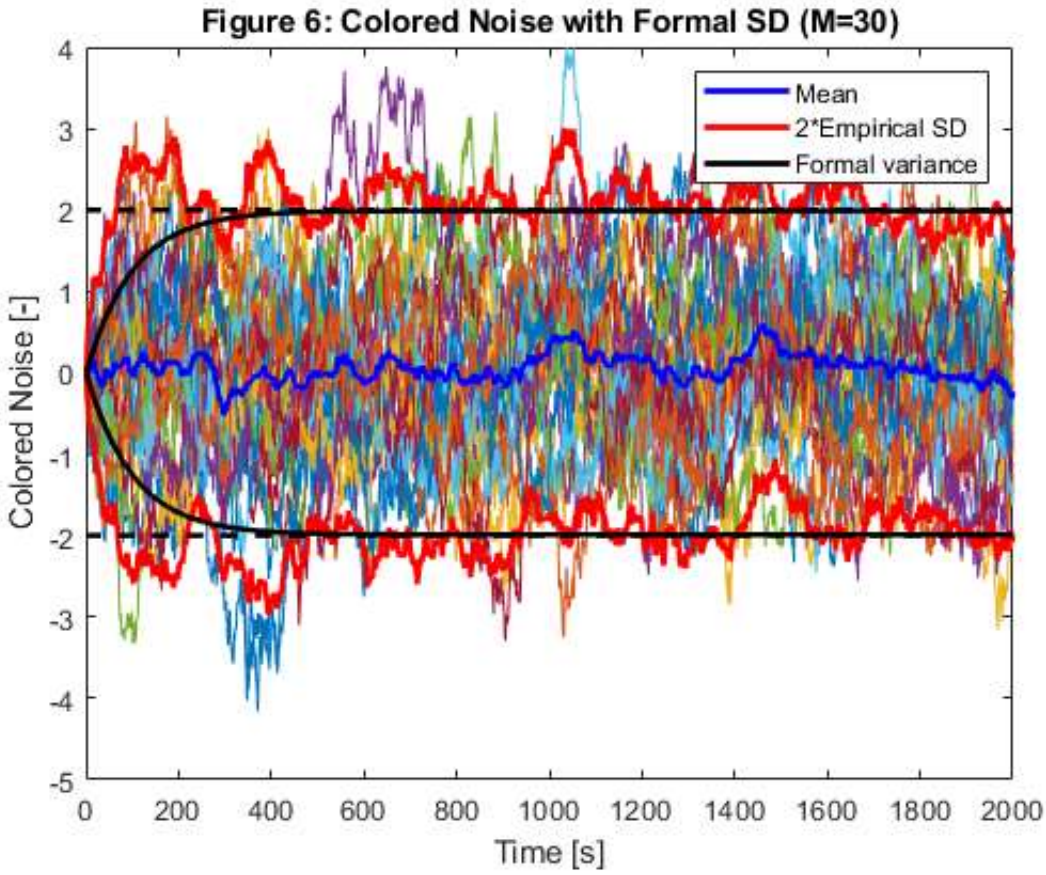
Note: the same colored noise simulation can be generated using the `randoplot` function, which returns a summary plot of a generated Gauss- Markov process. See `help randoplot` for more information.

Question II.b. Formal Variance

Propagation of error refers to the effect of variable uncertainity which accumulate due to the combined effect of the variables in the function. In practical applications, errors can accumulate if initial values are inexact or rounded measurements. Error propagation is generally expressed in terms of the standard deviation, which is the positive square root of the variance.

```
vx_g =zeros(n_g,1); % Initialize variance x
for k_g=2:n_g
    vx_g(k_g)=exp(-B*dt)*vx_g(k_g-1)+qk_g;
end
sx_g=sqrt(vx_g); % SD is sqrt of variance % Reassign

% Add recursively calculated variance to figure
a3=plot(vx_g, 'k', 'linewidth',2);
plot(-vx_g, 'k', 'linewidth',2);
title(['Figure 6: Colored Noise with Formal SD (M=', num2str(m_g), ')']);
legend([a1 a2 a3], 'Mean','2*Empirical SD','Formal variance')
hold off
```



