```
%%This class is a collection of functions to expand a Brownian motion wrt.
%%to different orthonormal bases and compute the Total Mean Squared Error
%%for the approximation with the n-th partial sum.
%%written by Tim Jaschek as a part of his bachelor thesis%%

%%Used to generate FIGURE 3 and the data for TABULAR 5.1 in this thesis %%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

classdef MSE
    properties (Constant)
        %Parameter for the partial sum
        n = 20;

        %Parameter for the accuracy of the Kernels
        N = 1000;
    end
    methods (Static)
        function approximation()
            %load the classes BrMo and Kernels
            BrMo;
            Kernels;
            %%%generate a Brownian Motion with N steps%%%%%
            p = 10000; %parameter for the accuracy of the BM
            X = BrMo.BrownianMotion(p,MSE.N);
            %%%get ORTHONORMAL BASES
            %%%the analytic KLT Eigenfunctions
            %%%the HaarWavelets
            %%%Eigenfunctions of Brownian Bridge
            %%%Eigenfunctions of Exponential Kernel
            [Z,lambda,psi] = BrMo.compute_KLT_components(MSE.N);
            psi = psi.';
            Haar = BrMo.Haar(1000);
            [lambda,Bridge] = Kernels.trapez_Sceme(Kernels.KMat(2,MSE.N));
            [lambda,Exp] = Kernels.trapez_Sceme(Kernels.KMat(3,MSE.N));
            %%%Compute the Approximations%%%%%
            X_hat = MSE.Approx(X,psi(:,1:MSE.n));
            X_tilde = MSE.Approx(X,Haar(:,1:MSE.n));
            X_bridge = MSE.Approx(X,Bridge(:,1:MSE.n));
            X_exp = MSE.Approx(X,Exp(:,1:MSE.n));
```

```
        %%%plot both%%%
        figure
        hold on;
        plot(linspace(1/MSE.N,1,MSE.N),X)
        plot(linspace(1/MSE.N,1,MSE.N),X_hat)
        plot(linspace(1/MSE.N,1,MSE.N),X_tilde)
        plot(linspace(1/MSE.N,1,MSE.N),X_bridge)
        %plot(linspace(1/MSE.N,1,MSE.N),X_exp)
        hold off;
    end
    function meansquare()
        %load the classes BrMo and Kernels
        BrMo;
        Kernels;

        disp('Compute TMSE of Karhunen-Loeve-Base...')
        [Z,lambda,psi] = BrMo.compute_KLT_components(MSE.N);
        psi = psi.';
        Mse = zeros(1,MSE.N);
        for i=1:30
            X = BrMo.BrownianMotion(5000,MSE.N);
            X_hat = MSE.Approx(X,psi(:,1:MSE.n));
            for j=1:MSE.N
                Mse(j) = Mse(j) + (X(j)-X_hat(j))^2;
            end
        end
        Mse = Mse/30;
        TMse = 0;
        h=1/MSE.N;
        for j=1:MSE.N-1
                TMse = TMse + h/2 * (Mse(j) + Mse(j+1));
        end
        TMse
        %%%%%%%%Haar%%%%%%
        disp('Compute TMSE of Haar-Base...')
        Haar = BrMo.Haar(1000);
        Mse = zeros(1,MSE.N);
        for i=1:30
            X = BrMo.BrownianMotion(5000,MSE.N);
            X_hat = MSE.Approx(X,Haar(:,1:MSE.n));
            for j=1:MSE.N
                Mse(j) = Mse(j) + (X(j)-X_hat(j))^2;
            end
        end
```

```
Mse = Mse/30;
TMse = 0;
h=1/MSE.N;
for j=1:MSE.N-1
        TMse = TMse + h/2 * (Mse(j) + Mse(j+1));
end
TMse
%%%%%%%%%Bridge%%%%%%
disp('Compute TMSE of Brownian-Bridge-Base...')
[lambda,Bridge] = Kernels.trapez_Sceme(Kernels.KMat(2,MSE.N));
Mse = zeros(1,MSE.N);
for i=1:30
    X = BrMo.BrownianMotion(5000,MSE.N);
    X_hat = MSE.Approx(X,Bridge(:,1:MSE.n));
    for j=1:MSE.N
        Mse(j) = Mse(j) + (X(j)-X_hat(j))^2;
    end
end
Mse = Mse/30;
TMse = 0;
h=1/MSE.N;
for j=1:MSE.N-1
        TMse = TMse + h/2 * (Mse(j) + Mse(j+1));
end
TMse
%%%%%%%%%Exponential%%%%%%
disp('Compute TMSE of Exponential-Base...')
[lambda,Exp] = Kernels.trapez_Sceme(Kernels.KMat(3,MSE.N));
Mse = zeros(1,MSE.N);
for i=1:30
    X = BrMo.BrownianMotion(5000,MSE.N);
    X_hat = MSE.Approx(X,Exp(:,1:MSE.n));
    for j=1:MSE.N
        Mse(j) = Mse(j) + (X(j)-X_hat(j))^2;
    end
end
Mse = Mse/30;
TMse = 0;
h=1/MSE.N;
for j=1:MSE.N-1
        TMse = TMse + h/2 * (Mse(j) + Mse(j+1));
end
```

```
            TMse
        end
        function X_hat = Approx(X,Phi);
            %%assume X has N steps on [0,1] and we have at least n Eigenvectors with N steps each%%%
            X_hat = zeros(1,MSE.N);
            A = zeros(1,MSE.n);
            h = 1/MSE.N;
            for i=1:MSE.n
                %%%compute the integrals via trapez sceme%%%
                for j=1:MSE.N-1
                    A(i) = A(i) + h/2 * (X(j)*Phi(j,i) + X(j+1)*Phi(j+1,i));
                end
            end
            for i=1:MSE.N
                X_hat(i) = dot(A,Phi(i,1:MSE.n));
            end
        end
    end
end
```