Online Appendix C

MATLAB Programs

This appendix contains a selection of basic MATLABm-file programs used in this text to produce figures and are listed here as sample code for readers. Since these m-files were used to produce figures for this book, they have more elaborate cosmetic figure enhancements, requiring full screen height, than would normally would be used for purely testing purposes.

C.1 Program: Uniform Distribution Simulation Histograms

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function uniform03fig1
% Fig. B.1 Book Illustration for Uniform RNG Simulation (3/2007):
clc % clear variables, but must come before globals,
%     else clears globals too.
clf % clear figures
fprintf('\nfunction uniform03fig1 OutPut:')
kfig = 0;
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
for m = 3:2:5
    kfig = kfig+1; figure(kfig);
    N=10^m;
    x=rand(N,1);
    xmean=mean(x);
    xstd=std(x);
    xmin = min(x);
    xmax = max(x);
    remean=(xmean*2-1)*100;
    restd=(xstd*sqrt(12)-1)*100;
    fprintf(...
        '\n fig=%i; m=%2i; N=%i; xmean=%f; xstd=%f; min(x)=%f; max(x)=%f'...
        ,kfig,m,N,xmean,xstd,xmin,xmax);
    fprintf('\n fig=%i; relerrmean=%f; relerrstd=%f;'...
```

bk0alf
2007/8/10
page C2


```
        ,kfig,remean,restd);
    nbins = 30; % min(fix(sqrt(10^m)),101);
    xmin = 0; xmax = 1;
    xbin1 = xmin; xbin2 = xmax; dxbin = (xbin2-xbin1)/nbins;
    xbin = xbin1+dxbin/2:dxbin:xbin2-dxbin/2;
    fprintf(...
        '\n fig=%i; #bins(x)=%4i; xbin in [%6f,%6f]; dxbin=%10f;'...
        ,kfig,nbins,xbin1,xbin2,dxbin)
    nx = hist(x,xbin); % Need Edge Oriented histc.
    bar(xbin,nx)
    axis tight
    title('Histogram for x = rand(N,1)'...
    ,'Fontsize',44,'FontWeight','Bold');
    ks = [0.1,0.8]; nxmax = max(nx);
    ytext=fix(ks(2)*nxmax); xtext=ks(1);
    textn=['N = ' int2str(N)];
    text(xtext,ytext,textn...
    ,'Fontsize',40,'FontWeight','Bold');
    ylabel('Uniform Bin Frequency'...
    ,'Fontsize',44,'FontWeight','Bold');
    xlabel('x, Uniform rand-Deviate'...
    ,'Fontsize',44,'FontWeight','Bold');
    patchobj = findobj(gca,'Type','patch');
    set(patchobj,'FaceColor','w','EdgeColor','k','linewidth',2);
    set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
    set(gcf,'Color','White','Position'...
        ,[scrsize(3)/ss(kfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
end
% End uniform03fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.2 Program: Normal Distribution Simulation Histograms

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function normal03fig1
% Fig. B.2 Book Illustration for Normal RNG Simulation (3/2007):
clc % clear variables, but must come before globals,
%     else clears globals too.
clf % clear figures
fprintf('\nfunction normal03fig1 OutPut:')
kfig = 0;
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
for m = 3:2:5
    kfig = kfig+1; figure(kfig);
    N=10^m;
    x=randn(N,1);
    xmean=mean(x);
```

```
    xstd=std(x);
    remean=xmean*100;
    restd=(xstd-1)*100;
    fprintf('\nNormal Random Deviate (MATLAB randn) Test:');
    fprintf('\n fig=%i; m=%2i; N=%i; xmean=%f; xstd=%f;'...
        ,kfig,m,N,xmean,xstd);
    fprintf('\n fig=%i; relerrmean=%f; relerrstd=%f;'....
        ,kfig,remean,restd);
    nbins = 50; % min(fix(sqrt(10^m)),101);
    xmin = min(x); xmax = max(x);
    xbin1 = xmin; xbin2 = xmax; dxbin = (xbin2-xbin1)/nbins;%
    fprintf('\n#bins(x)=%4i; xbin in [%6f,%6f]; dxbin=%10f;'...
        ,nbins,xbin1,xbin2,dxbin)
    xbin = xbin1+dxbin/2:dxbin:xbin2-dxbin/2;
    nx = hist(x,xbin); % Need Center Oriented hist.
    bar(xbin,nx)
    axis tight
    title('Histogram for x = randn(N,1)'...
    ,'Fontsize',44,'FontWeight','Bold');
    ks = [0.4,0.7];  nxmax = max(nx);
    xtext = xmax*(ks(1)-(kfig-1)*0.1); ytext=fix(ks(2)*nxmax);
    textn=['N = ' int2str(N)];
    text(xtext,ytext,textn...
        ,'Fontsize',40,'FontWeight','Bold');
    ylabel('Normal Bin Frequency'...
        ,'Fontsize',44,'FontWeight','Bold');
    xlabel('x, Normal randn-Deviate'...
        ,'Fontsize',44,'FontWeight','Bold');
    patchobj = findobj(gca,'Type','patch');
    set(patchobj,'FaceColor','w','EdgeColor','k','linewidth',2);
    set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
    set(gcf,'Color','White','Position'...
        ,[scrsize(3)/ss(kfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
end
% End normal03fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.3 Program: Lognormal Distribution Simulation Histograms

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function lognormal03fig1
% Fig. B.3 Book Illustration for LogNormal RNG Simulation (3/2007):
clc % clear variables, but must come before globals,
%     else clears globals too.
clf % clear figures
fprintf('\nfunction lognormal03fig1 OutPut:')
kfig = 0; mu = 0.0; sig = 0.5;
muln = exp(mu+sig^2/2);
```

```
sigln = muln*sqrt(exp(sig^2) -1);
nbins = 150;
fprintf('\n mu=%f; sig=%f; muln=%f; sigln=%f; nbins=%i'...
    ,mu,sig,muln,sigln,nbins);
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
for m = 3:2:5
    kfig = kfig+1; figure(kfig);
    N = 10^m;
    y = mu*ones(N,1) + sig*randn(N,1);
    x = exp(y);
    xmean=mean(x);
    xstd=std(x);
    remean=(xmean/muln - 1)*100;
    restd=(xstd/sigln - 1)*100;
    fprintf('\nLognormal Random Deviate (exp(mu+sig*randn)) Test:');
    fprintf('\n fig=%i; m=%2i; N=%i; xmean=%f; xstd=%f;'...
        ,kfig,m,N,xmean,xstd);
    fprintf('\n fig=%i; relerrmean=%f; relerrstd=%f;'...
        ,kfig,remean,restd);
    xmin = min(x); xmax = max(x);
    xbin1 = xmin; xbin2 = xmax; dxbin = (xbin2-xbin1)/nbins;%
    fprintf('\n#bins(x)=%4i; xbin in [%6f,%6f]; dxbin=%10f;'...
        ,nbins,xbin1,xbin2,dxbin)
    xbin = xbin1+dxbin/2:dxbin:xbin2-dxbin/2;
    nx = hist(x,xbin); % Need Center Oriented hist.
    bar(xbin,nx)
    axis tight
    title('Histogram for Lognormal x'...
        ,'Fontsize',44,'FontWeight','Bold');
    ks = [0.4,0.7];  nxmax = max(nx);
    xtext = xmax*(ks(1)-(kfig-1)*0.1); ytext=fix(ks(2)*nxmax);
    textn=['N = ' int2str(N)];
    text(xtext,ytext,textn...
        ,'Fontsize',40,'FontWeight','Bold');
    ylabel('Lognormal Bin Frequency'...
        ,'Fontsize',44,'FontWeight','Bold');
    xlabel('x, Lognormal Deviate'...
        ,'Fontsize',44,'FontWeight','Bold');
    patchobj = findobj(gca,'Type','patch');
    set(patchobj,'FaceColor','w','EdgeColor','k','linewidth',2);
    set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
    set(gcf,'Color','White','Position'...
        ,[scrsize(3)/ss(kfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
end
% End lognormal03fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.4 Program: Exponential Distribution Simulation Histograms

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function exponential03fig1
% Fig. B. 4 Book Illustration for Exponential RNG Simulation
% with mean one (3/2007):
clc % clear variables, but must come before globals,
%     else clears globals too.
clf % clear figures
fprintf('\nfunction exponential03fig1 OutPut:')
kfig = 0; mu = 1.0;
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
for m = 3:2:5
    kfig = kfig+1; figure(kfig);
    N=10^m;
    x=-mu*log(rand(N,1));
    xmean=mean(x);
    xstd=std(x);
    remean=(xmean/mu-1)*100;
    restd=(xstd/mu-1)*100;
    fprintf('\nExponential Random Deviate (MATLAB randn) Test:');
    fprintf('\n fig=%i; m=%2i; N=%i; xmean=%f; xstd=%f;'...
        ,kfig,m,N,xmean,xstd);
    fprintf('\n fig=%i; relerrmean=%f; relerrstd=%f;'...
        ,kfig,remean,restd);
    nbins = 50; % min(fix(sqrt(10^m)),101);
    xmin = 0; xmax = max(x);
    xbin1 = xmin; xbin2 = xmax; dxbin = (xbin2-xbin1)/nbins;%
    fprintf('\n#bins(x)=%4i; xbin in [%6f,%6f]; dxbin=%10f;'...
        ,nbins,xbin1,xbin2,dxbin)
    xbin = xbin1+dxbin/2:dxbin:xbin2-dxbin/2;
    nx = hist(x,xbin); % using centered defined bins,
    %                    rather than edge bins
    bar(xbin,nx)
    axis tight
    title('Histogram for x = -ln(rand(N,1))'...
        ,'Fontsize',44,'FontWeight','Bold');
    ks = [0.6,0.6];  nxmax = max(nx);
    xtext = xmax*ks(1); ytext=fix(ks(2)*nxmax);
    textn=['N = ' int2str(N)];
    text(xtext,ytext,textn...
        ,'Fontsize',40,'FontWeight','Bold');
    ylabel('Exponential Bin Frequency'...
        ,'Fontsize',44,'FontWeight','Bold');
    xlabel('x, Exponential random-Deviate'...
        ,'Fontsize',44,'FontWeight','Bold');
    patchobj = findobj(gca,'Type','patch');
    set(patchobj,'FaceColor','w','EdgeColor','k','linewidth',2);
```

bk0alfi
2007/8/10
page C6


```
    set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
    set(gcf,'Color','White','Position'...
        ,[scrsize(3)/ss(kfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
end
% End exponential03fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.5 Program: Poisson Distribution Versus Jump Counter k

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function poisson03fig1
% Fig. B.5 Book Illustration for Poisson distribution
% with 3 parameter values (3/2007):
clc % clear variables, but must come before globals,
%     else clears globals too.
fprintf('\nfunction poisson03fig1 OutPut:');
lv =[0.2,1.0,2.0,5.0]; nlam = 4;
nk = 10; kv = 0:nk;
pv = zeros(nk+1,nlam);
for ilam = 1:nlam
    pv(1,ilam) = exp(-lv(ilam));
    for k = 1:nk
        kv(k+1) = k;
        pv(k+1,ilam) = pv(1,ilam)*(lv(ilam))^k/factorial(k);
    end
end
kfig = 1; figure(kfig);
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
plot(kv,pv(:,1),'ko--',kv,pv(:,2),'k^:',kv,pv(:,3),'ks-.'...
    ,kv,pv(:,4),'kd-.'...
    ,'MarkerSize',10,'MarkerFaceColor','k','LineWidth',2)
title('Poisson Distributions: p_k(\Lambda)'...
        ,'Fontsize',44,'FontWeight','Bold');
ylabel('p_k(\Lambda)'...
        ,'Fontsize',44,'FontWeight','Bold');
xlabel('k, Poisson Counter'...
        ,'Fontsize',44,'FontWeight','Bold');
hlegend=legend('\Lambda = 0.2','\Lambda = 1.0','\Lambda = 2.'...
    ,'\Lambda = 5.','Location','NorthEast');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
    ,[scrsize(3)/ss(kfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
% End poisson03fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.6 Program: Binomial Distribution Versus Binomial Frequency f_1

