FinM 331/Stat 339 Financial Data Analysis, Winter 2010

Floyd B. Hanson, Visiting Professor

Email: fhanson@uchicago.edu

Master of Science in Financial Mathematics Program
University of Chicago

Lecture 2

6:30-9:30 pm, 11 January 2009, Ryerson 251 in Chicago

7:30-10:30 pm, 11 January 2009 at UBS in Stamford

7:30-10:30 am, 12 January 2009 at Spring in Singapore

2.1 Monte Carlo Method Methods Introduction,

An Application of Sampling and Asymptotic Limit Theorems:

- In many financial applications integral approximations are needed, such as expectation integrals applied for real, continuous, IID RVs, $\vec{X} = [X_i]_{n\times 1}$, with proper distribution $F_X(x) = \mathbf{Prob}[X_i \leq x]$ and proper probability density $f_X(x)$ on $[a,b]$ (e.g., if fully infinite, then $(-\infty, +\infty)$):

$$\begin{aligned} \mu_{g(X)} &\equiv \mathbf{E}_X[g(X)] \equiv \int_a^b g(y) f_X(y) dy \\ &\simeq \frac{1}{n}\sum_{i=1}^{n} g(X_i) \equiv \overline{g(X)}_n, \end{aligned} \tag{1}$$

  i.e., approximated by the sample mean, which is a RV with the X_i and all lower moments of the $g(X_i)$ are assumed to exist.

- The **sample mean** $\overline{g(X)}_n$ is an unbiased estimate of $\mu_{g(X)}$, i.e.,

$$\mu_{\overline{g(X)}_n} \equiv \mathbf{E}_X\left[\overline{g(X)}_n\right] \overset{\text{iid}}{=} \frac{1}{n}\sum_{i=1}^{n} \mu_{g(X)} = \mu_{g(X)}, \tag{2}$$

  noting that $\frac{1}{n}\sum_{i=1}^{n}$ is not an RV and $\mathbf{E}_X$ is a linear operator.

- The **variance of** $\overline{g(X)}_n$ is given by the variance of the sample mean formula (17) in Lect. 1, Sect. 1.3,

$$\sigma^2_{\overline{g(X)}_n} = \sigma^2_{g(X)}/n, \tag{3}$$

so the standard error or standard deviation is
$\mathbf{SE}[\overline{g(X)}_n] = \sigma_{\overline{g(X)}_n} = \sigma_{g(X)}/\sqrt{n}$.

- However, $\sigma_{g(X)}$ is the target of the method and is ***a priori*** unknown, but for a probabilistic estimate of the error the **unbiased sample variance** $g(X)$ can be used, i.e.,

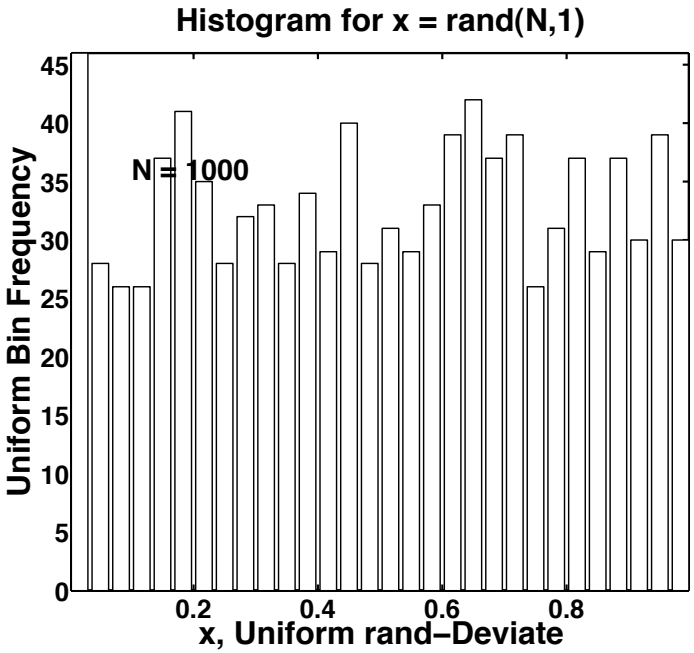
$$\widehat{s}^2_n = \frac{1}{n-1}\sum_{i=1}^{n}\Big(g(X_i) - \overline{g(X)}_n\Big)^2. \tag{4}$$

Hence, our estimate of the order of the Monte Carlo probabilistic error is the estimated $\mathbf{SE}$,

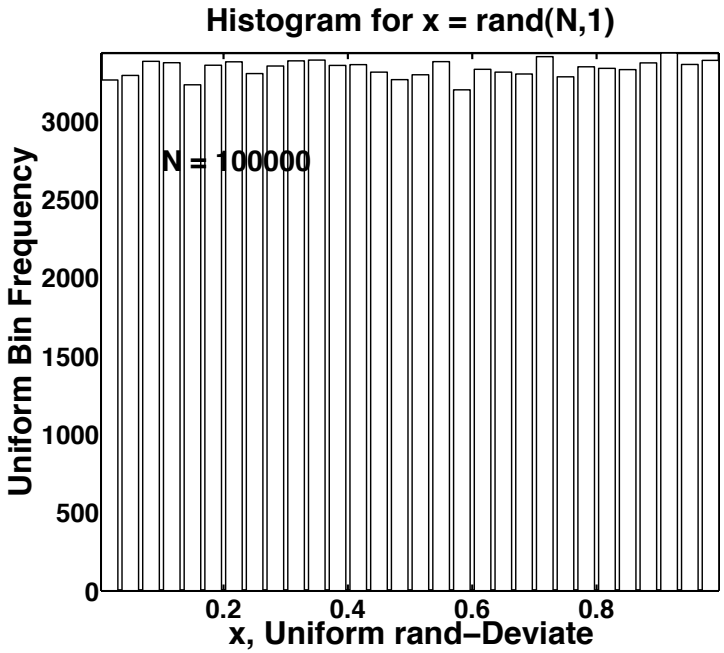
$$\widehat{\mathbf{SE}}_n\Big[\overline{g(X)}\Big] = \frac{\widehat{s}_n}{\sqrt{n}}. \tag{5}$$

- Often, the Monte Carlo estimate, itself, is written very loosely as "$\mu_{g(X)} = \overline{g(X)}_n \pm \widehat{s}_n/\sqrt{n}$", but the **order of one standard deviation margin of error** should be understood, i.e, $\mu_{g(X)} = \overline{g(X)}_n + \mathrm{O}(\widehat{s}_n/\sqrt{n})$.
- Hence, an Monte Carlo estimate can be computed by using an appropriate pseudorandom number generator like `rand(1,n)` for the row `n`-vector, **standard uniform distribution** on $[\delta, 1-\delta] \simeq (0^+, 1^-)$ in basic `MATLAB` or `unifrnd` in the `Statistics Toolbox`, where $\delta =$`eps`/2 and `eps` is the machine epsilon in MATLAB, and `randn(1,n)` for the **mean zero, unit variance, standard normal distribution** on $(-\infty, +\infty)$ in basic `MATLAB` or `normrnd(0,1,1,n)` in the `Statistics Toolbox`, for instance. In general, `unifrnd(a,b,m,n)` $\Leftrightarrow$ `a+(b-a)*rand(m,n)` and `normrnd(mu,sigma,m,n)` $\Leftrightarrow$ `mu+sigma*randn(m,n)` for `m X n` samples.

  {*Mileage will vary with other computational math/stat. systems.*}



(a) Sample size $N = 10^3$.

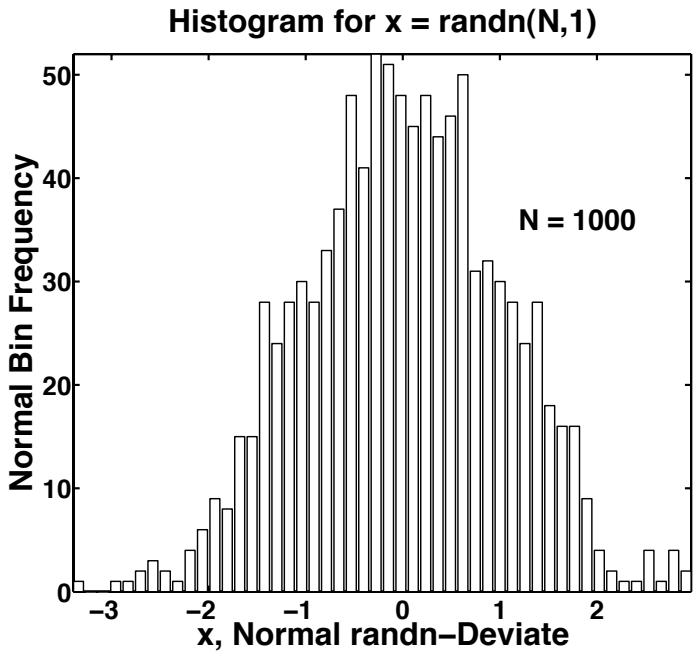


(b) Sample size $N = 10^5$.

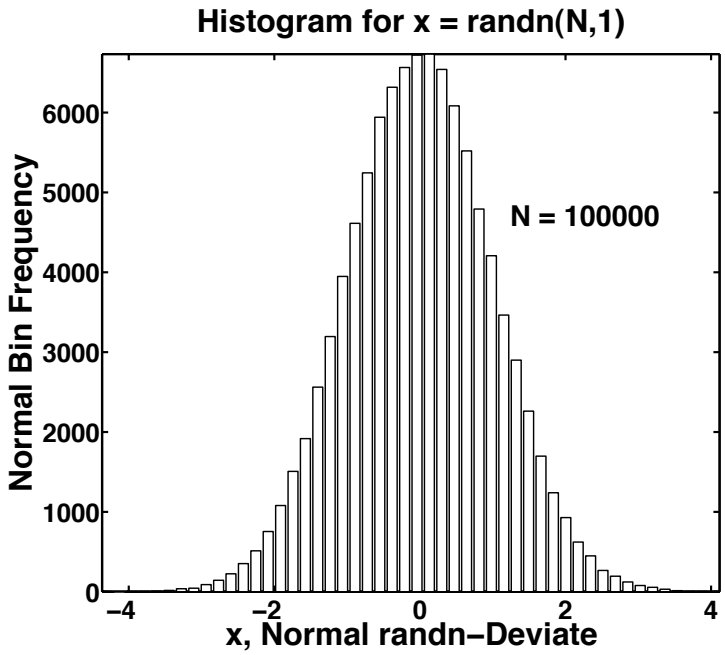
Figure 1: Histograms of simulations of the **standard uniform distribution** $f_X^{(u)}(x; 0, 1)$, $x \in (0, 1)$, using **`MATLAB rand`** for two different sample sizes N using ***sample uniform code*** (see also code copy in Chalk/Course Documents) in Hanson (2007) ***Online Appendix B Preliminaries***, page B6. **Hence, need large sample size to get a reasonable approximation to the distribution.**

Uniform code example for Figure 1 abbreviated to one page:

```
function uniform09fig
% FINM 331 Uniform Histogram Demo. from Hanson book (2007)
fprintf('\nfunction uniform09fig OutPut:')
for m = 3:2:5;
    N=10^m;
    rand('twister',3);
    x=rand(N,1);
    xmean=mean(x); xstd=std(x);
    xmin = min(x);  xmax = max(x);
    nbins = 30; % min(fix(sqrt(10^m)),101);
    xmin = 0; xmax = 1;%Adjust to theoretical [0,1] for graph;
    xbin1 = xmin; xbin2 = xmax; dxbin = (xbin2-xbin1)/nbins;
    xbin = xbin1+dxbin/2:dxbin:xbin2-dxbin/2;
    nx = hist(x,xbin); % Need Edge Oriented histc.
    bar(xbin,nx); axis tight;
    title('Histogram for x = rand(N,1)','Fontsize',44...
     ,'FontWeight','Bold');
% Code shorten for one page
end % For demo, use hist(rand(1e3,1),30), hist(rand(1e5,1),30)
    % in the command window.
```



(a) Sample size $N = 10^3$.