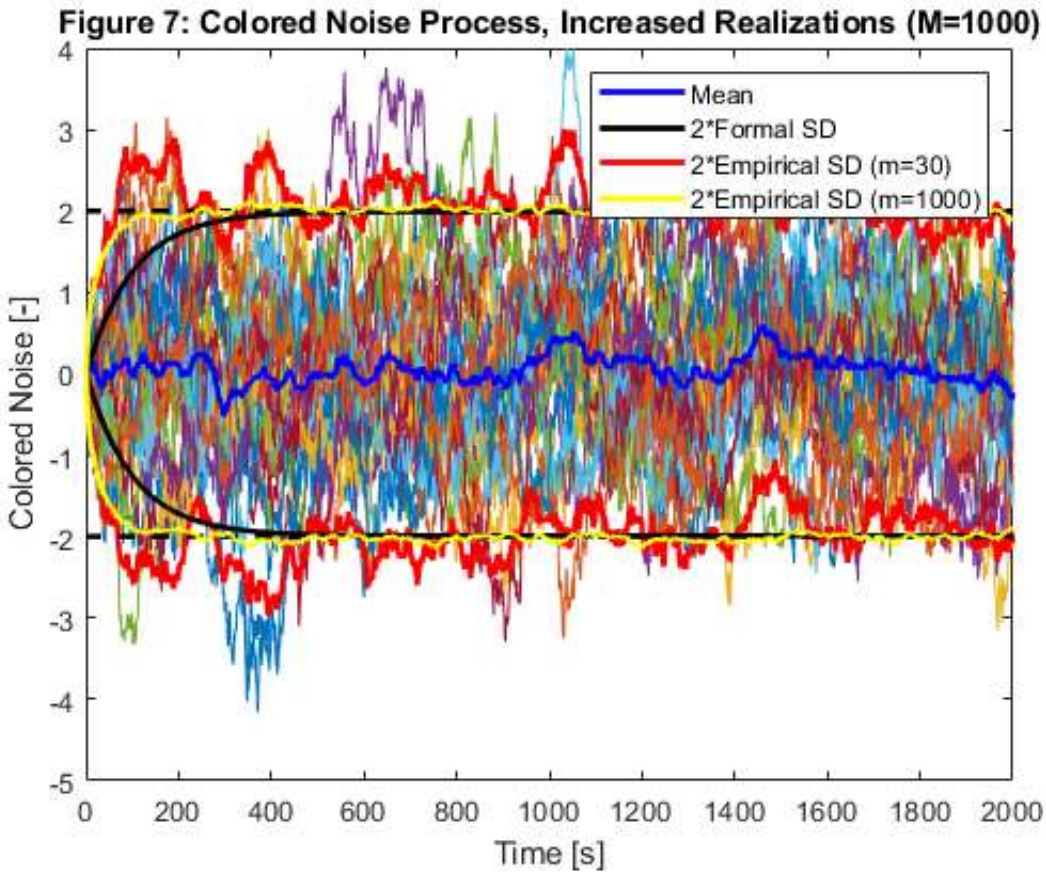
Computing formal variance recursively does not immediately yield a constant. It is only after about 500 runs increases that you can see the standard deviation stabalize around the anticipated constant. In the next section we greatly increase the number of simulation runs to if the results for the empirical standard deviation mroe closely approach this theoretical variance.

Question II.b. Increase number of realizations

```
m_g= 1000;              % Increase m from 30 to 1000
x_g=zeros(1,m_g);       % Initialize x to NaN, preallocate memory
w_g = sqrt(qk_g)*randn(n_g,m_g);

for k_g=2:n_g
    x_g(k_g,:)=exp(-B*dt)*x_g(k_g-1,:)+w_g(k_g,:);
end
estd_xg=std(x_g,0,2);

hold on
a4=plot(2*estd_xg, 'y', 'linewidth',2);
plot(-2*estd_xg, 'y', 'linewidth',2);
title(['Figure 7: Colored Noise Process, Increased Realizations (M=', num2str(m_g),')']);
legend([a1 a3 a2 a4], 'Mean','2*Formal SD','2*Empirical SD (m=30)','2*Empirical SD (m=1000)')
;
hold off
```



By increasing the number of realizations (m) from 30 to 1000, the empirical esimate is much closer to the formal standard deviation (std=1). This is a good way to check the random number generator since we see that for a really long sequence it is getting closer to the desired random distribution.

