Mathematical Modelling in the Natural Sciences SS21
Solutions to Exercises on Sheet 10

Exercises und Lecture Notes

Contents

• Exercise 1: Brownian Motion: Random Walk

◦ Task

Brownian motion of particles in a thin pipe is simulated. Within a a time interval of length Δt a particle can move left or right or remain stil. For such a step let δ be a random variable with

$$P(\delta = -1) = \alpha, \quad P(\delta = 0) = 1 - 2\alpha, \quad P(\delta = +1) = \alpha,$$

where $\alpha \in (0, \frac{1}{2}]$. Let $\{\delta_k\}_{k=1}^{\infty}$ be random variables which are independent and all distributed just as δ. Let $X(t; \Delta t)$ be a random variable representing the position of a Brownian particle at time t where

$$X(k \cdot \Delta t; \Delta t) = X((k-1) \cdot \Delta t; \Delta t) + \sqrt{D\Delta t/(2\alpha)} \cdot \delta_k, \quad k \in \mathbb{N}, \quad X(0; \Delta t) = 0.$$

(a) Show that for $\Delta t \to 0$, $X(t; \Delta t) \sim N(0, Dt)$, i.e., for $D = 1$, X approximates a Wiener process.

(b) Repeat the previous derivation but for many Brownian particles and show that the random variable $R_n(t; \Delta t, a, b)$ representing the relative number of n particles in an interval $[a, b)$ satisfies

$$P\left(\left|R_n(t; \Delta t, a, b) - \frac{1}{\sqrt{2\pi Dt}} \int_a^b e^{-x^2/(2Dt)} dx\right| < \epsilon\right) \overset{n\to\infty}{\longrightarrow} 1, \quad \forall \epsilon > 0.$$

(c) Carry out a Monte-Carlo simulation for many Brownian particles and compare the simulated densities with the theoretical results.

(d) From this Monte-Carlo simulation compute the numerical spectrum (`fft`) of the numerical derivative $\xi(t) = D_t X(t; \Delta t)$ (finite differences) for each particle and estimate the autocorrelation function of the white noise.

◦ Solution

* Part (a) Distribution, Position of Single Particle

The random variable δ satisfies

$$\mu_\delta = \mathbb{E}[\delta] = 0, \quad \sigma_\delta^2 = \mathbb{E}[\delta^2] = 2\alpha$$

and the random variable

$$X(t; \Delta t) = \sqrt{D\Delta t/(2\alpha)} \sum_{k=1}^{\lfloor t/\Delta t \rfloor} \delta_k$$

satisfies

$$\mu_{X(t;\Delta t)} = \mathbb{E}[X(t;\Delta t)] = \sqrt{D\Delta t/(2\alpha)} \sum_{k=1}^{\lfloor t/\Delta t \rfloor} \mu_{\delta_k} = 0$$

$$\sigma^2_{X(t;\Delta t)} = \mathbb{E}[X(t;\Delta t)^2] = D\Delta t/(2\alpha) \sum_{k=1}^{\lfloor t/\Delta t \rfloor} \sigma^2_{\delta_k} = \frac{\lfloor t/\Delta t \rfloor}{t/\Delta t} Dt.$$

By the central limit theorem,

$$P(a \le X(t;\Delta t) \le b) =$$

$$P\left(\sqrt{\frac{t/\Delta t}{\lfloor t/\Delta t \rfloor}} \frac{a}{\sqrt{Dt}} \le \frac{X(t;\Delta t) - \mu_{X(t;\Delta t)}}{\sigma_{X(t;\Delta t)}} \le \sqrt{\frac{t/\Delta t}{\lfloor t/\Delta t \rfloor}} \frac{b}{\sqrt{Dt}}\right)$$

$$\overset{\Delta t \to 0}{\longrightarrow} P\left(\frac{a}{\sqrt{Dt}} \le Z \le \frac{b}{\sqrt{Dt}}\right), \quad Z \sim N(0,1), \quad a < b, \quad a, b \in \mathbb{R}.$$

It follows that

$$P(a \le X(t;\Delta t) \le b) \overset{\Delta t \to 0}{\longrightarrow} \frac{1}{\sqrt{2\pi}} \int_{a/\sqrt{Dt}}^{b/\sqrt{Dt}} e^{-x^2/2} dx = \frac{1}{\sqrt{2\pi Dt}} \int_a^b e^{-x^2/(2Dt)} dx.$$

∗ Part (b) Distribution, Relative Number of Particles

For the representation of the event that the particle can be found in a spatial interval $[a, b)$, let $Y(t;\Delta t, a, b)$ be the random variable

$$Y(t;\Delta t, a, b) = \begin{cases} 1, & X(t;\Delta t) \in [a, b) \\ 0, & \text{else} \end{cases}$$

whose distribution can be approximated as Bernoulli,

$$P(Y(t;\Delta t, a, b) = 1) = P(X(t;\Delta t) \in [a, b)) \overset{\Delta t \to 0}{\longrightarrow} q(t; a, b)$$

$$P(Y(t;\Delta t, a, b) = 0) \overset{\Delta t \to 0}{\longrightarrow} 1 - q(t; a, b)$$

where

$$q(t; a, b) = \frac{1}{\sqrt{2\pi Dt}} \int_a^b e^{-x^2/(2Dt)} dx.$$

Therefore, the following hold approximately,

$$\mu_{Y(t;\Delta t,a,b)} = \mathbb{E}[Y(t;\Delta t, a, b)] \overset{\Delta t \to 0}{\longrightarrow} q(t; a, b)$$

$$\sigma^2_{Y(t;\Delta t,a,b)} = \mathbb{V}[Y(t;\Delta t, a, b)] \overset{\Delta t \to 0}{\longrightarrow} q(t; a, b)(1 - q(t; a, b)).$$

Now let $\{X_1(t;\Delta t), X_2(t;\Delta t), \dots\}$ be random variables which are independent and all distributed just as $X(t;\Delta t)$, where $X_i(t;\Delta t)$ represents the position of an ith Brownian particle at time t. Let $\{\delta_{i,k}\}$ be random variables which are independent and all distributed just as δ. At the kth step the random position of the ith is given with $D = 1$ by

$$X_i(k \cdot \Delta t; \Delta t) = X_i((k-1) \cdot \Delta t; \Delta t) + \sqrt{D\Delta t/(2\alpha)} \cdot \delta_{i,k}, \quad i, k \in \mathbb{N}, \quad X_i(0; \Delta t) = 0.$$

For the representation of whether the ith particle, $i \in \mathbb{N}$, can be found in the spatial interval $[a, b)$, let the random variables

$$Y_i(t; \Delta t, a, b) = \begin{cases} 1, & X_i(t) \in [a, b) \\ 0, & \text{sonst} \end{cases}$$

be given, which are independent and all distributed as $Y(t; \Delta t, a, b)$. Let

$$R_n(t; \Delta t, a, b) = \frac{1}{n} \sum_{i=1}^{n} Y_i(t; \Delta t, a, b)$$

be the random relative number of n particles which can be found in the spatial interval $[a, b)$ at time t. Then based upon previous estimates the following hold

$$\mu_{R_n(t;\Delta t,a,b)} = \mathbb{E}[R_n(t; \Delta t, a, b)] = \frac{1}{n} \sum_{i=1}^{n} \mu_{Y_i(t;\Delta t,a,b)} \overset{\Delta t \to 0}{\longrightarrow} q(t; a, b)$$

$$\sigma^2_{R_n(t;\Delta t,a,b)} = \mathbb{V}[R_n(t; \Delta t, a, b)] = \frac{1}{n^2} \sum_{i=1}^{n} \sigma^2_{Y_i(t:\Delta t,a,b)} \overset{\Delta t \to 0}{\longrightarrow} \frac{1}{n} q(t; a, b)(1 - q(t; a, b)).$$

It follows then from the central limit theorem that

$$P\left(\left|R_n(t; \Delta t, a, b) - \frac{1}{\sqrt{2\pi D t}} \int_a^b e^{-x^2/(2Dt)} dx\right| < \epsilon\right) = P\left(|R_n(t; \Delta t, a, b) - q(t; a, b)| < \epsilon\right)$$

$$\overset{\Delta t \to 0}{\longrightarrow} P\left(\frac{-\epsilon}{\sqrt{q(t; a, b)(1 - q(t; a, b))/n}} \leq \frac{R_n(t; \Delta t, a, b) - \mu_{R_n(t;\Delta t,a,b)}}{\sigma_{R_n(t;\Delta t,a,b)}} \leq \frac{\epsilon}{\sqrt{q(t; a, b)(1 - q(t; a, b))/n}}\right)$$

$$\overset{n \to \infty}{\longrightarrow} P\left(\frac{-\epsilon\sqrt{n}}{\sqrt{q(t; a, b)(1 - q(t; a, b))}} \leq Z \leq \frac{\epsilon\sqrt{n}}{\sqrt{q(t; a, b)(1 - q(t; a, b))}}\right), \quad Z \sim N(0, 1)$$

or with

$$\eta(t; a, b) = \frac{1}{\sqrt{2\pi D t}} \int_a^b e^{-x^2/(2Dt)} dx = \frac{1}{2}\left[\operatorname{erf}\left(\frac{b}{\sqrt{2Dt}}\right) - \operatorname{erf}\left(\frac{a}{\sqrt{2Dt}}\right)\right]$$

it follows

$$P\left(|R_n(t; \Delta t, a, b) - \eta(t; a, b)| < \epsilon\right) \overset{n \to \infty}{\longrightarrow} 1, \quad \forall \epsilon > 0.$$

The following Matlab code is used for parts (c) and (d).

```
h1 = figure(1); close(h1); h1 = figure(1);
set(h1,'Position',[20 20 1200 400]);          % setup figure

Np = 1000;                                    % number of particles
Nt = 128;                                     % number of time steps
T  = 1;                                       % final time
ti = linspace(0,T,Nt+1)';                     % time cell interfaces
ht = ti(2)-ti(1);                             % time step
tc = ti(1:Nt) + ht/2;                         % time cell centers
oi = linspace(0,2*pi,Nt+1);                   % frequency cell interfaces
ho = oi(2)-oi(1);                             % frequency step
om = oi(1:Nt) + ho/2;                         % frequency cell centers
```

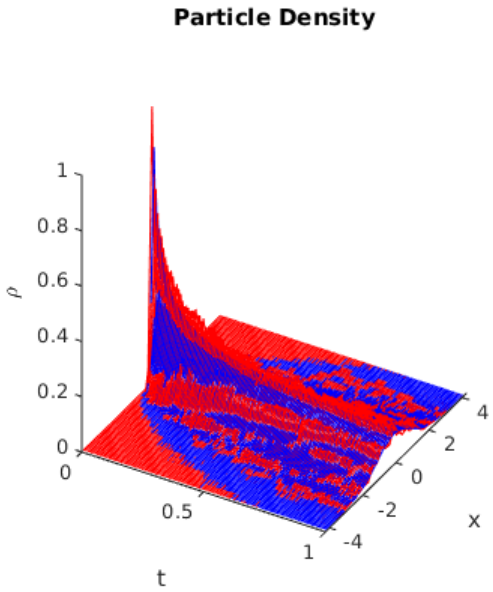
```
D  = 1;                                    % diffusivity
al = 1/3;                                  % P(delta = 0) = 1-2*al

X  = zeros(1,Np);                          % particle positions
Xv = X;

xmax = 3*sqrt(2*D*T);                      % spatial interval is
xmin = -xmax;                              %     [xmin,xmax]
Nx   = 2*round(round(sqrt(Np))/2)+1;       % number of spatial cells

xi = linspace(xmin,xmax,Nx+1);             % cell interfaces
hx = xi(2)-xi(1);
xc = xi(1:Nx)+hx/2;                        % cell centers

for i=1:Nt
    Xa = X;                                % save for stopping
    z  = rand(1,Np);                       % random, uniform [0,1]
    de = -(z < al) + (z > 1-al);           % step direction
    X  = X + sqrt(D*ht/(2*al))*de;         % new position
    Xv = [Xv;X];

    [Ni,yi] = histcounts(X,xi);            % numbers of particles in cells
    R = Ni/Np;                             % relative number
    r = (erf(xi(2:Nx+1)/sqrt(2*D*ti(i))) ... % theoretical prediction of R
       - erf(xi(1:Nx  )/sqrt(2*D*ti(i))))/2;

    subplot(1,3,1)                         % plot distributions
    hold on
    plot3(ti(i)*ones(size(xc)),xc,r,'b',ti(i)*ones(size(xc)),xc,R,'r')
    axis([0 T xmin xmax 0 1])
    view([30,30])
    title('Particle Density')
    xlabel('t')
    ylabel('x')
    zlabel('\rho')
    hold off;

    subplot(1,3,2)                         % plot positions
    hold on
    plot(X,ti(i)*ones(1,Np),'.')
    axis([xmin xmax 0 1])
    title('Particle Positions')
    xlabel('x')
    ylabel('t')
    hold off

    subplot(1,3,3)                         % plot variance
```

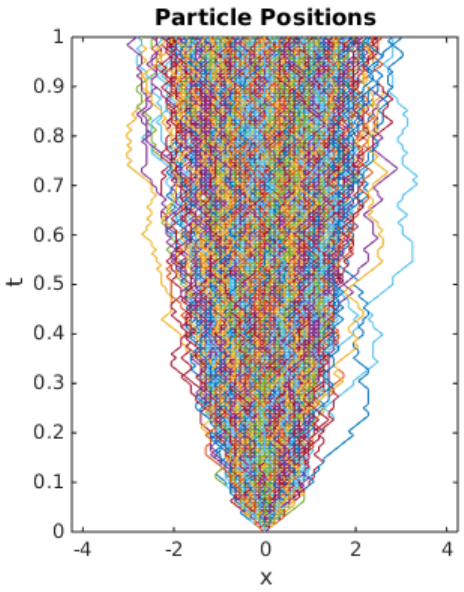
```
    plot(ti(1:i+1),std(Xv').^2)
    axis([0 T 0 T])
    title('Variance of Motion')
    xlabel('t')
    ylabel('E[X(t)^2]')

    drawnow;
end

h2 = figure(2); close(h2); h2 = figure(2);
set(h2,'Position',[20 20 1200 400]);         % setup figure

Xi = (Xv(2:Nt+1,:)-Xv(1:Nt,:))/ht;           % white noise
subplot(1,3,1)
plot(tc,Xi)
axis([0 T min(Xi(:)) max(Xi(:))])
xlabel('t')
ylabel('\xi(t)')
Xmu = mean(mean(Xi));                        % mean of time courses
Xsg = mean(std(Xi));                         % std of time courses
hold on;
plot(tc,Xmu*ones(1,Nt),'r', ...
     tc,(Xmu+Xsg)*ones(1,Nt),'g', ...
     tc,(Xmu-Xsg)*ones(1,Nt),'g', ...
     'LineWidth',2);
title(sprintf('White Noise\nE+/-S=%0.1e+/-%0.1e',Xmu,Xsg))
hold off

Ze = fft(Xi);                                % fft of time courses
Ze = Ze.*conj(Ze)/Nt;                        % spectrum of time courses
subplot(1,3,2)
plot(om,Ze)
hold on
Zmu = mean(mean(Ze));                        % mean of spectra
Zsg = mean(std(Ze));                         % std of spectra
plot(om,Zmu*ones(1,Nt),'r', ...
     om,(Zmu+Zsg)*ones(1,Nt),'g', ...
     om,(Zmu-Zsg)*ones(1,Nt),'g', ...
     'LineWidth',2);
axis([0 2*pi min(Ze(:)) max(Ze(:))])
xlabel('\omega')
ylabel('|F[\xi](\omega)|^2')
title(sprintf('Spectrum of White Noise\n E+/-S=%0.1e+/-%0.1e', ...
    Zmu,Zsg))
hold off

C = Xi*Xi'/Np;                               % autocorrelation matrix
subplot(1,3,3)
```

