ETH Zürich HS 2014
D-MATH
Prof. Dr. Johannes Muhle-Karbe

Coordinator
Ren Liu

Interest Rate Theory
Solutions Sheet 1

1. Note that

$$P(t,S)F(t;T,S) = \frac{P(t,T)-P(t,S)}{S-T},$$
$$F(T,S) = \frac{1}{S-T}\left(\frac{1}{P(T,S)}-1\right).$$

We build a self-financing trading strategy with time $t \leq T$ cost

$$\frac{P(t,T)-P(t,S)}{S-T},$$

yielding the value $F(T,S)$ at time S:

- At time $t \leq T$ buy one zero-coupon bond with maturity T at cost $P(t,T)$ and sell one zero-coupon bond with maturity S at cost $P(t,S)$.
  Total cost at time $t \leq T$: $P(t,T) - P(t,S)$.

- At time T receive 1 dollar for the maturing zero-coupon bond.
  Reinvest this 1 dollar to buy precisely $\frac{1}{P(T,S)}$ bonds with maturity S.

- At time S we receive $\frac{1}{P(T,S)} - 1$.

Scaling the strategy with $1/(S-T)$ yields the assertion.

2. For each $i = 1, \ldots, n$ we have

$$F(T_{i-1},T_i) = \frac{1}{\delta}\left(\frac{1}{P(T_{i-1},T_i)}-1\right),$$

Bitte wenden!

which yields the cash flow at time T_i for each caplet

$$\begin{aligned}\delta(F(T_{i-1},T_i)-\varkappa)^+ &= \delta\left(\frac{1}{\delta}\left(\frac{1}{P(T_{i-1},T_i)}-1\right)-\varkappa\right)^+\\ &= \left(\frac{1}{P(T_{i-1},T_i)}-(1+\delta\varkappa)\right)^+,\end{aligned}$$

and hence, the time T_{i-1} value of $\delta(F(T_{i-1},T_i)-\varkappa)^+$ *paid out at* T_i is given by

$$\begin{aligned}P(T_{i-1},T_i)\delta(F(T_{i-1},T_i)-\varkappa)^+ &= P(T_{i-1},T_i)\left(\frac{1}{P(T_{i-1},T_i)}-(1+\delta\varkappa)\right)^+\\ &= (1+\delta\varkappa)\left(\frac{1}{1+\delta\varkappa}-P(T_{i-1},T_i)\right)^+.\end{aligned}$$

This is $(1+\delta\varkappa)$ times the cash-flow at date T_{i-1} of a put option on a T_i-bond with strike price $\frac{1}{1+\delta\varkappa}$ and maturity T_{i-1}.

3. **a)** Applying the definition of the $AR(1)$ process recursively N times we get

$$\begin{aligned}X_t &= c+\varphi(c+\varphi\cdot X_{t-2}+\varepsilon_{t-1})+\varepsilon_t\\ &= \cdots\\ &= c\cdot\sum_{k=0}^{N-1}\varphi^k+\varphi^N X_{t-N}+\sum_{k=0}^{N-1}\varphi^k\varepsilon_{t-k}.\end{aligned}$$

Therefore, if X is stationary and has finite second moment, then

$$C(0):=Var[X_t]=\varphi^{2N}C(0)+\sum_{k=0}^{N-1}\varphi^k\cdot\sigma_\varepsilon^2 \tag{1}$$

which in turn implies that $\varphi^2<1$. On the other hand, if we assume that $|\varphi|<1$ holds. Then, by letting $N\to\infty$ we obtain

$$X_t=c\cdot\sum_{k=0}^{\infty}\varphi^k+\sum_{k=0}^{\infty}\varphi^k\varepsilon_{t-k}=\frac{c}{1-\varphi}+\sum_{k=0}^{\infty}\varphi^k\varepsilon_{t-k}. \tag{2}$$

Since the ε_t are assumed to be i.i.d, the stationarity follows.

b) The assertions are immediate consequences of (2) and (1).

c) Using Itô's formula the SDE can be solved explicitly

$$Y_t=y\cdot\exp(\mu t+\sigma W_t-\sigma^2t/2).$$

Siehe nächstes Blatt!