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function binomial03fig1
% Fig. B.6 Book Illustration for Binomial distribution (3/2007):
%     with 3 \pi_1 parameter values.
% pv(f_1) = p(f1,N-f1;\pi_1,1-\pi_1)
%         = Bi(N,f_1)*\pi_1^{f_1}*(1-\pi_1)^{N-f_1}
clc % clear variables, but must come before globals,
%     else clears globals too.
clf % clear figures
fprintf('\nfunction binomialfig03 OutPut:');
pi1v =[0.25,0.5,0.75]; npi1 = 3;
N = 10; f1v = 0:N; nfact = factorial(N);
pv = zeros(N+1,npi1);
for ipi = 1:npi1
    pi1 = pi1v(ipi);
    pv(1,ipi) = (1-pi1)^N;
    for f1 = 1:N
        pv(f1+1,ipi) = nfact/(factorial(f1)*factorial(N-f1))...
            *pi1^f1*(1-pi1)^(N-f1);
    end
end
kfig = 1; figure(kfig);
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
plot(f1v,pv(:,1),'ko--',f1v,pv(:,2),'k^:',f1v,pv(:,3),'ks-.'...
    ,'MarkerSize',10,'MarkerFaceColor','k','LineWidth',2)
title(...
    'Binomial Distributions: p_1(f_1) = p(f_1,N-f_1;\pi_1,1-\pi_1)'...
        ,'Fontsize',44,'FontWeight','Bold');
ylabel('p_1(f_1)'...
        ,'Fontsize',44,'FontWeight','Bold');
xlabel('f_1, Binomial Frequency'...
        ,'Fontsize',44,'FontWeight','Bold');
hlegend=legend('\pi_1 = 0.25','\pi_1 = 0.50','\pi_1 = 0.75'...
    ,'Location','NorthEast');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
    ,[scrsize(3)/ss(kfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
% End binomial03fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.7 Program: Simulated Diffusion $W(t)$ Sample Paths

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

bk0alfi
2007/8/10
page C8


```
function wiener06fig1
% Fig. 1.1a Book Illustrations for Wiener/Diffusion Process (3/2007)
%   RNG Simulation for t in [0,1] with sample variation.
%   Generation is by summing Wiener increments DW of even spacing Dt.
clc % clear workspace of prior output.
clear % clear variables, but must come before globals,
% else clears them.
fprintf('\nfunction wiener06fig1 OutPut:'); % print code figure name
nfig = 0;
N = 1000; TF = 1.0; Dt = TF/N; % Set initial time grid: Fixed Delta{t}.
NP = N + 1; % Number of points.
sqrtdt = sqrt(Dt);  % Set standard Wiener increment moments'
% for dX(t) = mu*dt + sigma*dW(t); here mu = 0, sigma = 1
%     and scaled dW(t) = sqrt(dt)*randn
% Begin Calculation:
tv = 0:Dt:TF; % time row-vector
nstate = 4; % number of states
jv = [1,2,3,4]; % selection of states; change when needed
DWv = zeros(nstate,N); Wv = zeros(nstate,NP); % DW & W vectors/arrays;
% Also sets initial Wv(j,1) = 0;
for j = 1:nstate
    randn('state',jv(j)); % Set initial state for repeatability;
    DWv(j,1:N) = sqrtdt*randn(1,N);  %Generate N random row-vector;
    for i=1:N % Simulated Sample paths by Increment Accumulation:
        Wv(j,i+1) = sum(DWv(j,1:i)); % Note Wv(j,1) = 0.0; effic. sum
    end
end
%%%%% Begin Plot:
nfig = nfig + 1;
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
fprintf('\n\nFigure(%i):  Diffusion Simulated Sample Paths(4)\n'...
    ,nfig)
figure(nfig)
marks = {'k-','k-o','k-^','k-x'}; % easier to change nstate marks
%
for j = 1:nstate
    plot(tv,Wv(j,1:NP),marks{j},'linewidth',2); hold on;
end
hold off
%
title('Diffusion Simulated Sample Paths (4)'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('W(t), Wiener State'...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('t, Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('State 1','State 2','State 3','State 4'...
    ,'Location','SouthWest');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
```

```
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
% [l,b,w,h]
% End wiener06fig1
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.8 Program: Simulated Diffusion $W(t)$ Sample Paths Showing Variation with Time Step Size

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function wiener06fig2
% Fig. 1.1b Book Illustration for Wiener/Diffusion Process (3/2007)
%   RNG Simulation for t in [0,1] with sample variation.
%   Generation is by summing Wiener increments DW of even spacing Dt.
clc % clear workspace of prior output.
clf % clear figures, else accumulative.
clear % clear variables, but must come before globals, else clears.
fprintf('\nfunction wiener06fig2 OutPut:'); % print code figure name
nfig = 1;
N = 1000; TF = 1.0; Dt = TF/N; % Set time grids: Several dt's.
NP = N+1; % Total number of Points.
% for dX(t) = mu*dt + sigma*dW(t); here mu = 0, sigma = 1
%     and scaled dW(t) = sqrt(dt)*randn
% Begin Calculation:
% nstate = 1; % number of states.
ndt = 3; % number of local dt's.
jv = [1,2,3,4]; % selection of states; change when needed
randn('state',jv(1));  % Set common initial state for repeatability
RN = randn(1,N); % common random sample of N points.
Wv = zeros(ndt,NP); % W array of local vectors;
% Also sets all Wv(kdt,1) = 0 for tv(1) = 0;
% recall MATAB is unit based.
ts = zeros(ndt,NP); % Declare maximal local time vectors;
%%%%% Begin Plot:
nfig = nfig + 1;
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
fprintf('\n\nFigure(%i):  Diffusion Simulated Sample Paths(4)\n'...
    ,nfig)
figure(nfig)
marks = {'k-','k-o','k-^','k-x'}; % easier to change nstate marks
%
for kdt = 1:ndt % Test Multiple Sample Paths with different dt's:
    S = 10^(kdt-1); % dt scalar factor;
    Ns = N/S; NPs = Ns+1; % Local counts;
    Dts = S*Dt; % Local time steps;
    sigs = sqrt(Dts); % Local diffusion scaling;
    ts(kdt,1:NPs) = 0:Dts:TF; % Local times;
```