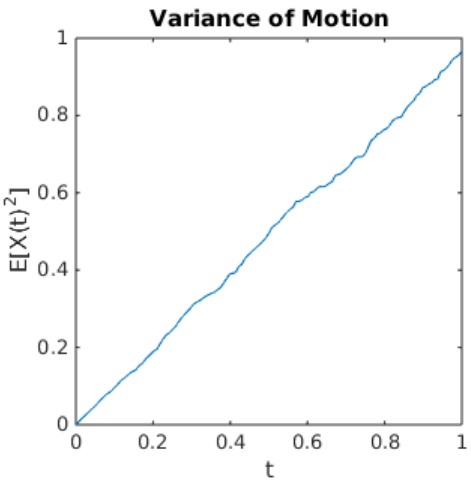
```
surf(tc,tc,C)
axis([0 T 0 T min(C(:)) max(C(:))])
xlabel('t')
ylabel('t')
zlabel('c(t_i-t_j)=E[\xi(t_i)\xi(t_j)]')
title('Autocorrelation Function')
```

* **Part (c) Simulation, Comparison to Theory**

Shown on the left are particle densities, where the theoretical prediction is shown in blue and the result of the Monte-Carlo simulation is shown in red. The simulated trajectories of Brownian particles are shown in the middle. Shown on the right is the expected variance $\mathbb{E}[X(t)^2] = Dt$ with $D = 1$. Note that with all other parameters held fixed, the match between the simulated and theoretical densities is best with $\alpha = 1/3$.



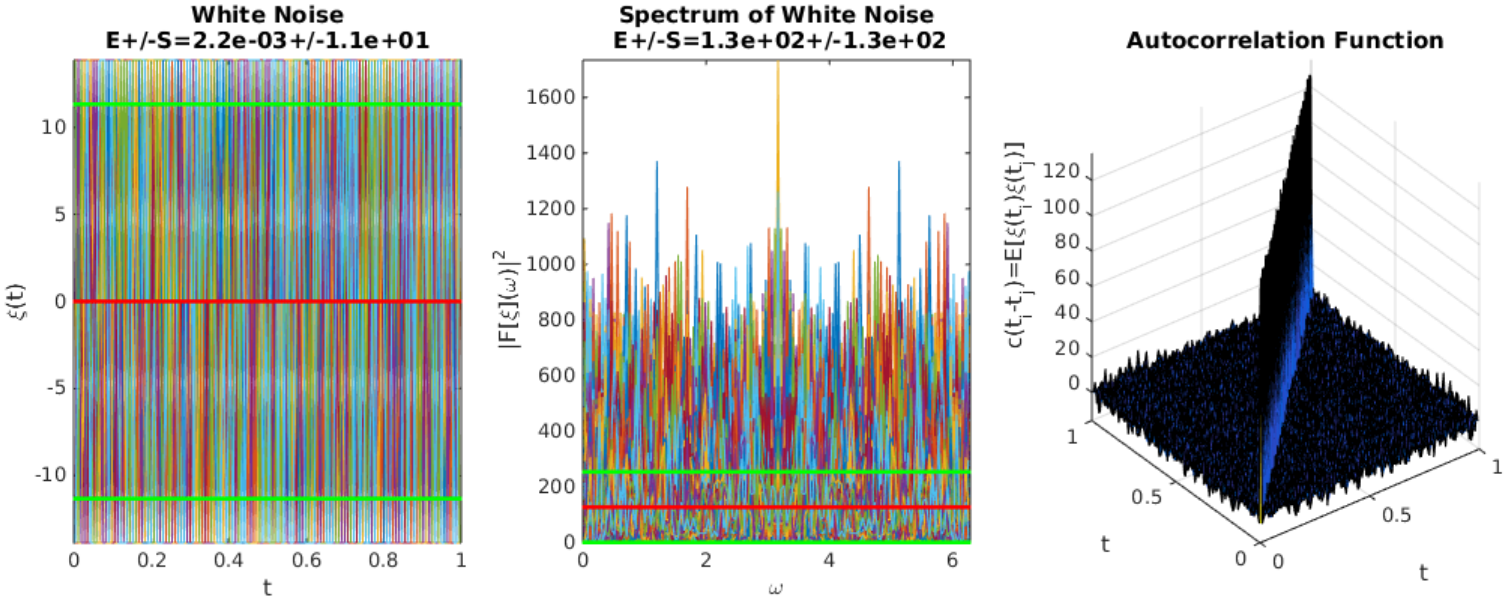




* **Part (d) White Noise and Autocorrelation**

In the next graphic the white noise $\xi(t) = D_t X(t; \Delta t)$ for each particle is shown on the left with a different color for each particle. The spectrum of each white noise time course is shown in the middle with a different color for each particle. Note that the spectrum is roughly flat, i.e., each frequency contributes more or less equally, where the mean energy is shown in red and the standard deviation in green. The autocorrleation function is shown on the right, where the peaks along the diagonal

indicate the independence of one time course from another, as predicted theoretically.



- ## Exercise 2: Brownian Motion: Diffusion

  - ### Task

Brownian motion of particles in a thin pipe is simulated with a continuous process. Let $X(t)$ be a random variable representing the position of a particle at time t with

$$P(X(t) \in [a,b]) = \int_a^b \rho(\xi,t)d\xi.$$

Let $\delta(\tau)$ be a random variable representing the change in position of the particle in a time interval of length τ with

$$P(\delta(\tau) \in [a,b]) = \int_a^b f(\xi,\tau)d\xi.$$

(a) Show that the densities ρ and f are related by the convolution,

$$\rho(x,t+\tau) = \int_{-\infty}^{+\infty} \rho(x-y,t)f(y,\tau)dy$$

and with $f(-x,\tau) = f(x,\tau)$ and $D\tau = \int_{-\infty}^{+\infty} x^2 f(x,\tau)dx$ it follows that

$$f(x,t) = \frac{e^{-x^2/(2Dt)}}{\sqrt{2\pi Dt}}.$$

(b) Carry out a Monte-Carlo simulation for many Brownian particles and compare the simulated densities with the theoretical results.

(c) From this Monte-Carlo simulation compute the numerical spectrum (`fft`) of the numerical derivative $\xi(t) = D_t X(t;\Delta t)$ (finite differences) for each particle and estimate the autocorrelation function of the white noise.

◦ Solution

∗ Part (a) Distribution, Position and Perturbation

It is assumed that f is even so that δ does not prefer one direction over the other. Once the probability density $\rho(x,t)$ for $X(t)$ is given, then it follows from

$$P(a \le X(t+\tau) \le b) = P(a \le X(t) + \delta(\tau) \le b) =$$
$$\int_{-\infty}^{+\infty} P(a \le X(t) + \xi \le b) f(\xi,\tau) d\xi = \int_{-\infty}^{+\infty} \left[\int_{a-\xi}^{b-\xi} \rho(\eta,t) d\eta \right] f(\xi,\tau) d\xi$$
$$= \int_a^b \left[\int_{-\infty}^{+\infty} \rho(x-\xi,t) f(\xi,\tau) d\xi \right] dx$$

that the probability density $\rho(x,t+\tau)$ for $X(t+\tau)$ is given by

$$\rho(x,t+\tau) = \int_{-\infty}^{+\infty} \rho(x-\xi,t) f(\xi,\tau) d\xi.$$

A Taylor expansion gives

$$\rho(x,t) + \tau\rho_t(x,t) + \cdots = \rho(x,t+\tau) = \int_{-\infty}^{+\infty} [\rho(x,t) - \rho_x(x,t)\xi + \rho_{xx}(x,t)\xi^2/2 + \cdots] f(\xi,\tau) d\xi$$
$$= \rho(x,t) \underbrace{\int_{-\infty}^{+\infty} f(\xi,\tau) d\xi}_{=1} - \rho_x(x,t) \underbrace{\int_{-\infty}^{+\infty} \xi f(\xi,\tau) d\xi}_{=0} + \rho_x(x,t) \int_{-\infty}^{+\infty} \tfrac{1}{2}\xi^2 f(\xi,\tau) d\xi + \cdots$$

where the integrals can be so simplified, first since f is a probability density, $\int_{-\infty}^{+\infty} f(\xi,\tau) d\xi = 1$ and secondly because f is even in ξ, i.e, $f(-\xi,\tau) = f(+\xi,\tau)$. There results

$$\rho_t(x,t) = D\rho_{xx}(x,t) \quad \text{mit} \quad D = \int_{-\infty}^{+\infty} \frac{\xi^2}{2\tau} f(\xi,\tau) d\xi.$$

With the initial conditions

$$\rho^\epsilon(x,0) = \begin{cases} 1/(2\epsilon), & x \in [-\epsilon, +\epsilon] \\ 0, & \text{otherwise} \end{cases}$$

satisfies

$$\rho^\epsilon(x,t) = \frac{1}{4\epsilon} \left[\operatorname{erf}\left(\frac{x+\epsilon}{\sqrt{4Dt}} \right) - \operatorname{erf}\left(\frac{x-\epsilon}{\sqrt{4Dt}} \right) \right], \quad \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-z^2} dz$$

the partial differential equation and the initial conditions, as an explicit calculation confirms. Moreover it holds for $t > 0$,

$$\lim_{\epsilon\to 0} \rho^\epsilon(x,t) = \lim_{\epsilon\to 0} \frac{1}{4\epsilon} \left[\operatorname{erf}\left(\frac{x+\epsilon}{\sqrt{2Dt}} \right) - \operatorname{erf}\left(\frac{x-\epsilon}{\sqrt{2Dt}} \right) \right] = \frac{1}{2} \frac{d}{dx} \operatorname{erf}\left(\frac{x}{\sqrt{2Dt}} \right) = \frac{e^{-x^2/(4Dt)}}{\sqrt{4\pi Dt}}$$

and therefore f is given by

$$\frac{e^{-x^2/(4D\tau)}}{\sqrt{4\pi D\tau}} \overset{\epsilon\to 0}{\longleftarrow} \rho^\epsilon(x,0+\tau) = \int_{-\infty}^{+\infty} \rho^\epsilon(x-\xi,0) f(\xi,\tau) d\xi = \frac{1}{2\epsilon} \int_{x-\epsilon}^{x+\epsilon} f(\xi,\tau) d\xi \overset{\epsilon\to 0}{\longrightarrow} f(x,\tau).$$

One confirms

$$\int_{-\infty}^{+\infty} \frac{\xi^2}{2\tau} f(\xi,\tau) d\xi = \frac{2D}{\sqrt{\pi}} \int_{-\infty}^{+\infty} \frac{\xi^2}{4D\tau} e^{-\xi^2/(4D\tau)} \frac{d\xi}{\sqrt{4Dt}} \overset{z=\xi/\sqrt{4D\tau}}{=} \frac{2D}{\sqrt{\pi}} \int_{-\infty}^{+\infty} z^2 e^{-z^2} dz$$

$$= \frac{2D}{\sqrt{\pi}} \left[-\frac{1}{2} z e^{-z^2} |_{-\infty}^{+\infty} + \frac{1}{2} \int_{-\infty}^{+\infty} e^{-z^2} dz \right] = \frac{D}{2} [\operatorname{erf}(+\infty) - \operatorname{erf}(-\infty)] = D$$

where

$$\frac{1}{2}\left[\mathrm{erf}(+\infty)-\mathrm{erf}(-\infty)\right]=\frac{1}{\sqrt{\pi}}\int_{-\infty}^{+\infty}e^{-z^2}dz=\left\{\left[\frac{1}{\sqrt{\pi}}\int_{-\infty}^{+\infty}e^{-x^2}dx\right]\left[\frac{1}{\sqrt{\pi}}\int_{-\infty}^{+\infty}e^{-y^2}dy\right]\right\}^{\frac{1}{2}}$$
$$\left\{\frac{1}{\pi}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}e^{-(x^2+y^2)}dxdy\right\}^{\frac{1}{2}}=\left\{\frac{1}{\pi}\int_{0}^{2\pi}\int_{0}^{+\infty}re^{-r^2}drd\theta\right\}^{\frac{1}{2}}=\left\{-e^{-r^2}|_0^{\infty}\right\}^{\frac{1}{2}}=1.$$

It follows in particular

$$\rho(x,\tau)=\int_{-\infty}^{+\infty}\rho(x-\xi,0)f(\xi,\tau)d\xi=\int_{-\infty}^{+\infty}\rho(x-\xi,0)\frac{e^{-\xi^2/(4D\tau)}}{\sqrt{4\pi D\tau}}d\xi.$$

∗ Part (b) Simulation, Comparison to Theory

The following Matlab code is used for parts (b) and (c).