Therefore, we have

$$\begin{aligned}\mathbb{E}[Y_t] &= y \cdot \exp(\mu t),\\ Var[Y_t] &= y^2 \exp(2\mu t)(\exp(\sigma^2 t) - 1).\end{aligned}$$

Since the right hand side diverges for $t \to \infty$, the process can not be stationary.

4. (i) see *driftedBM.m*, *AR1.m*, *poissonprocess.m* and *compoundpoissonprocess.m*

(ii) (a) We first show that $\mathbb{E}[e^{\sigma W_t}] = e^{\frac{1}{2}\sigma^2 t}$. Indeed,

$$\begin{aligned}\mathbb{E}[e^{\sigma W_t}] &= \int_{-\infty}^{\infty} \frac{1}{\sqrt{2\pi t}} e^{-\frac{1}{2t}x^2} \cdot e^{\sigma x} dx\\ &= \int_{\mathbb{R}} \frac{1}{\sqrt{2\pi t}} e^{-\frac{1}{2t}((x-\sigma t)^2 - \sigma^2 t^2)} dx\\ &= e^{\frac{1}{2}\sigma^2 t}.\end{aligned}$$

Hence,

$$\mathbb{E}[e^{X_1^{(a)}}] = e^{3+4/2} = 148.41$$

(b) Using Question 3 we get

$$\mathbb{E}[X_1^{(b)}] = \frac{c}{1-\varphi} = 0.5/(1-0.6) = 1.25$$

(c) Apply the definition of expectation we have

$$\begin{aligned}\mathbb{E}[e^{X_1^{(c)}}] &= \sum_{k=0}^{\infty} e^{-\lambda} \frac{\lambda^k}{k!} e^k\\ &= e^{-\lambda} \sum_{k=0}^{\infty} \frac{(\lambda e)^k}{k!}\\ &= e^{\lambda(e-1)}.\end{aligned}$$

Plugging in $\lambda = 2$ we get $\mathbb{E}[e^{X_1^{(c)}}] = 31.08$.

(d) For the compound Poisson process we have to condition on N_1, i.e.,

$$\begin{aligned}\mathbb{E}[e^{X_1^{(d)}}] &= \mathbb{E}[\mathbb{E}[e^{X_1^{(c)}}|N_1]]\\ &= \sum_{k=0}^{\infty} \mathbb{E}[e^{X_1^{(c)}}|N_t]\mathbb{P}[N_1 = k]\\ &= \sum_{k=0}^{\infty} e^{-\lambda} \frac{\lambda^k}{k!} \mathbb{E}[\Pi_{j=1}^{k} e^{Z_j}]\\ &= e^{-\lambda} \sum_{k=0}^{\infty} \frac{(\lambda\beta)^k}{k!}\\ &= e^{\lambda(\beta-1)},\end{aligned}$$

Bitte wenden!

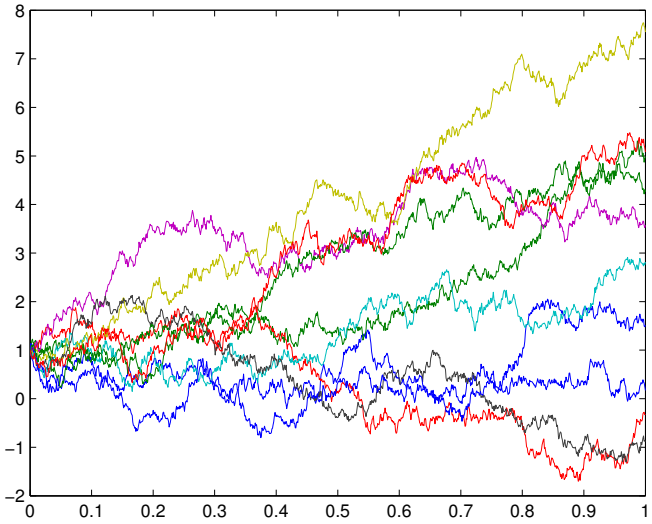
where $\beta = \mathbb{E}[e^{Z_1}] = 0.5(e + e^{-1})$. Plugging in $\lambda = 2$ we get $\mathbb{E}[e^{X_1^{(d)}}] = 2.96$.