Question II.c Matlab functions

This section uses a series of Matlab functions to return:

- the empirical mean and standard deviation;
- formal standard deviation;
- a matrix with the realisations.

Simplifying assumptions:

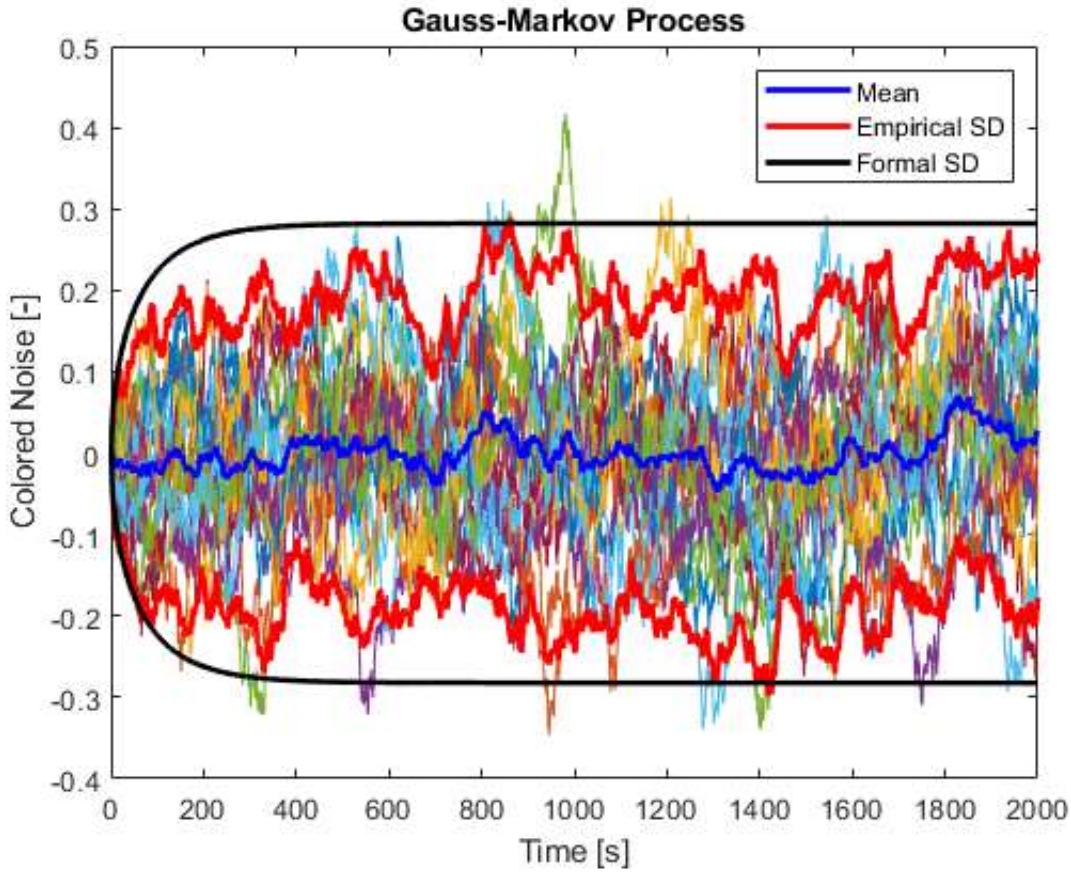
- dt = 1
- std_kx =1

```
% Specify Input Parameters:
clc; clear all;
n=2000; m=20; T=100; B = 1/T;

% Call Functions
[y, qk] = randomatrix(2000,20,100);   % Create a random matrix (y)
empirical_mean = empirimean(y);       % Get empirical statistics
empirical_stddev = empiristd(y);      % Get empirical statistics
formal_stddev = formalstd(n, T);      % Formal, not dependant on y
```