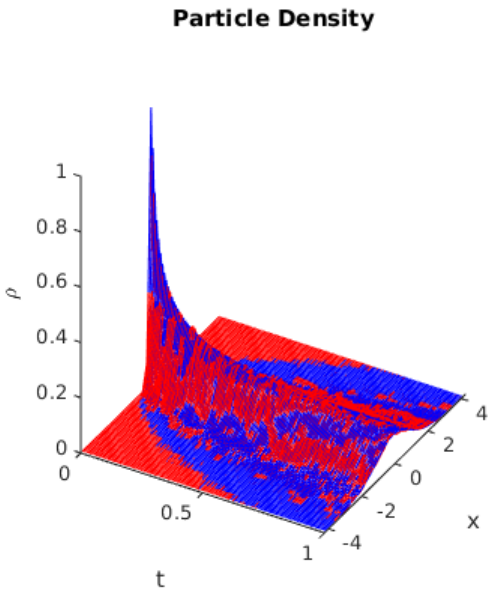
```
h1 = figure(1); close(h1); h1 = figure(1);
set(h1,'Position',[20 20 1200 400]);          % setup figure

Np = 1000;                                    % number of particles
Nt = 128;                                     % number of time steps
T  = 1;                                       % final time
ti = linspace(0,T,Nt+1)';                     % time cell interfaces
ht = ti(2)-ti(1);                             % time step
tc = ti(1:Nt) + ht/2;                         % time cell centers
oi = linspace(0,2*pi,Nt+1);                   % frequency cell interfaces
ho = oi(2)-oi(1);                             % frequency step
om = oi(1:Nt) + ho/2;                         % frequency cell centers

D  = 1;                                       % diffusivity

X  = zeros(1,Np);                             % particle positions
Xv = X;

xmax = 3*sqrt(2*D*T);                         % spatial interval is
xmin = -xmax;                                 %     [xmin,xmax]
Nx   = 2*round(round(sqrt(Np))/2)+1;          % number of spatial cells

xi = linspace(xmin,xmax,Nx+1);                % cell interfaces
hx = xi(2)-xi(1);
xc = xi(1:Nx)+hx/2;                           % cell centers

for i=1:Nt
    Xa = X;                                   % save for stopping
    de = randn(1,Np);                         % random, N(0,1)
    X  = X + sqrt(D*ht)*de;                   % new position
    Xv = [Xv;X];

    [Ni,yi] = histcounts(X,xi);               % numbers of particles in cells
```

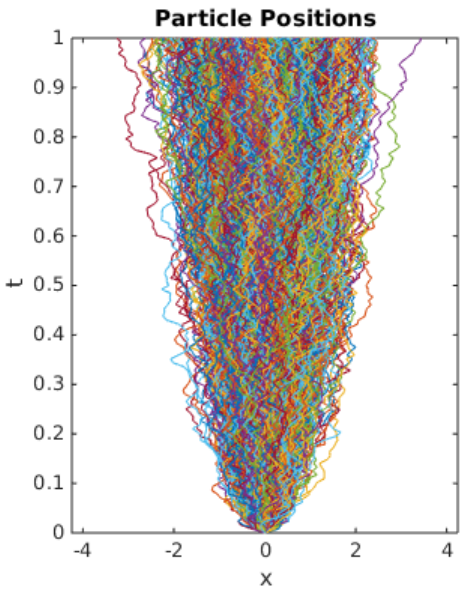
```
    R = Ni/Np;                                % relative number
    r = (erf(xi(2:Nx+1)/sqrt(2*D*ti(i))) ... % theoretical prediction of R
       - erf(xi(1:Nx  )/sqrt(2*D*ti(i))))/2;

    subplot(1,3,1)                            % plot distributions
    hold on
    plot3(ti(i)*ones(size(xc)),xc,r,'b',ti(i)*ones(size(xc)),xc,R,'r')
    axis([0 T xmin xmax 0 1])
    view([30,30])
    title('Particle Density')
    xlabel('t')
    ylabel('x')
    zlabel('\rho')
    hold off;

    subplot(1,3,2)                            % plot positions
    hold on
    plot(X,ti(i)*ones(1,Np),'.')
    axis([xmin xmax 0 1])
    title('Particle Positions')
    xlabel('x')
    ylabel('t')
    hold off

    subplot(1,3,3)                            % plot variance
    plot(ti(1:i+1),std(Xv').^2)
    axis([0 T 0 T])
    title('Variance of Motion')
    xlabel('t')
    ylabel('E[X(t)^2]')

    drawnow;
end

h2 = figure(2); close(h2); h2 = figure(2);
set(h2,'Position',[20 20 1200 400]);          % setup figure

Xi = (Xv(2:Nt+1,:)-Xv(1:Nt,:))/ht;            % white noise
subplot(1,3,1)
plot(tc,Xi)
axis([0 T min(Xi(:)) max(Xi(:))])
xlabel('t')
ylabel('\xi(t)')
Xmu = mean(mean(Xi));                         % mean of time courses
Xsg = mean(std(Xi));                          % std of time courses
hold on;
plot(tc,Xmu*ones(1,Nt),'r', ...
     tc,(Xmu+Xsg)*ones(1,Nt),'g', ...
```

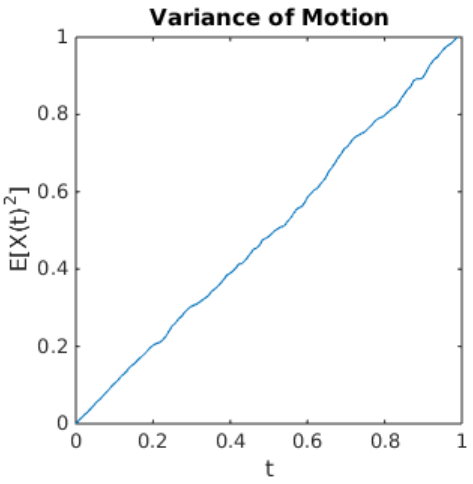
```
     tc,(Xmu-Xsg)*ones(1,Nt),'g', ...
     'LineWidth',2);
title(sprintf('White Noise\nE+/-S=%0.1e+/-%0.1e',Xmu,Xsg))
hold off

Ze = fft(Xi);                                % fft of time courses
Ze = Ze.*conj(Ze)/Nt;                        % spectrum of time courses
subplot(1,3,2)
plot(om,Ze)
hold on
Zmu = mean(mean(Ze));                        % mean of spectra
Zsg = mean(std(Ze));                         % std of spectra
plot(om,Zmu*ones(1,Nt),'r', ...
     om,(Zmu+Zsg)*ones(1,Nt),'g', ...
     om,(Zmu-Zsg)*ones(1,Nt),'g', ...
     'LineWidth',2);
axis([0 2*pi min(Ze(:)) max(Ze(:))])
xlabel('\omega')
ylabel('|F[\xi](\omega)|^2')
title(sprintf('Spectrum of White Noise\n E+/-S=%0.1e+/-%0.1e', ...
    Zmu,Zsg))
hold off

C = Xi*Xi'/Np;                               % autocorrelation matrix
subplot(1,3,3)
surf(tc,tc,C)
axis([0 T 0 T min(C(:)) max(C(:))])
xlabel('t')
ylabel('t')
zlabel('c(t_i-t_j)=E[\xi(t_i)\xi(t_j)]')
title('Autocorrelation Function')
```

Shown on the left are particle densities, where the theoretical prediction is shown in blue and the result of the Monte-Carlo simulation is shown in red. The simulated trajectories of Brownian particles

are shown in the middle. Shown on the right is the expected variance $\mathbb{E}[X(t)^2] = Dt$ with $D = 1$.

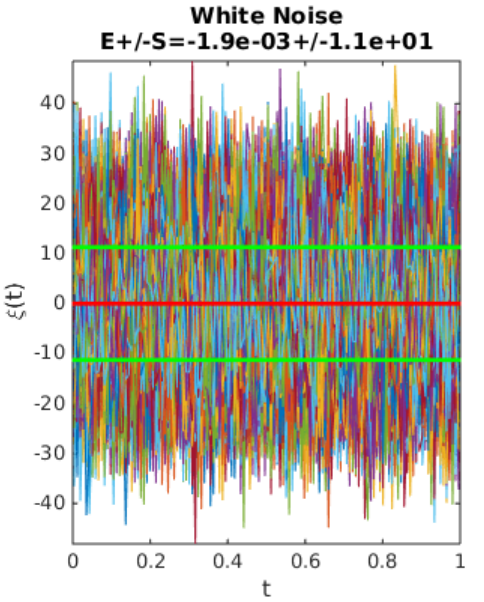


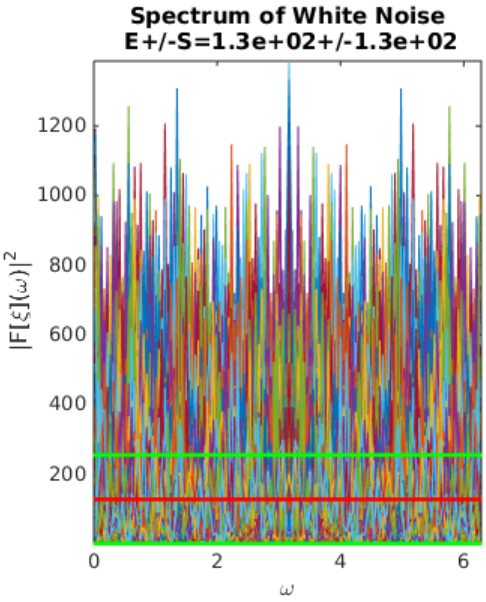


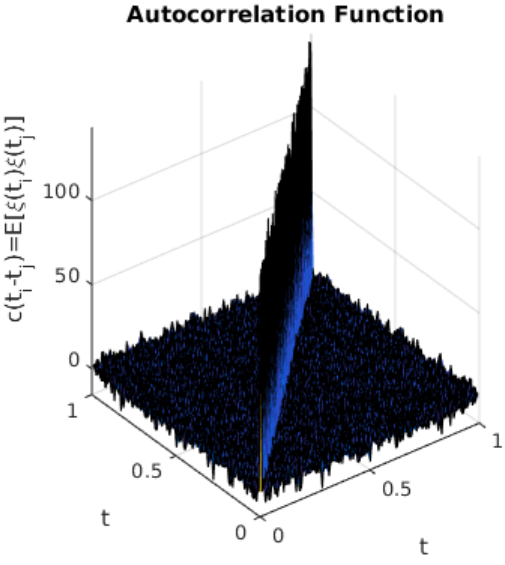


∗ **Part (c) White Noise and Autocorrelation**

In the next graphic the white noise $\xi(t) = D_t X(t; \Delta t)$ for each particle is shown on the left with a different color for each particle. The spectrum of each white noise time course is shown in the middle with a different color for each particle. Note that the spectrum is roughly flat, i.e., each frequency contributes more or less equally, where the mean energy is shown in red and the standard deviation in green. The autocorrleation function is shown on the right, where the peaks along the diagonal indicate the independence of one time course from another, as predicted theoretically.







• Exercise 3: Brownian Motion: Random Basis Expansion

◦ Task

Let the Haar functions be given by

$$h_0(t) = \chi_{[0,1]}(t), \quad h_1(t) = \chi_{[0,1/2]}(t) - \chi_{[1/2,1]}(t)$$

$$h_k(t) = \begin{cases} 2^{n/2}, & (k-2^n) & < 2^n t < & (k-2^n+1/2) \\ -2^{n/2}, & (k-2^n+1/2) & < 2^n t < & (k-2^n+1) \\ 0, & \text{otherwise.} & & \end{cases} \qquad n = \lfloor \log_2(k) \rfloor$$

Let the Schauder functions be given by

$$s_k(t) = \int_0^t h_k(\tau) d\tau, \quad k \in \mathbb{N}_0$$

For $K \in \mathbb{K}$ let $\{A_k\}_{k=0}^K$ be a sequence of random variables which are independent and identically distributed with $A_k \sim N(0, D)$ with $D > 0$ or typically $D = 1$. Define

$$W(t) = \sum_{k=0}^{K} A_k s_k(t).$$

(a) Show that the Haar functions are given by

$$h_k(t) = 2^{\frac{\lfloor \log_2(k) \rfloor}{2}} h_1(2^{\lfloor \log_2(k) \rfloor}(t+1) - k), \quad k \in \mathbb{N}.$$

and that the Schauder functions are given by

$$s_k(t) = 2^{-\frac{\lfloor \log_2(k) \rfloor}{2}} s_1(2^{\lfloor \log_2(k) \rfloor}(t+1) - k), \quad k \in \mathbb{N}_0.$$

(b) Show that for a temporal grid $\{t_i = i/N_t\}_{i=0}^{N_t}$ with $\log_2(N_t) \in \mathbb{N}$, the Schauder functions evaluate to $s_k(t_i) = 0$ for $k > K = N_t - 1$.

(c) Write a Matlab code to compute the approximate Wiener process $W(t)$ on a temporal grid $\{t_i = i/N_t\}_{i=0}^{N_t}$ with $\log_2(N_t) \in \mathbb{N}$ and $K = N_t - 1$.

(d) Repeat the random calculation of $W(t)$ many times and compute the numerical spectrum (`fft`) of the numerical derivative $\xi(t) = D_t W(t)$ (finite differences) for each instantiation and estimate the autocorrelation function of the white noise.

◦ Solution

∗ Part (a) Haar and Schauder Functions

According to the claimed formula for the Haar functions,

$$h_k(t) = 2^{\frac{\lfloor \log_2(k) \rfloor}{2}} h_1(2^{\lfloor \log_2(k) \rfloor}(t+1) - k), \quad k \in \mathbb{N}$$

there holds for $k = 1$ and $n = \lfloor \log_2(k) \rfloor = 0$,

$$h_1(t) = 2^0 h_1(2^0(t+1) - 1) = h_1(t).$$

For $k > 1$, $n = \lfloor \log_2(k) \rfloor$ and

$$2^n \hat{t} \in (k - 2^n, k - 2^n + \tfrac{1}{2}) \quad \text{or} \quad \hat{\tau} = 2^n(\hat{t} + 1) - k \in (0, \tfrac{1}{2})$$

the formula leads to the evaluation

$$h_k(\hat{t}) = 2^{n/2} h_1(2^n(\hat{t} + 1) - k) = 2^{n/2} h_1(\hat{\tau}) = 2^{n/2} \cdot (+1)$$

and for

$$2^n \check{t} \in (k - 2^n + \tfrac{1}{2}, k - 2^n + 1) \quad \text{or} \quad \check{\tau} = 2^n(\check{t} + 1) - k \in (\tfrac{1}{2}, 1)$$

the formula leads to the evaluation

$$h_k(\check{t}) = 2^{n/2} h_1(2^n(\check{t} + 1) - k) = 2^{n/2} h_1(\check{\tau}) = 2^{n/2} \cdot (-1).$$

For $2^n t \notin (k - 2^n, k - 2^n + 1)$ ist $\tau = 2^n(t + 1) - k \notin (0, 1)$, und $h_k(t) = 2^{-n/2} h_1(\tau) = 0$. With this characterization, the Schauder functions are given with

$$\sigma = 2^{\lfloor \log_2(k) \rfloor}(\tau + 1) - k \quad \text{and} \quad 2^{\frac{\lfloor \log_2(k) \rfloor}{2}} d\tau = 2^{-\frac{\lfloor \log_2(k) \rfloor}{2}} d\sigma,$$

according to the calculation

$$\begin{aligned} s_k(t) = 2^{\frac{\lfloor \log_2(k) \rfloor}{2}} \int_0^t h_1(2^{\lfloor \log_2(k) \rfloor}(\tau + 1) - k) d\tau &= 2^{-\frac{\lfloor \log_2(k) \rfloor}{2}} \int_0^{2^{\lfloor \log_2(k) \rfloor}(t+1)-k} h_1(\sigma) d\sigma \\ &= 2^{-\frac{\lfloor \log_2(k) \rfloor}{2}} s_1(2^{\lfloor \log_2(k) \rfloor}(t + 1) - k). \end{aligned}$$

∗ Part (b) Nyquist Property for Basis Functions

Choose $N_t = 2^m$, $m \in \mathbb{N}$ and $K = N_t - 1$. Suppose $k \geq K + 1$ and set $n = \lfloor \log_2(k) \rfloor$ with $n \geq m$ since

$$2^m = N_t = K + 1 = 2^{\lfloor \log_2(K+1) \rfloor} \leq 2^n = 2^{\lfloor \log_2(k) \rfloor} \leq 2^{\log_2(k)} = k < 2^{\lfloor \log_2(k) \rfloor - 1} = 2^{n+1}.$$

Fix $t_i = i/N_t$ with $i \in \{0, \ldots, N_t\}$. For $s_k(t_i) \neq 0$ to hold, it must be that

$$\tau_i = 2^n(t_i + 1) - k \in (0, 1)$$

but this cannot be since τ_i is always an integer,

$$\tau_i = 2^n \left(\frac{i}{N_t} + 1 \right) - k = \frac{2^n}{N_t}(i + N_t) - k = 2^{n-m}(i + 2^m) - k \in \mathbb{Z}.$$

∗ Part (c) Simulation, Comparison to Theory

The following Matlab code is used to simulate Wiener processses with Schauder functions.

