EE 6383 – Nonlinear Control Systems

Topic 5: Digital Simulation of Stochastic Nonlinear Systems: the Monte Carlo Method

Dr. James H. Taylor
Department of Electrical and Computer Engineering
University of New Brunswick
e-mail: jtaylor@unb.ca
website: www.ece.unb.ca/jtaylor

21 May 2019

Topic Outline

- Motivation
- Stochastic Systems: Basic Concepts
- Performance Measures (Statistics)
- Monte Carlo Analysis (Sample Statistics)
- Confidence Bounds
- Role of Assumed Probability Density
- Example

References:

- Gelb, A. (Ed.), *Applied Optimal Estimation,* MIT Press, 1974.
- Taylor, J. H., "Statistical Performance Analysis of Nonlinear Stochastic Systems by the Monte Carlo Method", *Mathematics and Computers in Simulation,* Vol. 23, Jan. 1981.

Motivation

- Many systems must deal with random inputs (e.g., a radar pointing and tracking system for randomly-moving targets).
- Many systems are perturbed by random effects:
  - Random initial conditions (e.g., launch conditions)
  - Random disturbances (e.g., wind gusts acting on an aircraft)
  - Random measurement noise (e.g., radar scintillation)

  Examples: trains (random rail irregularities); vehicles of all kinds; manufacturing processes and systems; biological processes and ecosystems; . . . randomness is /bf everywhere

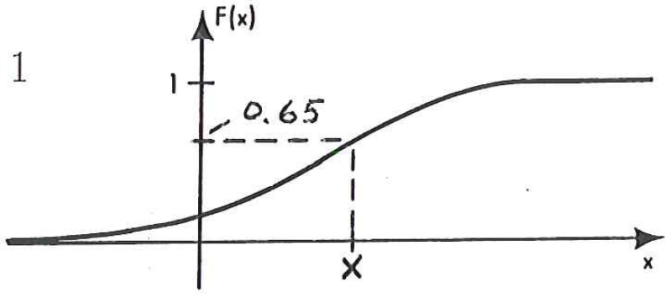
Such systems must be simulated, analyzed, and designed, taking the statistical nature of their behaviour into account. Monte Carlo simulation is a very valuable technique in the nonlinear case.

However, note that standard **covariance analysis** gives an **exact** solution for the statistical nature of the behaviour of a **linear** system with random inputs; this will be presented in another module.

Random Variables: Basic Properties

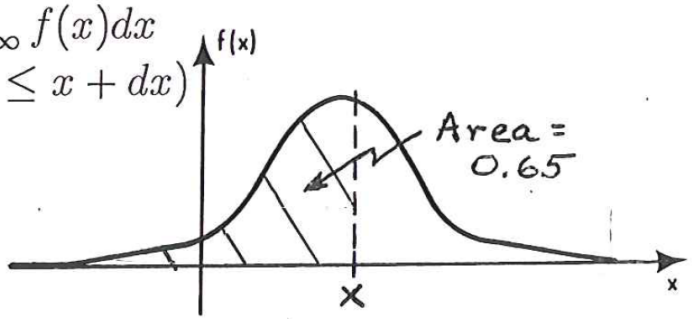
- **Random Variable** - a variable (denoted x) which takes on a random value (denoted X); an outcome of some random experiment(s).
  - **Probability Distribution Function:**

$F(x) = \text{Prob}\,(X \leq x)$
$F(-\infty) = 0\,; \;\; F(\infty) = 1$



  - **Probability Density Function:**

$f(x) = \frac{dF}{dx}\,; \;\; F(x) = \int_{-\infty}^{x} f(x)dx$
$f(x)dx = \text{Prob}\,(x < X \leq x + dx)$



- **Joint Functions:**

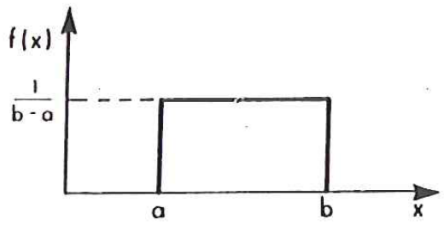
$$F(x,\, y) \;=\; \text{Prob}\,(X \leq x \text{ and } Y \leq y)$$
$$f(x,\, y) \;=\; \frac{\partial^2 F}{\partial x \partial y}$$

Note: $F_x(x) = F(x,\, \infty)$ and $f_x(x) = \int_{-\infty}^{\infty} f(x,\, y)\, dy$; if x and y are independent then $f(x,\, y) = f_x(x) f_y(y)$ etc.