bk0alli
2007/8/10
page C10


```
    for i = 1:Ns % Simulated Sample paths by Increment Accumulation:
        Wv(kdt,i+1) =  Wv(kdt,i) + sigs*RN(1,i*S);
    end
    plot(ts(kdt,1:NPs),Wv(kdt,1:NPs),marks{kdt},'linewidth',2);
    hold on;
end
%
hold off
%
title('Diffusion Simulations: \Delta{t} Effects'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('W(t), Wiener State'...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('t, Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('\Delta{t} = 10^{-3}, N = 1000'...
    ,'\Delta{t} = 10^{-2}, N = 100'...
    ,'\Delta{t} = 10^{-1}, N = 10','Location','SouthWest');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
% [l,b,w,h]
% End wiener06fig2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.9 Program: Simulated Simple Poisson $P(t)$ Sample Paths

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function poisson03fig2
% Fig. 1.2a Book Illustration for Simple Poisson/Jump Process (3/2007)
%   RNG Simulation for P(t) = 1:K jumps with sample variation.
%   Generation is by Poisson Jump Exponentially distributed
%      jump time increments T(k+1)-T(k), T(k+1) = kth jump time,
%      T(1) := 0.
%
clc % clear variables, but must come before globals,
%     else clears globals too.
fprintf('\nfunction poisson03fig2 OutPut:')
nfig = 0;
K = 10; KP = 2*K +1; % Include sample of K jumps only.
p = zeros(KP,1); kstates = 4; LT = zeros(KP,kstates);
% Begin Calculation:
for kstate = 1:kstates; % Test Multiple Simulated Sample Paths:
    LT(1,kstate) = 0; p(1) = 0; % Set initial scaled jump time
    %                             and jump count.
    rand('state',kstate);  % Set initial state for repeatability
    %                        or path change.
```

```
    DTe = -log(rand(K,1));  % Generate random vector of
    %                         K exponential variates.
    for k = 1:K % Simulated sample scaled jump times
        %         LT(k+1) = lambda*T(k+1):
        LT(2*k,kstate) = LT(2*k-1,kstate) + DTe(k);
        LT(2*k+1,kstate) = LT(2*k,kstate);
        p(2*k) = p(2*k-1);
        p(2*k+1) = p(2*k-1) + 1;
    end
end
% Begin Plot:
nfig = nfig + 1;
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
fprintf('\n\nFigure(%i):  Simulated Jump Sample Paths\n',nfig)
figure(nfig)
plot(LT(1:KP,1),p,'k-',LT(1:KP,2),p,'k:',LT(1:KP,3),p,'k-.'...
    ,LT(1:KP,4),p,'k--','LineWidth',2);
title('Simulated Simple Jump Sample Paths'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('P(t), Poisson State'...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('\lambda t, Scaled Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('Sample 1','Sample 2','Sample 3','Sample 4'...
    ,'Location','Southeast');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
% End poisson03fig2
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.10 Program: Simulated Simple Incremental Poisson $\Delta P(t)$ Sample Paths

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function poisson03fig3
% Fig. 1.2b Book Illustration for Simple Incremental (3/2007)
%   Poisson/Jump Process RNG Simulation for
%      Delta{P}(t) = P(t+Delta{t})-P(t) = 1:K jumps
%   with sample variation.
%   Generation is by Poisson Jump Zero-One Law:
%         Prob(Delta{P}(t)=0] = 1-lambda*dt,
%      assuming sufficiently small Delta{t}'s.
%
clc % clear variables, but must come before globals,
%     else clears globals too.
```

```
clf % clear figures, else accumulative.
fprintf('\nfunction delpois03fig3 OutPut:')
nfig = 3;
figure(nfig);
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
marks = {'k-','k:','k-.','k--'};
% marks = {'k-o','k:s','k-.^','k--d'};
K = 10; KS = 500;  KP = KS + 1; % Include first K jumps from total
%                                 KS sample only.
kstates = 4; DP = zeros(KP,kstates); DT = zeros(KP,kstates);
% Begin Calculation:
for kstate = 1:kstates; % Test Multiple Simulated Sample Paths:
    k = 0; DP(1,kstate) = 0.0; DT(1,kstate) = 0;  % Set initial
    %                                               jump parms.
    rand('state',kstate-1);  % Set initial state for repeatability
    %                          or path change.
    xu = rand(KS,1);  % Generate random vector of K uniform variates.
    dt = 0.05; lambda = 1.0; % Set time step and jump rate.
    ldt = lambda*dt; % one jump prob.
    xl = (1-ldt)/2;  xr = (1+ldt)/2; % Set centered jump probability
    %                                  thresholds, using centered
    %                                  part of uniform distribution
    %                                  to avoid open end point bias.
    ip = 0; % Set plot counter.
    for i = 1:KS % Simulated sample scaled jump times
        %          LT(k+1) = lambda*T(k+1):
        ip = ip + 1;
        if xu(i) <= xr && xu(i) >= xl % Get jump if prob. in [xl,xr].
            k = k + 1;
            DP(ip+1,kstate) = DP(ip,kstate);
            DT(ip+1,kstate) = DT(ip,kstate) + dt;
            ip = ip + 1;
            DP(ip+1,kstate) = DP(ip,kstate) + 1;
            DT(ip+1,kstate) = DT(ip,kstate);
        else
            DP(ip+1,kstate) = DP(ip,kstate);
            DT(ip+1,kstate) = DT(ip,kstate) + dt;
        end
        if k == K
            KP = ip + 1;
            fprintf('\n kstate = %i; i = %i points; k = %i jumps;'...
                ,kstate-1,i,k);
            break;
        end
    end
    plot(DT(1:KP,kstate),DP(1:KP,kstate),marks{kstate}...
        ,'LineWidth',2), hold on
end
% Begin Plot:
```

```
fprintf('\n\nFigure(%i):  Simulated Small \Delta{t} Jump Paths\n'...
    ,nfig)
title('Simulated Small \Delta{t} Simple Jump Sample Paths'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('\Delta{P}(t), Poisson State'...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('t,  Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('Sample 1','Sample 2','Sample 3','Sample 4',0);
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
% End poisson03fig3
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.11 Program: Simulated Diffusion Integrals $\int(dW)^2(t)$ by Itô Partial Sums

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function intdwdw
% Fig. 2.1 Example MATLAB code for integral of (dW)^2 (3/2007):
clc % clear variables;
t0 = 0.0; tf = 1.0;
n = 1.0e+4; nf = n + 1; % set time grid: (n+1) subintervals
dt = (tf-t0)/nf;  %       and (n+2) points;
% replace these particular values according the application;
sqrtdt = sqrt(dt); % dW(i) noise time scale so E[dW] = 0;
kstate = 1; randn('state',kstate); % Set randn state
%    for repeatability;
dW = sqrtdt*randn(nf,1); % simulate (n+1)-dW(i)'s sample;
t = t0:dt:tf; % get time vector t;
W = zeros(1,nf+1); % set initial diffusion noise condition;
sumdw2 = zeros(1,nf+1); % set initial integral sum;
for i = 1:nf % simulate integral sample path.
    W(i+1) = W(i) + dW(i); % sum diffusion noise;
    sumdw2(i+1) = sumdw2(i) + (dW(i))^2; % sum whole integrand;
end
fprintf('\n\nFigure 1: int[(dW)^2](t) versus t\n');
nfig = 1;
figure(nfig);
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
plot(t,sumdw2,'k-',t,t,'k--','LineWidth',2); % plot sum and t;
title('\int(dW)^2(t) Simulations versus t'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('\int(dW)^2(t) and t, States'...
    ,'FontWeight','Bold','Fontsize',44);
```

```
xlabel('t, Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('\int(dW)^2(t)','t'...
    ,'Location','Southeast');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
% End intdwdw
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.12 Program: Simulated Diffusion Integrals $\int g(W,t)dW$: Direct Case by Itô Partial Sums

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function intgwtdw
% Fig. 4.1 Book code example for int[g(w,t)dw] on [t0,t]
%   by RNG Simulation (3/2007):
%   Generation is by summing g(W(i),t(i))dW(i) of even spacing dt
%      for i=0:n, but converted to from index base 0 to base 1:
%      matlab[G(W(i),T(i))DW(i);i=1:N+1]...
%             = math[g(W(i),t(i))dW(i);i=0:n].
%   Sample g(w,t) = exp(w-t/2) with exact integral g(w,t) - 1
%      on [0,t].
clc % clear variables, but must come before globals,
%         else clears globals too.
clf % clear figures
fprintf('\nfunction intgwtdw OutPut:')
nfig = 0; % figure counter.
TF= 2.0; T0 = 0; N = 20000;  NI = N+1; dt = (TF-T0)/NI; % Set initial
%                     time grid:  Fixed Delta{t}.
sqrtdt = sqrt(dt); % Set std. Wiener increment time scale.
%
% Begin Sample Path Calculation:
kstate = 1;
randn('state',kstate);  % set randn state for repeatability.
DW = sqrtdt*randn(1,NI); % Generate normal random vector of N+1
%                           samples for dW(t).
T = zeros(1,NI+1); T(1) = 0;  % set T initially.
W = zeros(1,NI+1);  % Set W(1) in place of W(0) = 0 for base 1 vector.
S = zeros(1,NI+1);  % Set integral sum initially.
gv = zeros(1,NI+1); gv(1) = g(W(1),T(1)); % Set integrand initially.
Err = zeros(1,NI+1); % Set Error initially.
for i = 1:NI % Simulated Sample paths by Increment Accumulation:
    T(i+1) = T(i) + dt;
    W(i+1) = W(i) + DW(i);
    gv(i+1) = g(W(i+1),T(i+1));
    S(i+1) = S(i) + gv(i)*DW(i);% integrand g defined in subfunction.
```

```
    Err(i+1) = S(i+1) - (gv(i+1) -gv(1)); % CAUTION: FOR KNOWN g HERE!
end
T(1,NI+1) = TF;  % Correct for final cumulative time rounding errors.
% Begin Plot:
nfig = nfig + 1;
fprintf('\n\nFigure(%i):  int[g](t) versus t Simulations\n',nfig)
figure(nfig);
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
plot(T,S,'k-',T,W,'k-.',T,Err,'k--','LineWidth',2);
title('\int g(W,t)dW(t) for g = exp(W(t)-t/2)'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('\int g(W,t)dW(t), W(t), g(W(t),t) - g(0,0)'...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('t, Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('\int g(W,t)dW(t)','W(t)','Error(t)'...
    ,'Location','SouthWest');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
% End Main
function gv = g(W,T)
% Example g(W(t),t) = exp(W(t) - t/2); exact integral = g(W(t),t) - 1.
gv = exp(W - T/2);
% End intgwtdw
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.13 Program: Simulated Diffusion Integrals $\int g(X,t)dW$: Chain Rule

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function intgxtdw
% Fig. 4.2 Book code example for int[g(x(t),t)dw] on [t0,t] (3/2007)
%   by RNG Simulation:
%   Generation is by summing g(X(i),t(i))dW(i) of even spacing dt for
%      i=0:n, but converted to from index base zero to base one:
%      matlab[G(X(i),T(i))DW(i);i=1:N+1] = math[g(X(i),t(i))dW(i);i=0:n].
%   Chain Rule Form: Int[gdW](t) = G(W,t)-G(0,0) - Int[(g_t+0.5*g_w)(w,t)dt];
%      G_w(w,t) = g(w,t), G_{ww}(w,t) = g_w(w,t).
%   Sample Test Code for various g's, here for g(x,t) = exp(x) and x = w.
clc % clear variables, but must come before globals,
%     else clears globals too.
clf % clear figures
fprintf('\nfunction intgxtdw OutPut:')
nfig = 0; % figure counter.
TF= 2.0; T0 = 0; N = (TF-T0)*10000;  NI = N+1; dt = (TF-T0)/NI;
```

```
% Set initial time grid: Fixed Delta{t}, Scaled to [T0,TF] with N.
sqrtdt = sqrt(dt); % Set standard Wiener increment time scale.
% Begin Sample Path Calculation:
kstate = 1;
randn('state',kstate);  % set randn state for repeatability.
dW = sqrtdt*randn(1,NI); % Generate normal random vector of N+1
%                          samples for dW(t).
t = zeros(1,NI+1); t(1) = T0;  % set T initially.
W = zeros(1,NI+1);  % Set W(1) in place of W(0) = 0 or base 1 vector.
X = zeros(1,NI+1);  % Set integral sum initially.
gv = zeros(1,NI+1); gv(1) = g(X(1),t(1)); % Set integrand initially.
sdw = zeros(1,NI+1); sdt = zeros(1,NI+1); % Set integral sum initially.
ev = zeros(1,NI+1); % Set Error initially.
for i = 1:NI % Simulated Sample paths by Increment Accumulation:
    t(i+1) = i*dt;
    W(i+1) = W(i) + dW(i);
    X(i+1) = W(i+1); % Set State for this g Example.
    gv(i+1) = g(X(i+1),t(i+1));
    sdw(i+1) = sdw(i) + gv(i)*dW(i);% integrand g in subfunction.
    sdt(i+1) = sdt(i) - gthgw(X(i+1),t(i+1))*dt;% gthgw in subfunction.
    ev(i+1) = sdw(i+1) - exact(X(i+1),t(i+1)) - sdt(i+1);
    % CAUTION: For given g only!
end
t(NI+1) = TF;  % Correct for final cumulative time rounding errors.
% Begin Plot:
nfig = nfig + 1;
fprintf('\n\nFigure(%i):  int[g](t) versus t Simulations\n',nfig)
figure(nfig)
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
plot(t,sdw,'k-',t,W,'k-.',t,ev,'k--','LineWidth',2);
title('\int g(X,t)dW(t) for g = exp(X), X = W'...
    ,'FontWeight','Bold','Fontsize',44);
ylabel('\int g(X,t)dW(t), X = W(t) and Error(t)'...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('t, Time'...
    ,'FontWeight','Bold','Fontsize',44);
hlegend=legend('\int g(X,t)dW(t)','X = W(t)','Error(t)'....
    ,'Location','SouthWest');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
% End Main
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gv = g(x,t)  % ignore caution that t might not be used here.
% Sample g(X(t),t) only, e.g.,
%1% gv = exp(x-t/2); % x = w.
%2% gv = exp(x); % x = w.
```

```
%3% gv = x; % x = w.
gv = exp(x);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function exactv = exact(x,t)
% Sample g(X(t),t) exact integrals only, e.g.,
%1% exactv = exp(x-t/2) - 1; % i.e., G(w,t)-G(0,0), x=w,
%      G(w,t)=exp(w-t/2).
%2% exactv = exp(x) - 1; % i.e., G(w,t)-G(0,0), x=w, G(w,t)=exp(w).
%3% exactv = 0.5*(x^2-t);  % i.e., G(w,t)-G(0,0), x=w,
%      G(w,t)=0.5*(w^2-t).
exactv = exp(x) - 1; % i.e., G(w,t)-G(0,0), x=w, G(w,t)=exp(w).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function gthgwv = gthgw(x,t)
% Reg. Correction Int. of (G_t+0.5*G_{ww})(X(t),t), G_w = g.
%1% gthgwv = 0; % i.e., g=exp(x-t/2)=G, G_t=-0.5*G, G_{ww}=G.
%2% gthgwv = 0.5*exp(x); % i.e., G=g=exp(w), G_t=0, G_{ww}=g_w=exp(w).
%3% gthgw 0; % i.e., g=x=w, G=0.5*(w^2-t), G_t=-0.5, G_{ww}=g_w=1;
gthgwv = 0.5*exp(x); % i.e., G=g=exp(w), G_t=0, G_{ww}=g_w=exp(w).
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% End intgxtdw
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.14 Program: Simulated Linear Jump-Diffusion Sample Paths

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function linjumpdiff03fig1
% Figs. 4.3, 4.4, 4.5 Book Illustrations for Linear
%   (Geometric) Jump-Diffusion (3/2007) SDE RNG Simulation
%   with constant coefficients for t in [0,1]
%   with sample variation:
%        DX(t) = X(t)*(mu*Dt + sig*DW(t) + nu*DP(t),
%            X(0) = x0.
%   Or log-state:
%        DY(t) = (mu-sig^2/2)*Dt + sig*DW(t) + log(1+nu)*DP(t),
%       Y(0) = log(x0).
%   Generation is by summing Wiener increments DW of even spacing Dt
%     with Poisson jump increment added at correct time increment.
%   Sufficiently SMALL increments assumed, so zero-one jump law is
%       appropriate.
%   Allows Separate Driver Input and Special Jump
%       or Diffusion Handling.
clc % clear variables, but must come before globals,
    %    else clears globals too.
clf % clear figures
fprintf('\nfunction linjumpdiff03fig1 OutPut:');
%%% Initialize input to jdsimulator
N = 1000; T = 1.0; % Set initial time grid: Fixed Delta{t}.
mu = 0.5; sig = 0.10; nu = -0.10; lambda = 3.0;
```