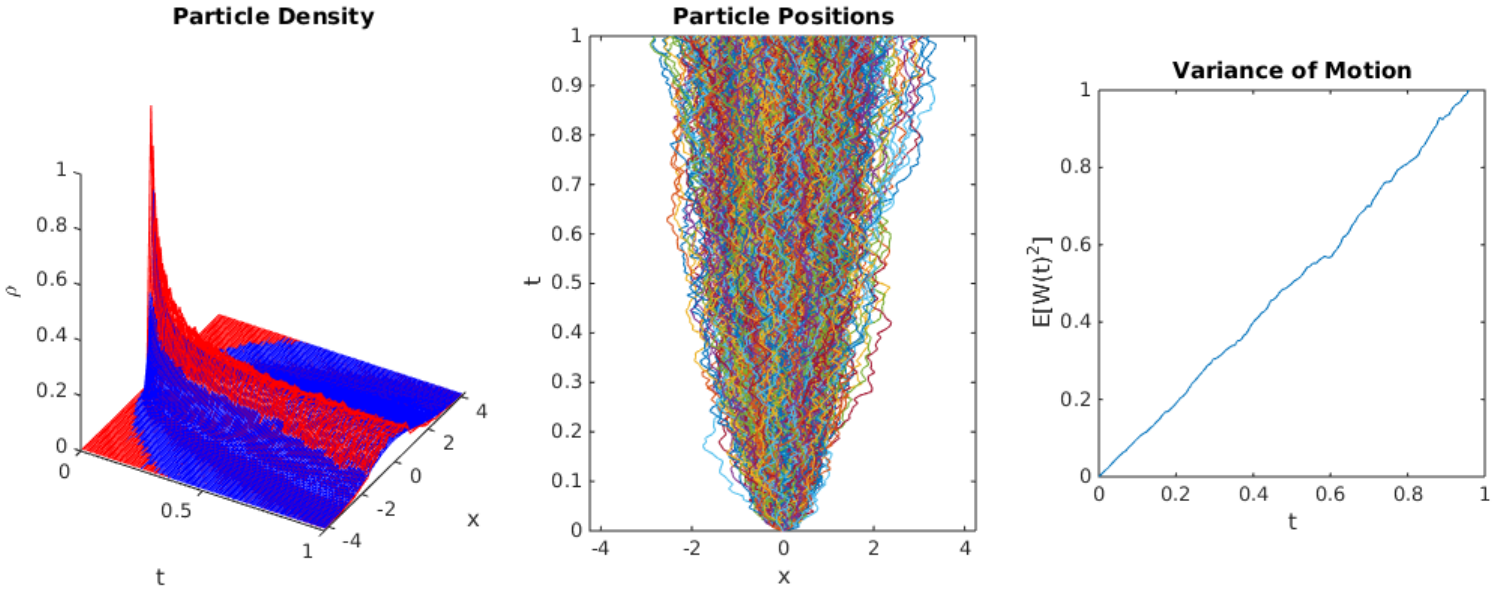
```
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[20 20 1200 400]);         % setup figure

% Haar functions
    h0 = @(t) ((0 <= t) & (t <= 1));
    h1 = @(t) ((0 <= t) & (t <= 1/2)) - ((1/2 < t) & (t < 1));
    h  = @(k,t) 2.^(floor(log2(k))/2).*h1(2.^(floor(log2(k))).*(t+1)-k);
% Schauder functions
```

```
s0 = @(t) t.*((0 <= t) & (t <= 1));
s1 = @(t) t.*((0 <= t) & (t <= 1/2)) + (1-t).*((1/2 < t) & (t < 1));
s  = @(k,t) 2.^(-floor(log2(k))/2).*s1(2.^(floor(log2(k))).*(t+1)-k);

Np = 1000;                                  % number of particles
Nt = 128;                                   % number of time steps
T  = 1;                                     % final time
ti = linspace(0,T,Nt+1)';                   % time cell interfaces
ht = ti(2)-ti(1);                           % time step
tc = ti(1:Nt) + ht/2;                       % time cell centers
oi = linspace(0,2*pi,Nt+1);                 % frequency cell interfaces
ho = oi(2)-oi(1);                           % frequency step
om = oi(1:Nt) + ho/2;                       % frequency cell centers

D  = 1;                                     % diffusivity

xmax = 3*sqrt(2*D*T);                       % spatial interval is
xmin = -xmax;                               %     [xmin,xmax]
Nx   = 2*round(round(sqrt(Np))/2)+1;        % number of spatial cells

xi = linspace(xmin,xmax,Nx+1);              % cell interfaces
hx = xi(2)-xi(1);
xc = xi(1:Nx)+hx/2;                         % cell centers

K = 2^(ceil(log2(Nt)))-1;                   % number of samples

Wv = [];                                    % save Wiener processes
for p=1:Np
    A = sqrt(D)*randn(1,K+1);               % normal random coefficients
    W = A(1)*s0(ti) + A(2)*s1(ti);
    for k=2:K                               % sum over Schauder basis
        W = W + A(k+1)*s(k,ti);
    end
    Wv = [Wv,W];
end

for i=1:Nt
    [Ni,yi] = histcounts(Wv(i,:),xi);       % numbers of particles in cells
    R = Ni/Np;                              % relative number
    r = (erf(xi(2:Nx+1)/sqrt(4*D*ti(i))) ... % theoretical prediction of R
       - erf(xi(1:Nx  )/sqrt(4*D*ti(i))))/2;

    subplot(1,3,1)                          % plot distributions
    hold on
    plot3(ti(i)*ones(size(xc)),xc,r,'b',ti(i)*ones(size(xc)),xc,R,'r')
    axis([0 T xmin xmax 0 1])
    view([30,30])
    title('Particle Density')
```

```
    xlabel('t')
    ylabel('x')
    zlabel('\rho')
    hold off;

    subplot(1,3,2)                          % plot positions
    hold on
    plot(Wv(i,:),ti(i)*ones(1,Np),'.')
    axis([xmin xmax 0 1])
    title('Particle Positions')
    xlabel('x')
    ylabel('t')
    hold off

    subplot(1,3,3)                          % plot variance
    plot(ti,std(Wv').^2)
    axis([0 T 0 T])
    title('Variance of Motion')
    xlabel('t')
    ylabel('E[W(t)^2]')

    drawnow;
end

h2 = figure(2); close(h2); h2 = figure(2);
set(h2,'Position',[20 20 1200 400]);        % setup figure

Wi = (Wv(2:Nt+1,:)-Wv(1:Nt,:))/ht;          % white noise
subplot(1,3,1)
plot(tc,Wi)
axis([0 T min(Wi(:)) max(Wi(:))])
xlabel('t')
ylabel('\xi(t)')
Wmu = mean(mean(Wi));                       % mean of time courses
Wsg = mean(std(Wi));                        % std of time courses
hold on;
plot(tc,Wmu*ones(1,Nt),'r', ...
     tc,(Wmu+Wsg)*ones(1,Nt),'g', ...
     tc,(Wmu-Wsg)*ones(1,Nt),'g', ...
     'LineWidth',2);
title(sprintf('White Noise\nE+/-S=%0.1e+/-%0.1e',Wmu,Wsg))
hold off

Ze = fft(Wi);                               % fft of time courses
Ze = Ze.*conj(Ze)/Nt;                       % spectrum of time courses
subplot(1,3,2)
plot(om,Ze)
hold on
```

```
Zmu = mean(mean(Ze));                          % mean of spectra
Zsg = mean(std(Ze));                           % std of spectra
plot(om,Zmu*ones(1,Nt),'r', ...
     om,(Zmu+Zsg)*ones(1,Nt),'g', ...
     om,(Zmu-Zsg)*ones(1,Nt),'g', ...
     'LineWidth',2);
axis([0 2*pi min(Ze(:)) max(Ze(:))])
xlabel('\omega')
ylabel('|F[\xi](\omega)|^2')
title(sprintf('Spectrum of White Noise\n E+/-S=%0.1e+/-%0.1e', ...
    Zmu,Zsg))
hold off

C = Wi*Wi'/Np;                                 % autocorrelation matrix
subplot(1,3,3)
surf(tc,tc,C)
axis([0 T 0 T min(C(:)) max(C(:))])
xlabel('t')
ylabel('t')
zlabel('c(t_i-t_j)=E[\xi(t_i)\xi(t_j)]')
title('Autocorrelation Function')
```

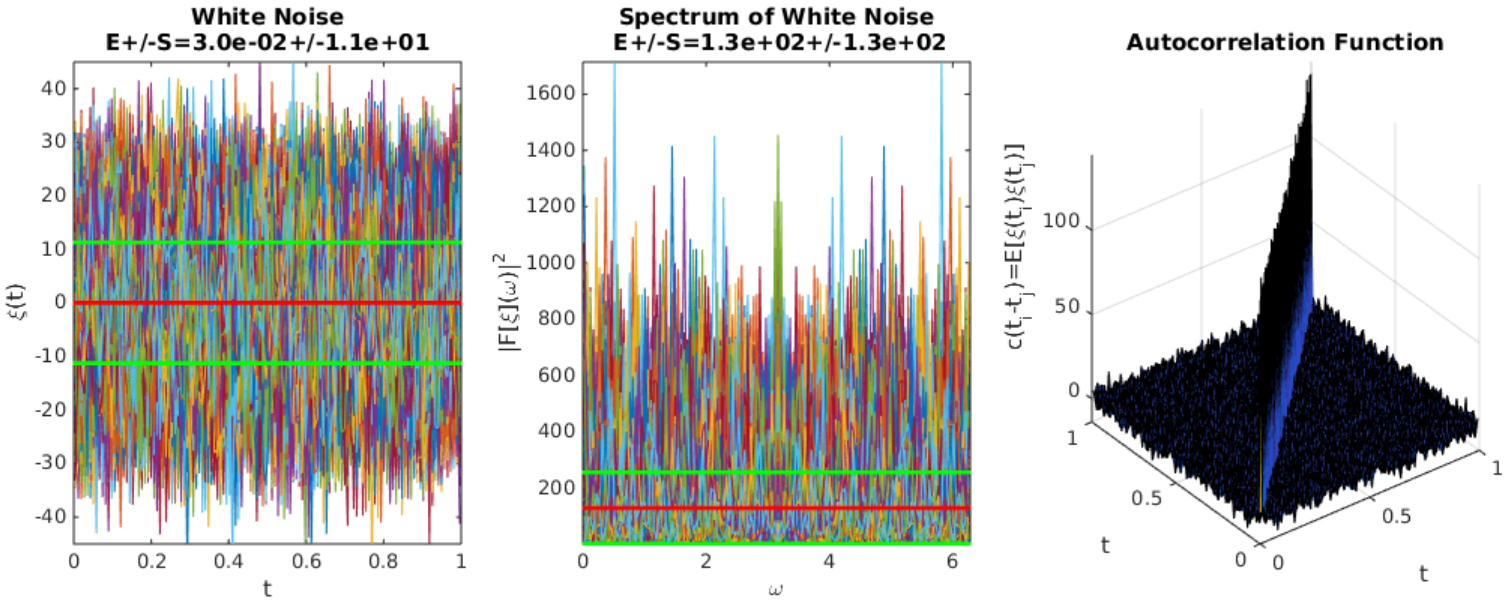
Shown on the left are particle densities, where the theoretical prediction is shown in blue and the result of the Monte-Carlo simulation is shown in red. The simulated trajectories of Brownian particles are shown in the middle. Shown on the right is the expected variance $\mathbb{E}[X(t)^2] = Dt$ with $D = 1$.



∗ Part (d) White Noise and Autocorrelation

In the next graphic the white noise $\xi(t) = D_t X(t; \Delta t)$ for each particle is shown on the left with a different color for each particle. The spectrum of each white noise time course is shown in the middle with a different color for each particle. Note that the spectrum is roughly flat, i.e., each frequency

contributes more or less equally, where the mean energy is shown in red and the standard deviation in green. The autocorrleation function is shown on the right, where the peaks along the diagonal indicate the independence of one time course from another, as predicted theoretically.



• Exercise 4: Stochastic Differential Equation, Explicit Solution

◦ Task

Let $S(t)$ be the random price of a stock at time $t \geq 0$. Let the evolution of the price be modelled by the SDE,

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), \quad S(0) = s_0$$

where $\mu > 0$ is the drift, $\sigma > 0$ is the volatility and $W(t)$ is a Wiener process with $\mathbb{E}[W(t)] = 0$ and $\mathbb{E}[W^2(t)] = t$, $t \geq 0$.

(a) Show that the solution is given by

$$S(t) = s_0 \exp[\sigma W(t) + (\mu - \sigma^2/2)t]$$

(b) Write a Matlab code to solve the SDE by regarding it as an ODE with values of the Wiener process $W(t)$ being generated as in the previous exercises. Compare the result with the known solution.

(c) Repeat the random calculation of $S(t)$ many times, compute the expected values $\mathbb{E}[S(t)]$ and compare these with the expected values of the known solution.