Typical Probability Densities

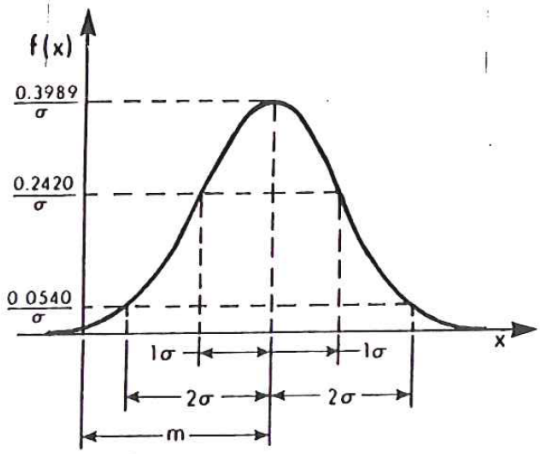
- **Uniform Density Function**

$$f(x) = \begin{cases} \frac{1}{b-a}, & a \le x < b \\ 0, & \text{elsewhere} \end{cases}$$



- **Normal Density Function**

$$f(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left[-\frac{(x-m)^2}{2\sigma^2}\right]$$



- **Multivariate Normal Density Function**

$$f(x_1, x_2, \ldots) = \frac{1}{\sqrt{(2\pi)^n |P|}} \exp\left[-\tfrac{1}{2}(x-m)^T P^{-1}(x-m)\right]$$

where $m = \mathrm{E}[\,x\,]$ (mean vector)
and $P = \mathrm{E}\,[\,(x-m)\,(x-m)^T\,]$ (covariance matrix)

Expected Values

General definitions:

