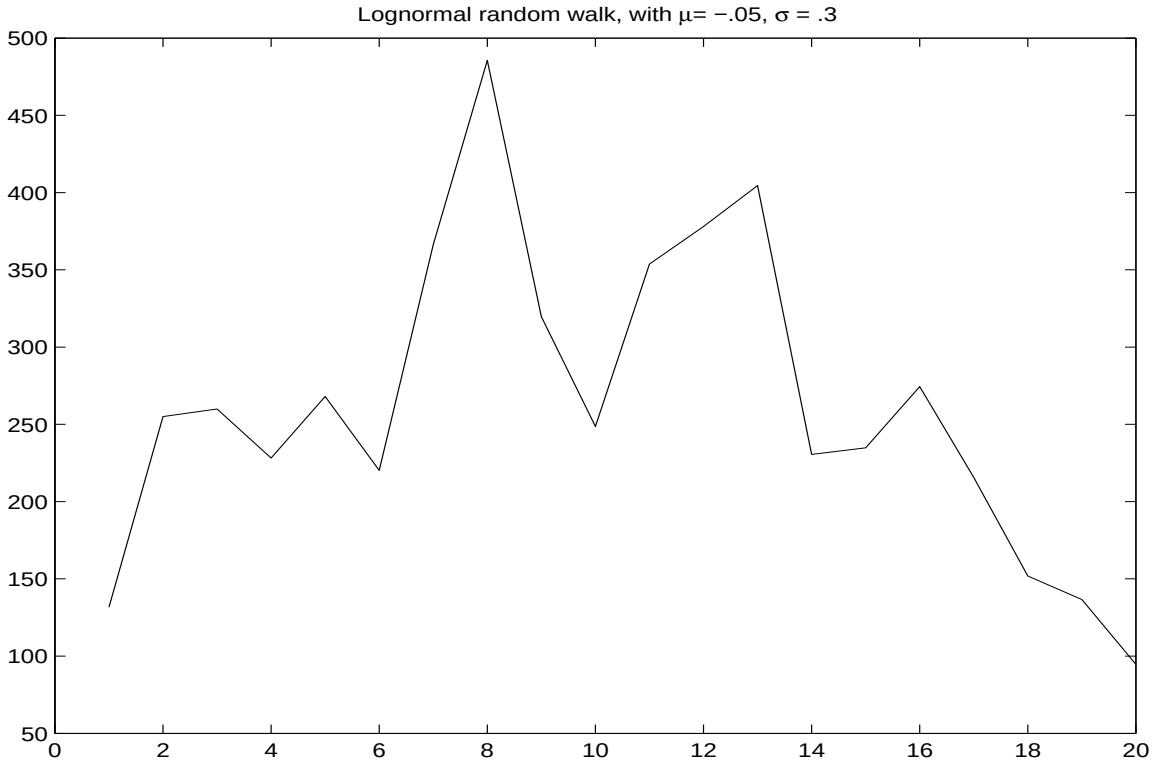
STOCK OPTION MODELS and VERIFICATION

Stock Option Model Simulation

- Assumptions:

  a) stock price at time n is (a *log normal random walk*): $S_n = S_0 e^{X_1+X_2+\cdots+X_n}$ with iid X_i's, $X_i \sim Normal(\mu, \sigma^2)$;



  b) option is to purchase at price K at some time $n \leq N$;

  c) if gain or profit is $S_n - K$, what is a good strategy for deciding when to purchase?

STOCK OPTION MODEL CONTINUED

d) optimal strategy when $\alpha = \mu + \sigma^2/2 \geq 0$: wait until N; expected profit is $E[max(S_N - K, 0)]$, easy to simulate; need Normal RV's for $X_i's$.

```
S0 = 100; mu = .005; s = .1; K = 105; N = 12; I = 10000;
S = S0*exp( sum( mu + s*randn(N,I) ) ); X = max(S-K,0);
disp([mean(X) 2*std(X)/sqrt(I)])
       19.079       0.61147
```

Note: this can be written as an integral.

e) if $\alpha < 0$, a "reasonably good" strategy is, letting $P_m = S_{N-m}$,
to exercise the option at time $N - m$ if $P_m > K$ and, for $i = 1, \ldots, m$,

$$P_m > K + P_m e^{i\alpha}\Phi(\sigma\sqrt{i} + b_i) - K\Phi(b_i),$$

with $b_i = \frac{i\mu - \ln(K/P_m)}{\sigma\sqrt{i}}$ and $\Phi(x) = \frac{1}{\sqrt{2\pi}}\int_{-\infty}^{x} e^{-t^2/2}dt$;
the option is exercised if $P_m - K >$ expected later values;

the goal of simulation is to determine expected profit $E[max(S_{N-m} - K, 0)]$.