◦ Solution

∗ Part (a) Stock Formula

Recall Itô's Formula for $S(t) = u(X(t), t)$ given that $dX(t) = b(X(t))dt + B(X(t))dW(t)$,

$$\begin{cases} dX(t) &= bdt + BdW, & X(0) = x_0 \\ dS(t) &= [u_t + u_x b + \frac{1}{2}u_{xx}B^2]dt + u_x B dW, & S(0) = s_0 \end{cases}$$

where with the short form is meant,

$$b = b(X(t)), \quad B = B(X(t)), \quad u = u(X(t), t), \quad dW = dW(t).$$

To satisfy the SDE,

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), \quad S(0) = s_0$$

the following must hold

$$\left\{ \begin{array}{rcccc} u_t + u_x b + \frac{1}{2} u_{xx} B^2 & = & \mu S & = & \mu u \\ u_x B & = & \sigma S & = & \sigma u \end{array} \right.$$

Let $B = \sigma$, $b = 0$ and $x_0 = 0$. Then according to the second equation in the system, u must satisfy

$$u_x = u, \quad s_0 = u(x_0, 0) = u(0, 0).$$

With $u = u_x$ it follows that $u = u_x = u_{xx}$ holds, and according to the first equation in the system, u must satisfy

$$u_t = (\mu - \sigma^2/2)u \quad \text{or} \quad u(X, t) = f(X)\exp((\mu - \sigma^2/2)t).$$

Inserting this form into the previous equation gives

$$0 = u_x - u = [f'(X) - f(X)]\exp((\mu - \sigma^2/2)t), \quad s_0 = u(0, 0) = f(0)$$

or $f(X) = s_0 e^X$ and

$$u(X, t) = s_0 \exp(X + (\mu - \sigma^2/2)t).$$

The solution to the SDE

$$dX = \sigma dW, \quad X(0) = 0$$

is $X(t) = \sigma W(t)$, and hence the solution to the SDE for the stock price is

$$S(t) = s_0 \exp[\sigma W(t) + (\mu - \sigma^2/2)t].$$

∗ Part (b) Simulation, Comparison to Stock Formula

The evolution of the random stock price is calculated with the following Matlab code.

```
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[20 20 1600 800]);          % setup figure

% Haar functions
    h0 = @(t) ((0 <= t) & (t <= 1));
    h1 = @(t) ((0 <= t) & (t <= 1/2)) - ((1/2 < t) & (t < 1));
    h  = @(k,t) 2.^(floor(log2(k))/2).*h1(2.^(floor(log2(k))).*(t+1)-k);
% Schauder functions
    s0 = @(t) t.*((0 <= t) & (t <= 1));
    s1 = @(t) t.*((0 <= t) & (t <= 1/2)) + (1-t).*((1/2 < t) & (t < 1));
    s  = @(k,t) 2.^(-floor(log2(k))/2).*s1(2.^(floor(log2(k))).*(t+1)-k);

    Np = 128;                                     % number of particles
    Nt = 128;                                     % number of time steps
    T  = 1;                                       % final time
    ti = linspace(0,T,Nt+1)';                     % time cell interfaces
```

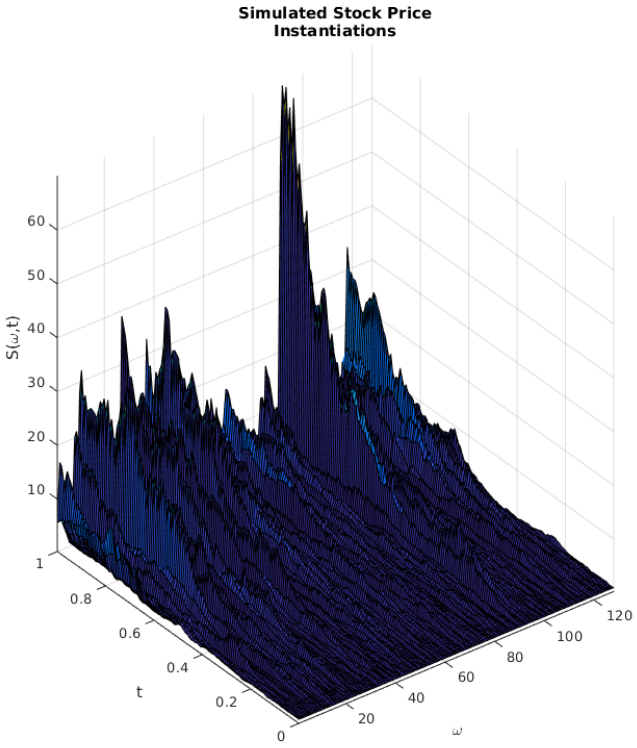
```
ht = ti(2)-ti(1);                               % time step
tc = ti(1:Nt) + ht/2;                           % time cell centers
oi = linspace(0,2*pi,Nt+1);                     % frequency cell interfaces
ho = oi(2)-oi(1);                               % frequency step
om = oi(1:Nt) + ho/2;                           % frequency cell centers

D  = 1;                                         % diffusivity

xmax = 3*sqrt(2*D*T);                           % spatial interval is
xmin = -xmax;                                   %     [xmin,xmax]
Nx   = 2*round(round(sqrt(Np))/2)+1;            % number of spatial cells

xi = linspace(xmin,xmax,Nx+1);                  % cell interfaces
hx = xi(2)-xi(1);
xc = xi(1:Nx)+hx/2;                             % cell centers

K = 2^(ceil(log2(Nt)))-1;                       % number of samples

Wv = [];                                        % save Wiener processes
for p=1:Np
    A = sqrt(D)*randn(1,K+1);                   % normal random coefficients
    W = A(1)*s0(ti) + A(2)*s1(ti);
    for k=2:K                                   % sum over Schauder basis
        W = W + A(k+1)*s(k,ti);
    end
    Wv = [Wv,W];
end

mu = 2;                                         % drift
sg = 1;                                         % volatility
s0 = 1;                                         % initial value
Wt = (Wv(2:Nt+1,:)-Wv(1:Nt,:))/ht;              % white noise
S  = zeros(Nt+1,Np);
S(1,:) = s0;
for i=1:Nt
    S(i+1,:) = S(i,:) + ht*S(i,:).*(mu + sg*Wt(i,:));
end
P = s0*exp(sg*Wv + (mu-sg^2/2)*kron(ti,ones(1,Np)));
subplot(1,2,1)
surf(1:Np,ti,S)
axis tight
xlabel('\omega')
ylabel('t')
zlabel('S(\omega,t)')
title(sprintf('Simulated Stock Price\nInstantiations'))
subplot(1,2,2)
surf(1:Np,ti,P)
axis tight
```

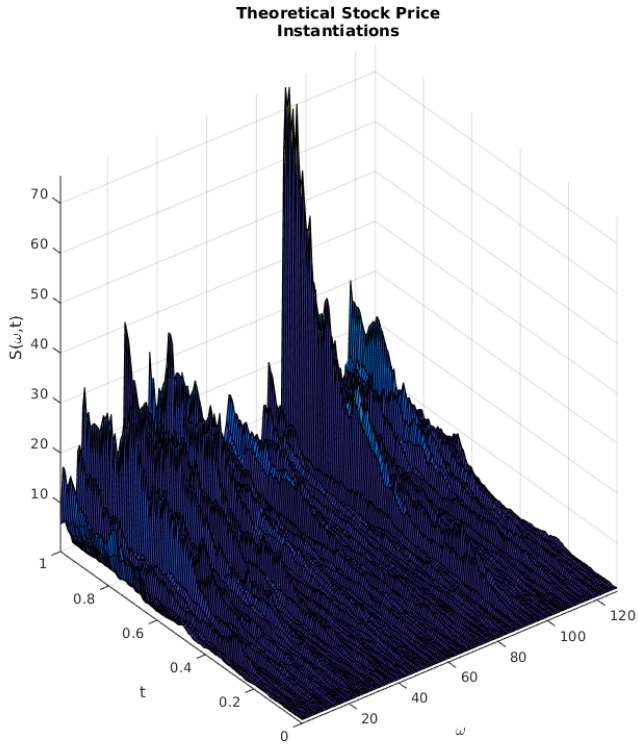
```
xlabel('\omega')
ylabel('t')
zlabel('S(\omega,t)')
title(sprintf('Theoretical Stock Price\nInstantiations'))

h2 = figure(2); close(h2); h2 = figure(2);
set(h2,'Position',[20 20 800 400]);          % setup figure
subplot(1,2,1)
plot(ti,mean(S,2))
axis tight
xlabel('t')
ylabel('E[S](t)')
title(sprintf('Simulated Stock Price\nExpectation'))
subplot(1,2,2)
plot(ti,mean(P,2))
axis tight
xlabel('t')
ylabel('E[S](t)')
title(sprintf('Theoretical Stock Price\nExpectation'))
```

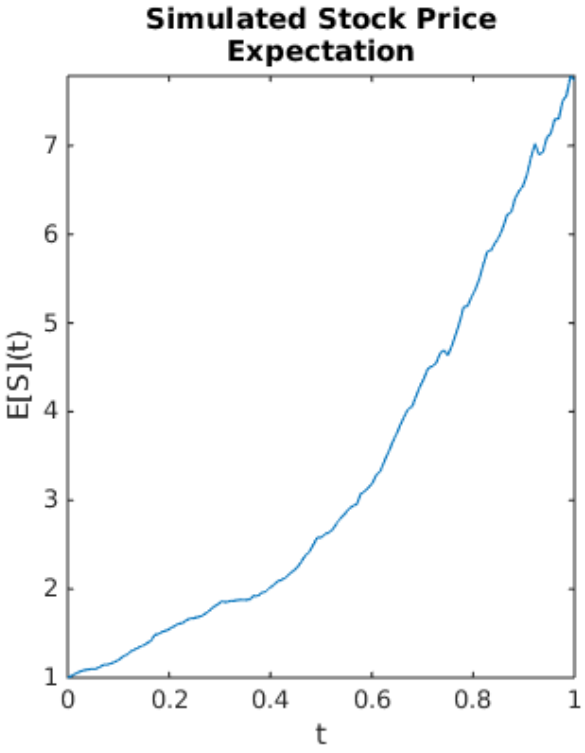
The relative norm of the difference between the simulated and the theoretical solutions for $\mu = 2$ and $\sigma = 1$ is roughly 5%, and for $\mu = 1$ and $\sigma = 2$ it is roughly 18%.

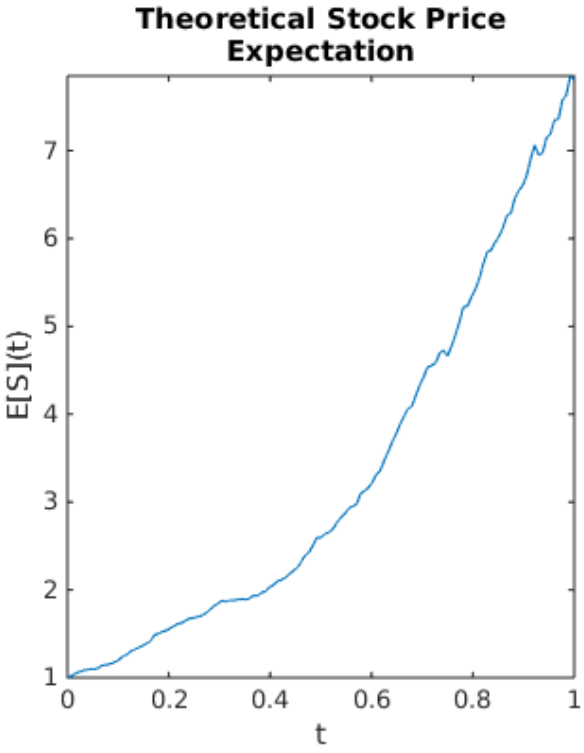
$*$ Part (c) Simulation, Comparison of Expected Values

The results of the calculations are demonstrated grafically as follows. First with with $\mu = 2$ and $\sigma = 1$ the drift slightly dominates the volatility, and an exponential growth in the stock price is evident.

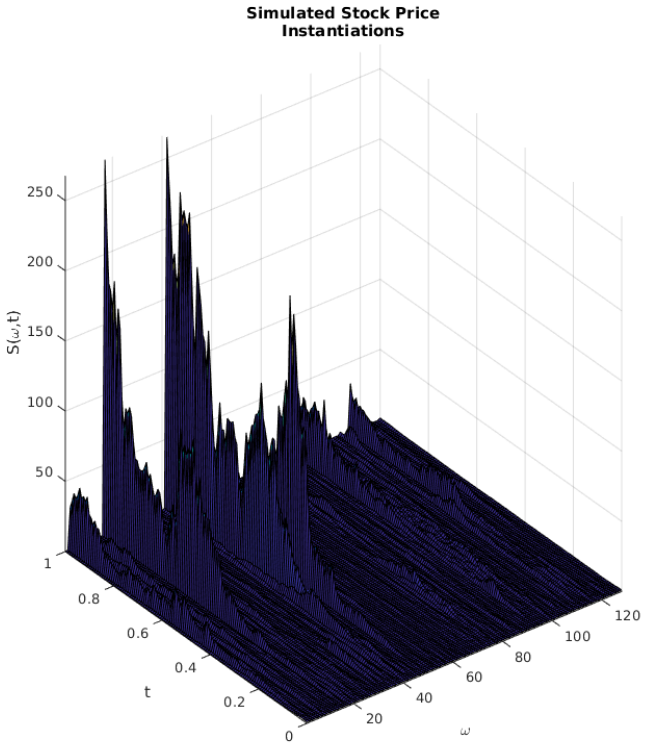


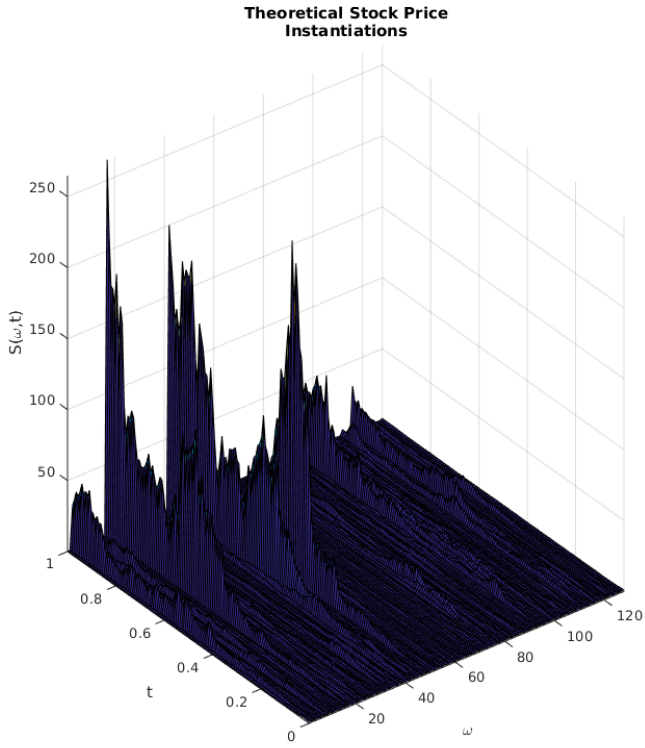


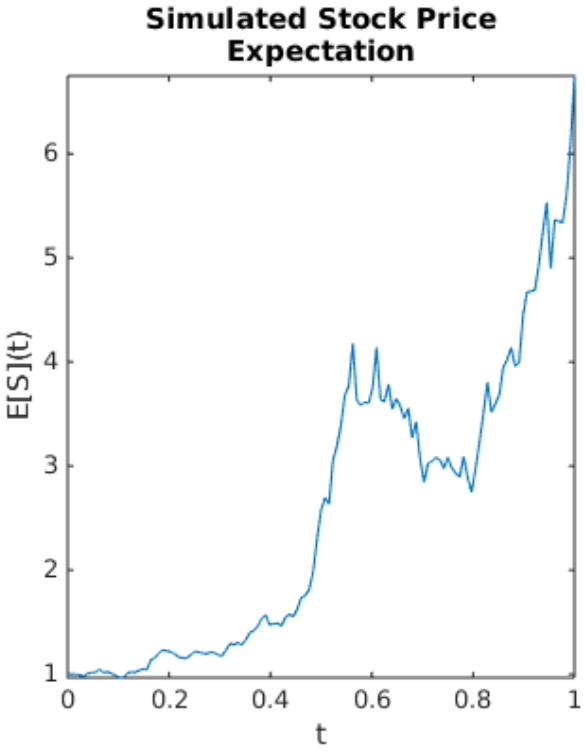


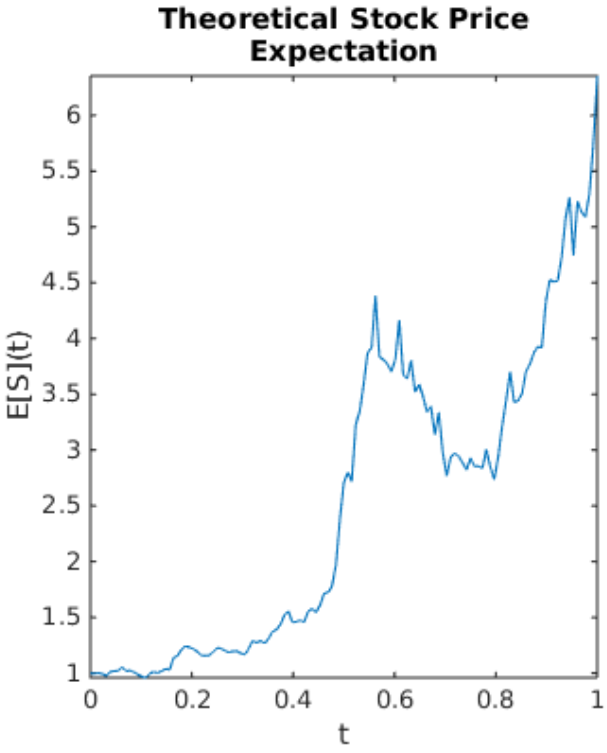


For the following case that $\mu = 1$ and $\sigma = 2$ hold, the volatility slightly dominates the drift, and there are more oscillations in the growth pattern of the stock price.