(b) Sample size $N = 10^5$.

Figure 2: Histograms of simulations of the **standard normal distribution** with mean 0 and variance 1 using **`MATLAB randn`** with 50 bins for two sample sizes N. The histogram for the large sample size of $N = 10^5$ in Subfigure 2(b) exhibits a better approximation to the theoretical normal density $f_X^{(n)}(x; 0, 1)$ using ***normal sample code*** (see also code copy in Chalk/Course Documents) in Hanson (2007) ***Online Appendix B Preliminaries***, page B9. Note that **`randn(1e5,1)`** $\in (-5, 5)$, approximately, and **`normpdf(5,0,1)=1.4867e-6`**.

Normal code example for Figure 2 abbreviated to one page:

```
function normal09fig
% FINM 331 Normal Histogram Demo. from Hanson book (2007)
for m = 3:2:5
    N=10^m;
    x=randn(N,1);
    xmean=mean(x);
    xstd=std(x);
    xmin = min(x); xmax = max(x);
    xbin1 = xmin; xbin2 = xmax; dxbin = (xbin2-xbin1)/nbins;%
    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);
    ks = [0.4,0.7];  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('Normal Bin Frequency','Fontsize',44);
    xlabel('x, Normal randn-Deviate' ,'Fontsize',44);
end% For demo., use hist(rand(1e3,1),30), hist(rand(1e5,1),30).
```

- ***Being innovative can be helpful.*** If the desired integral is of the form,

$$\int_a^b h(x)dx, \tag{6}$$

given a, b and a **moderately varying** $h(x)$, then let

$$f_X(x) = 1/(b-a) \tag{7}$$

be a ***uniform density*** on (a, b),

$$g(x) = (b-a) * h(x) \tag{8}$$

where $*$ denotes multiplication in **MATLAB**, forming a substitute expectation for the original integral (9),

$$\int_a^b g(x) f_X(x)dx = \mu_{g(X)} \equiv \mathrm{E}_X[g(X)]. \tag{9}$$

The sample mean is evaluated using the uniform variate,

```
X =a*ones(n,1)+(b-a)*rand(n,1);
```
(10)

an n-vector (**MATLAB is most efficient in matrix-vector form**).

The **sample mean** is

```
gmean = mean(g(X));
```
(11)

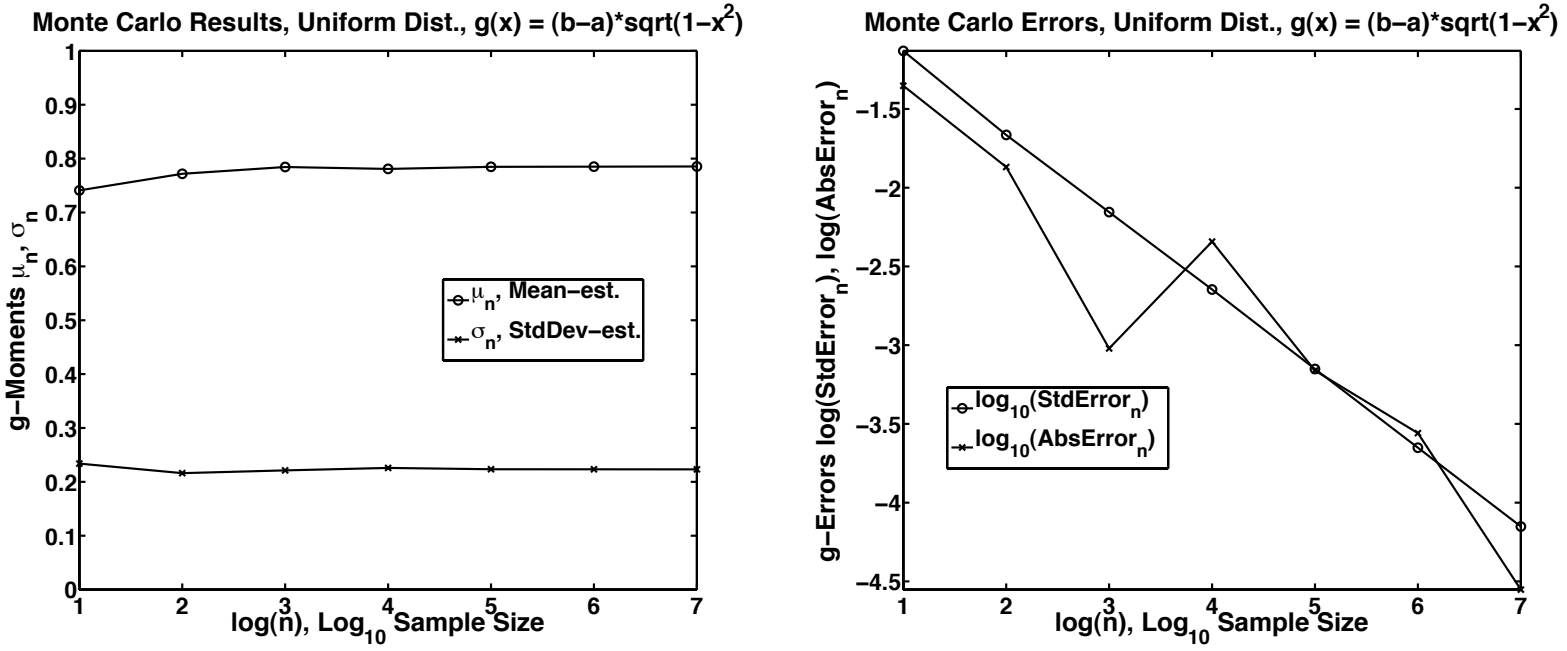
and the approximate integral is

hintegral = mean(g(X))/(b-a); (12)

The **estimated standard error (SE) of** $\overline{g(X)}$ is

$$\sigma_{\overline{g(X)}} = \frac{\sigma_{g(X)}}{\sqrt{n}} \simeq \frac{\widehat{s_n}}{\sqrt{n}} = \widehat{\mathrm{SE}}_n\left[\overline{g(X)}\right], \tag{13}$$

replacing the theoretical standard deviation $\sigma_{g(X)}$, which is usually unknown, by the feasibly estimated (unbiased as a variance) sample standard deviation $\widehat{s_n}$. The estimated standard error for the Monte Carlo approximation of the original will be $\widehat{\mathrm{SE}}_n\left[\overline{g(X)}\right]/(b-a)$.



(a) Mean & Std. Dev.

(b) Std. & Abs. Errors.

Figure 3: Monte Carlo resuls for ***uniform rand-deviates*** for $f(x) = 1/(b-a)$ and $g(x) = (b-a) * h(x)$ on (a, b) as a function of sample size N using ***sample MCM uniform code*** (see also code copy in Chalk/Course Documents) in Hanson (2007) ***applied stochastics text***, page 265.

- ***Uniform Monte Carlo code example*** for Figure 3 abbreviated to one page:

```
function mcm1test09
% FINM 331 mcm1test:  Monte Carlo Method (2009),
% integral of h(x) = sqrt(1-x^2); on [a,b]:
a = 0; b = +1; % -1 <= a < b <= +1;
IntExact = 0.5*(asin(b)-asin(a))+0.5*(b*sqrt(1-b^2)-a*sqrt(1-a^2));
MugExact = IntExact;
Sigg = sqrt((b-a)^2*(1-(b^2+a*b+a^2)/3)-MugExact^2); kmax = 7;
n = zeros(1,kmax); meang = zeros(1,kmax); sigg = 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);
   g = (b-a)*sqrt(1-x.^2); % vectorized g;
   meang(k) = mean(g); % E[g(X)];
   sigg(k) = std(g); % sqrt(sigmag^2),sigmag^2 = unbiased Var(g );
   sigdrn(k) = sigg(k)/sqrt(n(k));
   error(k) = abs(meang(k)-MugExact);
   fprintf('%1i %8i %6.4f   %6.4f %9.3e %9.3e  %9.3e %9.3e\n'...
       ,k,n(k),meang(k),MugExact,sigg(k),Sigg,sigdrn(k),error(k))
end
% end mcm1test.m  (For plot code see source)
```

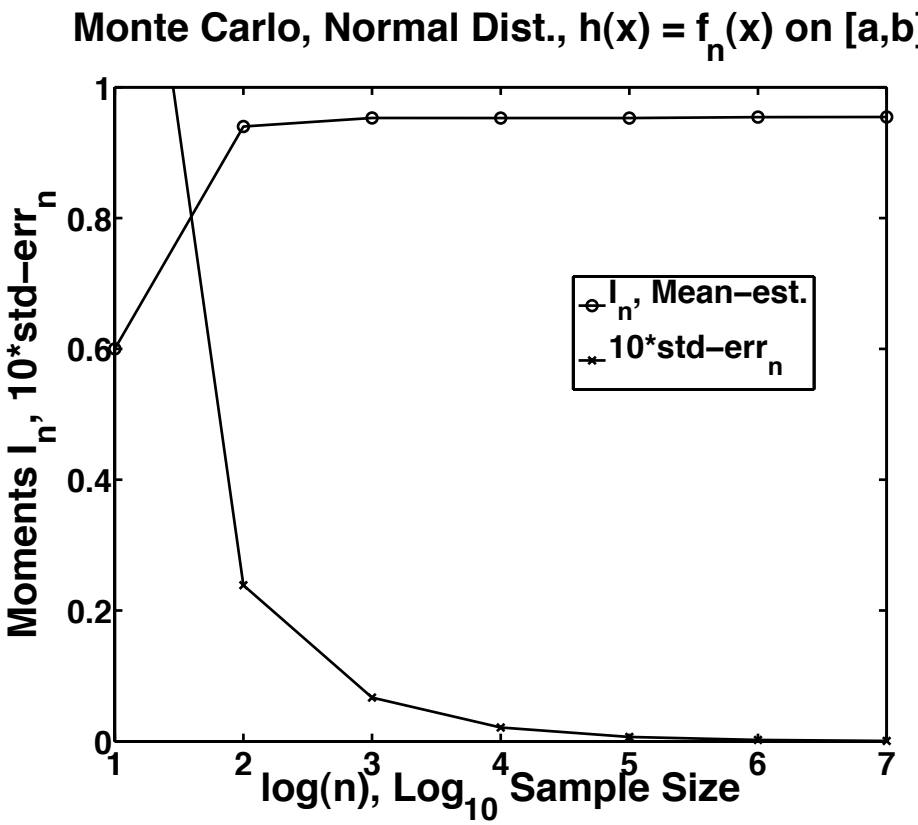


Figure 4: Monte Carlo results for ***normal randn-deviates*** for the standard normal distribution on (a, b) as a function of sample size N by the **von Neumann method of acceptance and rejection** to account for the reduced sample size on a finite interval (observations outside the interval are not counted) using ***sample MCM uniform code*** (see also code copy in Chalk/Course Documents) in Hanson (2007) ç page 272.