5. see nelsonsiegel.m

6. Matlab Files

```
function [simulatedvalue, theoreticalvalue]= driftedBM
% Matlab simulation drifted BM X= 1+2t+2W_t
tic
%% parameter input
% horizon
T=1;
% sample size
Nplot=10;
Nsimu=10^5;
% grid points
M=10^3;
% volatility
sigma=2;
driftcoeff=2;

% theoretical value
theoreticalvalue= exp(1+driftcoeff*T+sigma^2/2*T);
%% Simulation
% BM
BM = [zeros(1,Nplot);sqrt(T/M)*cumsum(randn(M,Nplot))];
% the process X^(a)
timegrid= 0:T/M:T;
drift=repmat(timegrid',1,Nplot);
Xa=1+driftcoeff*drift+sigma*BM;

%plot the sample paths
plot(timegrid,Xa(:,:))

%compute simulated value
normalrv=sqrt(T)*randn(1,Nsimu);
simulatedvalue= mean(exp(1+driftcoeff*T+sigma*normalrv))
    ;

%estimated variance
estvar= var(exp(1+driftcoeff*T+sigma*normalrv));
% confidence interval using CLT
```

Siehe nächstes Blatt!




```
36 cfplus=simulatedvalue+1.96*sqrt(estvar/Nsimu);
37 cfminus=simulatedvalue -1.96*sqrt(estvar/Nsimu);
38
39 disp('Exact value:')
40 disp(theoreticalvalue)
41 disp('Estimated value: ')
42 disp(simulatedvalue)
43 disp('Confidence interval: ')
44 disp([cfminus,cfplus])
45 toc
```

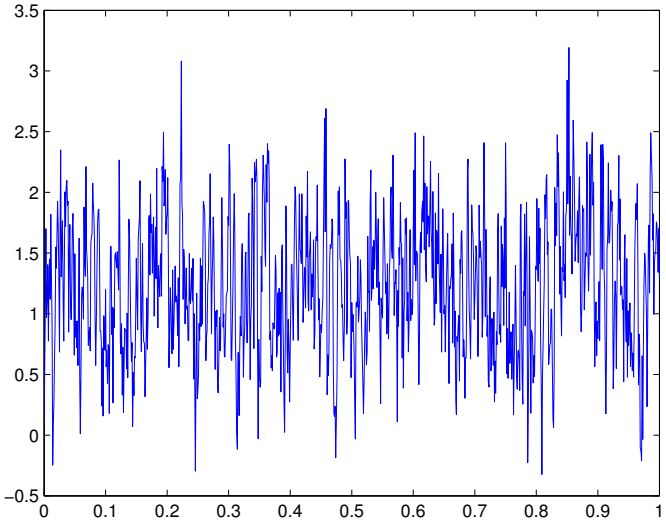
```
1 function [simulatedvalue,theoreticalvalue]= AR1
2 % In this exercise we simulated N paths of a AR(1)
      process
3 % X_t = c+ \varphi X_{t-1}+ \varepsilon_t, \varepsilon_t
      ~ i.i.d.
4 % N(0,\sigma^2)
5 tic
6 %% parameter input
7 % horizon
8 T=1;
9 % sample size
10 Nsimu=10^5;
11 Nplot=Nsimu;
12 % grid points
13 M=10^3;
```

Bitte wenden!

```
% volatility stdnrv = sigma
stdnrv=sqrt(0.2);
c=0.5;
phi=0.6;
AR0= 0.1;
% time step
dt= T/M;

% theoretical value
theoreticalvalue= c/(1-phi);
%% Simulation
AR = [AR0*ones(M,Nplot)];
% the process X^(b)
for i =1:M
    AR(i+1,:)=c+ phi*AR(i,:)+stdnrv.*randn(1,Nplot);
end

%plot the first sample path
timegrid= 0:dt:T;
plot(timegrid,AR(:,1))

%compute simulated value
simulatedvalue= mean(AR(end,:));

%estimated variance
estvar= var(AR(end,:));
% confidence interval using CLT
cfplus=simulatedvalue+1.96*sqrt(estvar/Nsimu);
cfminus=simulatedvalue-1.96*sqrt(estvar/Nsimu);

disp('Exact value:')
disp(theoreticalvalue)
disp('Estimated value: ')
disp(simulatedvalue)
disp('Confidence interval: ')
disp([cfminus,cfplus])
toc
```

```
function [simulatedvalue,theoreticalvalue]=
    poissonprocess
% In this exercise we simulated 10 paths of Poisson
    process with
% itensity lambda=2
```

Siehe nächstes Blatt!