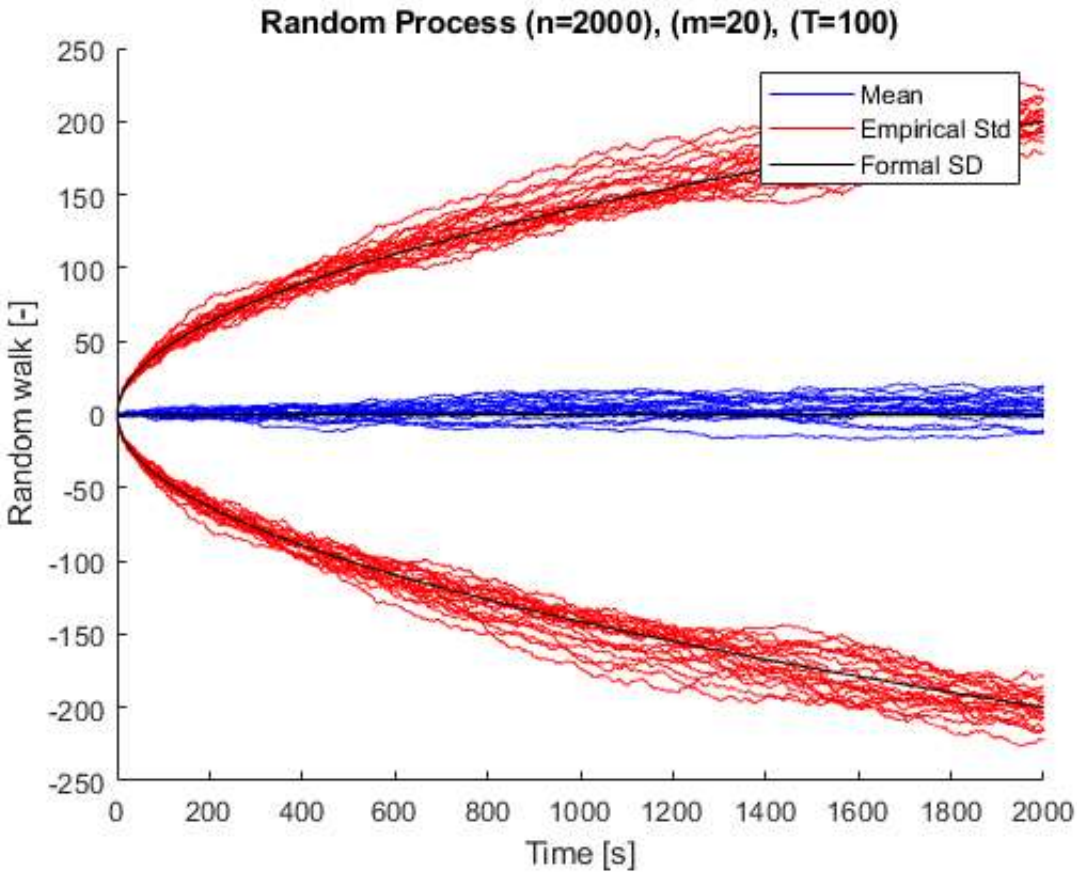
```
randoplots(y, empirical_mean, empirical_stddev,formal_stddev); % Plot
```

```
The variance (qk) for T=100 is: 0.00019801.
```



To easily summarize multiple sequences, can use the `randomanlysis` function. Note that this function removes individual signals for clarity.

```
x = randomanalysis(n, m, T);
```



II. d Correlation Periods

The last portion of this assignment is to examine seven Gauss-Markov series, by varying their correlation periods (T). The RANDOMATRIX function created in Part II.c is used to study series for T= 1, 10, 50, 100, 500, 1000, and 5000.