bk0alff
2007/8/10
page C18


```
% set constant parameters.
%
jdsimulator(mu,sig,nu,lambda,N,T);
%
% END INPUT FOR JUMP-DIFFUSION SIMULATOR.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function jdsimulator(mu,sig,nu,lambda,N,T)
idiff = 1; ijump = 1;
if sig == 0, idiff = 0; end
if nu == 0, ijump = 0; end
nfig = 0; % figure counter.
NI = N+1;  Dt = T/NI;
iv = 2; % iv=1 for *(1+/-sqrt(Var[X])) or iv=2 for *exp(+/-sqrt(Var[Y]).
sqrtdt = sqrt(Dt); % Set standard Wiener increment moments.
muddt = (mu - sig^2/2)*Dt; % Get Ito diffusion corrected drift term.
lognu = log(1 + nu); % Get log of relative jump term amplitude.
% Begin Sample Path Calculation:
t = 0:Dt:T; kstates = 4; x0 = 1.0;
XS = zeros(NI+1,kstates); % declare global state vector.
for kstate = 1:kstates % Test Multiple Simulated Sample Paths:
    if idiff == 1
      randn('state',kstate-1); % Set initial normal state
                               %    for repeatability.
      DW = sqrtdt*randn(NI,1); % Generate normal random vector
                               %    of N samples for DW(t).
    end
    if ijump == 1
      rand('state',kstate-1); % Set initial uniform state
                              %    for repeatability.
      DU = rand(NI,1); % Generate Uniform random vector of N
                       %    DP(t) samples.
      ldt = lambda*Dt; % one jump prob.
      ul = (1-ldt)/2;  ur = (1+ldt)/2; % Set centered jump
                                       %   probability thresholds,
    end
     XS(1,kstate) = x0; % Set initial state.
    % using centered part of uniform distribution
    %     to avoid open end point bias.
    YS = zeros(1,NI+1); % Set initial exponent.
    for i = 1:NI % Simulated Sample paths by Increment Accumulation:
        YS(i+1) = YS(i) + muddt;
        if idiff == 1, YS(i+1) = YS(i+1)+ sig*DW(i); end
        if ijump == 1
          if DU(i) <= ur && DU(i) >= ul % Get jump if prob. in [ul,ur]:
             YS(i+1) = YS(i+1) + lognu;
          end
        end
        XS(i+1,kstate) = x0*exp(YS(i+1));% Invert exponent to get state.
    end
end
```

```
% Compute Mean State Path and +/- One Std. Deviation:
XT(1) = x0; XB(1) = x0;
XM = zeros(1,NI+1); XM(1) = x0; % Set initial state mean.
XT = zeros(1,NI+1); XT(1) = x0; % Set initial state top spread.
XB = zeros(1,NI+1); XB(1) = x0; % Set initial state bottom spread.
muxexp = mu + lambda*nu;
if iv == 1, sigxexp = sig^2 + lambda*nu^2; end
if iv == 2, sigyexp = sig^2 + lambda*(log(1+nu))^2; end
for i = 1:NI
    XM(i+1) = x0*exp(muxexp*t(i+1));
    if iv == 1
        V = sqrt(exp(sigxexp*t(i+1)) - 1);
        XT(i+1) = XM(i+1)*(1 + V);
        XB(i+1) = XM(i+1)*(1 - V);
    end
    if iv == 2
        V = exp(sqrt(sigyexp*t(i+1)));
        XT(i+1) = XM(i+1)*V;
        XB(i+1) = XM(i+1)/V;
    end
end
% Begin Plot:
nfig = nfig + 1;
kjd = 4 - 2*idiff - ijump;
NP = N + 2;
stitle = {'Linear Jump-Diffusion Simulations'...
        ,'Linear Diffusion Simulations'...
        ,'Linear Jump Simulations'};
sylabel = {'X(t), Jump-Diffusion State','X(t), Diffusion State'...
        ,'X(t), Jump State'};
fprintf('\n\nFigure(%i):  Linear Jump-Diffusion Simulations\n',nfig)
figure(nfig)
scrsize = get(0,'ScreenSize'); % figure spacing for target screen
ss = [5.0,4.0,3.5]; % figure spacing factors
plot(t,XS(1:NP,1),'k-'...
    ,t,XS(1:NP,2),'k-'...
    ,t,XS(1:NP,3),'k-'...
    ,t,XS(1:NP,4),'k-'...
    ,t,XM(1:NP),'k--'...
    ,t,XT(1:NP),'k-.'...
    ,t,XB(1:NP),'k-.','LineWidth',2);
title(stitle(kjd)...
    ,'FontWeight','Bold','Fontsize',44);
ylabel(sylabel(kjd)...
    ,'FontWeight','Bold','Fontsize',44);
xlabel('t, Time'...
    ,'FontWeight','Bold','Fontsize',44);
if iv == 1
hlegend=legend('X(t) Sample 1','X(t) Sample 2','X(t) Sample 3'...
    ,'X(t) Sample 4','E[X](t)','(E[X]*(1+V))(t)','(E[X]*(1-V))(t)'...
```

```
    ,'Location','NorthWest');
end
if iv == 2
hlegend=legend('X(t) Sample 1','X(t) Sample 2','X(t) Sample 3'...
    ,'X(t) Sample 4','E[X](t)','(E[X]*V)(t)','(E[X]/V)(t)'...
    ,'Location','NorthWest');
end
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss(nfig) 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
%
% End JDSimulator Code
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.15 Program: Simulated Linear Mark-Jump-Diffusion Sample Paths

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function linmarkjumpdiff06fig1
% Fig. 5.1 Book Illustration for Linear Distributed-Jump Diffusion (3/2007)
%   SDE RNG Simulation with variable coefficients for t in [0,1]
%   with sample variation:
%        DX(t) = X(t)*(mu(t)*Dt + sig(t)*DW(t) + nu(Q)*DP(t),
%            X(0) = x0.
%   Or log-state:
%        DY(t) = (mu(t)-sig^2(t)/2)*Dt + sig(t)*DW(t) + Q*DP(t),
%       Y(0) = log(x0) and Q = ln(1+nu(Q)).
%   Generation is by summing Wiener increments DW of even spacing Dt
%     with Poisson jump increment added at correct time increment.
%   Sufficiently SMALL increments assumed, so zero-one jump law is
%       appropriate.
%   For demonstration purposes, Q will be assumed to be
%     (qdist =1) UNIFORMLY distributed on (qparm1,qparm2)=(a,b)
%     OR
%     (qdist=2) NORMALLY distributed with (qparm1,qparm2)=(muj,sj2).
%   Allows Separate Driver Input and Special Jump
%       or Diffusion Handling.
clc % clear variables, but must come before globals,
    %    else clears globals too.
clf % clear figures
fprintf('\nfunction linjumpdiff06fig1 OutPut:');
%%% Initialize input to jdsimulator with sample parameters:
N = 1000; t0 = 0; T = 2.0; % Set initial time grid: Fixed Delta{t}.
idiff = 1; ijump = 1; x0 = 1.0;
qdist = 1; a = -2; b = +1; qparm1 = a; qparm2 = b; %e.g., Uniform
%OR E.G., Normal distribution:
%qdist = 2; muj = 0.28; sj2 = +0.15; qparm1 = muj; qparm2 = sj2;
```

```
% set constant parameters.
fprintf('\n  N=%i; x0=%6.3f; t0=%6.3f; T=%6.3f;',N,x0,t0,T);
fprintf('\n  qdist=%i*; qparm1=%6.3f; qparm2=%6.3f;'...
    ,qdist,qparm1,qparm2);
fprintf('\n  * qdist=1 for uniform Q-distribution.');
fprintf('\n  * qdist=2 for normal Q-distribution.');
%
jdsimulator(idiff,ijump,qdist,qparm1,qparm2,N,x0,t0,T);
%
% END INPUT FOR JUMP-DIFFUSION SIMULATOR.
%%%
```

bk0alff
2007/8/10
page C22


```
        Q = qparm1+(qparm2-qparm1)*rand(1,N);
    elseif qdist == 2 %Generate Normal random mark vector Q samples.
     sj = sqrt(qparm2); Q = qparm1+sj*randn(1,N);
    end
    ul = (1-ldtv)/2;  ur = 1-ul; % Set vector centered jump
                                 %   probability thresholds,
    end
    YS = zeros(1,N+1); % preallocate state exponent for efficiency.
    XS(1,k) = x0; % Set kth initial state.
    for i = 1:N % Simulated Sample paths by Increment Accumulation:
        YS(i+1) = YS(i) + muddt(i); % Add dY-drift:
        % Add diffusion increment:
        if idiff == 1, YS(i+1) = YS(i+1)+ sv(i)*DW(i); end
        % Using centered part of uniform distribution, with
        %     acceptance-rejection, to avoid open end point bias:
        if ijump == 1
            if DU(i) <= ur(i) && DU(i) >= ul(i) % Jump if in [ul,ur]
               YS(i+1) = YS(i+1) + Q(i); % If jump, +jump amplitude.
            end % Else no jump, so do not add anything.
        end
        XS(i+1,k) = x0*exp(YS(i+1));% Invert exponent to get state.
    end % i
end % k
% Sample Mean State:
XSM = zeros(1,NP);
for i = 1:NP
    XSM(i) = mean(XS(i,:));
end
% Begin Plot:
scrsize = get(0,'ScreenSize');
ss = 5.2; dss = 0.2; ssmin = 3.0;
kfig = kfig + 1;
stitle = {'Linear Mark-Jump-Diffusion Simulations'...
        ,'Linear Diffusion Simulations'...
        ,'Linear Mark-Jump Simulations'};
sylabel = {'X(t), Jump-Diffusion State','X(t), Diffusion State'...
        ,'X(t), Jump State'};
slegend = {'X(t), State 1','X(t), State 5'...
        ,'X(t), State 9','X(t), State 10'...
        ,'XM(t), th. Mean=E[X(t)]','XSM(t), Sample Mean'};
fprintf('\n\nFigure(%i):  Linear Jump-Diffusion Simulations\n',kfig)
figure(kfig)
plot(tv,XS(1:NP,1),'k+-'...
    ,tv,XS(1:NP,2),'k:'...
    ,tv,XS(1:NP,3),'k--'...
    ,tv,XS(1:NP,4),'k-.'...
    ,tv,XM(1:NP),'k-',tv,XSM(1:NP),'k.-'...
    ,'LineWidth',2); % Add for more States?
title(stitle(kjd),'FontWeight','Bold','Fontsize',44);
ylabel(sylabel(kjd),'FontWeight','Bold','Fontsize',44);
```

```
xlabel('t, Time','FontWeight','Bold','Fontsize',44);
hlegend=legend(slegend,'Location','NorthEast');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
ss = max(ss - dss,ssmin);
set(gcf,'Color','White','Position'...
   ,[scrsize(3)/ss 60 scrsize(3)*0.60 scrsize(4)*0.80]);
%[l,b,w,h]
%
% End JDSimulator Code
%
% linear Time-Dependent SDE Coefficient Functions:
%   (Change with application; functions must be vectorizable,
%       using vector element dot operations or vector functions.)
%
function v = mu(t)
% drift coefficient example, change with applications:
v = 0.1*sin(t);
% end mu(t)
%
function v = sigma(t)
% drift coefficient example, change with applications:
v = 1.5*exp(-0.01*t);
% end sigma(t)
%
function v = lambda(t)
% drift coefficient example, change with applications:
v = 3.0*exp(-t.*t);
% end lambda(t)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.16 Program: Curse of Dimensionality

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function curseofdim
%
%  Fig. 8.1 Bellman's Curse of dimensionality illustrated (3/2007):
%
clc
%
nxmax = 10;
nx = 1:nxmax; % dimensions
Nx = 32; % nodes per dimension
NDV = nx.*Nx.^nx; % Order of typical size array (diagonal case);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
```