- ***Normal Monte Carlo with Acceptance Sampling code example***

  applied to finite range normal in Figure 4:

```
function mcm2acceptreject09
% FINM 331: Normal Monte Carlo with Acceptance-Rejection Sampling.
%   I = int(h(x),x=a..b), h(x) = f(x) = exp(-x^2/2)/sqrt(2pi);
a = -2; b = 2; % limits of integration;
f = inline ('exp(-x.*x/2)./sqrt(2*pi)','x');  % x in [a,b]
kmax = 7; nmc = 10^kmax;  nac = 0;% select MCM random sample size;
x = randn(1,nmc); %  nmc X 1 normal distribution;
kv = zeros(1,kmax); Ihatn = zeros(1,kmax); stderrn = zeros(1,kmax);
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;
        Ihatn(k) = nac/n; % Estimate Integral
        stderrn(k) = sqrt(Ihatn(k)*(1-Ihatn(k))/(n-1));
fprintf ('%2i   %8i  %8.6f  %9.3e\n',k,n,Ihatn(k),stderrn(k));
    end
end
% end mcm2acceptreject.m (For plot part of code see source.)
```

- **Acceptance-Rejection Sampling Method of von Neumann (1951)**: Often integrals will have a density $f_X(x)$ defined on $[a,b]$, but the integral is defined on a subset $[c,d] \subset [a,b]$ or in the more complicated case there is **NO random number generator** available for $f_X(x)$. In the simpler case, the **accepted** (ac) integral is

$$\begin{aligned} I^{(\mathrm{ac})}[g] &\equiv \int_c^d g(x) f_X(x) dx = C^{(\mathrm{ac})} \mathrm{E}[g(X)|X \in (c,d)] \\ &= C^{(\mathrm{ac})} \mu_{g(X)}^{(\mathrm{ac})}, \end{aligned} \tag{14}$$

where $C^{(\mathrm{ac})} = I^{(\mathrm{ac})}[g] \equiv \int_c^d f_X(x)dx = F_X(d) - F_X(c)$ is the **renormalization constant**, $f_X^{(\mathrm{ac})}(x) = f_X(x)/C^{(\mathrm{ac})}$ is the **accepted density** and $\mu_{g(X)}^{(\mathrm{ac})}$ is the mean of $g(X)$ on the accepted (ac) interval. Given the usual IID sample $\vec{X} = [X_i]_{n\times 1}$ on $[a,b]$ such that $n = n^{(\mathrm{ac})} + n^{(re)}$, i.e., $n^{(\mathrm{ac})}$ is the **number of accepted observations** $\vec{X}^{(\mathrm{ac})} = [X_i^{(\mathrm{ac})}]_{n^{(\mathrm{ac})}\times 1}$ in $[c,d]$, while $n^{(re)}$ is the **rejected number**. **Both numbers are random since they vary with the sample.**

Also, let $1^{(\mathrm{ac})}(x) = 1$ if $x \in [c, d]$ and otherwise 0 be the **accepted indicator function**, so $g^{(\mathrm{ac})}(x) \equiv g(x)1^{(\mathrm{ac})}(x)$ is the function on $[a, b]$, where we will work. The **unbiased sample approximation to the accepted normalization constant** is

$$C^{(\mathrm{ac})} = I^{(\mathrm{ac})}[1] \simeq \frac{1}{n}\sum_{i=1}^{n} 1^{(\mathrm{ac})}(X_i) = \frac{n^{(\mathrm{ac})}}{n} \equiv \widehat{I}_n^{(\mathrm{ac})}[1] = \widehat{C}_n^{(\mathrm{ac})}, \quad (15)$$

since

$$\mathrm{E}_X[\widehat{I}_n^{(\mathrm{ac})}[1]] = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_X[1^{(\mathrm{ac})}(X_i)] = I^{(\mathrm{ac})}[1]. \quad (16)$$

Then, the **accepted sample mean** of the integral of $g(X)$ is

$$\begin{aligned} \widehat{I}_n^{(\mathrm{ac})}[g] &\equiv \frac{1}{n}\sum_{i=1}^{n} g(X_i)1^{(\mathrm{ac})}(X_i) = \frac{1}{n}\sum_{i=1}^{n^{(\mathrm{ac})}} g(X_i^{(\mathrm{ac})}) \\ &= \frac{\widehat{C}_n^{(\mathrm{ac})}}{n^{(\mathrm{ac})}} \sum_{i=1}^{n^{(\mathrm{ac})}} g(X_i^{(\mathrm{ac})}), \end{aligned} \quad (17)$$

since by accepted expectations it yields the **unbiased estimate** of the integral, i.e.,

$$\mathrm{E}_X\left[\widehat{I}_n^{(\mathrm{ac})}[g]\right] = \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_X\left[g(X_i)1^{(\mathrm{ac})}(X_i)\right] = I^{(\mathrm{ac})}[g]. \quad (18)$$

Similarly, the variance is by the usual IID calculation,

$$\begin{aligned}\sigma^2_{\widehat{I}_n^{(\mathrm{ac})}} &= \mathrm{Var}_X\left[\widehat{I}_n^{(\mathrm{ac})}\right] = \mathrm{E}_X\left[\left(\widehat{I}_n^{(\mathrm{ac})} - \mu_{\widehat{I}_n^{(\mathrm{ac})}} 1^{(\mathrm{ac})}(X_i)\right)^2\right] \\ &\overset{\mathrm{iid}}{=} \frac{1}{n^2}\sum_{i=1}^n \mathrm{E}_X\left[\left(g^{(\mathrm{ac})}(X_i) - \mu_{g^{(\mathrm{ac})}(X)}\right)^2 1^{(\mathrm{ac})}(X_i)\right] \\ &= \frac{\sigma^2_{g^{(\mathrm{ac})}(X)}}{n},\end{aligned} \tag{19}$$

The **unbiased sample variance** of $g^{(\mathrm{ac})}(X)$ is

$$\widehat{s}^2_n\left[g^{(\mathrm{ac})}\right] = \frac{1}{n-1}\sum_{i=1}^n \left(g^{(\mathrm{ac})}(X_i) - \overline{g^{(\mathrm{ac})}(X)}_n 1^{(\mathrm{ac})}(X_i)\right)^2 \tag{20}$$

and thus the **standard error** of $g(X)$ can be approximated by

$$\mathrm{SE}\left[\widehat{I}_n^{(\mathrm{ac})}\right] = \frac{\sigma_{g^{(\mathrm{ac})}(X)}}{\sqrt{n}} \simeq \frac{\widehat{s}_n\left[g^{(\mathrm{ac})}\right]}{\sqrt{n}}. \tag{21}$$

In Figure 4, $g(x) = 1$, so

$$\mathrm{SE}_n^{(\mathrm{ac})}\left[\widehat{I}_n^{(\mathrm{ac})}\right] \simeq \sqrt{\frac{1}{n}\frac{n^{(\mathrm{ac})}}{n}\left(1 - \frac{n^{(\mathrm{ac})}}{n}\right)}. \tag{22}$$

Remarks: The details are left to the viewer.

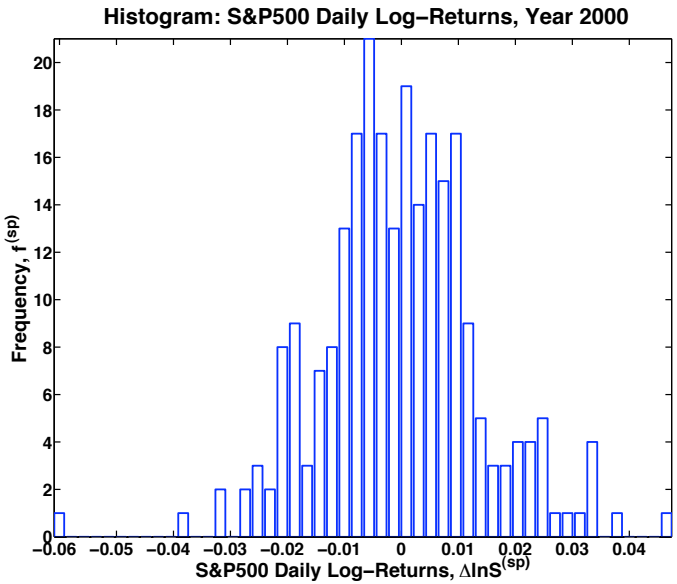
- ***Monte Carlo Risk-Neutral European Call Pricing code example***: Note automatic acceptance and rejection by max function.

```
function mcm4eurocall
% Monte Carlo Risk-Neutral Pricing for European Call Options (2009),
% Adapted from D.J. Higham, Comp.Sci.&Engr., Nov/Dec2004, pp.72-79.
clc; clear
fprintf('Monte Carlo Pricing for European Call Options (2009)');
A0 = 2; K = 1; r = 0.05; sigma = 0.25; T = 3; N = 1.e6; PC = 95/100;
randn('state',50);
A = A0*exp((r-0.5*sigma^2)*T+sigma*sqrt(T)*randn(N,1)); %AssetPrice
C = exp(-r*T)*max(A-K,0); % Call Prices with Acceptance by Max;
Cmean = mean(C);
SE_C = std(C)/sqrt(N); % Standard Error for Call Cmean
width = norminv((1+PC)/2)*SE_C; % 100*PC% CI Bandwidth
CImin = Cmean-width; % Confidence Inteval left
CImax = Cmean+width; % Confidence Inteval right
fprintf('\nCImin=%6.4f < Cmean=%6.4f < CImax=%6.4f;\n'...
,CImin,Cmean,CImax);
%%%%%%%% Delete this function if  have Statistics Toolbox %%%%
function  z = norminv(p)
z = -sqrt(2)*erfcinv(2*p);
% end mcm3eurocall.m
```

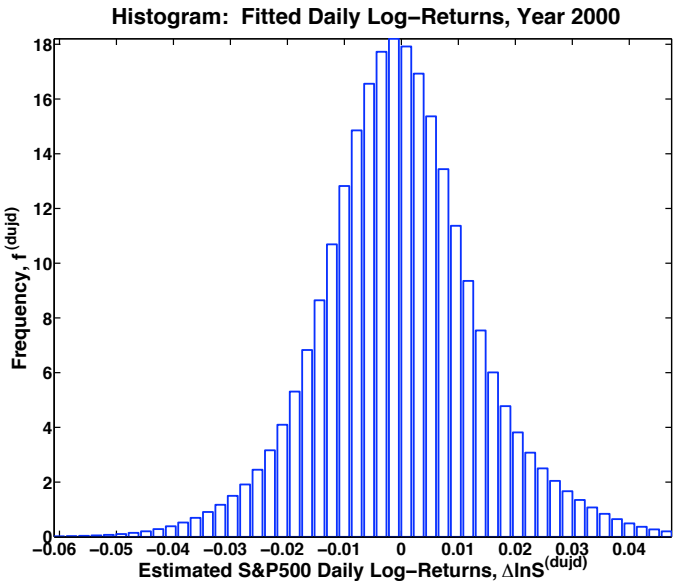
Output: CImin=1.1436 $<$ Cmean=1.1453 $<$ CImax=1.1471;

Remark: Cmean is close to Black-Scholes answer, $C_{\mathbf{bs}} = \mathbf{1.1447}$*, and within CI.*

• 2.2. Evidence of Jumps and Time-Dependence:



(a) Histogram for ***S&P500 (sp) data***.