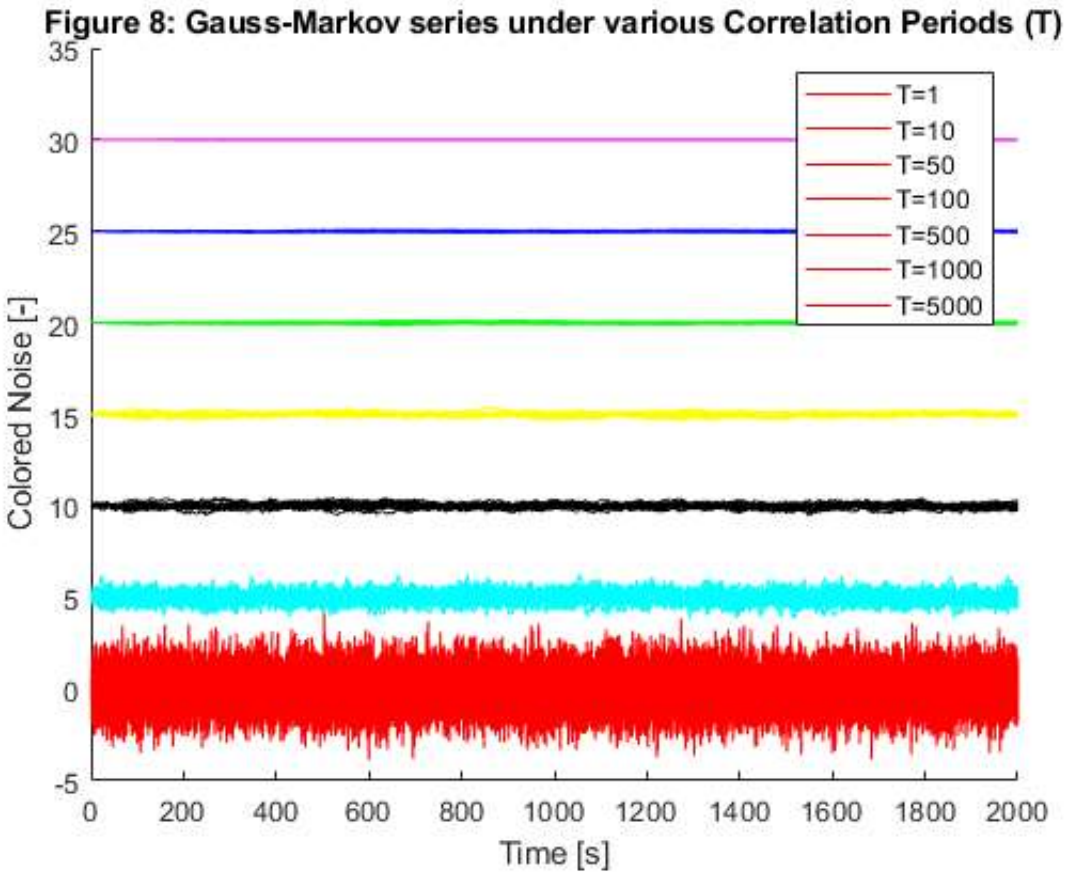
```
t_1 = randomatrix(n,m,1);
t_10 = randomatrix(n,m,10);
t_50 = randomatrix(n,m,50);
t_100 = randomatrix(n,m,100);
t_500 = randomatrix(n,m,500);
t_1000 = randomatrix(n,m,1000);
t_5000 = randomatrix(n,m,5000);

figure(5); hold on;
hh1 = plot(t_1 , 'r', 'DisplayName','T=1' );
hh2 = plot(t_10 + 5, 'c', 'DisplayName','T=10' );
hh3 = plot(t_50 + 10, 'k', 'DisplayName','T=50' );
hh4 = plot(t_100 + 15, 'y','DisplayName','T=100' );
hh5 = plot(t_500 + 20, 'g', 'DisplayName','T=500' );
hh6 = plot(t_1000 + 25, 'b', 'DisplayName','T=1000' );
hh7 = plot(t_5000 + 30, 'm', 'DisplayName','T=5000' );


% Figure Formatting
xlabel('Time [s]');
ylabel('Colored Noise [-]');
title(['Figure 8: Gauss-Markov series under various Correlation Periods (T)']);
%legend show;
%legend([hh1 hh2 hh3 hh4 hh5 hh6 hh7], 'T=1', 'T=10','T=50','T=100','T=500','T=1000','T=5000'
```

```
)
legend(hh1, 'T=1', 'T=10','T=50','T=100','T=500','T=1000','T=5000')
```

```
The variance (qk) for T=1 is: 0.86466.
The variance (qk) for T=10 is: 0.018127.
The variance (qk) for T=50 is: 0.00078421.
The variance (qk) for T=100 is: 0.00019801.
The variance (qk) for T=500 is: 7.984e-06.
The variance (qk) for T=1000 is: 1.998e-06.
The variance (qk) for T=5000 is: 7.9984e-08.
Warning: Ignoring extra legend entries.
```



Published with MATLAB® R2018b