bk0alfi
2007/8/10
page C24


```
nfig = 1;
figure(nfig);
%
plot(nx,NDV,'k-'...
    ,'linewidth',5);
axis tight
title('Curse of Dimensionality: n_x exp(n_x ln(N_x)) for N_x=32'...
        ,'Fontsize',44,'FontWeight','Bold');
xlabel('n_x, State Dimensions'...
        ,'Fontsize',44,'FontWeight','Bold');
ylabel('NDV, Problem Size'...
        ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Nxmax = 64;
Nx3 = 1:Nxmax; % nodes per dimension
ln2Nx3 = log2(Nx3); %log_2(Nodes)
BPGB8 = 8/1024^4;
NDV3 = zeros(nxmax,Nxmax);
for i=1:nxmax
    for j=1:Nxmax
        %Order of typical size array (diagonal case):
        NDV3(i,j) = nx(i)*Nx3(j)^nx(i)*BPGB8;
    end
end
%
nfig = nfig+1;
figure(nfig);
% Caution: mesh takes reverse arguments for y & x in z:
mesh(ln2Nx3,nx,NDV3,'linewidth',2)
colormap(gray);
axis tight
title('Curse of Dimensionality: N_{DV} = 8n_x exp(n_x ln(N_x)) in GB'...
        ,'Fontsize',28,'FontWeight','Bold');
ylabel('  n_x, State Dimensions'...
        ,'Fontsize',28,'FontWeight','Bold');
xlabel('log_2(N_x), log_2(Nodes)'...
        ,'Fontsize',28,'FontWeight','Bold');
zlabel('N_{DV}, Problem Size Order (GB)'...
        ,'Fontsize',28,'FontWeight','Bold');
set(gca,'Fontsize',28,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
log2NDV3 = zeros(nxmax,Nxmax);
for i=1:nxmax
```

```
    for j=1:Nxmax
        %Order of typical size array (diagonal case):
        log2NDV3(i,j) = log2(8*nx(i)*Nx3(j)^nx(i));
    end
end
%
nfig = nfig+1;
figure(nfig);
% Caution: mesh takes reverse arguments for y & x in z:
mesh(ln2Nx3,nx,log2NDV3,'linewidth',2)
colormap(gray);
axis tight
title('Curse of Dimensionality: N_{DV} = log_2(8*n_x exp(n_x ln(N_x)))'...
        ,'Fontsize',28,'FontWeight','Bold');
ylabel('  n_x, State Dimensions'...
        ,'Fontsize',28,'FontWeight','Bold');
xlabel('log_2(N_x), log_2(Nodes)'...
        ,'Fontsize',28,'FontWeight','Bold');
zlabel('log_2(N_{DV}), Log2(ProblemSize)'...
        ,'Fontsize',28,'FontWeight','Bold');
set(gca,'Fontsize',28,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position'...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
% End curseofdim
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.17 Program: Euler–Maruyama Simulations for Linear Diffusion SDE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sdeeulersim
% Figs. 9.2, 9.3 Euler-Maruyama Simulations (3/2007):
%   Linear, Time-Dep. Coeff. SDE,
%   dX(t) = X(t)(mu(t)dt+sigma(t)dW(t)), X(0) = x0, t0 < t < tf,
%   Given  Initial data: x0, t0, tf, Nt; functions: f, g
clc
x0 = 1; t0 = 0; tf = 5; Nt = 2^10;
randn('state',8);
DT = tf/Nt; sqrtdt = sqrt(DT);
t = t0:DT:tf;
DW = randn(1,Nt)*sqrtdt;
W = cumsum(DW); % Note: omits initial zero value; count is off by 1;
%
Xexact = zeros(1,Nt+1); Xexact(1) = x0; % initialize
for k = 1:Nt % Exact formula to fine precision for exact consistency:
    Xexact(k+1) = xexact(x0,t(k+1),W(k));
end
% Lumped coarse sample from fine sample:
```

bk0alfi
2007/8/10
page C26


```
L = 2^3; NL = Nt/L; KL = 0:L:Nt; DTL = L*DT; tL = t0:DTL:tf;
fprintf('(N_t,NL)=(%i,%i); Size(t,KL,tL)=[(%i,%i);(%i,%i);(%i,%i)];'...
    ,Nt,NL,size(t),size(KL),size(tL));
Xeul = zeros(1,NL+1); Xeul(1) = x0;
Xdiff = zeros(1,NL+1); Xdiff(1) = Xeul(1) - Xexact(1);
for k = 1:NL % Euler-Maruyama formula to coarse precision:
    DWL = sum(DW(1,KL(k)+1:KL(k+1)));
    Xeul(k+1) = Xeul(k) + f(Xeul(k),tL(k))*DTL+g(Xeul(k),tL(k))*DWL;
    Xdiff(k+1) = Xeul(k+1) - Xexact(KL(k+1));
end
%
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%
nfig = 1;
figure(nfig);
plot(tL,Xeul,'k--','linewidth',3); hold on
plot(t,Xexact,'k-','linewidth',3); hold off
axis([t0 tf 0 max(max(Xeul),max(Xexact))]);
title('Euler-Maruyama and Exact Linear SDE Simulations'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('t, Time'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('X(t), State'...
    ,'Fontsize',44,'FontWeight','Bold');
legend('Xeul(t): Euler','Xexact(t): Exact','Location','NorthWest');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
Xdiffmax = max(abs(Xdiff));
fprintf('\nMaximal Euler-Exact Absolute Error:');
fprintf('\n     max(abs(Xeul(TL)-Xexact(TL)))=%8.2e=%8.2e*DTL;\n'...
    ,Xdiffmax,Xdiffmax/DTL);
% (N_t,NL) = (1024,128); Size(t,KL,tL) = [(1,1025);(1,129);(1,129)];
% Maximal Euler-Exact Abs. Error:
%      max(abs(Xeul(TL)-Xexact(TL))) = 1.31e+00 = 3.36e+01*DTL;
%
nfig = nfig+1;
figure(nfig);
plot(tL,Xdiff,'k-','linewidth',3);
axis tight;
title('Euler and Exact Linear SDE Simulations Error'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('t, Time'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Xeul(t)-Xexact(t), Error'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
```

```
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = f(x,t)
mu = 1/(1+0.5*t)^2;
y = mu*x;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = g(x,t)  % for general case, ignore "t not used" warning.
sig = 0.5;
y = sig*x;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = xexact(x0,t,w)
% exact solution if available for general linear SDE:
mubar = 2-2/(1+0.5*t); sig = 0.5; sig2bar = sig^2*t/2;
y = x0*exp(mubar-sig2bar + sig*w);
%
% end sdeeulersim.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.18 Program: Milstein Simulations for Linear Diffusion SDE

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function sdemilsteinsim
% Figs. 9.4, 9.5, 9.6 Milstein SDE Simulations (3/2007):
%   Linear, Time-Dep. Coeff. SDE
%   dX(t) = X(t)(mu(t)dt+sigma(t)dW(t)), X(0) = x0, t0 < t < tf,
% Given  Initial data: x0, t0, tf, Nt; functions: f, g
clc
x0 = 1; t0 = 0; tf = 5; Nt = 2^12;
randn('state',8);
DT = tf/Nt; sqrtdt = sqrt(DT);
t = t0:DT:tf;
DW = randn(1,Nt)*sqrtdt;
W = cumsum(DW); % Note: omits initial zero value; count if off by 1;
Xexact = zeros(1,Nt+1); Xexact(1) = x0; % initialize
for k = 1:Nt % Exact formula to fine precision for exact consistency:
    Xexact(k+1) = xexact(x0,t(k+1),W(k));
end
% Lumped coarse sample from fine sample:
L = 2^3;
NL = Nt/L; KL = 0:L:Nt; DTL = L*DT; tL = t0:DTL:tf;
fprintf('(N_t,NL)=(%i,%i); Size(t,KL,tL)=[(%i,%i);(%i,%i);(%i,%i)];'...
    ,Nt,NL,size(t),size(KL),size(tL));
Xmil = zeros(1,NL+1); Xmil(1) = x0;
Xeul = zeros(1,NL+1); Xeul(1) = x0;
```