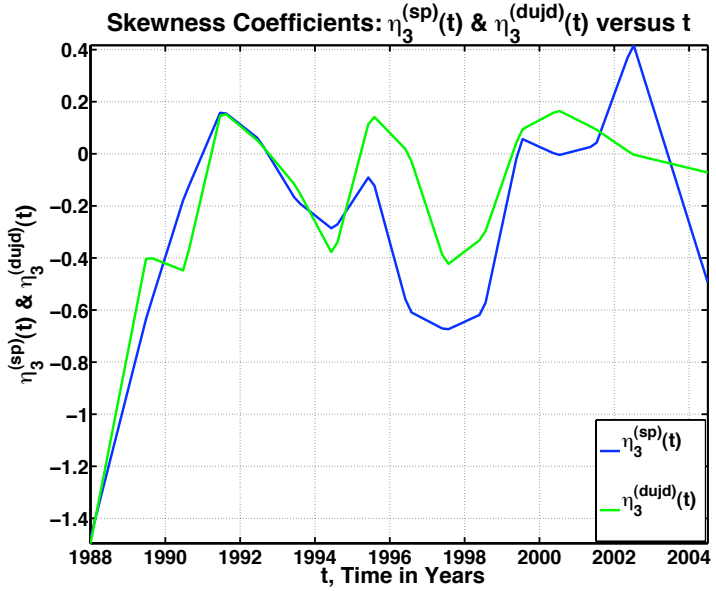


(b) Histogram for ***DUJD (dujd) model***.

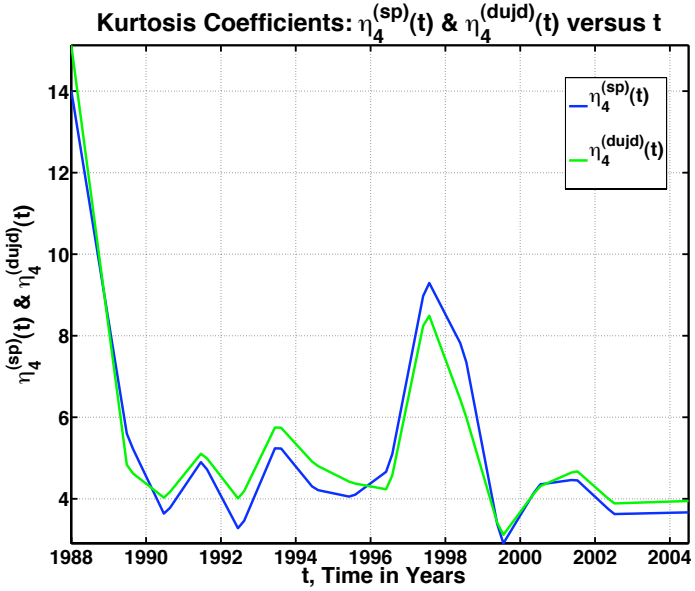
Figure 5: Histogram frequencies using year 2000 closings for ***S&P500 (sp) data*** and ***double-uniform jump-diffusion (dujd) model*** from ***Zhu & Hanson, 2006 book chapter***. Notice skew asymmetry, fat tails, peakedness and finite extent.

Higher Central Moments with $M_X^{(k)} \equiv \mathrm{E}_X[(X-\mu_X)^k]$, $k \geq 2$, with $M_X^{(2)} = \sigma_X^2$:

- ***Skewness***, $\eta_X^{(3)} \equiv M_X^{(3)}/\sigma_X^3 \neq 0$ ***(normal value)***, is a scaled 3rd central moment measure of asymmetry. Often in financial markets, $\eta_X^{(3)} < 0$ due to the usual dominance of crashes over rallies and develops after a sufficient amount of time, but can be positive during some time periods.
- ***Kurtosis***, $\eta_X^{(4)} \equiv M_X^{(4)}/\sigma_X^4 > 3$ ***(normal value)***, is a scaled 4th central moment measure of fat tails, associated with high crowns (peaks). Excess Kurtosis over the normal value, $\delta\eta_X^{(4)} \equiv \eta_X^{(4)} - 3$, due to nonnormal processes develops after a sufficient amount of time, but can be found to be small, i.e., data is nearly normal, over shorter periods.



(a) Skewness Coefficient $\eta_X^{(3)}(t)$.



(b) Kurtosis Coefficients $\eta_X^{(4)}(t)$.

Figure 6: Estimation (midyear smoothing) of coefficients of skewness and kurtosis for ***S&P500 (sp) data*** and ***double-uniform jump-diffusion (dujd) model*** from ***Zhu & Hanson, 2006 book chapter***. Notice time-dependence and variable comparisons. Data is S&P 500 daily adjusted closings in 1988-2004.

- Aside from high frequency financial data (more than daily closings), the ***number of daily closings per year are not that large***, on average there are $n \simeq 252$ market daily closings per year, or roughly $\Delta t \simeq 1/252 \simeq 4 \times 10^{-3}$ market years between daily closings, so the realistic limits n just large and Δt just small are not quite the purely mathematical limits, $n \to +\infty$ and $\Delta t \to 0^+$, respectively, for higher order approximations, except when there is high frequency data (data between closings).
- Market parameters are ***Time Dependent***, i.e., in discrete time, $\mu \Delta t = \mu_i \Delta t$ and $\sigma\sqrt{\Delta t} = \sigma_i \sqrt{\Delta t}$ for $i = 1 : n$ time-steps, so that the cumulative log-return, **as a summed time series**,

$$\log(A_{n+1}) = \log(A_1) + \sum_{i=1}^{n} \log(1 + \mu_i \Delta t + \sigma_i \sqrt{\Delta t} Z_i), \quad (23)$$

  are the multiplicative stochastic model ***log random variables (RVs)***, but are ***no longer the sum of identically distributed (ID)***, in general.

- Stochastic ***Jumps such as Crashes and Bubbles*** are relatively rare and are nonnormal processes that include large deviations with probability greater those predicted by the (log-)normal distribution. Such jumps are documented for large number of financial instruments. Merton (1976) pioneered the first ***Jump-Diffusion*** finance model, for option pricing. In discrete asset data form, with constant parameters, the model, **as a time series**, is

$$A_{i+1} = A_i \left(1 + \mu \Delta t + \sigma \sqrt{\Delta t} Z_i + \sum_{j=1}^{Y_i} \nu_{i,j} \right) , \qquad (24)$$

for $i = 1 : n$, where Y_i is a **Poisson jump counting RV** with $\mathrm{E}[Y_i] = \lambda \Delta t = \mathrm{Var}[Y_i]$, λ is a **constant jump rate**, the $\nu_{i,j}$ are **IID jump-amplitude RVs**, while the Z_i are usually distributed as a standard normal distribution, $F_Z^{(n)}(z; 0, 1)$, such that the $\{Z_i, Y_i, \nu_{i,j}\}$ are pairwise independent.

The **Poisson distribution**, for for $k = 0, 1, 2, \ldots$ jumps in a Δt time-step, is

$$p_k(\lambda\Delta t) \equiv \mathbf{Prob}[Y_i = k] = e^{-\lambda\Delta t}\frac{(\lambda\Delta t)^k}{k!}. \tag{25}$$

So, in the case of a **small** Poisson parameter, $\lambda\Delta t \ll 1$, the Poisson is approximated by a ***Bernoulli process*** which satisfies a **zero-one law** such that $Y_i = 0$ with probability, $1 - \lambda\Delta t$ and $Y_i = 1$ with probability $\lambda\Delta t$ and with negligible error $\mathbf{O^2}(\lambda\Delta t) = \mathbf{O}((\lambda\Delta t)^2)$. In this **zero-one jump** case, the jump-diffusion model becomes a much easier problem,

$$A_{i+1} = A_i\left(1 + \mu\Delta t + \sigma\sqrt{\Delta t}Z_i + \nu_{i,1}Y_i\right), \tag{26}$$

for $i = 1 : n$, since $\sum_{j=1}^{Y_i}\nu_{i,j} = \nu_{i,1}Y_i$ if Y_i takes on only values zero and one, often denoting failure and success events, like tails of heads in a coin toss, for the general Bernoulli problem.