Simulated Stock Price
Instantiations
S(ω,t)
t
ω


Theoretical Stock Price
Instantiations
S(ω,t)
t
ω


Simulated Stock Price
Expectation
E[S](t)
t


Theoretical Stock Price
Expectation
E[S](t)
t

• Exercise 5: Riemann Sums and Itô Integral

◦ Task

Let values of a Wiener process $W(t)$ be generated as in the previous exercises. Write a Matlab code to approximate and to compute the expected value of the integral for final times T ranging from 0 to some $T_{\max} > 0$

$$\int_0^T W(t)dW(t)$$

using Riemann sums

$$R(V_m, \lambda) = \sum_{k=0}^{m-1} W(\tau_k(\lambda))[W(t_{k+1}) - W(t_k)]$$

where $0 = t_0 < t_1 < \cdots < t_m = T$ and $\tau_k(\lambda) = (1-\lambda)t_k + \lambda t_{k+1}$ for $\lambda \in [0, 1]$.

(a) Show that the result depends upon λ.

(b) Show that for $\lambda = 0$ the result agrees with the formula of Itô as $m \to \infty$.

◦ Solution

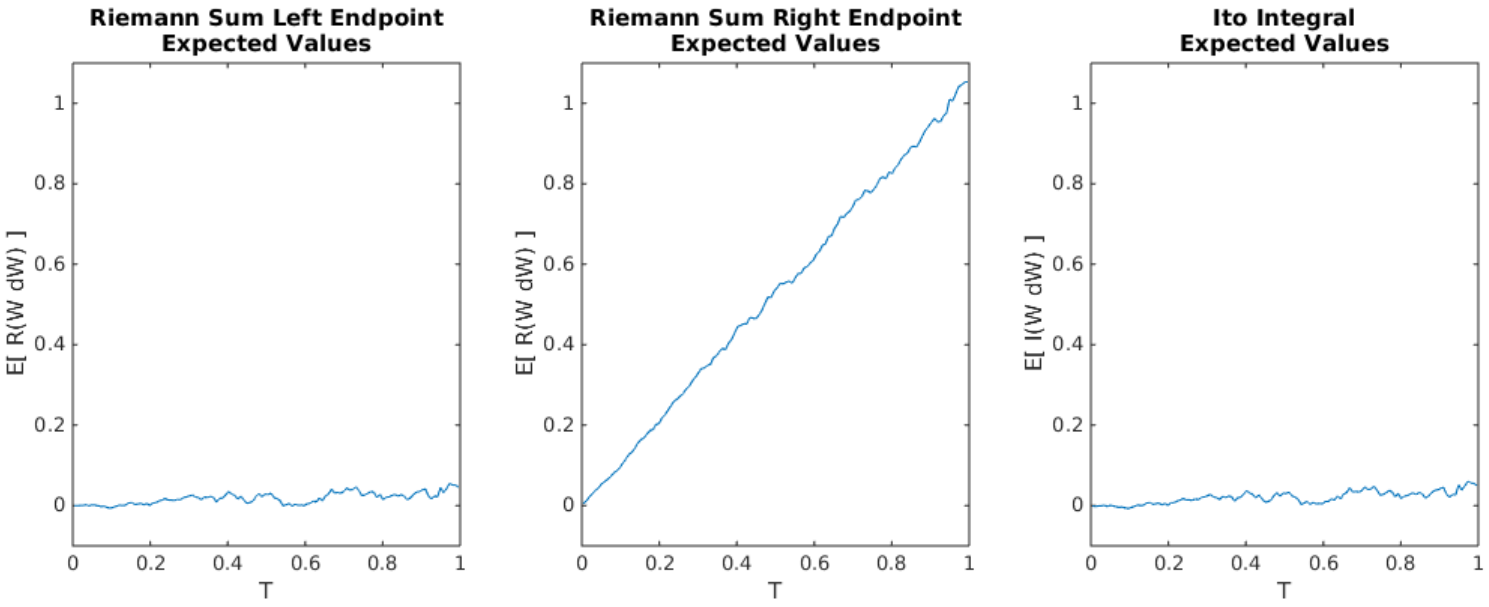
The following Matlab code is used to approximate the stochastic integral with Riemann sums.

```
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[20 20 1200 400]);          % setup figure

% Haar functions
    h0 = @(t) ((0 <= t) & (t <= 1));
    h1 = @(t) ((0 <= t) & (t <= 1/2)) - ((1/2 < t) & (t < 1));
    h  = @(k,t) 2.^(floor(log2(k))/2).*h1(2.^(floor(log2(k))).*(t+1)-k);
% Schauder functions
    s0 = @(t) t.*((0 <= t) & (t <= 1));
    s1 = @(t) t.*((0 <= t) & (t <= 1/2)) + (1-t).*((1/2 < t) & (t < 1));
    s  = @(k,t) 2.^(-floor(log2(k))/2).*s1(2.^(floor(log2(k))).*(t+1)-k);

    Np = 128;                                     % number of particles
    Nt = 128;                                     % number of time steps
    T  = 1;                                       % final time
    ti = linspace(0,T,Nt+1)';                     % time cell interfaces
    ht = ti(2)-ti(1);                             % time step
    tc = ti(1:Nt) + ht/2;                         % time cell centers
    oi = linspace(0,2*pi,Nt+1);                   % frequency cell interfaces
    ho = oi(2)-oi(1);                             % frequency step
    om = oi(1:Nt) + ho/2;                         % frequency cell centers

    D  = 1;                                       % diffusivity

    xmax = 3*sqrt(2*D*T);                         % spatial interval is
    xmin = -xmax;                                 %     [xmin,xmax]
    Nx   = 2*round(round(sqrt(Np))/2)+1;          % number of spatial cells
```

```
xi = linspace(xmin,xmax,Nx+1);              % cell interfaces
hx = xi(2)-xi(1);
xc = xi(1:Nx)+hx/2;                         % cell centers

K = 2^(ceil(log2(Nt)))-1;                   % number of samples

Wv = [];                                    % save Wiener processes
for p=1:Np
    A = sqrt(D)*randn(1,K+1);               % normal random coefficients
    W = A(1)*s0(ti) + A(2)*s1(ti);
    for k=2:K                               % sum over Schauder basis
        W = W + A(k+1)*s(k,ti);
    end
    Wv = [Wv,W];
end

dW = Wv(2:Nt+1,:)-Wv(1:Nt,:);               % Wiener increments

R1 = Wv(1,:).*dW(1,:);                      % Riemann sum left endpoint
for i=2:Nt                                  %     over all times
    R1 = [R1;sum(Wv(1:i,:).*dW(1:i,:),1)];
end
R1m = mean(R1,2);                           % expected values

subplot(1,3,1)                              % plot result
plot(tc,R1m)
axis tight
xlabel('t')
ylabel('E[ R(W dW) ]')
title(sprintf('Riemann Sum Left Endpoint\nExpected Values'))

R2 = Wv(2,:).*dW(1,:);                      % Riemann sum right endpoint
for i=2:Nt                                  %     over all times
    R2 = [R2;sum(Wv(2:(i+1),:).*dW(1:i,:),1)];
end
R2m = mean(R2,2);                           % expected values

subplot(1,3,2)                              % plot result
plot(tc,R2m)
axis tight
xlabel('t')
ylabel('E[ R(W dW) ]')
title(sprintf('Riemann Sum Right Endpoint\nExpected Values'))

                                            % Ito integral
R3 = 0.5*Wv(2:Nt+1,:).^2 - 0.5*kron(ti(2:Nt+1),ones(1,Np));
R3m = mean(R3,2);                           % expected values
```

```
subplot(1,3,3)                                  % plot result
plot(1:Np,R3m)
axis tight
xlabel('t')
ylabel('E[ I(W dW) ]')
title(sprintf('Ito Integral\nExpected Values'))
```

The results of the calculations are demonstrated graphically as follows.



∗ Part (a) Wiener Independence Property

(a) The left and middle plots above show respectively the results of using evaluations at the left and right endpoint of subintervals in the Riemann sum, and the results are conspicuously different. That the middle plot be linear agrees with the property of a Wiener process that in $W(t_{i+1})[W(t_{i+1}) - W(t_i)] = [W(t_{i+1}) - W(t_i)]^2 + W(t_i)[W(t_{i+1}) - W(t_i)]$ the random variables $W(t_i)$ and $W(t_{i+1}) - W(t_i)$ are independent while $[W(t_{i+1}) - W(t_i)]^2$ is distributed as $N(0, dt)$.

∗ Part (b) Itô Integral from Riemann Sum

The right plot shows the calculation of the Itô integral, which clearly agrees with the result on the left as predicted theoretically. That these integrals be zeros agrees with the property of a Wiener process that in $W(t_i)[W(t_{i+1}) - W(t_i)]$ the random variables $W(t_i)$ and $W(t_{i+1}) - W(t_i)$ are independent.

• Exercise 6: Stopping Times

◦ Task

For $\Omega = (-1, +1)$ and $Lu = \frac{1}{2}u_{xx}$ let u be the solution to

$$-Lu = 1 \text{ in } \Omega, \quad u = 0 \text{ on } \partial\Omega.$$

For $x \in \Omega$ let $X(t) = W(t) + x$ be a Brownian motion through Ω starting at x and define the stopping time $\tau_x = \inf\{t \geq 0 : X(t) \in \partial\Omega\}$.

(a) Write a Matlab code to compute a trajectory $X(t)$ by generating values of the Wiener process $W(t)$ as in the previous exercises.

(b) Verify the claim that $u(x)$ agrees with $\mathbb{E}[\tau_x]$, $\forall x \in \Omega$.

(c) With $g(x) = 2 - 2x^2$ and $f(x) = 1$ let

$$J_x(\theta) = \mathbb{E}\left[g(X(\min\{\tau_x, \theta\}) + \int_0^{\min\{\tau_x,\theta\}} f(X(s))ds\right]$$

represent the expected cost of halting the Brownian motion at time $\min\{\tau_x, \theta\}$. Show that u above is also the value function $u(x) = \inf_\theta J_x(\theta)$ and that the optimal stopping time is $\theta_x^\star = \tau_x$. On the other hand, given $f(x) = 1$ and $g_\epsilon(x) = \max\{0, 2(1-\epsilon)^2 - 2x^2\}$, show that $\theta_x^\star = \inf\{t \geq 0 : X(t) \in \partial\Omega_\epsilon\}$ where $\Omega_\epsilon = (-1+\epsilon, +1-\epsilon)$.

◦ Solution

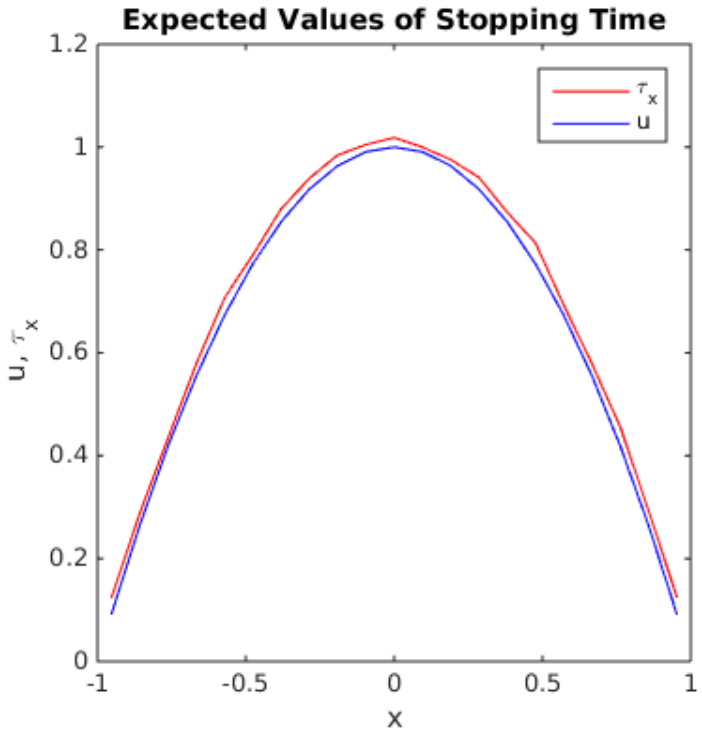
∗ Part (a) Simulation from Wiener Process

The following Matlab code is used to calculate the Wiener processes and the associated stopping times, and the expected values of the stopping times are compared to the solution of the boundary value problem.

```
h1 = figure(1); close(h1); h1 = figure(1);
set(h1,'Position',[20 20 400 400]);          % setup figure

Np = 10000;                                  % number of particles
Nt = 10000;                                  % number of time steps
T  = 10;                                     % final time
ti = linspace(0,T,Nt+1)';                    % time cell interfaces
ht = ti(2)-ti(1);                            % time step
tc = ti(1:Nt) + ht/2;                        % time cell centers

D  = 1;                                      % diffusivity
al = 1/2;                                    % P(delta = 0) = 1-2*al

xmax = +1;                                   % spatial interval is
xmin = -1;                                   %     [xmin,xmax]
Nx   = 21;                                  % number of spatial cells

xi = linspace(xmin,xmax,Nx+1);               % cell interfaces
hx = xi(2)-xi(1);
xc = xi(1:Nx)+hx/2;                          % cell centers
u  = 1-xc.^2;                                % value function

X  = kron(xc',ones(1,Np));                   % particle positions
tx = zeros(Nx,Np) + 4;                       % stopping times
```

```
for i=1:Nt
    z  = rand(Nx,Np);                          % random, uniform [0,1]
    de = -(z < al) + (z > 1-al);               % step direction
    X  = X + sqrt(D*ht/(2*al))*de;             % new positions
    tx = min(tx,ti(i+1)).*((X <= xmin) | (X >= xmax)) ...
       + tx.*((xmin < X) & (X < xmax));
    Etx = mean(tx,2);
    plot(xc,Etx,'r',xc,u,'b')
    drawnow;
end
Etx = mean(tx,2);
plot(xc,Etx,'r',xc,u,'b')
legend('\tau_x','u')
xlabel('x')
ylabel('u, \tau_x')
title('Expected Values of Stopping Time')
```

The results of the calculation are demonstrated graphically as follows.