- **Ensemble Average**

$$E[g(x)] = \int_{-\infty}^{\infty} g(x) f(x) dx \tag{1}$$

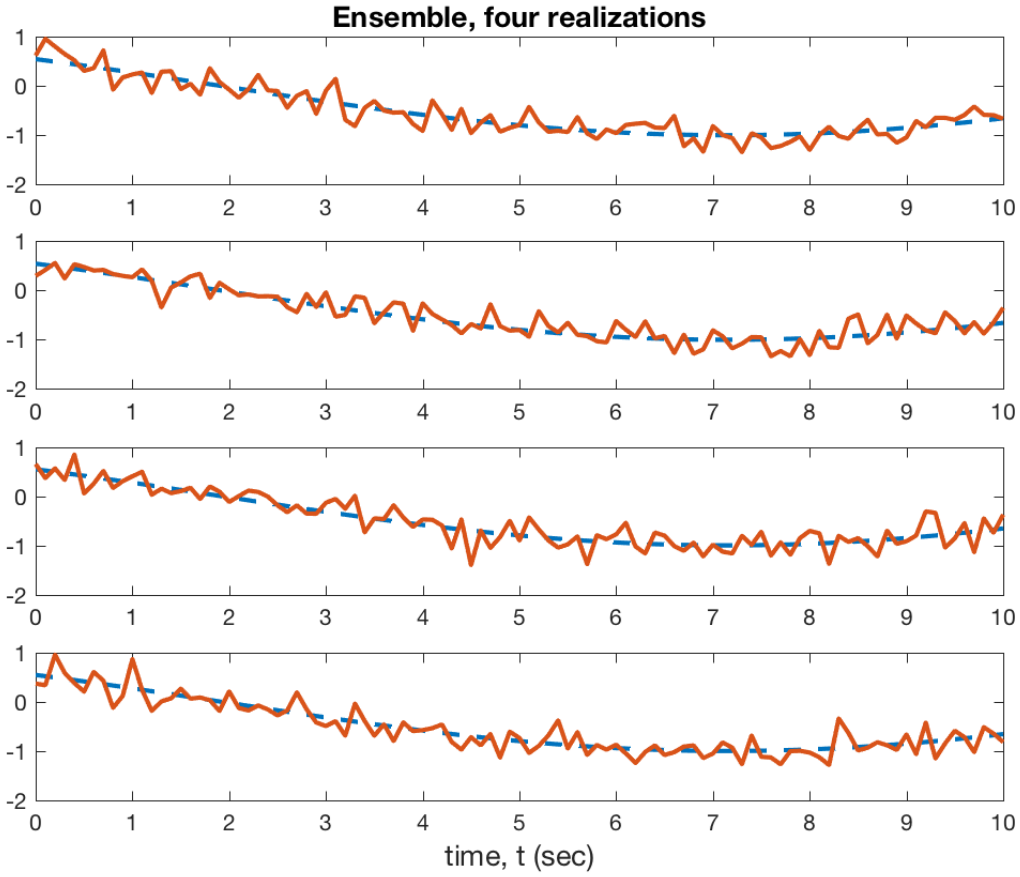
- **Time Average**

$$E[g(x)] = \lim_{t \to \infty} \frac{1}{2T} \int_{-T}^{T} g(x(t)) dt \tag{2}$$

- The form of $g(x)$ is arbitrary; standard cases are: $g(x) = x \rightarrow$ mean of x, $g(x) = x^2 \rightarrow$ mean-squared value of x, etc.
- Ensemble and time averages are the same for *stationary ergodic processes*; as an obvious counterexample, a random bias is not ergodic.
- In this module we will focus on generating performance statistics by **ensemble averaging** a substantial number of instances of trial scenarios, usually of **nonstationary random processes**.

Random Processes

Members of an Ensemble {x(t)}



An **ensemble** is a set of experiments / trials where the deterministic effects are the same but the random processes (initial conditions or random inputs) differ in each trial.

Basic Statistical Measures

- Mean or expected value of x: $m = E[x]$
- Mean-squared value of x: $E[x^2]$
- Root-mean-square (RMS) value of x: $\sqrt{E[x^2]}$
- Variance of x: $p = E[(x-m)^2]$
- Standard deviation of x: $\sigma = \sqrt{E[x-m)^2]}$
- Correlation coefficient of x and y:

$$\rho = \frac{E[xy] - E[x]E[y]}{\sigma_x \sigma_y}$$

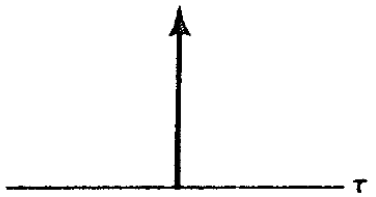
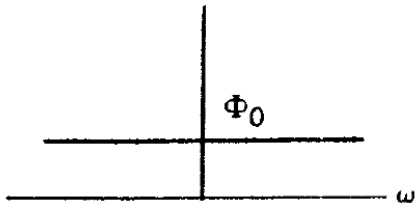
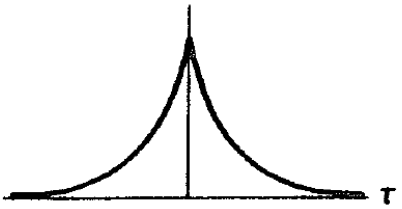
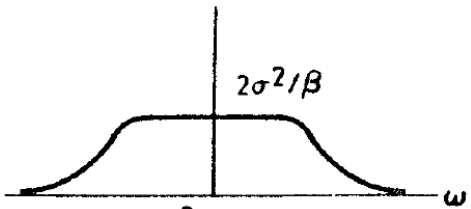
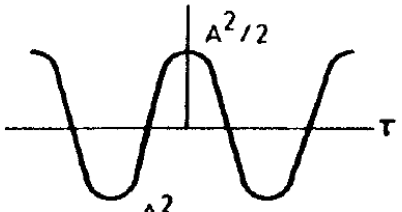
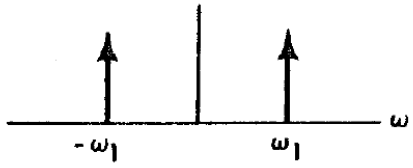
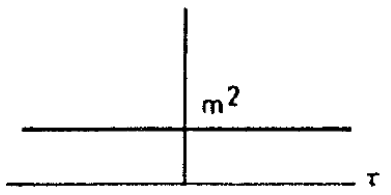
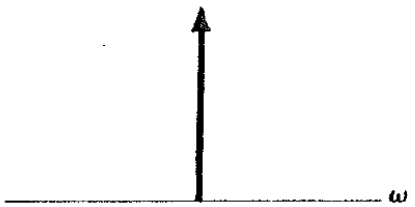
  for independent random variables, $\rho = 0$; if $x = \alpha y$, $\rho = 1$
- Autocorrelation of x: $\phi_{xx}(\tau) = E[x(t)x(t-\tau]$
- In general,