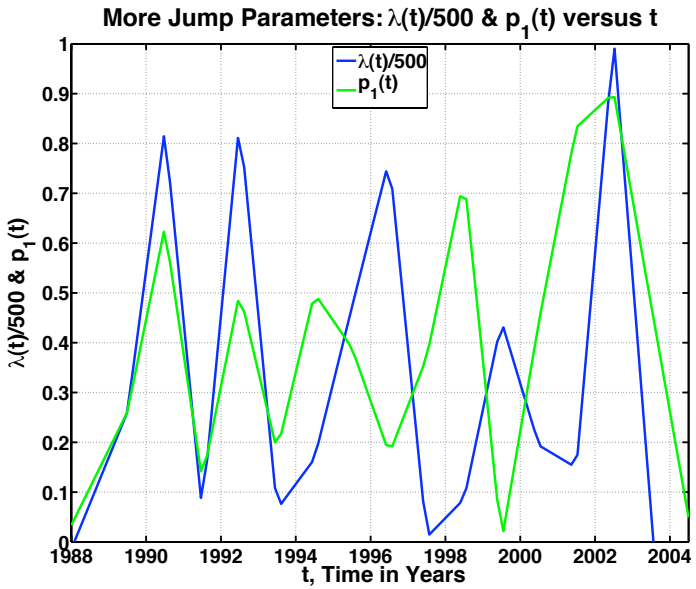
- The conversion to the log form is not as simple as for the pure diffusion Z_i multiplicative case with drift μ and it is necessary to go back to the original continuous time formulation. There are independent chain rules for the continuous diffusion process and the discontinuous jump process, since **at an instantaneous jump there is no time ($\Delta t = 0$) for the continuous change**. Hence, for the Bernoulli **0 – 1** jump law with $\lambda \Delta t \ll 1$,

$$\log(A_{i+1}) = \log(A_i) + \log(1 + \mu \Delta t + \sigma \sqrt{\Delta t} Z_i + \nu_i Y_i); \quad (27)$$

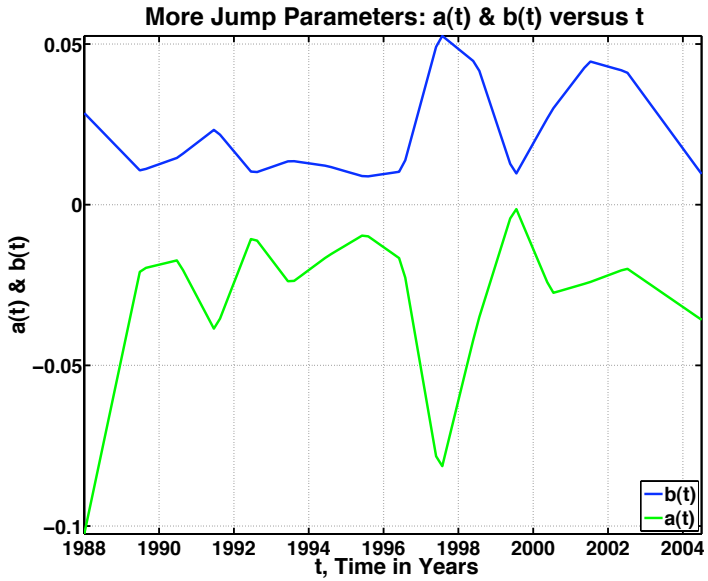
$$\log(A_{i+1}) \simeq \log(A_i) + \sigma \sqrt{\Delta t} Z_i + (\mu - \sigma^2 Z_i^2/2) \Delta t + \log(1 + \nu_i) Y_i, \quad (28)$$

where the two step approximation is first with $\{Y_i = 0, 0 < \Delta t \ll 1\}$ and then with $\{Y_i = 1, \Delta t = 0\}$. We will take (28) as our Bernoulli jump-diffusion model as given. The error $O^{3/2}(\Delta t)$ is negligible in probability. The log-asset prices A_i are distributed as a **Poisson mixture of normal distributions according to the jump counts k** (Hanson, 2007), causing a distribution with **fat tails**.

• 2.2.5 More Evidence of Jumps and Time-Dependence:

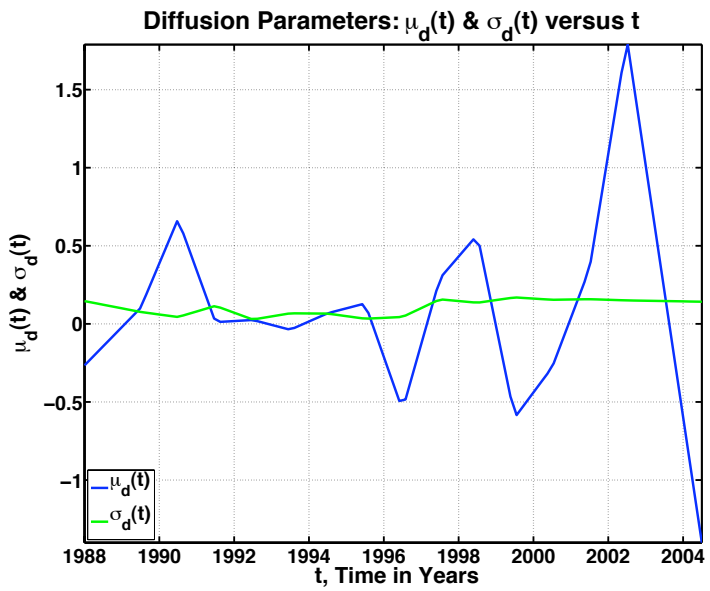


(a) $\boldsymbol{\lambda(t)}$ & $\boldsymbol{p_1(t)}$.

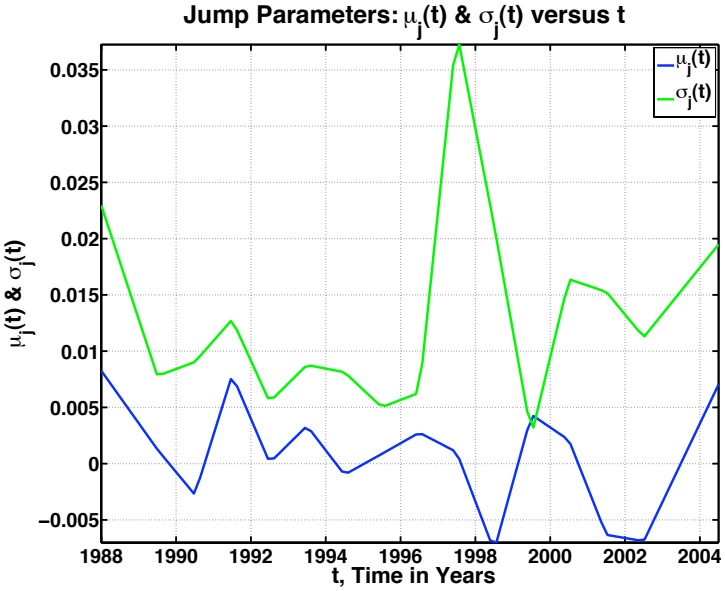


(b) $\boldsymbol{a(t)}$ & $\boldsymbol{b(t)}$.

Figure 7: Estimation (midyear smoothing) of DUJ model parameters jump rate $\boldsymbol{\lambda(t)}$, negative-jump probability $\boldsymbol{p_1(t)}$with lower bound $\boldsymbol{a(t)}$, and positive-jump upper bound $\boldsymbol{b(t)}$ from ***Zhu & Hanson, 2006 book chapter***. Notice time-dependence, but caution since smoothing results are flawed at the far right and far left. Data is S&P 500 Index daily adjusted closings, 1988-2004.



(a) $\mu_d(t)$ & $\sigma_d(t)$ for diffusion (d).



(b) $\mu_j(t)$ & $\sigma_j(t)$ for jumps (j).

Figure 8: Estimation (midyear smoothing) of DUJD model mean and standard deviation parameters, $(\mu_d(t), \sigma_d(t))$ for diffusion component and $(\mu_j(t), \sigma_j(t))$ for jump component from *Zhu & Hanson, 2006 book chapter*. Notice time-dependence and that the ***both diffusion and jump standard deviations are much more variable than the means***, respectively.

- **2.3. Asset Models with Stochastic Volatility:**
  - Empirical evidence indicates that the asset price A_i volatility, $\sigma = \sqrt{V_i}$, is random leading to a **stochastic-volatility diffusion (SVD) model, with stochastic variance V_i**, taking the form of **coupled square-root stochastic noise models**, **as a double time series**,

$$\log(A_{i+1}) \simeq \log(A_i) + (\mu - V_i/2)\Delta t + \sqrt{V_i \Delta t} Z_i^{(a)}, \qquad (29)$$
$$V_{i+1} = V_i + \kappa_v(\theta_v - V_i)\Delta t + \sigma_v \sqrt{V_i \Delta t} Z_i^{(v)},$$

  where $V_i \geq \min(V_i) = \varepsilon_v > 0$ such that $\Delta t \ll \varepsilon_v \ll 1$ **to avoid the zero-variance singularity** as well as stochastic calculus and simulation problems (Hanson, 2009). Here, κ_v is the deterministic mean rate of decay, θ_v is the deterministic equilibrium level, σ_v is the volatility of volatility and $\rho_{a,v} = \mathrm{Cov}\left[Z_i^{(a)}, Z_i^{(v)}\right]$ is the correlation coefficient between the $Z_i^{(a)}$ and $Z_i^{(v)}$ standard normally distributed noise.

- The problem is that the ***stochastic variance and hence the stochastic volatility cannot be directly estimated***, so are ***called latent variables*** and special methods are needed to otherwise estimate these stochastic variables.
- Also, T. G. Andersen et al. (2007) have found that ***volatility jumps are highly important*** in bond yields, equity return indices and foreign exchange rates.

- ***Main Topic 2. Exploratory Data Analysis Tools: Analytical and Computational***
  - ***2.3. Pseudo-Random Number Generators (RNGs) for Financial Applications:***
    - ***Pseudo-RNG means Deterministically Simulated RNGs:***

      There is no such thing as purely random number generators and the basic procedure to simulate the generation of so-called random or **pseudo-random numbers** is purely deterministic using basic algebraic operations with large multipliers truncated by modular arithmetic with a large modulus called the **linear congruential method**, sometimes followed by techniques like shuffling the raw results to reduce bias (see any ***Numerical Recipes handbook***). There are other methods, such as those for **quasi-random number generators** that have more deterministic representations, but are more efficient for generating very large samples of these simulated random numbers.

- ***More on Pseudo-RNGs:***

  Specialists in this area work to develop new, more reliable and efficient simulated random numbers, such as the the ***Mersenne Twister*** that is used as an option in the current MATLAB uniform random generator **`rand`** or the Statistics Toolbox **`normrnd`**.

  MATLAB has not always been on the leading edge in computational systems for the applied sciences, because decades ago the **`rand`** developed for the early PCs was used beyond its lifetime and a bad bias appeared in the output of **`rand`** using more powerful and precise PCs.

- ***Uniform Random Generation:***

  The uniform or rectangular distribution is the most basic distribution and its (pseudo-)random number generator is used to generate random numbers for other distributions by transformation or other techniques. It is also the best example of fat tails. The ***uniform distribution on*** $[a,b]$ is theoretically given by its density,

$$f_Z^{(u)}(z;a,b) = \frac{1}{b-a}\left\{ \begin{array}{ll} 1, & a \le z \le b \\ 0, & else \end{array} \right\}, \tag{30}$$

  where $\mu_Z = (b+a)/2$ and $\sigma_Z^2 = (b-a)^2/12$, or by the distribution itself,

$$F_Z^{(u)}(z;a,b) = \frac{1}{b-a}\left\{ \begin{array}{ll} 0, & z \le a \\ z-a, & a \le z \le b \\ b-a, & b \le z \end{array} \right\}. \tag{31}$$

- ***MATLAB Uniform RNGs:***

  In **MATLAB** (similar in R, S and other systems), the basic uniform random number generator (RNG) is on the open interval $(0,1)$ (almost all RNG are on open intervals), i.e., for $F_Z^{(u)}(z;0^+,1^-)$ or more precisely on $[\delta, 1-\delta] \simeq (0^+,1^-)$, where $\delta = \texttt{eps}/2$ and **eps** is the **machine epsilon** in **MATLAB** as mentioned last time. The usual form on (a,b) is

$$Z = a + (b-a) * \texttt{rand}(m,n); \Longleftrightarrow Z = \texttt{unifrnd}(a,b,m,n); \quad (32)$$

  where *(m,n)* is the integer size of the random matrix generated and **unifrnd** is the **MATLAB Statistics Toolbox** uniform RNG with enhanced arguments. *Caution:* "**unifrnd(mu,sigma,n);**" means the same as "**unifrnd(mu,sigma,n,n);**" or an **n**th order random matrix, so use "**unifrnd(mu,sigma,1,n);**" or "**unifrnd(mu,sigma,n,1);**" for row and column vectors, respectively. For histogram illustration of **rand** simulations using *book sample uniform code*, see this Lecture, pp. 5-6.