bk0alff
2007/8/10
page C28


```
Xdiff = zeros(1,NL+1); Xdiff(1) = Xmil(1) - Xexact(1);
Xmileul = zeros(1,NL+1); Xmileul(1) = Xmil(1) - Xeul(1);
for k = 1:NL % Milstein and Euler formulas to coarse precision:
    DWL = sum(DW(1,KL(k)+1:KL(k+1)));
    Xmil(k+1)=Xmil(k)+f(Xmil(k),tL(k))*DTL+g(Xmil(k),tL(k))*DWL...
        +0.5*g(Xmil(k),tL(k))*gx(Xmil(k),tL(k))*(DWL^2-DTL);
    Xeul(k+1)=Xeul(k)+f(Xeul(k),tL(k))*DTL+g(Xeul(k),tL(k))*DWL;
    Xdiff(k+1) = Xmil(k+1) - Xexact(KL(k+1));
    Xmileul(k+1) = Xmil(k+1) - Xeul(k+1);
end
%
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%
nfig = 1;
figure(nfig);
plot(tL,Xmil,'k--','linewidth',3); hold on
% plot(tL,Xeul,'k:','linewidth',3); hold on
plot(t,Xexact,'k-','linewidth',3); hold off
axis([t0 tf 0 max(max(max(Xmil),max(Xeul)),max(Xexact))]);
title('Milstein and Exact Linear SDE Simulations'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('t, Time'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('X(t), State'...
    ,'Fontsize',44,'FontWeight','Bold');
hlegend = legend('Xmil(t): Milstein','Xexact: Exact'...
    ,'Location','Best');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
Xdiffmax = max(abs(Xdiff));
fprintf('\nMaximal Milstein-Exact Absolute Error:');
fprintf('\n     max(abs(Xmil(TL)-Xexact(TL)))=%8.2e=%8.2e*DTL;\n'...
    ,Xdiffmax,Xdiffmax/DTL);
% (N_t,NL) = (1024,128); Size(t,KL,tL) = [(1,1025);(1,129);(1,129)];
% Maximal Milstein-Exact Absolute Error:
%      max(abs(Xmil(TL)-Xexact(TL))) = 1.23e+00 = 3.16e+01*DTL;
% Maximal Milstein-Euler Absolute Error:
%     max(abs(Xmil(TL)-Xeul(TL))) = 9.54e-01 = 2.44e+01*DTL;
%
nfig=nfig+1;
figure(nfig);
plot(tL,Xdiff,'k-','linewidth',3);
axis tight;
title('Milstein and Exact SDE Simulations Error'...
```

```
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('t, Time'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Xmil(t)-Xexact(t), Error'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
Xmileulmax = max(abs(Xmileul));
fprintf('\nMaximal Milstein-Euler Absolute Error:');
fprintf('\n     max(abs(Xmil(TL)-Xeul(TL))) = %8.2e = %8.2e*DTL;\n'...
    ,Xmileulmax,Xmileulmax/DTL);
%
nfig=nfig+1;
figure(nfig);
plot(tL,Xmileul,'k-','linewidth',3);
axis tight;
title('Milstein and Euler SDE Simulations Difference'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('t, Time'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Xmil(t)-Xeul(t), Difference'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = f(x,t)
mu = 1/(1+0.5*t)^2;
y = mu*x;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = g(x,t) % for general case, ignore "t not used" warning.
sig = 0.5;
y = sig*x;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = gx(x,t) % for general case, ignore "x,t not used" warning.
sig = 0.5;
y = sig;
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function y = xexact(x0,t,w)
% exact solution if available for general linear SDE:
mubar = 2-2/(1+0.5*t); sig = 0.5; sig2bar = sig^2*t/2;
y = x0*exp(mubar-sig2bar + sig*w);
```

```
%
% end sdemilsteinsim.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.19 Program: Monte Carlo Simulation Comparing Uniform and Normal Errors

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mcm0unifnorm
% Example Output:  Book Chapter 9 Sims, Section Monte Carlo (3/2007);
%   compare error for uniform and normal on
%   int(exp(-x^2/2)/sqrt(2*pi),x=-R..R)
clc
clear
%
global R V
%
fprintf('Compare Uniform and Normal Monte Carlos:\n');
n = 100; srtn = sqrt(n);
R=5; V = 2*R;
fprintf('\nn=%i; R=%6.4f; V=%6.4f;\n',n,R,V);
% erfc(x) = 2\sqrt(pi)*int(exp(-t^2),t=x..infty);
% normcdf(x)=0.5*erfc(-x/sqrt(2));
exact = 0.5*(erfc(-R/sqrt(2))-erfc(+R/sqrt(2)));
sig2uexact = 2.5/sqrt(pi)*(erf(R)-erf(-R))-exact^2;
sig2nexact = exact*(1-exact);
fprintf('\nexact integral = %10.8f;',exact);
fprintf('\nsig2unifexact = %9.4e; sigunifexact = %9.4e;'...
    ,sig2uexact,sqrt(sig2uexact));
fprintf('\nsig2normexact = %9.4e; signormexact = %9.4e;\n'...
    ,sig2nexact,sqrt(sig2nexact));
U = -R+V*rand(1,n);
X = randn(1,n);
fuv = zeros(1,n); fnv = zeros(1,n);
for i=1:n
    fuv(i)=fu(U(i));
    fnv(i)=fn(X(i));
end
% Monte Carlo estimators:
sun = mean(fuv);
snn = mean(fnv);
fprintf('\nsunifn=%10.8f; snormn=%10.8f;',sun,snn);
fprintf('\nsunifnabserror=%9.4e%%; snormnabserror=%9.4e%%;'...
    ,sun-exact,snn-exact-1);
fprintf('\nsunifnrelerror=%9.4e%%; snormnrelerror=%9.4e%%;\n'...
    ,100*(sun/exact-1),100*(snn/exact-1));
% Monte Carlo variance estimators:
sig2un = var(fuv);  % MATLAB var(x); gives unbiased variance
sig2nn = var(fnv);
```

```
fprintf('\nsig2unifn=%9.4e; sig2normn=%9.4e;',sig2un,sig2nn);
fprintf('\nsig2unifnabserror=%9.4e%%; sig2normnabserror=%9.4e%%;'...
    ,sig2un-sig2uexact,sig2nn-sig2nexact);
fprintf('\nsig2unifnrelerror=%9.4e%%; sig2normnrelerror=%9.4e%%;\n'...
    ,100*(sig2un/sig2uexact-1),100*(sig2nn/sig2nexact-1));
% std. errors:
seunifexact = sqrt(sig2uexact)/srtn;
senormexact = sqrt(sig2nexact)/srtn;
seunifn = sqrt(sig2un)/srtn;
senormn = sqrt(sig2nn)/srtn;
fprintf('\nstderrunifexact=%9.4e; stderrnormexact=%9.4e;'...
    ,sqrt(sig2uexact)/srtn,sqrt(sig2nexact)/srtn);
fprintf('\nstderrunifn=%9.4e; stderrnormn=%9.4e;'...
    ,sqrt(sig2un)/srtn,sqrt(sig2nn)/srtn);
fprintf('\nstderrunifndiff=%9.4e; stderrnormndiff=%9.4e;\n'...
    ,seunifn-seunifexact,senormn-senormexact);
%
%%%%%
function y = fu(x)
global V
y = V*exp(-x.*x/2)/sqrt(2*pi);
%%%
function y = fn(x)
global R
y = 1;
if abs(x)>R, y=0; end
%%%%%
%
% end mcm0unifnorm.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.20 Program: Monte Carlo Simulation Testing Uniform Distribution

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mcm1test
% Figs. 9.8a, 9.8b mcm1test:  Monte Carlo Method (3/2007),
%     nx = 1 dim, uniform dist,
%     I = int(F(x),x=a..b), F(x) = sqrt(1-x^2), -1 <= a < b <= +1;
%     technically, f(x) = (b-a)F(x) = (b-a)*sqrt(1-x^2) to account for
%     uniform density phi(x) = 1/(b-a) on [a,b], so I = meanf.
%
clc; clear
%
fprintf('Monte Carlo Test of 1-dim Uniform Dist. on (a,b)');
fprintf('\n   with F(x)=sqrt(1-x^2) and f(x) = (b-a)F(x):\n');
a = 0; b = +1; % -1 <= a < b <= +1;
% integral of f(x) = sqrt(1-x^2); on [a,b]:
```

bk0alfi
2007/8/10
page C32


```
IntExact = 0.5*(asin(b)-asin(a))+0.5*(b*sqrt(1-b^2)-a*sqrt(1-a^2));
MufExact = IntExact;
Sigf = sqrt((b-a)^2*(1-(b^2+a*b+a^2)/3)-MufExact^2);
fprintf('\nk        n muhatn mufExact   sighatn     Sigf    stderrn');
fprintf('  AbsErrorf\n');
kmax = 7;
n = zeros(1,kmax); meanf = zeros(1,kmax); sigf = zeros(1,kmax);
sigdrn = zeros(1,kmax); error = zeros(1,kmax);
for k = 1:kmax
   rand('state',0); % set state or seed
   n(k) = 10^k; % sample size, k = log10(n(k)) ;
   x = a+(b-a)*rand(n(k),1); % get n(k) X 1 random sample on (a,b);
   f = (b-a)*sqrt(1-x.^2); % vectorized f;
   meanf(k) = mean(f); % E[f(X)];
   sigf(k) = std(f); % sqrt(sigmaf^2), sigmaf^2 = unbiased variance of f;
   sigdrn(k) = sigf(k)/sqrt(n(k));
   error(k) = abs(meanf(k)-MufExact);
   fprintf('%1i %8i %6.4f   %6.4f %9.3e %9.3e  %9.3e %9.3e\n'...
       ,k,n(k),meanf(k),MufExact,sigf(k),Sigf,sigdrn(k),error(k))
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%
nfig = 1;
figure(nfig);
kv = 1:kmax;
plot(kv,meanf,'k-o','linewidth',3,'MarkerSize',12);
hold on
plot(kv,sigf,'k-x','linewidth',3,'MarkerSize',12);
axis([min(kv) max(kv) 0 1]);
title('Monte Carlo Results, Uniform Dist., F(x) = sqrt(1-x^2)'...
    ,'Fontsize',36,'FontWeight','Bold');
xlabel('log(n), Log_{10} Sample Size'...
    ,'Fontsize',36,'FontWeight','Bold');
ylabel('f-Moments \mu_n, \sigma_n'...
    ,'Fontsize',36,'FontWeight','Bold');
legend('\mu_n, Mean-est.','\sigma_n, StdDev-est.','Location','Best');
set(gca,'Fontsize',32,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
hold off
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
nfig = nfig+1;
figure(nfig);
kv = 1:kmax;
plot(kv,log10(sigdrn),'k-o','linewidth',3,'MarkerSize',12);
hold on
```

```
plot(kv,log10(error),'k-x','linewidth',3,'MarkerSize',12);
% ymin = min(min(log10(sigdrn)),min(log10(error)));
% ymax = max(max(log10(sigdrn)),max(log10(error)));
axis tight; %axis([min(kv) max(kv) ymin ymax]);
title('Monte Carlo Errors, Uniform Dist., F(x) = sqrt(1-x^2)'...
    ,'Fontsize',36,'FontWeight','Bold');
xlabel('log(n), Log_{10} Sample Size'...
    ,'Fontsize',36,'FontWeight','Bold');
ylabel('f-Errors log(StdError_n), log(AbsError_n)'...
    ,'Fontsize',36,'FontWeight','Bold');
legend('log_{10}(StdError_n)','log_{10}(AbsError_n)','Location','Best');
set(gca,'Fontsize',32,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
hold off
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% end mcm1test.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.21 Program: Monte Carlo Acceptance-Rejection Technique

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mcm2acceptreject
% Fig. 9.10 mcm2acceptreject: Monte Carlo Method (3/2007),
%     nx = 1 dim, normal dist.,
%     I = int(F(x),x=a..b), F(x) = exp(-x^2/2)/sqrt(2pi),
%           -1 <= a < b <= +1;
%     f(x) = I_{x in [a,b]} to account for truncated integral,
%     so I = meanf.
%
clc; clear
%
fprintf('Monte Carlo and Finite Difference Comparison:');
fprintf('\n   including Acceptance-Rejection Technique Application,');
fprintf('\n   with Normal Dist. on (a,b)');
fprintf('\n   and with int(F(x),x=a..b), F(x) = exp(-x^2/2)/sqrt(2pi);\n');
%
a = -2; b = 2; % limits of integration;
nfd = 100; % number of finite difference steps;
kmax = 7;
nmc = 10^kmax; % select Monte Carlo random sample size;
F = inline ('exp(-x.*x/2)./sqrt(2*pi)','x');  % x in [a,b]
%   Thus, relative to the normal density, f(x)={1, x in [a,b]; 0, else};
h = (b - a)/nfd; % step size;
% Trapezoid Rule (see also MATLAB trapz(x,y) built-in function):
trap = (F(a)+F(b))/2;
```

```
for i = 1:nfd-1,
    trap = trap+F(a+i*h);
end
trap = h*trap;
fprintf('\n%3i-point Trapezoidal Rule: I(-1,1) = %.6f\n',nfd+1,trap);
% Simpson's (1/3) Rule:
simp = F(a)+F(b);
for i = 1:nfd-1
    if mod(i,2)
        simp =simp+ 4*F(a + i*h);
    else
        simp=simp+2*F(a + i*h);
    end
end
simp = h*simp/3;
fprintf('\n%3i-point Simpson''s rule: I(-1,1) = %.6f\n',nfd+1,simp);
% MATLAB quad built-in function (adaptive Simpson's rule,
%                   default 1.e-6 accuracy):
tol = 1.e-9;
quadfn = quad(F,a,b,tol);
fprintf('\n%7.1e-accurate quad: = %.6f\n',tol,quadfn);
% Direct von Neumann Acceptance-Rejection Technique:
fprintf('\nMonte Carlo results by von Neumann''s Acceptance-Rejection technique:\n');
fprintf('\n k          n    muhatn    stderrn\n');
nac = 0;
x = randn(nmc,1); % MATLAB nmc X 1 normal distribution;
for n = 1:nmc
    if (x(n) >= a) & (x(n) <= b)
        nac = nac + 1; % counts accepted points;
    end
    if (n==10)|(n==100)|(n==1000)|(n==10000)|(n==100000)|(n==1000000)|(n==nmc)
        k = log10(n);
        kv(k) = k;
        muhatn(k) = nac/n;
        stderrn(k) = sqrt(muhatn(k)*(1-muhatn(k))/(n-1));
        fprintf ('%2i   %8i  %8.6f  %9.3e\n',k,n,muhatn(k),stderrn(k));
    end
end
fprintf('\n 101-pt. trap: %8.6f  %9.3e*',trap,abs(trap-quadfn));
fprintf('\n 101-pt. simp: %8.6f  %9.3e*',simp,abs(simp-quadfn));
fprintf('\n    accurate:  %8.6f  %9.3e*',quadfn,abs(quadfn-quadfn));
fprintf('\n * Absolute Errors\n');
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%
nfig = 1;
figure(nfig);
kv = [1:kmax];
```

```
plot(kv,muhatn,'k-o','linewidth',3,'MarkerSize',12); hold on
plot(kv,stderrn*10,'k-x','linewidth',3,'MarkerSize',12); hold off
axis([min(kv) max(kv) 0 1]);
title('Monte Carlo Results, Normal Dist., F(x) = \phi_n(x) on [a,b]'...
    ,'Fontsize',36,'FontName','Helvetica','FontWeight','Bold');
xlabel('log(n), Log_{10} Sample Size'...
    ,'Fontsize',32,'FontName','Helvetica','FontWeight','Bold');
ylabel('Moments \mu_n, 10*std-err_n'...
    ,'Fontsize',32,'FontName','Helvetica','FontWeight','Bold');
legend('\mu_n, Mean-est.','10*std-err_n','Location','Best');
set(gca,'Fontsize',28,'FontName','Helvetica','FontWeight'...
    ,'Bold','linewidth',3);
set(gcf,'Color','White','Position'...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% end mcm2acceptreject.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.22 Program: Monte Carlo Multidimensional Integration

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function mcm3multidim
% Fig. 9.11 mcm3multidim: Monte Carlo Multidimensional Integration,
%     nx = 2:5 dims, normal distribution,
%     I = int(F(x),x=a..b), F(x) = exp(-sum(x.^2)/2)/sqrt(2*pi)^nx,
%     so f(x) = I_{a<= x <= b}, indicator function using vector
%     inequalities  (3/2007).
%
clc; clear
%
fprintf('Monte Carlo Multidimenstional Integration:');
fprintf('\n   including Acceptance-Rejection Technique Application,');
fprintf('\n   with Normal Dist. on (a,b)');
fprintf('\n   and with int(F(x),x=a..b), F(x) = exp(-x.^2/2)');
fprintf('/sqrt(2pi)^nx;\n');
%
nxmax = 5; % dimension
kmax = 6; % power of 10
% f = inline ('exp(-sum(x.*x)/2) / sqrt(2*pi)^length(x)','x');
muhatn = zeros(kmax,nxmax); stderrn = zeros(kmax,nxmax);
for nx = 2:nxmax
    a = -2*ones(1,nx); % lower vector limit
    b = 2*ones(1,nx); % upper vector limit
    nmc = zeros(1,kmax);
    for k = 1:kmax
        nmc(k) = 10^k; % sample size
```