∗ Part (b) Comparison to PDE Solution

As seen in the graphic, the solution $u(x) = 1 - x^2$ to the boundary value problem evidently agrees with $\mathbb{E}[\tau_x]$.

∗ **Part (c) Optimal Stopping Time**

The optimality condition for the minimization of $J_x(\theta)$ is

$$\min\{f + Lu, g - u\} = 0 \text{ in } \Omega, \quad u = g \text{ on } \partial\Omega.$$

Since

$$g(x) - u(x) = 2(1 - x^2) - (1 - x^2) = (1 - x^2) = u(x) \geq 0 \text{ in } \Omega$$

and

$$f + Lu = 1 + \tfrac{1}{2}u_{xx} = 0 \text{ in } \Omega, \quad u(x) - g(x) = 0 \text{ on } \partial\Omega$$

the function u satisfies the optimality conditions. Thus, the closed set

$$S = \{x \in \bar{\Omega} : u(x) = g(x)\} = \partial\Omega$$

is the stopping set. For $x \in \bar{\Omega}$ the optimal stopping time is

$$\theta_x^* = \inf\{t \geq 0 : X(t) \in S\} = \inf\{t \geq 0 : X(t) \in \partial\Omega\} = \tau_x.$$

On the other hand, given $f(x) = 1$, $g_\epsilon(x) = \max\{0, 2(1-\epsilon)^2 - 2x^2\}$ and $\Omega_\epsilon = (-1 + \epsilon, +1 - \epsilon) \subset \Omega$, the function

$$u_\epsilon(x) = \begin{cases} (1-\epsilon)^2 - x^2, & x \in \Omega_\epsilon \\ 0, & x \in \Omega \backslash \Omega_\epsilon \end{cases}$$

satisfies the optimality conditions

$$g_\epsilon(x) - u_\epsilon(x) = \begin{cases} (1-\epsilon)^2 - x^2, & x \in \Omega_\epsilon \\ 0, & x \in \Omega \backslash \Omega_\epsilon \end{cases} = u_\epsilon(x) \geq 0 \text{ in } \Omega$$

and

$$f + Lu_\epsilon = 1 + \tfrac{1}{2}(u_\epsilon)_{xx} = 0 \text{ in } \Omega, \quad u_\epsilon(x) - g_\epsilon(x) = 0 \text{ on } \partial\Omega \ (\subset \bar{\Omega}\backslash\Omega_\epsilon).$$

Thus, the closed set

$$S_\epsilon = \{x \in \bar{\Omega} : u_\epsilon(x) = g_\epsilon(x)\} = \bar{\Omega}\backslash\Omega_\epsilon$$

is the stopping set. For $x \in \bar{\Omega}$ the optimal stopping time is

$$\theta_x^* = \inf\{t \geq 0 : X(t) \in S_\epsilon\} = \inf\{t \geq 0 : X(t) \in \partial\Omega_\epsilon\}.$$

• Exercise 8: Call Option Pricing

◦ **Task**

Let $S(t)$ be the random price of a stock as indicated in exercise (3),

$$dS(t) = \mu S(t)dt + \sigma S(t)dW(t), \quad S(0) = s_0$$

where $\mu > 0$ is the drift, $\sigma > 0$ is the volatility and $W(t)$ is a Wiener process with $\mathbb{E}[W(t)] = 0$ and $\mathbb{E}[W^2(t)] = t$, $t \geq 0$. For a European call option let a strike price $p > 0$ and a strike time $T > 0$ be given. Let $r > 0$ be the interest rate corresponding to a risk-free bond $B(t) = e^{rt}$. Let the call option pricing or payoff function $u(s,t)$ at time t for a stock value s be given by the Black-Scholes-Merton final and boundary value problem,

$$\begin{cases} u_t + rsu_s + \frac{1}{2}\sigma^2 s^2 u_{ss} - ru = 0, & s > 0, \quad t \in (0,T) \\ u(s,T) = (s-p)^+, & s \geq 0, \quad t \in [0,T] \\ u(s,t) = 0, & s = 0, \quad t \in [0,T]. \end{cases}$$

(a) Show that the exact solution to the Black-Scholes-Merton problem is given by

$$\begin{aligned} u(s,t) &= \frac{s}{2}\left[1+\operatorname{erf}\left(\frac{\ln(s/p)+(r+\sigma^2/2)(T-t)}{\sqrt{2}\sigma\sqrt{T-t}}\right)\right] \\ &- \frac{p}{2}\left[1+\operatorname{erf}\left(\frac{\ln(s/p)+(r-\sigma^2/2)(T-t)}{\sqrt{2}\sigma\sqrt{T-t}}\right)\right]e^{-r(T-t)} \end{aligned}$$

(b) Choose numerical values for all named parameters and solve the Black-Scholes-Merton problem with a truncated boundary numerically,

$$\begin{cases} u_t + rsu_s + \frac{1}{2}\sigma^2 s^2 u_{ss} - ru = 0, & s\in(0,2p), & t\in(0,T) \\ u(s,T) = (s-p)^+, & s\in[0,2p], & t\in[0,T] \\ u(s,t) = 0, & s=0, & t\in[0,T]. \\ u_s(s,t) = 1, & s=2p, & t\in[0,T]. \end{cases}$$

where the Neumann boundary condition has been introduced at a right boundary $s_{\max} = 2p$ to avoid an otherwise infinite domain. Indicate the sought call option price $u(s_0, 0)$.

(c) Recall the solution $S(t) = s_0 \exp[\sigma W(t) + (\mu - \sigma^2/2)t]$ and compute the randomly evolving call option payoff according to $C(t) = u(S(t), t)$.

(d) For the portfolio $\Pi(t) = \phi(t)S(t) - \psi(t)B(t)$ compute the stock holdings owned and the bond holdings owed, respectively,

$$\phi(t) = u_s(S(t),t), \quad \psi(t) = e^{-rt}[u_s(S(t),t)S(t) - u(S(t),t)].$$

With these dynamic holdings, compute the randomly evolving portfolio value $\Pi(t)$ and compare it with the randomly evolving call option value $C(t)$. Verify that the portfolio is self-financing.

◦ Solution

∗ Part (a) Exact Solution to Black-Scholes-Merton

Define the transformed variables $x = \ln(s/p)$, $\tau = T - t$ and the function $v(x,\tau) = u(pe^x, T-\tau)$ which then satisfies the initial and boundary value problem

$$\begin{cases} v_\tau - \frac{1}{2}\sigma^2 v_{xx} + (\sigma^2/2 - r)v_x + rv = 0, & x\in\mathbb{R}, & \tau\in(0,T) \\ v(x,\tau) = p(e^x-1)^+, & x\in\mathbb{R}, & \tau = 0 \\ v(x,\tau) = 0, & x\to-\infty, & \tau\in[0,T] \end{cases}$$

Next with $\alpha = r/\sigma^2 - 1/2$ and $\beta = r/2 + \sigma^2/8 + r^2/(2\sigma^2)$ define the function $w(x,\tau) = e^{\alpha x + \beta\tau}v(x,\tau)$ which satisfies the initial value problem for the diffusion equation

$$w_\tau - \tfrac{1}{2}\sigma^2 w_{xx} = 0, \quad w(x,0) = pe^{\alpha x}(e^x-1)^+$$

solved by the convolution

$$w(x,\tau) = \frac{1}{\sqrt{2\pi\sigma^2\tau}}\int_{-\infty}^{+\infty}\exp\left(-\frac{(x-\xi)^2}{2\sigma^2\tau}\right)w(\xi,0)d\xi$$

This is the basis for the claimed error function formulation. These characterizations are verified by a direct calculation of u, v and w with the error function formula shown above for u.

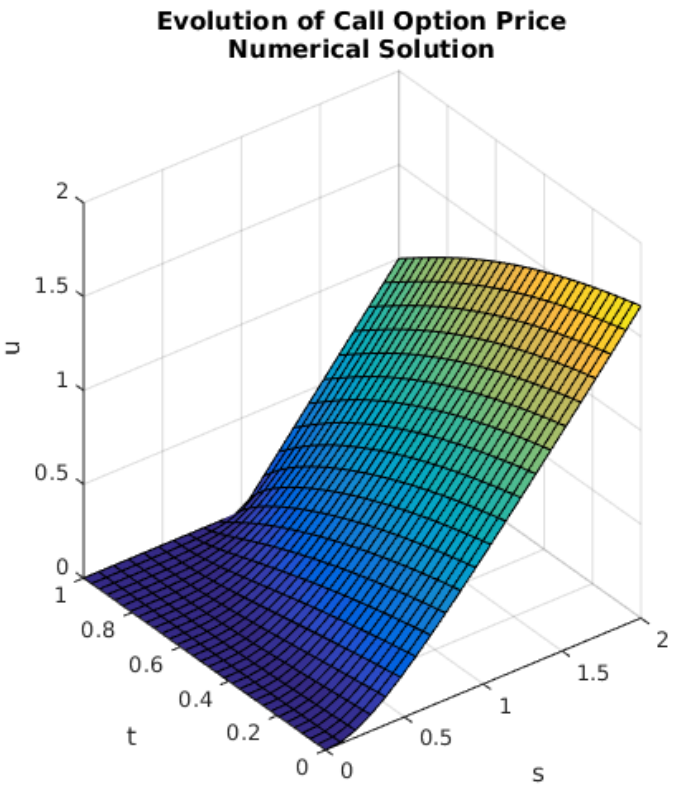
* **Part (b) Numerical Solution to Black-Scholes.Merton**

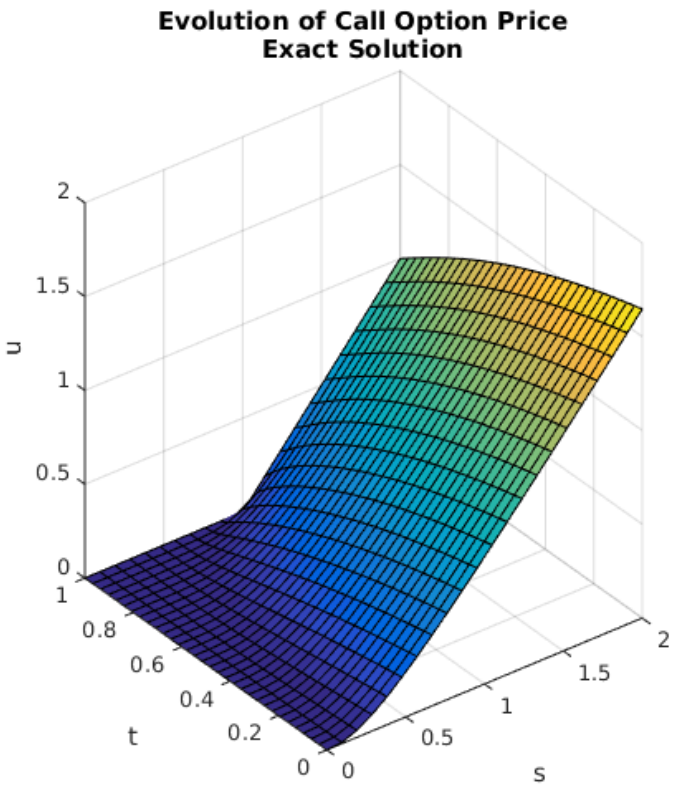
The final and boundary value problem on the truncated domain is solved with the following Matlab code.

```
h1 = figure(1); close(h1); h1 = figure(1);
set(h1,'Position',[100 10 1000 500]);        % setup figure

p  = 1;                                      % strike price
r  = 1;                                      % interest rate
sg = 1;                                      % volatility
mu = 1;                                      % drift

Ns   = 20;                                   % number of price cells
smin = 0;                                    % stock price in
smax = 2*p;                                  %     [smin,smax]
s    = linspace(smin,smax,Ns+1)';            % stock price interfaces
hs   = s(2)-s(1);                            % cell width

Nt   = 30;                                   % number of time points
T    = 1;                                    % strike time
t    = linspace(0,T,Nt+1);                   % temporal cell interfaces
ht   = t(2)-t(1);                            % time step

I  = speye(Ns+1);
ds = spdiags(ones(Ns+1,1),1,Ns,Ns+1) ...
   - spdiags(ones(Ns+1,1),0,Ns,Ns+1);        % d/ds
ds = ds/hs;
L  = ds'*ds;                                 % Lu ~= -u_ss
L(end,:) = L(end,:)/(smax^2*sg^2/2);
b  = zeros(Ns+1,1); b(1) = 1;
Bd = spdiags(b,0,Ns+1,Ns+1);
L  = (I-Bd)*L;                               % Dirichlet BCs left

K  = spdiags(ones(Ns+1,1),+1,Ns+1,Ns+1) ...  % convection,
   - spdiags(ones(Ns+1,1),-1,Ns+1,Ns+1);     %     central differencing
K(1,1) = -2; K(1,2) = +2;
K(Ns+1,Ns) = -2; K(Ns+1,Ns+1) = +2;
K  = K/(2*hs);

A  = (sg^2/2)*spdiags(s.^2,0,Ns+1,Ns+1)*L ...
   - r*spdiags(s,0,Ns+1,Ns+1)*K ...
   + r*I;                                    % evolution operator

b  = zeros(Ns+1,1); b(Ns+1) = 1;
Bn = spdiags(b,0,Ns+1,Ns+1);
un = (sg^2/2)*spdiags(s.^2,0,Ns+1,Ns+1)*(L*s);
un = Bn*un;                                  % artificial Neumann BC, right
```

```
u  = zeros(Ns+1,Nt+1);
u(:,Nt+1) = max(0,s-p);                        % final time condition

for k=Nt:-1:1                                  % backward time stepping
    u(:,k) = (I + ht*A) \ (u(:,k+1) + ht*un);
end

subplot(1,2,1)                                 % plot results
surf(s,t,u')
xlabel('s')
ylabel('t')
zlabel('u')
title(sprintf('Evolution of Call Option Price\nNumerical Solution'))

ue = zeros(Ns+1,Nt+1);                         % exact solution
ss = kron(s,ones(1,Nt+1));
t(Nt+1) = t(Nt+1) - eps;
tt = kron(ones(Ns+1,1),t);
dn = (sqrt(2)*sg*sqrt(T-tt));
q1 = (log(ss/p) + (r+sg^2/2)*(T-tt))./dn;
q2 = (log(ss/p) + (r-sg^2/2)*(T-tt))./dn;
ue = (ss/2).*(1 + erf(q1)) ...
   -  (p/2).*(1 + erf(q2)).*exp(-r*(T-tt));

subplot(1,2,2)                                 % plot results
surf(s,t,ue')
xlabel('s')
ylabel('t')
zlabel('u')
title(sprintf('Evolution of Call Option Price\nExact Solution'))

h2 = figure(2); close(h2); h2 = figure(2);
set(h2,'Position',[100 10 400 400]);        % setup figure
plot(s,ue(:,1))
xlabel('s')
ylabel('u(s,0)')
title('Call Option Price')
```