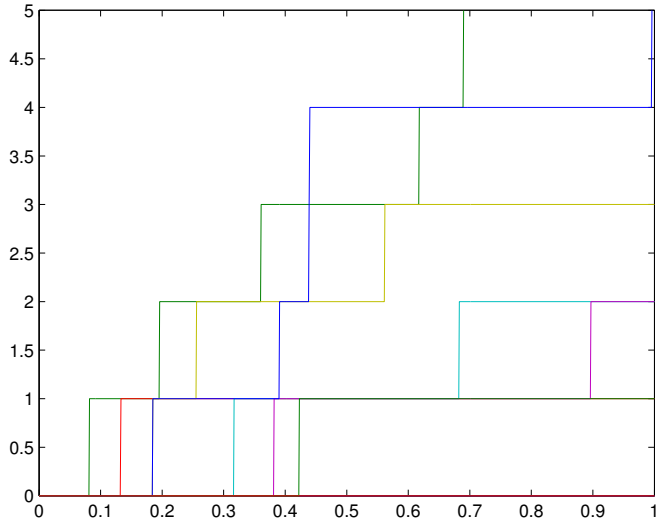

```
4 tic
5 %% parameter input
6 % horizon
7 T=1;
8 % sample size
9 Nplot=10;
10 Nsimu=10^5;
11 % grid points
12 M=10^3;
13 % intensity
14 lambda=2;
15
16 % theoretical value
17 theoreticalvalue= exp(lambda*(exp(1)-1));
18 %% Simulation
19 % method 1
20 figure(1)
21 % number of total jumps at T
22 NT= poissrnd(lambda*T,1,Nplot);
23 temp1= zeros(max(NT),Nplot);
24 for i =1:Nplot
25     %condition on N_T, the jump times are ordered
          uniformlly distributed
26     temp1(1:NT(i),i)= sort(T*rand(1,NT(i)));
27     stairs([0;temp1(1:NT(i),i)],(0:NT(i))');
28     hold on
```

Bitte wenden!

```
29 end
30 hold off
31
32 % method 2
33 PP = [zeros(1,Nplot);cumsum(poissrnd(lambda*T/M,M,Nplot)
       )];
34 % the process X^(c)
35 timegrid= 0:T/M:T;
36 Xc=PP;
37
38 figure(2)
39 %show the first 10 sample paths
40 plot(timegrid,Xc(:,:))
41
42
43 %compute simulated value
44 poissonrv=poissrnd(lambda*T,1,Nsimu);
45 simulatedvalue= mean(exp(poissonrv));
46
47 %estimated variance
48 estvar= var(exp(poissonrv));
49 % confidence interval using CLT
50 cfplus=simulatedvalue+1.96*sqrt(estvar/Nsimu);
51 cfminus=simulatedvalue-1.96*sqrt(estvar/Nsimu);
52
53 disp('Exact value:')
54 disp(theoreticalvalue)
55 disp('Estimated value: ')
56 disp(simulatedvalue)
57 disp('Confidence interval: ')
58 disp([cfminus,cfplus])
59 toc
```

```
1 function [simulatedvalue,theoreticalvalue]=
      compoundpoissonprocess
2 % In this exercise we simulated 10 paths of compound
      Poisson process
3 % X^(c)_t= \sum_{i=1}^N_t Z_i with itensity lambda=2 and
       Z_i = 1/-1 with
4 % prob=0.5
5 tic
6 %% parameter input
7 % horizon
```

Siehe nächstes Blatt!