STOCK OPTION MODELS CONTINUED

- Stock Option Model Simulation Algorithm:
  Input: μ, σ, K, N, S_0, $\alpha = \mu + \sigma^2/2$;
  Initialize: $m = N$, $P = S_0$, $I = 0$;
  While $I = 0$ and $m > 0$, update P using:

  $$\text{compute } V = max_{i=1}^{m} \left(K(1 - \Phi(b_i)) + Pe^{i\alpha}\Phi(\sigma\sqrt{i} + b_i) \right)$$

  if $P < max(K, V)$, generate $X \sim Normal(\mu, \sigma^2)$,
  and reset $P = Pe^X$, $m = m - 1$,
  otherwise set $I = 1$;
  EndWhile
  Output If $I = 1$, output $P - K$, otherwise output 0.
- Repeated runs provide $E[P - K]$;
  other data could be collected, e.g. average $E[N - m]$.
- Implementation needs:
  a) Software for Normal cdf, e.g. text algorithm or Matlab "normcdf";
  b) Software for $Normal(\mu, \sigma^2)$ RVs, e.g.
  from text $Normal(0, 1)$ algorithm or Matlab "randn".

STOCK OPTION MODELS CONTINUED

Matlab

```
function [ PK Nm ] = option( mu, sig, N, K, S0 )
% Stock Option Simulation
%   Example: option( -.04, .25, 20, 100, 100 )
  m = N; P = S0; I = 0; al = mu + sig^2/2;
  while I == 0 & m > 0
    sgi = sig*sqrt([1:m]); b = ( mu*[1:m] - log(K/P) )./sgi;
    V = max( K*(1-normcdf(b)) + P*exp(al*[1:m] ).*normcdf(sgi+b));
    if P < max(K,V); X = mu + sig*randn; P = P*exp(X); m = m - 1;
    else, I = 1;
    end
  end,        PK = max( P - K, 0); Nm = N - m;
%  end option
```

- Example: $\mu = -.04$, $\sigma = .25$, $N = 20$, $S_0 = K = 100$.

```
for i = 1 : 1000
  [X(i) NM(i)] = option(-.04,.25,20,100,100);
end,
disp([mean(X) 2*std(X)/sqrt(1000) mean(NM)])
   32.955       1.0282       16.638
```

STOCK OPTION MODELS CONTINUED

- Another option model: Asian option has

$$S_n = S_{n-1} e^{(r-\frac{\sigma^2}{2})\delta+\sigma\sqrt{\delta}X_n},$$

with $\delta = T/N$, $X_n \sim Normal(0,1)$ and with profit

$$e^{-rT} \max(\frac{1}{N+1}\sum_{i=0}^{N} S_i - K, 0).$$

Also has integral representation.

VERIFICATION and TESTING

Verification and Testing for Simulation Programs

- Standard Program Debugging:
  a) initially develop programs with good structure,
  including use of functions and some comments;
  b) display of intermediate results to find bugs, check for consistency.
- Testing Principles:
  a) test individual modules or functions separately;
  b) test special input cases for known results, if possible;
  c) use theoretical results to verify approximate simulations;
  d) use detailed trace results to verfiy simulation accuracy.

VERIFICATION and TESTING EXAMPLE

Single-Server Queue Example

```
function [A D Tp] = snglsv( T, lam )
% Returns arrival times A, departure times D, and overtime Tp
%  Single-Server Q Simulation
  t = 0; na = 0; nd = 0; n = 0; ta = -log(rand)/lam; td = inf;
  while ta <= T % time left for more arrivals
    if ta <= td, t = ta; n = n + 1; % new arrival
      na = na + 1; A(na) = t; ta = t - log(rand)/lam;
      if n == 1, td = t + G; end,
    else, t = td; n = n - 1; % departure
      nd = nd + 1; D(nd) = t;
      if n >  0, td = t + G; else, td = inf; end
    end
  end % no more arrivals, empty the Q
  while n > 0, t = td; nd = nd + 1; D(nd) = t;
    n = n - 1; td = t + G;
  end,        Tp = max(t-T,0);
% end snglsv
function Y = G;  % Uniform between .2 and .3
   Y =  .2 + .1*rand;
% end G = uniform(.2,.3)
```

SS QUEUE EXAMPLE CONT

Sample Output: with code modified for
fixed interarrival time = 1/lam, and service time = .25:

```
[A,D,OT] = snglsvt(8,3); disp([A; D]); disp(OT)
 0.33333  0.66667     1      1.3333   1.6667    2
 0.58333  0.91667     1.25   1.5833   1.9167    2.25
     ... 24 total arrivals ...
  6.3333  6.6667      7      7.3333   7.6667    8
  6.5833   6.9167     7.25   7.5833   7.9167    8.25
0.25
```

Sample Output with actual function:

```
[A,D,OT] = snglsv(8,3); disp([A; D]); disp(OT)
0.42035  0.65946  0.87783  0.91594  1.0987
0.66493  0.91209  1.1605   1.3804   1.6201

... 27 total arrivals ...

6.3914   6.4858    6.608   7.9071  7.9328
6.643    6.9398    7.1588  8.1947  8.4424
0.44236
```

Sample Output with multiple runs:

```
clear
for K = [100 1000]
  for i=1:K, [A,D,OT(i)] = snglsv(8,3);
     T(i) = mean(D-A); L(i) = length(D);
  end
  disp([K mean(T), mean(OT) mean(L)])
end
 100      0.5112       0.34443        24.28
1000      0.51974      0.3559         24.014
clear
for K = [100 1000]
  for i=1:K, [A,D,OT(i)] = snglsv(8,4); % Change lam
     T(i) = mean(D-A); L(i) = length(D);
  end
  disp([K mean(T), mean(OT) mean(L)])
end
 100      0.89043       1.0463        31.57
1000      0.88592       1.0604        31.885
```