bk0alfi
2007/8/10
page C36


```
        nac = 0;
        for n = 1:nmc(k)
            x = randn(1,nx); % 1Xnmc vector normal distribution;
            if (x >= a) & (x <= b) % accept-reject technique
                nac = nac + 1; % counts accepted points;
            end
        end
        muhatn(k,nx) = nac/nmc(k);
        stderrn(k,nx) = sqrt(muhatn(k,nx)*(1-muhatn(k,nx))/(nmc(k)-1));
    end
end
%
fprintf('\nMonte Carlo results in mutlidimension,');
fprintf('\n    by von Neumann''s Acceptance-Rejection technique:\n');
fprintf('\nMonte Carlo Mean Estmate, muhatn:');
fprintf('\n k        n      nx=2      nx=3      nx=4      nx=5\n');
for k = 1:kmax
    fprintf ('%2i %8i  %8.6f  %8.6f  %8.6f  %8.6f\n'...
        ,k,nmc(k),muhatn(k,2:nxmax));
end
fprintf('\nMonte Carlo Std Error Estmate, stderrn:');
fprintf('\n k        n       nx=2       nx=3       nx=4       nx=5\n');
for k = 1:kmax
    fprintf ('%2i %8i  %9.3e  %9.3e  %9.3e  %9.3e\n'...
        ,k,nmc(k),stderrn(k,2:nxmax));
end
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%
nfig = 1;
figure(nfig);
kv = 1:kmax;
plot(kv,muhatn(:,2),'k-o'...
    ,kv,muhatn(:,3),'k-x'...
    ,kv,muhatn(:,4),'k-+'...
    ,kv,muhatn(:,5),'k-*'...
    ,'linewidth',3,'MarkerSize',14);
axis([min(kv) max(kv) 0.5 1]);
title('Monte Carlo Means, Normal Dist., F(x) = \phi_n(x) on [a,b]'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('log(n), Log_{10} Sample Size'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Mean Estimates, \mu_n'...
    ,'Fontsize',44,'FontWeight','Bold');
hlegend=legend('nx = 2','nx = 3','nx = 4','nx = 5','Location','Best');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
```

```
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
nfig = nfig+1;
figure(nfig);
kv = 1:kmax;
plot(kv,stderrn(:,2),'k-o'...
    ,kv,stderrn(:,3),'k-x'...
    ,kv,stderrn(:,4),'k-+'...
    ,kv,stderrn(:,5),'k-*'...
    ,'linewidth',3,'MarkerSize',14);
axis tight; % axis([min(kv) max(kv) 0 0.2]);
title('Monte Carlo Std. Errors, Normal Dist. on [a,b]'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('log(n), Log_{10} Sample Size'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Std. Errors, stderr_n'...
    ,'Fontsize',44,'FontWeight','Bold');
hlegend=legend('nx = 2','nx = 3','nx = 4','nx = 5','Location','Best');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%
% end mcm3multidim.m
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.23 Program: Regular and Bang Control Examples

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function RegCtrlExample6p1a
% Figs. A.1a, A.1b, A.2a, A.2b Book Illustration
%     for Regular Control & Bang Control Examples:
%     Bang Control is Also part of Exercise 3
% Linear State Dynamics, Power Utility.
%
clc % clear variables, but must come before globals,
    %    else clears globals too.
clf % clear figures
fprintf('\nExample 6.1: Regular & Bang Control OutPuts:');
%
global gamma mu0 x0 tf
%
nfig = 0;
gammav = [0.5,2.0]; % i.e., [Regular,Bang] examples
ng = 2;
mu0 = 0.10;
```

```
x0 = 10;
t0 = 0; tf = 1.0;
dt = 0.01;
t = t0:dt:tf;
umin = 0; umax = 10; du = (umax-umin)/100;
uv = umin:du:umax;
fprintf('\n(t0,tf)=(%f,%f); x0=%f; mu0=%f;\n(umin,umax)=(%f,%f); du=%f;'...
    ,t0,tf,x0,mu0,umin,umax,du);
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%
for i = 1:ng
    gamma = gammav(i);
    if i == 1
        K00 = 10; K0 = K00;
        K01 = 1.0;
        K02 = 20;
fprintf('\n GK(%f)=%e; GK(%f)=%e;',K01,GK(K01),K02,GK(K02));
        [Z,Gval,exitflag] = fzero(@GK,[K01,K02]);
fprintf('\ngamma=%f; Z=%2e;  Gval=%e; exitflag=%e;'...
    ,gamma,Z,Gval,exitflag);
        if isnan(Z) == 0
            K0 = Z;
        end
        ureg = K0; % us = ureg;
        xreg = X(t,ureg);
        xf = X(tf,ureg);
        lamf = xf^(gamma-1);
        lamreg = LAM(t,ureg,lamf);
        Sf = S(xf,tf);
fprintf('\ngamma=%f; Ureg=%f; xf =%f; lamf=%f; Sf=%f;' ...
    ,gamma,K0,xf,lamf,Sf);
        nfig = nfig + 1;
        fprintf('\n\nFigure(%i):  Regular Control Example\n',nfig)
        figure(nfig)
        plot(t,xreg,'k-',t,lamreg,'k--','LineWidth',4);
        title('Regular Control Maximum Example (a)'...
            ,'Fontsize',44,'FontWeight','Bold');
        ylabel('X^*(t) and \lambda^*(t), Optimal States'...
            ,'Fontsize',44,'FontWeight','Bold');
        xlabel('t, time'...
            ,'Fontsize',44,'FontWeight','Bold');
        hlegend=legend('X^*(t) Optimal Wealth State' ...
        ,'\lambda^*(t) Optimal Co-State','Location','Best');
set(hlegend,'Fontsize',36,'FontName','Helvetica','FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
nfig = nfig + 1;
```

```
    K = (x0*exp((mu0-ureg)*tf))^gamma;
    hv = Hc(uv,K);
fprintf('\n\nFigure(%i): Hamiltonian Regular Example\n',nfig)
figure(nfig)
plot(uv,hv,'k-','LineWidth',4);
title('Hamiltonian Regular Example'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Hc(u), Hamiltonian'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('u, Control'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
    end
%%%%%%%%%%%%%
    if i == 2
        K = (x0*exp((mu0-umin)*tf))^gamma;
        Hmin = Hc(umin,K);
        K = (x0*exp((mu0-umax)*tf))^gamma;
        Hmax = Hc(umax,K);
        ubang = umin; % Hbang = umax;
        if Hmax > Hmin, ubang = umax; end
        xbang = X(t,ubang); % us = ubang;
        xf = X(tf,ubang);
        lamf = xf^(gamma-1);
        lambang = LAM(t,ubang,lamf);
fprintf('\ngamma=%f; ubang=%f; xf=%e; lamf=%e;'...
    ,gamma,ubang,xf,lamf);
fprintf('\ngamma=%f; ubang=%f; Hmin=%e; Hmax=%e;'...
    ,gamma,ubang,Hmin,Hmax);
        nfig = nfig + 1;
        fprintf('\n\nFigure(%i): Bang Control Example\n',nfig)
        figure(nfig)
        plot(t,xbang,'k-',t,lambang,'k--','LineWidth',4);
        title('Bang Control Maximum Example (b)'...
            ,'Fontsize',44,'FontWeight','Bold');
        ylabel('X^{bang}(t) and \lambda^{bang}(t), Bang States'...
            ,'Fontsize',44,'FontWeight','Bold');
        xlabel('t, time'...
            ,'Fontsize',44,'FontWeight','Bold');
        hlegend=legend('X^{bang}(t) Bang Wealth State' ...
            ,'\lambda^{bang}(t) Bang Co-State','Location','Best');
set(hlegend,'Fontsize',36,'FontName','Helvetica','FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%%%
nfig = nfig + 1;
    K = (x0*exp((mu0-ubang)*tf))^gamma;
```

```
    hv = Hc(uv,K);
fprintf('\n\nFigure(%i): Hamiltonian Bang Example\n',nfig)
figure(nfig)
plot(uv,hv,'k-','LineWidth',4);
title('Hamiltonian Bang Example'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('Hc(u), Hamiltonian'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('u, Control'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
    end
%
end
%
% End Optimal Example Code
%
function cv = C(u)
global gamma
cv = u.^gamma/gamma;
%
function sv = S(x,t) % general form, ignore t not used warning
global gamma
sv = x.^gamma/gamma;
%
function gv = GK(K0)
global gamma mu0 x0 tf
gv = K0^(gamma-1)-(x0*exp((mu0-K0)*tf))^gamma;
%
function xv = X(t,K0)
global mu0 x0
xv = x0*exp((mu0-K0)*t);
%
function lamv = LAM(t,K0,lamf)
global mu0 tf
lamv = lamf*exp(-(mu0-K0)*(t-tf));
%
function hu = Hc(u,K) % Hamiltonian for fixed or bang control u.
global mu0
hu = C(u) + K*(mu0-u);
%
% End RegCtrlExample6p1a
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.24 Program: Simple Optimal Control Example

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function SimpleOptExample
% Figs. A.3a, A.3b Book Illustration for Simple Opt Principle Example:
% Static, Quadratic Control (3/2007).
%
clc % clear variables, but must come before globals,
    %    else clears globals too.
clf % clear figures
fprintf('\nExample:  Static, Quadratic Control OutPut:');
%
dx = 0.1;
x = -3:dx:+3;
us = ustar(x);
cs = C(x,us);
nfig = 0;
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
%%%%%%%%%%%%%%%
nfig = nfig + 1;
fprintf('\n\nFigure(%i):  Static Optimal Example Control\n',nfig)
figure(nfig)
plot(x,us,'k-','LineWidth',4);
axis([-3 3 -2 2]);
title('Static Optimal Example Control'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('u^*(x), Optimal Control'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('x, State'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
%%%%%%%%%%%%%%%
nfig = nfig + 1;
fprintf('\n\nFigure(%i):  Static Optimal Example Cost\n',nfig)
figure(nfig)
plot(x,cs,'k-','LineWidth',4);
title('Static Optimal Example Cost'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('C^*(x), Optimal Cost'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('x, State'...
    ,'Fontsize',44,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%
```