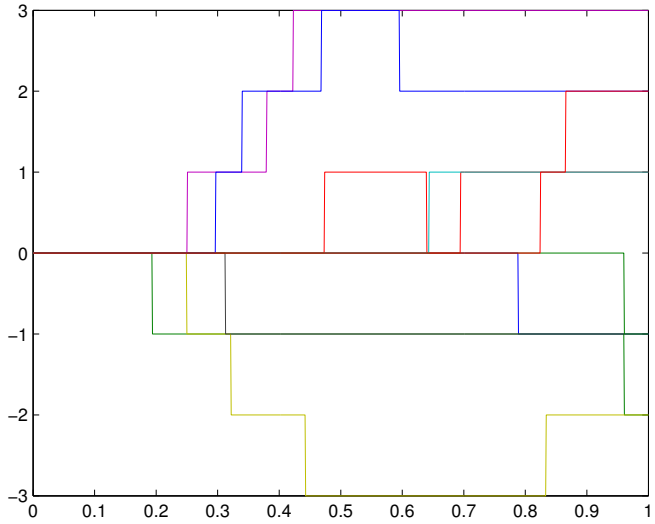



```
 8 T=1;
 9 % sample size
10 Nplot=10;
11 Nsimu=10^5;
12 % grid points
13 M=10^3;
14 % intensity
15 lambda=2;
16 % expection of e^Z
17 beta = 0.5*(exp(1)+exp(-1));
18
19 % theoretical value
20 theoreticalvalue= exp(lambda*T*(beta-1));
21 %% Simulation
22 % compound poisson process
23 figure(1)
24 % method 1
25 % number of total jumps at T
26 NT= poissrnd(lambda*T,1,Nplot);
27 temp1= zeros(max(NT),Nplot);
28 for i =1:Nplot
29     %condition on N_T, the jump times are ordered
          uniformlly distributed
30     temp1(1:NT(i),i)= sort(T*rand(1,NT(i)));
31     % simulate Z
32     stairs([0;temp1(1:NT(i),i)],cumsum([0,2*unidrnd(2,1,
```

Bitte wenden!

```
          NT(i))-3])')
33      hold on;
34  end
35  hold off;
36
37
38  % method 2
39  figure(2)
40  CPP = [zeros(1,Nplot);cumsum(poissrnd(lambda*T/M,M,Nplot
        ).*(2*unidrnd(2,M,Nplot)-3))];
41  % the process X^(a)
42  timegrid= 0:T/M:T;
43  Xc=CPP;
44
45  %show the first 10 sample paths
46  plot(timegrid,Xc(:,1:10))
47
48  %compute simulated value
49  % the poisson process N
50  poissonrv=poissrnd(lambda*T,1,Nsimu);
51  maxN=max(poissonrv);
52  % the jump distribution Z
53  bernoullirv= 2*unidrnd(2,maxN,Nsimu)-3;
54  % place holder
55  temp= zeros(1,Nsimu);
56  for i=1:maxN
57      % we only consider those numbers for which the
          poisson process is > 0
58      index= (poissonrv>0);
59      temp(index)=temp(index)+bernoullirv(i,index);
60      poissonrv(index)=poissonrv(index)-1;
61  end
62  simulatedvalue= mean(exp(temp));
63
64
65  %estimated variance
66  estvar= var(exp(temp));
67  % confidence interval using CLT
68  cfplus=simulatedvalue+1.96*sqrt(estvar/Nsimu);
69  cfminus=simulatedvalue-1.96*sqrt(estvar/Nsimu);
70
71  disp('Exact value:')
```

Siehe nächstes Blatt!