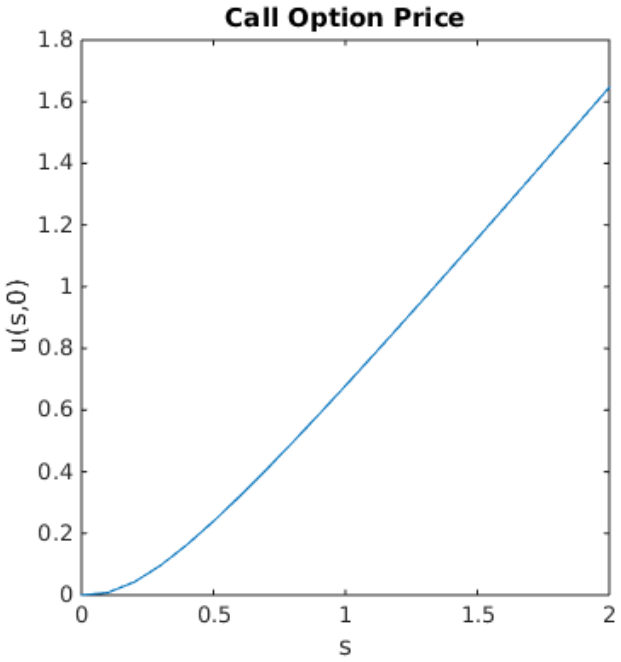
The results are demonstrated graphically as follows.





With the parameters $\mu = 1$, $\sigma = 1$, $r = 1$, $T = 1$ and $p = 1$, the call option price $u(s_0, 0)$ is given for

values of $s_0 \in [0, 2]$ by the foremost curve in the surface above, also seen in the following graphic.



∗ **Part (c) Simulation of Stock Value**

The following Matlab code is used to simulate the course of a stock, a bond, their holdings, a call option value and a matching portfolio value. The time courses of their expected values are shown graphically below.

```
    h1 = figure(1); close(h1); h1 = figure(1);
    set(h1,'Position',[20 20 1500 1000]);          % setup figure

% Haar functions
    h0 = @(t) ((0 <= t) & (t <= 1));
    h1 = @(t) ((0 <= t) & (t <= 1/2)) - ((1/2 < t) & (t < 1));
    h  = @(k,t) 2.^(floor(log2(k))/2).*h1(2.^(floor(log2(k))).*(t+1)-k);
% Schauder functions
    s0 = @(t) t.*((0 <= t) & (t <= 1));
    s1 = @(t) t.*((0 <= t) & (t <= 1/2)) + (1-t).*((1/2 < t) & (t < 1));
    s  = @(k,t) 2.^(-floor(log2(k))/2).*s1(2.^(floor(log2(k))).*(t+1)-k);

% stock parameters
    p  = 2;                                        % strike price
    T  = 1;                                        % strike time
    sg = 1;                                        % volatility
    r  = 1;                                        % interest rate
    mu = 1;                                        % drift
    S0 = 1;                                        % initial stock value
```

```
% call option price
    R  = @(s,t) sqrt(2)*sg*sqrt(T - t);
    Q  = @(s,t) (log(s/p) + (r + sg^2/2)*(T - t))/R(s,t);
    P  = @(s,t) (log(s/p) + (r - sg^2/2)*(T - t))/R(s,t);
    U  = @(s,t) (s/2)*(1 + erf(Q(s,t))) ...
              - (p/2)*(1 + erf(P(s,t)))*exp(-r*(T - t));
    Us = @(s,t) exp(-Q(s,t)^2)/(R(s,t)*sqrt(pi)) ...
              - exp(-P(s,t)^2)/(R(s,t)*sqrt(pi))*exp(-r*(T - t))*(p/s) ...
              + (1 + erf(Q(s,t)))/2;

    Np = 100;                                       % number of instances
    Nt = 128;                                       % number of time steps
    ti = linspace(0,T,Nt+1)';                       % time cell interfaces
    ti(Nt+1) = ti(Nt+1)-eps;
    ht = ti(2)-ti(1);                               % time step

    D  = 1;                                         % diffusivity
    K = 2^(ceil(log2(Nt)))-1;                       % number of samples
    Wv = [];                                        % save Wiener processes
    for l=1:Np
        A = sqrt(D)*randn(1,K+1);                   % normal random coefficients
        W = A(1)*s0(ti) + A(2)*s1(ti);
        for k=2:K                                   % sum over Schauder basis
            W = W + A(k+1)*s(k,ti);
        end
        Wv = [Wv,W];
    end
                                                    % random stock value
    S = S0*exp(sg*Wv + (mu-sg^2/2)*kron(ti,ones(1,Np)));
    B = kron(exp(r*ti),ones(1,Np));                 % bond value
    C = zeros(Nt+1,Np);                             % call option price
    for i=1:Nt+1
        for j=1:Np
            C(i,j) = U(S(i,j),ti(i));
        end
    end
    phi = zeros(Nt+1,Np);                           % stock holdings
    for i=1:Nt+1
        for j=1:Np
            phi(i,j) = Us(S(i,j),ti(i));
        end
    end
    psi = zeros(Nt+1,Np);                           % bond holdings
    for i=1:Nt+1
        for j=1:Np
            psi(i,j) = (phi(i,j)*S(i,j) - C(i,j))/B(i,j);
        end
    end
```

```
    Pi = phi.*S - psi.*B;                          % portfolio

    dPi = Pi(2:Nt+1,:) - Pi(1:Nt,:);
    dS  =  S(2:Nt+1,:) -  S(1:Nt,:);
    dB  =  S(2:Nt+1,:) -  S(1:Nt,:);               % self-financing residual
    SF  = dPi - phi(1:Nt,:).*dS + psi(1:Nt,:).*dB;

    Sm   = mean(S,2);                              % expected values vs time
    phim = mean(phi,2);
    psim = mean(psi,2);
    Bm   = mean(B,2);
    Cm   = mean(C,2);
    Pim  = mean(Pi,2);

    subplot(2,3,1)                                 % plot results
        plot(ti,Sm,'b',ti,p*ones(Nt+1,1),'r:')
        xlabel('t')
        ylabel('E[S]')
        title('Stock Value')
    subplot(2,3,2)
        plot(ti,Bm)
        xlabel('t')
        ylabel('E[B]')
        title('Bond Value')
    subplot(2,3,3)
        plot(ti,Cm)
        xlabel('t')
        ylabel('E[C]')
        title('Call Option Payoff')
    subplot(2,3,4)
        plot(ti,phim)
        xlabel('t')
        ylabel('E[\phi]')
        title('Stock Holdings')
    subplot(2,3,5)
        plot(ti,psim)
        xlabel('t')
        ylabel('E[\psi]')
        title('Bond Holdings')
    subplot(2,3,6)
        plot(ti,Pim)
        xlabel('t')
        ylabel('E[Pi]')
        title('Portfolio Value')

disp(sprintf('\n|E[C-Pi]| = %0.1e',norm(Cm(:)-Pim(:))));

disp(sprintf('\nInitial Call Option Price = %0.1e',Cm(1)))
```

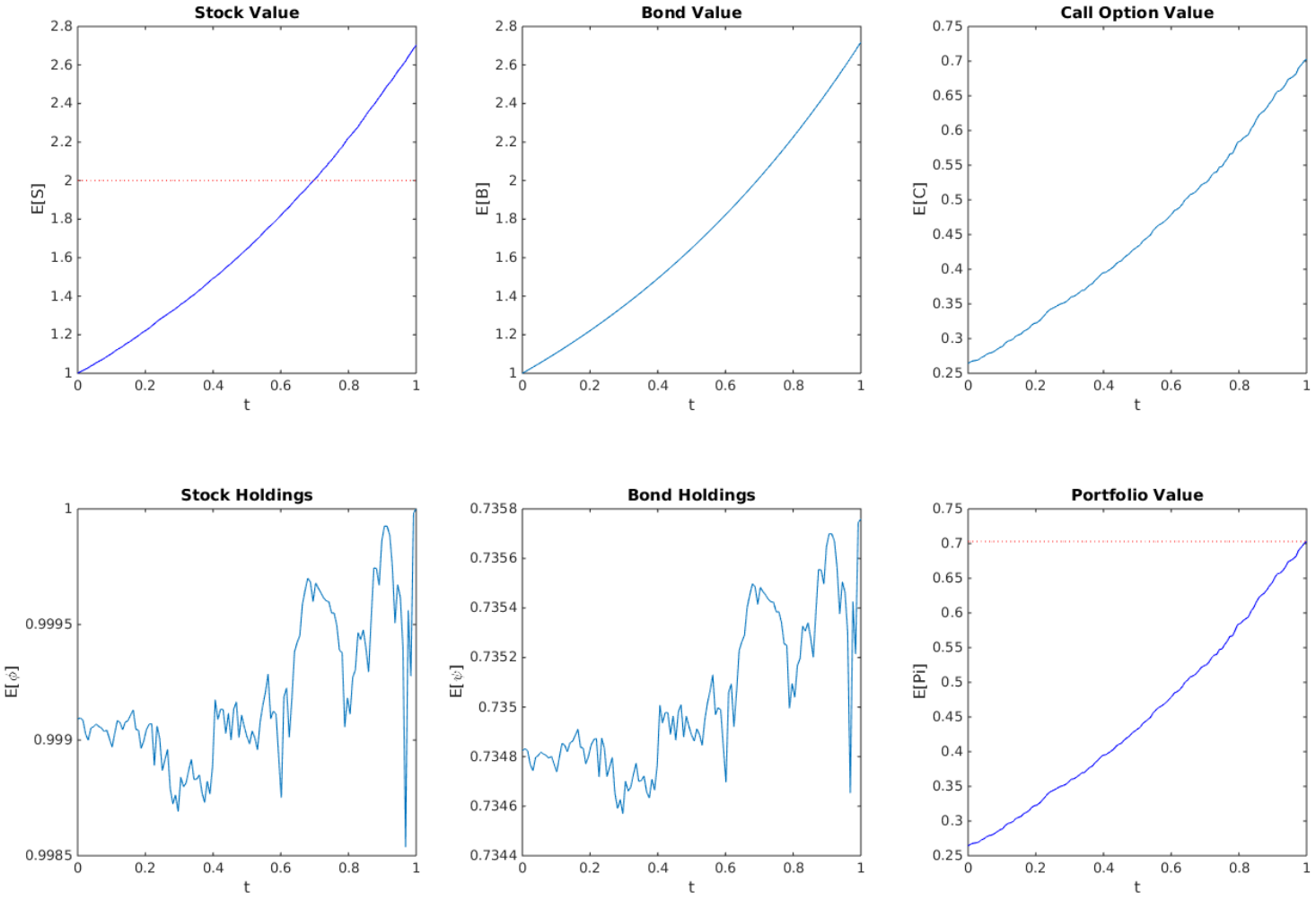
```
%
disp(sprintf('\nInitial Stock Assets = %0.1e',phim(1)*Sm(1)))
disp(sprintf('Final Stock Assets = %0.1e',phim(end)*Sm(end)))
%
disp(sprintf('\nInitial Bond Debts = %0.1e',psim(1)*Bm(1)))
disp(sprintf('Final Bond Debts = %0.1e',psim(end)*Bm(end)))
%
disp(sprintf('\nActual End-Payoff = %0.1e',max(0,Sm(end)-p)))
disp(sprintf('Stochastic End-Payoff = %0.1e',Cm(end)))

disp(sprintf('\nExpectation of Self-Financing Residual = %0.1e',mean(SF(:))))
```

∗ **Part (d) Comparison of Portfolio and Call Option Value**

For the following cases the parameters $\mu = 1$, $r = 1$, $T = 1$, $p = 1$ and $s_0 = 1$ are used.

In the first case, the volatility is set low at $\sigma = 0.1$ so that the anticipated agreement among quantities can be verified. The strike price is shown marked in red in the upper left graph together with the stock value. The difference between the stock value and the strike price at the end is $\mathbb{E}[S(T)] - p = 0.7$, which is the payoff of the call option. The initial price for the call option is $\mathbb{E}[C(0)] = 0.26 = \mathbb{E}[\Pi(0)]$, and this initial value grows with the portfolio to the asset of $\mathbb{E}[\Pi(T)] = 0.7$ at the end, which matches the actual payoff $\mathbb{E}[S(T)] - p$ as well as the stochastic call option value $\mathbb{E}[C(T)] = 0.7$ at the end. This break-even threshold, which the portfolio value must reach, is shown in red in the lower right graph.

The expected value of the self-financing residual is $\mathbb{E}[d\Pi - \phi dS + \psi dB] = 0.000087$.



In the next case, the volatility is set higher at $\sigma = 1$, and the anticipated agreement among quantities is not as evident as in the last example. The strike price is shown marked in red in the upper left graph together with the stock value. The difference between the stock value and the strike price at the end is $\mathbb{E}[S(T)] - p = 1.2$, which is the payoff of the call option. The initial price for the call option is $\mathbb{E}[C(0)] = 0.48 = \mathbb{E}[\Pi(0)]$, and this initial value grows with the portfolio to the asset of $\mathbb{E}[\Pi(T)] = 1.7$ at the end, which matches the stochastic call option value $\mathbb{E}[C(T)] = 1.7$ at the end, but it is signifantly larger than the the actual payoff $\mathbb{E}[S(T)] - p = 1.2$. This excess is seen in the lower right graph as the blue curve exceeds the red threshold. Therefore, the value of portfolio is more than adequate to cover the actual payoff. The expected value of the self-financing residual is

$\mathbb{E}[d\Pi - \phi dS + \psi dB] = 0.0042.$