bk0alt
2007/8/10
page C42


```
% End Optimal Example Code
function us = ustar(x)
us = max(-1,min(+1,x));
%
function cv = C(x,u)
cv = 2 + x + 0.5*(u-x).^2;
%
% End SimpleOptExample
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.25 Program: Bang-Bang Control with Control Switching Example

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function bangbangdetctrl05fig1
% Fig. A.4 Book Illustration for solution of Bang-Bang Control (3/2007)
%   Problem Example in Deterministic Optimal Control Chapter.
%   Application is a leaky reservoir with linear state dynamics:
%       \dot{X}(t) = -a X(t)+U(t),   X(0) = x_0,
%   integral and point constraints,
%       \int_0^{t_f}U(t)dt = K >0, 0\leq U(t)\leq M, K\leq M t_f,
%   and running cost:
%       J[X]=\int_0^{t_f} X(t) dt
%   Technique is to make integral constraint into a new state,
%       X_2(t) = U(t), X_2(0) = 0, X_2(t_f) = K, so X_1(t) = X(t).
clc % clear variables, but must come before globals,
    %    else clears globals too.
clf % clear figures
fprintf('\nfunction bangbangdetctrl05fig1 OutPut:');
%%% Initialize input:
N = 100; t0 = 0; tf = 2.0; dt = (tf-t0)/N;% Set time grid parameters.
a = 0.6; M = 2; K = 2.4; x0 = 1.0; % set constant parameters:
% get optimal control, optimal state and optimal multiplier sum:
t = 0:dt:tf; % time grid;
ts = K/M; % switch time for U* = M to U* = 0;
fprintf('\nSwitch time = %f',ts);
nt = size(t,2);
x = x0*exp(-a*t); C2 = -(1-exp(-a*(tf-ts)))/a; expf = exp(-a*tf);
y = exp(a*t); ys = exp(a*ts); Mda = M/a;
lam1 = (1/a)*(1-expf*y); lamsum = C2+lam1;
u = zeros(nt);
for i=1:nt
    if t(i)<=ts
        u(i) = M;
        x(i) = x(i)+Mda*(1-1/y(i));
    else
        u(i) = 0;
        x(i) = x(i)+Mda*(ys-1)/y(i);
```

```
    end
end
% Begin Plot:
nfig = 1;
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
fprintf('\n\nFigure(%i):  Bang-Bang Control Problem Example\n',nfig)
figure(nfig)
plot(t,u,'k-.',t,x,'k--',t,lamsum,'k:',t,zeros(size(t)),'k-' ...
    ,'LineWidth',4);
title('Bang-Bang Control Example'...
    ,'Fontsize',44,'FontWeight','Bold');
ylabel('U^*,  X^*,  \lambda_1^*+\lambda_2^*'...
    ,'Fontsize',44,'FontWeight','Bold');
xlabel('t, Time'...
    ,'Fontsize',44,'FontWeight','Bold');
text(ts,-0.15,'t_s','Fontsize',36,'FontWeight','Bold');
hlegend=legend('U^*(t), Control',...
    'X^*(t), State','\lambda_1*+\lambda_2*','Location','Best');
set(hlegend,'Fontsize',36,'FontWeight','Bold');
set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
set(gcf,'Color','White','Position' ...
    ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
%%%%
% End  bangbangdetctrl05fig1
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

C.26 Program: Singular Control Examples

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

function SingCtrlExample63
% Figs. A.5a, A.5b Book Illustration for Singular Control Example (3/2007):
% Linear State Dynamics, Power Utility.
%
clc % clear variables, but must come before globals,
    %    else clears globals too.
clf % clear figures
fprintf('\nExample:  Singular Control OutPut:');
%
global mu0 p0 c0 delta0 x0 tf xsing umax TF
%
nfig = 0;
scrsize = get(0,'ScreenSize');
ss = [3.0,2.8,2.6,2.4,2.2,2.0];
mu0 = 0.08; delta0 = 1.8*mu0; umax = 2*mu0; umin = 0;
p0 = 5.0; c0dp0 = 4.0; c0 = c0dp0*p0;
tf = 15.0; %t0 = 0;
```

```
xsing = c0dp0/(1 - mu0/delta0);
using = mu0;
TF = tf + log(1-mu0*(delta0+umax-mu0)/(delta0*umax))/(delta0+umax-mu0);
% Assuming TF \geq T0max or TF \geq T0min;
x0v = [xsing+1,xsing-1]; nx0 = 2;
% cannot predeclare dynamic t,x,u vectors : ignore mlint warnings.
for ix0 = 1:nx0
    x0 = x0v(ix0);
    T0max = -log(xsing/x0)/(umax-mu0);
    T0min = +log(xsing/x0)/mu0;
    fprintf('\n x0=%f; c0/p0=%f; xsing=%f; XF(tf)=%f;',x0,c0dp0,xsing...
         ,XF(tf));
    fprintf('\n T0min=%f, T0max=%f; TF=%f; tf=%f;',T0min,T0max,TF,tf);
    dt = 0.01; % Comparison time step size;
    if x0 > xsing
        fprintf('\n Max0Case: x0 = %f > xsing = %f;',x0,xsing);
        n0 = ceil(T0max/dt); dt0 = T0max/n0;
        % t , x, u are dynamic vectors
        for i = 1:n0+1
            t(i) = (i-1)*dt0;
            x(i) = X0max(t(i));
            u(i) = umax;
        end
        ns = ceil((TF-T0max)/dt); dts = (TF-T0max)/ns;
        for i = 2:ns+1
            it = i + n0;
            t(it) = T0max+(i-1)*dts;
            x(it) = xsing;
            u(it) = using;
        end
        nf = ceil((tf-TF)/dt); dtf = (tf-TF)/nf;
        for i = 2:nf+1
            it = i + n0 + ns;
            t(it) = TF+(i-1)*dtf;
            x(it) = XF(t(it));
            u(it) = umax;
        end
        nt = n0 + ns + nf + 1;
    end
    if x0 < xsing
        fprintf('\n Min0Case: x0 = %f < xsing = %f;',x0,xsing);
        n0 = ceil(T0min/dt); dt0 = T0min/n0;
        for i = 1:n0+1
            t(i) = (i-1)*dt0;
            x(i) = X0min(t(i));
            u(i) = umin;
        end
        ns = ceil((TF-T0min)/dt); dts = (TF-T0min)/ns;
        for i = 2:ns+1
            it = i + n0;
```

```
            t(it) = TOmin+(i-1)*dts;
            x(it) = xsing;
            u(it) = using;
        end
        nf = ceil((tf-TF)/dt); dtf = (tf-TF)/nf;
        for i = 2:nf+1
            it = i + n0 + ns;
            t(it) = TF+(i-1)*dtf;
            x(it) = XF(t(it));
            u(it) = umax;
        end
        nt = n0 + ns + nf + 1;
    end
    t(nt) = tf;
%%%%%%%%%%%%%%
    nfig = nfig + 1;
    fprintf('\n\nFigure(%i): Singular Control Example\n',nfig)
    figure(nfig)
    plot(t,x,'k-','LineWidth',4);
    if ix0 == 1, htitle=title('Singular Control Example (a)'); end
    if ix0 == 2, htitle=title('Singular Control Example (b)'); end
    if ix0 == 3, htitle=title('Singular Control Example (c)'); end
    if ix0 == 4, htitle=title('Singular Control Example (d)'); end
    set(htitle,'Fontsize',44,'FontWeight','Bold')
    ylabel('X^*(t), Optimal State'...
    ,'Fontsize',44,'FontWeight','Bold');
    xlabel('t, time'...
    ,'Fontsize',44,'FontWeight','Bold');
    hlegend=legend('X^*(t) Optimal State','Location','Best');
    set(hlegend,'Fontsize',36,'FontWeight','Bold');
    axis([0 tf 0 xsing+2]);
    set(gca,'Fontsize',36,'FontWeight','Bold','linewidth',3);
    set(gcf,'Color','White','Position' ...
        ,[scrsize(3)/ss(nfig) 70 scrsize(3)*0.60 scrsize(4)*0.80]);
end
%
% End Optimal Example Code
%
% function fv = F(x,u,t)
% global mu0 p0 c0 delta0 x0 tf xsing umax TF
% fv = (mu0-u)*x;
%
% function cv = C(x,u,t)
% global mu0 p0 c0 delta0 x0 tf xsing umax TF
% cv = exp(-delta0*t)*max(p0*x-c0,0)*u;
%
% function sv = S(x,t)
% global mu0 p0 c0 delta0 x0 tf xsing umax TF
% sv = 0;
%
```

```
function xv = XF(t)
global mu0 xsing umax TF
xv = xsing*exp(-(umax-mu0)*(t-TF));
%
function xv = X0max(t)
global mu0 x0 umax
xv = x0*exp(-(umax-mu0)*t);
%
function xv = X0min(t)
global mu0 x0
xv = x0*exp(mu0*t);
%
% End SingCtrlExample63
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```