- ***Matrix Laboratory Background:***

  Note that **`MATLAB`** stands for ***"MATrix LABoratory"***, not Math Laboratory, but is also a highly vectorized computational system and will perform the best if you can ***make your m-file code as much in matrix-vector form as possible***, avoid the use of **`for`** and other loops, if possible or practical.

  `MATLAB` is so popular in some areas, especially for the powerful toolboxes in the area, that the area users, such as control engineering, believe that **`MATLAB`** was develop just for them. However, it was originally developed around 1980 or earlier by ***Cleve Moler***, an University of New Mexico ***Applied Math*** Professor and Stanford PhD who worked in computational linear algebra and who built a computational system for himself in ***FORTRAN*** (***C was only a computer systems language*** then) to compute his results and graphically display them and eventual let his friends and colleagues use it and finally going commercial with it.

- ***Rand Seeds, State and Streams:***

  Each call to **`rand`** produces an different state or seed of random numbers, but this can be controlled but specifying the 'state' or 'seed', for example,

$$\texttt{rand}('state', k); Z = a + (b - a) * \texttt{rand}(m, n); \qquad (33)$$

  where ***'state'*** is a literal and ***k*** is a specified positive integer state, so if ***k*** is fixed so will the set of random numbers produced. The ***'seed'*** produces slightly different random numbers. See **`MATLAB help`** for more information or for more sophisticated control of randomstreams see **`RandStream`**.

- ***More Statistics Toolbox Uniform RNG – `unifrnd`:***

  The **`MATLAB Statistics Toolbox`** (comes with the **`MATLAB Student Version`**) provides a more compact form for the $m \times n$ uniform RNG on (a, b), actually a trivial repackaging of the above vanilla form as

  ```
  Z = unifrnd(a,b,m,n);
  ```
  (34)

  As with **`rand`** higher order random arrays beyond order 2 (matrices) can be generated, e.g, **`unifrnd(a,b,m,n,p)`** for a 3rd order, $m \times n \times p$ array.

- ***More Uniform Distribution Statistics:***

  The coefficient of skew is $\eta_3 = 0$, i.e., **skewless** and the coefficient of kurtosis is $\eta_4 = 1.8 < 3$, i.e., **negative excess kurtosis over normal or platykurtic**, flatter than normal and very much so.

- ***Normal Random Generation – Normal Probability Density Function:***

  The **normal distribution is a good representation of the central part of the distribution** of equities and other underlying financial assets when log-prices are studied, the asset prices themselves are usually log-normally distributed. Recall that the **Central Limit Theorem** (Lect. 1, pp. 39ff) predicts that is should be the limit of the scaled sample mean in the limit of a very large number of independent observations with common $\mu_z = \mathrm{E}_Z[Z]$ and $\sigma_z^2 = \mathrm{Var}_Z[Z]$ statistics.

  The **normal distribution on** $(-\infty, +\infty)$ is theoretically given by its density,

$$f_Z^{(n)}(z; \mu_z, \sigma_z^2) = \frac{1}{\sqrt{2\pi\sigma_z^2}} \exp\left(-(z-\mu_z)^2/\left(2\sigma_z^2\right)\right), \quad (35)$$

  where **`normpdf(z,mu,sigma)`** is the **`MATLAB Statistics Toolbox`** **normal probability density function (PDF)**.

- ***Normal Cumulative Distribution Function (CDF):***

  The **normal CDF** for normal RV $Z^{(n)}$ is given by

$$F_Z^{(n)}\left(z;\mu_z,\sigma_z^2\right) \equiv \mathbf{Prob}[Z^{(n)} \le z] = \int_{-\infty}^{z} f_Z^{(n)}\left(x;\mu_z,\sigma_z^2\right)\,dx \tag{36}$$

  and computed with the convenient **`Statistics Toolbox`** function

```
fz = normcdf(z,mu,sigma);
```
(37)

  given normal parameters **`mu`** and **`sigma`**.

  For the ***discretized Black-Scholes (1973) model***, the change in the log-asset can be written in Δt-**precision** (i.e., neglecting $\mathrm{o}(\Delta t)$):

$$\Delta \log(A_i) \overset{\Delta t}{=} \mu_{\log}\Delta t + \sigma\sqrt{\Delta t}Z_i \overset{\text{dist}}{=} F_Z^{(n)}\left(z;\mu_{\log}\Delta t,\sigma^2\Delta t\right); \tag{38}$$

  where $\mu_{\log} \equiv \mu - \sigma^2/2$, σ and $\Delta t \ll 1$ are constants, while ***Merton's (1973) versions are much more general***.

- ***Statistics Toolbox Normal RNG:***

  With this Toolbox, the scaled, compact form of the ***normal RNG*** is **`normrnd`** with the usual syntax,

  ```
  Z = normrnd(mu,sigma,m,n);
  ```
  (39)

  where (m, n) is the integer size of the random matrix generated.

- ***Vanilla MATLAB Normal RNG:***

  In vanilla **`MATLAB`**, the basic normal random number generator (RNG) **`randn`** is similar to the uniform **`rand`**, except for that except that it is usually generated from two uniform RVs by the elegant ***Box-Muller algorithm***. The usual form on on the real line is

  ```
  Z = mu + sigma*randn(m,n);
  ```
  (40)

  where **`(m,n)`** is the integer size of the random matrix generated. This form is the likely the underlying function that defines **`normrnd`**. For histogram illustration of **`randn`** using ***book sample normal code***, see this Lecture, pp. 7–8.

- ***Normal Distribution Inverse Function:***

  Another important function is the inverse normal CDF,

$$z = \left(F_Z^{(n)}\right)^{-1}(p;\mu_z,\sigma_z^2) \quad \ni \quad p = F_Z^{(n)}(z;\mu_z,\sigma_z^2)\,, \tag{41}$$

  where p is the given probability. The **`Statistics Toolbox`** function can be used in the form

```
z = norminv(p,mu,sigma);
```
(42)

  If the `Statistics Toolbox` is not available (it does not come with the nonstudent version of **`MATLAB`**), so it may useful to be aware of the relationship to the **complementary error function**,

$$\texttt{erfc}(x) = \frac{1}{\sqrt{\pi}}\int_x^\infty \exp(-t^2)dt \tag{43}$$

  and its inverse, i.e., for the standard normal CDF, is defined as

```
normcdf(x) = 0.5*erfc(exp(-x/sqrt(2)));
```
(44)

  with normal distribution inverse

```
norminv(x) = -sqrt(2)*erfcinv(2*p);
```
(45)

- ***Computational Caution for Normal CDF:***

  Caution*: The normal CDF is very sensitive to the approximation used and less precise approximations can lead to gross errors in the case of small x values in the tails, so making your own approximation for* **`normcdf`** *is not recommended, but the MATLAB complementary error function* **`erfc`** *was developed by the legendary special function expert, Argonne National Lab's Jim Cody.*

- ***More Normal Statistics:***

  The normal **coefficient of skew** is $\eta_3 = 0$, i.e., **skewless** and the **coefficient of kurtosis** is $\eta_4 = 3$, i.e., **no excess kurtosis** and called **mesokurtic**.

- ***Exponential Distribution:***

  The **exponential distribution** on the positive real line is important since the **time between successive jumps of a Poisson process is exponential distributed**, i.e. ***tells what the expected time to the next crash or bubble if the jump rate were known***. Also, the related **double exponential distribution** on the full real line is used by ***S. Kou & H. Wang, 2004*** to model jump amplitude distribution, as ***Z. Zhu & F. Hanson, 2005*** has used both single and double uniformly distributed jump amplitudes ***to model crashes and bubbles in the market***.

- ***Exponential Probability Density and Cumulative Distribution Functions:***

  The **exponential distribution on** $[0,\infty)$ is theoretically given by its density,

$$f_Z^{(e)}(z;\lambda)=\lambda e^{-\lambda z}\left\{\begin{array}{ll}1, & z\geq 0\\ 0, & else\end{array}\right\}, \tag{46}$$

  with $\mu_Z=1/\lambda$ and $\sigma_Z^2=1/\lambda^2$ statistics, or by the distribution itself,

$$F_Z^{(e)}(z;\lambda)\equiv \mathrm{Prob}[Z^{(e)}\leq z]=\left(1-e^{-\lambda z}\right)\left\{\begin{array}{ll}1, & z\geq 0\\ 0, & else\end{array}\right\}, \tag{47}$$

  for exponentially distributed RV, $Z^{(e)}$, where the single parameter of the distribution is the **rate** λ, the reciprocal of the mean μ_Z. The $\{\eta_3=2>0,\eta_4=9>3\}$ are the coefficients of **skew** and **kurtosis**, so has **positive skew** and **positive excess kurtosis** or **leptokurtic**.

- ***Exponential Distribution RNGs:***

  With the **`Statistics Toolbox`**, the scaled version compact exponential random number generator **`exprnd`** is given by

  ```
  Z = exprnd(mu,m,n);
  ```
  (48)

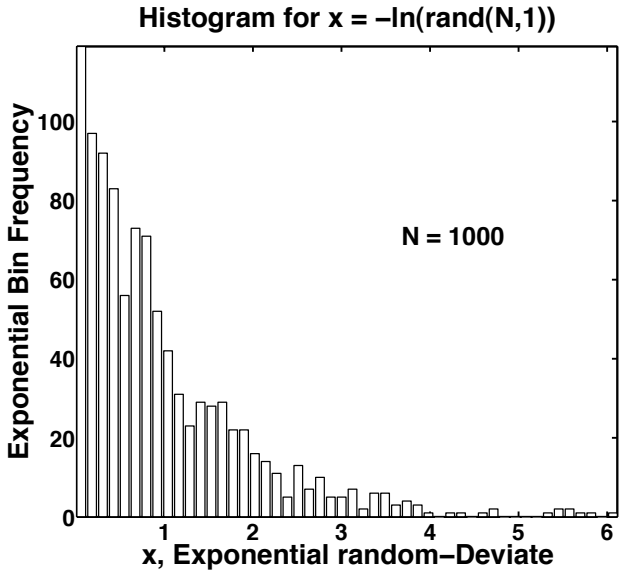
  where **`mu=1/lambda`** is the mean and **`(m,n)`** is the array size.