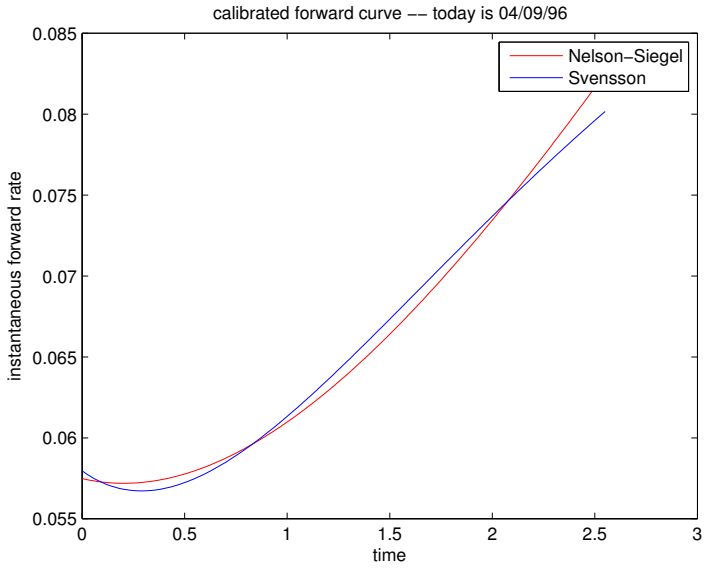

```
72 disp(theoreticalvalue)
73 disp('Estimated value: ')
74 disp(simulatedvalue)
75 disp('Confidence interval: ')
76 disp([cfminus,cfplus])
77 toc
```

```
 1 function [zoptns,zoptsv,resns,ressv]=nelsonsiegel
 2 % Given a finite number of bond prices we plot the
     calibrated forward curve
 3 %using Nelson Siegel family/ Svensson family.
 4
 5 %% Input
 6 %p= [n*1] bond price; n = number of bonds
 7 % Table 3.2 first 3 bonds
 8 %C= [n*N] CF matrix; N = number of cash flow dates (e.g
     ., coupon payments)
 9 %T= [N*1] payment tenor
10 p=[103.82;106.04;118.44];
11 C=zeros(length(p),1+3+6);
12 C(3,1)=6.125;
13 C(1,2)=105;
14 C(2,3)=4.875;
15 C(3,4)=6.125;
16 C(2,5)=4.875;
17 C(3,6)=6.125;
```

Bitte wenden!

```
18 C(2,7)=104.875;
19 C(3,8)=6.125;
20 C(3,9)=6.125;
21 C(3,10)=106.125;
22
23 % number of days between today 04/09/96 and the next CF
       date
24 T=[22;72;137;203;318;387;502;568;752;933];
25 % day count convention= actual/365
26 T=T/365;
27
28 %% Optimization
29 % bond prices d(z) using Nelson-Siegel Ansatz
30 % z is four-dimensional
31 funns = @(z,C) C*(exp(-(z(1)*T+z(2)/z(4)*(1-exp(-z(4)*T)
       )-z(3)/z(4)*T.*exp(-z(4)*T)+z(3)/(z(4)^2)*(1-exp(-z(4)
       *T)))));
32
33 % bond prices d(z) using Svensson Ansatz
34 % z is six-dimensional
35 funsv = @(z,C) C*(exp(-(z(1)*T+z(2)/z(5)*(1-exp(-z(5)*T)
       )-z(3)/z(5)*T.*exp(-z(5)*T)+z(3)/(z(5)^2)*(1-exp(-z(5)
       *T))-z(4)/z(6)*T.*exp(-z(6)*T)+z(4)/(z(6)^2)*(1-exp(-z
       (6)*T)))));
36
37 % numerical search for the least squared problem
38 % min || C*d(z) - p|| over z
39
40 z0ns=0.05*ones(4,1);
41 z0sv=0.1*ones(6,1);
42 [zoptns,resns]= lsqcurvefit(@(z,C) funns(z,C),z0ns,C,p);
43 [zoptsv,ressv]= lsqcurvefit(@(z,C) funsv(z,C),z0sv,C,p);
44
45
46 % plot the forward curve
47 figure(1)
48 time = 0:0.01:T(end);
49 NSforward= zoptns(1)+(zoptns(2)+zoptns(3)*time).*exp(-
       zoptns(4)*time);
50 SVforward= zoptsv(1)+(zoptsv(2)+zoptsv(3)*time).*exp(-
       zoptsv(5)*time)+zoptsv(4)*time.*exp(-zoptsv(6)*time);
51 plot(time,NSforward,'r-',time,SVforward,'b-')
```

Siehe nächstes Blatt!




```
52 legend('Nelson–Siegel','Svensson')
53 xlabel('time')
54 ylabel('instantaneous forward rate')
55 title('calibrated forward curve –– today is 04/09/96')
56 end
```