  There is no corresponding vanilla **`MATLAB`** version, but in ***Hanson's Online Appendix B Preliminaries*** we have

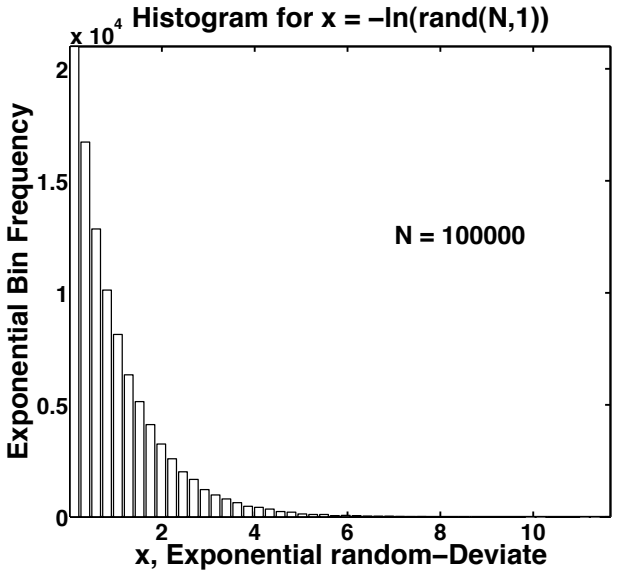
  ```
  X = -log(rand(m,n))/lambda;
  ```
  (49)

  using an efficient uniform distribution transformation using **`MATLAB`**'s **`rand`**.

  A histogram illustration of **`x=-mu*log(rand(N,1))`** simulations follows using ***book sample exponential code***.



(a) Sample size $N = 10^3$.



(b) Sample size $N = 10^5$.

Figure 9: Histograms of simulations of the **standard exponential distribution** with mean $\mu = 1/\lambda = 1$ using **`MATLAB rand`** and **`log`** with 50 bins for two sample sizes N. The histogram for the large sample size of $N = 10^5$ in Subfigure 9(b) exhibits a better approximation to the theoretical exponential density $f_Z^{(e)}(z;1)$ using ***exponential sample code*** in ***Online Appendix B Preliminaries***, page B15 and in Chalk/Course Documents.

- ***More Exponential Distribution Functions:***

  Other useful exponential distribution functions are **`expcdf`**, **`expinv`** and **`exppdf`**, whose syntax can be found from **`MATLAB help`** with the **`Statistics Toolbox`**.

  There are also several useful generic functions where the user specifies the distribution nickname as a literal in quotes of over 20 distribution options and some are **`random`**, **`cdf`** and **`pdf`**. The random integer generator **`randi`** provides an way of selection random component subscripts from a nonrandom vector of observations.

  In ***Hanson's Online Appendix B Preliminaries*** there is more information on the properties of some of these distributions, including ***means, variances, skews and kurtosis formulas***.

- ***Poisson Random Generation:***

  Siméon-Denis Poisson (pronounced like *pwah-sohn*) developed the Poisson distribution following a study of the frequency of **mule kicks** in the French Army, but ironically his name means **fish** in French and has nothing to do with the word **poison**, though it is a rare event that a student asks about those "poison variables".

  The **Poisson distribution** represents integer or discrete events, which we will call jump events or just jumps that we wish to include in models with large market crashes and rallies. The **Poisson distribution** is defined independent of times such that for jump counts $k = 0,1,2,\ldots$, the **probability** that there are exactly k **jumps** with **parameter** $\Lambda > 0$ is given by

$$p_k(\Lambda) \equiv \text{Prob}[Z^{(p)} = k] = e^{-\Lambda}\frac{\Lambda^k}{k!} \tag{50}$$

  where $k!$ is the ***factorial function*** such that $0! \equiv 1$ and recursively, $(k+1)! = (k+1)\cdot k!$.

- ***Poisson Distribution Properties:***

  The properties of the Poisson distribution follow quickly from those for the exponential function

$$\exp(x) \equiv e^x \overset{\text{taylor}}{=} \sum_{k=0}^{\infty} \frac{x^k}{k!}, \tag{51}$$

  which is **uniformly convergent** making the interchange of calculus operations very legitimate.

  The **Poisson expectation** of a function of a Poisson RV, $g(Z^{(p)})$, is defined asvspace*-0.5em

$$\mathbf{E}_Z\left[g(Z^{(p)})\right] \equiv e^{-\Lambda} \sum_{k=0}^{\infty} \frac{\Lambda^k}{k!} g(k), \tag{52}$$

  so from the properties of the basic exponential Taylor series, $\mathbf{E}[1] = 1$ trivially conserving probability since $\sum_{k=0}^{\infty} \Lambda^k/k! = \exp(\Lambda)$.

- ***Poisson Distribution Mean:***

  Similarly, but also using an interchange of differentiation and summation,

$$\begin{aligned}\mu = \mu^{(p)} \equiv \mathbf{E}\left[Z^{(p)}\right] &\equiv e^{-\Lambda}\sum_{k=0}^{\infty}\frac{\Lambda^k}{k!}k \\ &\overset{\text{ident}}{=} e^{-\Lambda}\cdot\Lambda\frac{d}{d\Lambda}\sum_{k=0}^{\infty}\frac{\Lambda^k}{k!} \overset{\text{taylor}}{=} e^{-\Lambda}\cdot\Lambda\frac{d}{d\Lambda}e^{\Lambda} \\ &\overset{\text{loe}}{=} \Lambda,\end{aligned} \tag{53}$$

  since the natural exponential has the property that it is its own derivative, $d\exp(x)/dx = \exp(x)$, by one of the **laws of exponents (LOE)** we have $\exp(a)\cdot\exp(b) \overset{\text{loe}}{=} \exp(a+b)$, and the identity $\Lambda\frac{d}{d\Lambda}\sum_{k=0}^{\infty}\frac{\Lambda^k}{k!} \overset{\text{ident}}{=} \sum_{k=0}^{\infty}\frac{\Lambda^k}{k!}k$ through uniform convergence.

- ***Poisson Distribution Variance and Beyond:***

  The **Poisson variance** reflects this one parameter distribution and is

$$\begin{aligned}\sigma^2 = \left(\sigma^{(p)}\right)^2 \equiv \mathrm{E}\left[\left(Z^{(p)} - \Lambda\right)^2\right] &\overset{\text{ident}}{=} E\left[\left(Z^{(p)}\right)^2\right] - \Lambda^2 \\ &= e^{-\Lambda}\sum_{k=0}^{\infty}\frac{\Lambda^k}{k!}k^2 - \Lambda^2 = e^{-\Lambda}\left(\Lambda\frac{d}{d\Lambda}\right)^2 e^{\Lambda} - \Lambda^2 \\ &= e^{-\Lambda}\Lambda\tfrac{d}{d\Lambda}\Lambda e^{\Lambda} - \Lambda^2 = e^{-\Lambda}\Lambda\left(\Lambda e^{\Lambda} + e^{\Lambda}\right) - \Lambda^2 \\ &= \Lambda(\Lambda+1) - \Lambda^2 = \Lambda,\end{aligned} \tag{54}$$

  where power of k inside the summand has been replaced by the same power of the operator $\Lambda\frac{d}{d\Lambda}$ outside and the only a single power of differentiation was performed at each step. Note that a bit of analysis and algebra has been used. For **higher central moments**, identities can also be used to form recursions in terms of the regular moments using the binomial theorem with given lower ones, such as $\mathrm{E}[(Z-\Lambda)^3] = \mathrm{E}[Z^3] - 3\Lambda\mathrm{E}[Z^2] + 2\Lambda^3$ for reducing the third order central moment to the regular one, given $\mathrm{E}[Z^2] = \Lambda(\Lambda+1)$.

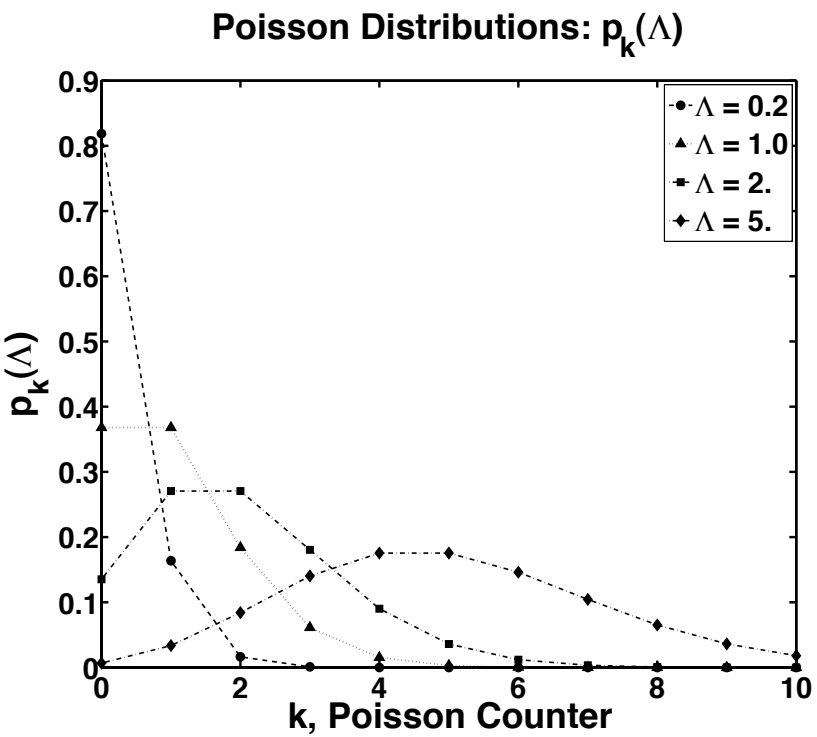


Figure 10: ***Poisson distributions*** $p_k(\Lambda)$ with respect to the Poisson counter variable k for parameter values $\Lambda = 0.2, 1.0, 2.0,$ **and** 5.0. The discrete values are connected by dashed, dotted, and dash-dotted lines to help visualize the distribution the parameter form, but notice the **normal-like** form for $\Lambda = 5.0$ all using ***Poisson sample code*** with **MATLAB** recursions in ***Online Appendix B Preliminaries***, page B21 and in Chalk/Course Docs.

- ***Statistics Toolbox Poisson Functions:***
  - `K = poissrnd(Lambda,m,n);` produces an $\mathtt{m}\times\mathtt{n}$ matrix of random Poisson counts `K` for Poisson parameter $\Lambda = \mathtt{Lambda}$.
  - `P = poisspdf(X,Lambda);` produces a continuous probability matrix (or vector or scalar) `Y` for a corresponding continuous counting matrix (or vector or scalar) `X` for integer components and else 0, i.e.,

$$Y = p_X(\Lambda)1_{X\in\{0,1,2,\dots\}}, \tag{55}$$

    say, in the scalar case where $1_{\mathcal{S}}$ is the indicator function for set $\mathcal{S}$. For Example, a poorly documented curve of one of the curves in Fig. 10 on the previous page can quickly be produced by

```
X = 0:10; Lambda = 5.0; P = poisspdf(X,Lambda);
```

    but this can fixed up to make a professional presentation.

- Also `[LambdaHat,LambdaCI] = poissfit(X,alpha)`, given input n-vector data `X` and confidence interval value `alpha` will output Poisson parameter estimate `LambdaHat` by the sample mean and a `100(1-alpha)%` confidence interval (CI).

- ***Poisson Stochastic Processes as a Time Series:***

  For modeling crashes, it is necessary to introduce time but here discrete time, as in a time series, and a constant jump amplitude so we are not restricted to unit jumps, i.e., $Z_i = c \cdot \Delta P_i$ and $\Lambda = \lambda \Delta t$, $\lambda > 0$ and $\Delta t > 0$. Let the log-return be

$$J_i = c \cdot \Delta P_i, \tag{56}$$

  such that ΔP_i is an (integer) counting process and

$$p_k(\lambda \Delta t) = \mathrm{Prob}[\Delta P_i = k] = e^{-\lambda \Delta t} \frac{(\lambda \Delta t)^k}{k!}, \tag{57}$$

  where $\mathrm{E}[\Delta P_i] = \lambda \Delta t = \mathrm{Var}[\Delta P_i]$, but $\mathrm{E}[J_i] = c\lambda \Delta t$ and $\mathrm{Var}[J_i] = c^2 \lambda \Delta t$ since the J_i process is not a unit amplitude process like ΔP_i unless $c = 1$, so the mean-variance equivalence property does not hold for even a more general Poisson process.

  P.S. ***Process means a function of time***, but can be deterministic (nonrandom) or stochastic (random).

- ***Exponential Distribution of Time Between Poisson Jumps:***

  Lemma: Let ΔP_j be a simple Poisson process with fixed jump rate $\lambda > 0$ and T_j be the jth jump-time, then the time between jumps $\Delta T_j \equiv T_{j+1} - T_j$ conditioned on T_j for $j = 0, 1, 2, \ldots$ with $T_0 \equiv 0$ is

$$F_{\Delta T_j}(\Delta t; \lambda) \equiv \text{Prob}[\Delta T_j \leq \Delta t \,|\, T_j] = 1 - e^{-\lambda \Delta t}. \quad (58)$$

  Proof: The basic idea is that the above conditional probability will be the same as the probability there will be at least one jump in Δt,

$$\begin{aligned}
\text{Prob}[\Delta T_j \leq \Delta t | T_j] &\overset{\text{prob}}{=}^{\text{total}} 1 - \text{Prob}[\Delta T_j > \Delta t | T_j] \\
&\overset{\text{idea}}{=}^{\text{basic}} 1 - \text{Prob}[\Delta P_j = 0 | T_j] \\
&\overset{\text{T}_\text{j}}{=}^{\text{indep}} 1 - \text{Prob}[\Delta P_j = 0] \\
&\overset{\text{j}}{=}^{\text{indep}} 1 - \text{Prob}[\Delta P_0 = 0] \\
&\overset{\text{p}_0(\lambda)}{=} 1 - e^{-\lambda \Delta t} = F_Z^{(e)}(\Delta t; 1/\lambda),
\end{aligned} \quad (59)$$

  where $\Delta P_j \equiv P_{j+1} - P_j$ and which proves the hypothesis.

- ***Compound Poisson Processes:***

  Simple Poisson processes are just too simple to have sufficient character for real world applications financial applications, because large market movements beyond diffusions come in many sizes. Even in ***Merton's (1976)*** pioneering jump-diffusion model generalizing Black-Scholes-Merton (1973), he used compound processes, in which in discrete log-return form of the jump term, restricted to single jumps, at the ith time-step is

  $$J_i = \tilde{\nu}_i \Delta P_i, \tag{60}$$

  where the $\tilde{\nu}_i = \log(1+\nu_i)$ from $\nu_i > -1$ are **IID RVs independent of the Poisson jump counting process ΔP_i given a jump** and in the jump-diffusion independent of the diffusion. So a compound Poisson is **doubly stochastic**, but you can think of the **Poisson process randomly generating a jump-time as well as a jump-amplitude instantaneously**. However, in the discrete case, there could be **no jumps but no more than one jump likely** in the time-interval $(t_i - \Delta t, t_i]$ if $\lambda \Delta t \ll 1$, where the jump-rate is λ.

- ***Compound Poisson Statistics:***

  For IID $\tilde{\nu}_i$, let $\mathbf{E}[\tilde{\nu}_i] \equiv \mu_{\tilde{\nu}}$ and $\mathbf{Var}[\tilde{\nu}_i] \equiv \sigma^2_{\tilde{\nu}}$, then

$$\mathbf{E}[J_i] = \mathbf{E}[\tilde{\nu}_i \Delta P_i] \stackrel{\text{ind}}{=} \mathbf{E}[\tilde{\nu}_i] \cdot \mathbf{E}[\Delta P_i] = \mu_{\tilde{\nu}} \lambda \Delta t \tag{61}$$

  and

$$\begin{aligned}
\mathbf{Var}[J_i] = & \mathbf{E}[(\tilde{\nu}_i \Delta P_i - \mu_{\tilde{\nu}} \lambda \Delta t)^2] \\
\stackrel{\text{devs}}{=} & \mathbf{E}[((\tilde{\nu}_i - \mu_{\tilde{\nu}})(\Delta P_i - \lambda \Delta t) \\
& + \mu_{\tilde{\nu}}(\Delta P_i - \lambda \Delta t) + \lambda \Delta t (\tilde{\nu}_i - \mu_{\tilde{\nu}}))^2] \\
\stackrel{\text{alg}}{=} & \mathbf{E}\big[(\tilde{\nu}_i - \mu_{\tilde{\nu}})^2 (\Delta P_i - \lambda \Delta t)^2 \\
& + 2\mu_{\tilde{\nu}}(\tilde{\nu}_i - \mu_{\tilde{\nu}})(\Delta P_i - \lambda \Delta t)^2 \\
& + 2\lambda \Delta t (\tilde{\nu}_i - \mu_{\tilde{\nu}})^2 (\Delta P_i - \lambda \Delta t) \\
& + 2\mu_{\tilde{\nu}} \lambda \Delta t (\tilde{\nu}_i - \mu_{\tilde{\nu}})(\Delta P_i - \lambda \Delta t) \\
& + \mu^2_{\tilde{\nu}}(\Delta P_i - \lambda \Delta t)^2 + (\lambda \Delta t)^2 (\tilde{\nu}_i - \mu_{\tilde{\nu}})^2] \\
\stackrel{\text{ind}}{=} & \sigma^2_{\tilde{\nu}} \lambda \Delta t + \mu^2_{\tilde{\nu}} \lambda \Delta t + \sigma^2_{\tilde{\nu}} (\lambda \Delta t)^2 \\
\stackrel{\text{alg}}{=} & \left(\sigma^2_{\tilde{\nu}}(1 + \lambda \Delta t) + \mu^2_{\tilde{\nu}}\right) \lambda \Delta t \stackrel{\Delta t}{=} \mathbf{E}\big[\tilde{\nu}_i^2\big] \lambda \Delta t.
\end{aligned} \tag{62}$$

- ***More on Compound Poisson Processes:***

  Note the with the IID assumption for the $\tilde{\nu}_i$, the mean of the compound, bilinear process is a bilinear product of means of the jump amplitude and the number of jumps on the average, making a lot of sense, but for the variance of the compound process is a ***bilinear combination of the 2nd moment*** $(\mathrm{E}[\tilde{\nu}_i^2])$ ***and the common jump variance*** $=$ ***mean jump count*** $(\lambda\Delta t)$, to ***precision-*** Δt, i.e., here neglecting $\mathrm{O}((\Delta t)^2) = \mathrm{o}(\Delta t)$ for $\Delta t \ll 1$.

- **Why do we throw away the $(\Delta t)^2$ terms in models?**

  Well, that is almost never explained in calculus. If we have an fixed interval $[0, T]$ and break it up into n parts of size $\Delta t = T/n$, then by elementary integral rules (**integrals are the ultimate solution form**),

$$\int_0^T (dt)^2 \simeq \sum_{i=1}^{n} (\Delta t)^2 = n(\Delta t)^2 = n\left(\frac{T}{n}\right)^2 = \frac{T^2}{n} \to 0^+, \quad (63)$$

  as $n \to +\infty$, **so $\mathrm{O}((\Delta t)^2) = \mathrm{o}(\Delta t)$ simply does not count and neither does $\mathrm{O}\left((\Delta t)^{3/2}\right) = \mathrm{o}(\Delta t)$ from diffusions**.

- ***Question about Change of Variables for Simple or Compound Poisson Processes:***

  The question about how the discrete jump-diffusion equation on Eq. (27) went from multiplicative (geometric) form,

$$A_{i+1} = A_i(1+\mu\Delta t+\sigma\sqrt{\Delta t}Z_i+\nu_i Y_i), \tag{64}$$

  to the approximate log-form on Eq. (28) rewritten with ($\Delta\log(A_i)$),

$$\Delta\log(A_i) \simeq (\mu-\sigma^2/2)\Delta t+\sigma\sqrt{\Delta t}Z_i+\log(1+\nu_i)Y_i \tag{65}$$

  and is very briefly explained in the text on this slide, saying that discontinuous ($\Delta^{(d)}$) or jump changes are calculated differently from continuous ($\Delta^{(c)}$) or diffusion changes. Diffusion change is by taking pure logs of the continuous part when $\Delta t \ll 1$,

$$\Delta^{(c)}\log(A_i) \simeq (\mu-\sigma^2 Z_i^2/2)\Delta t+\sigma\sqrt{\Delta t}Z_i), \tag{66}$$

  while the former is by difference of logs of the discrete part, pre-jump from post-jump (using $\log(1+\nu_i Y_i) \overset{\text{zol}}{=}_{\text{lol}} \log(1+\nu_i)Y_i$),

$$\Delta^{(d)}\log(A_i) = \log(A_i(1+\nu_i Y_i)) - \log(A_i) \overset{\text{zol}}{=}_{\text{lol}} \log(1+\nu_i)Y_i, \tag{67}$$

  and finally, $\Delta\log(A_i) \simeq \Delta^{(c)}\log(A_i)+\Delta^{(d)}\log(A_i)$.

** Reminder: Lecture 2 Homework Posted in Chalk Assignments, due by Lecture 3 in Chalk Assignments!*

*** Summary of Lecture 2:**

1. **Simple Monte Carlo Methods.**
2. **Pseudo-RNGs and Distribution Properties**
3. **MATLAB RNGs, PDFs, CDFs, ...**
4. **Uniform, Normal, Exponential Distributions, ...**
5. **Poisson Distribution and Applications**
6. **Compound Poisson if time permits**