$$E[x_1 + x_2 + \ldots + x_n] = E[x_1] + E[x_2] + \ldots + E[x_n]$$

- For *independent* random variables,

$$E[x_1 x_2 \ldots x_n] = E[x_1]E[x_2] \ldots E[x_n]$$

Common Random Processes

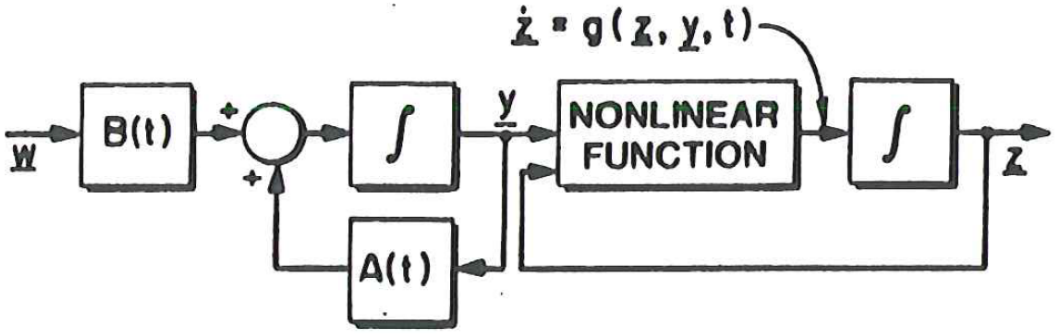
PROCESS	AUTOCORRELATION FUNCTION	POWER SPECTRAL DENSITY
WHITE NOISE	 $\phi_{xx}(\tau) = \phi_0 \delta(\tau)$	 $\Phi_{xx} = \Phi_0$
MARKOV PROCESS	 $\phi_{xx}(\tau) = \sigma^2 e^{-\beta\lvert\tau\rvert}$	 $\Phi_{xx} = \frac{2\beta\sigma^2}{\omega^2+\beta^2}$
SINUSOID	 $\phi_{xx}(\tau) = \frac{A^2}{2}\cos\omega_1\tau$	 $\Phi_{xx} = \frac{\pi}{2}A^2\left[\delta(\omega-\omega_1)+\delta(\omega+\omega_1)\right]$
RANDOM BIAS	 $\phi_{xx}(\tau) = m^2$	 $\Phi_{xx} = 2\pi m^2 \delta(\omega)$

System Models

- General form:

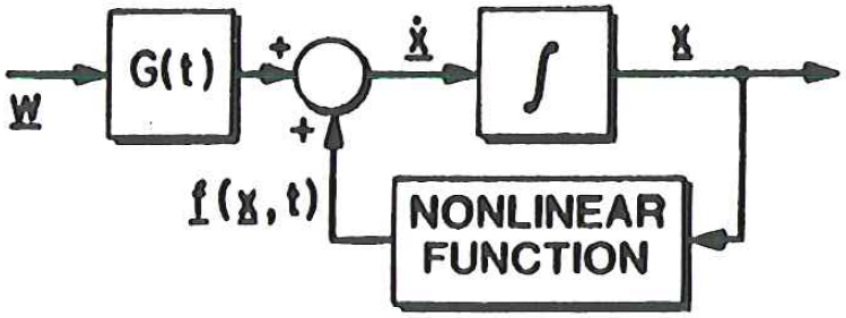
$$\begin{aligned} \dot{z} &= g(z, y, t) \\ y &= \text{random input} \end{aligned}$$

where y **cannot** be white noise; if obtained as shown below, y is first-order Markov (FOM) if w is white noise.



- Equivalent form if y is a FOM:

$$\dot{x} = f(x, t) + G(t)w \tag{3}$$



where equivalence is established via

$$x = \begin{bmatrix} z \\ - - - \\ y \end{bmatrix} \; ; \; f = \begin{bmatrix} g \\ - - - \\ A(t)y \end{bmatrix} \; ; \; G = \begin{bmatrix} O \\ - - - \\ B(t) \end{bmatrix}$$

System Models (Cont'd)

Given Eqn. (3):

$$\dot{x} = f(x,t) + G(t)w$$

we need a **statistical characterization** of the inputs and initial conditions:

- Random input characterization:

$$\begin{aligned} b(t) &= E[w(t)] \\ u &= w - b \\ Q(t)\delta(t-\tau) &= E[(u(t)u^T(\tau)] \end{aligned} \tag{4}$$

- Initial condition statistics:

$$\begin{aligned} m_0 &= E[x(0)] \\ P_0 &= E[(x(0) - m_0)\,(x(0) - m_0)^T] \end{aligned} \tag{5}$$

Each specification contains a **mean** and a **random** component.

Simulating "White Noise"

To simulate $u(t)$ as a **white noise with spectral density** $\mathbf{Q}(t)$ we use a standard random number generator[1] to create an independent sequence of random vectors $u(kh),\ k = 0,\ 1,\ 2,\ \ldots$ satisfying

$$E[u(kh)] = 0\,,\quad E[u(kh)u^T(kh)] = \frac{1}{h}\,Q(kh) \tag{6}$$

Then, define $u(t)$ by

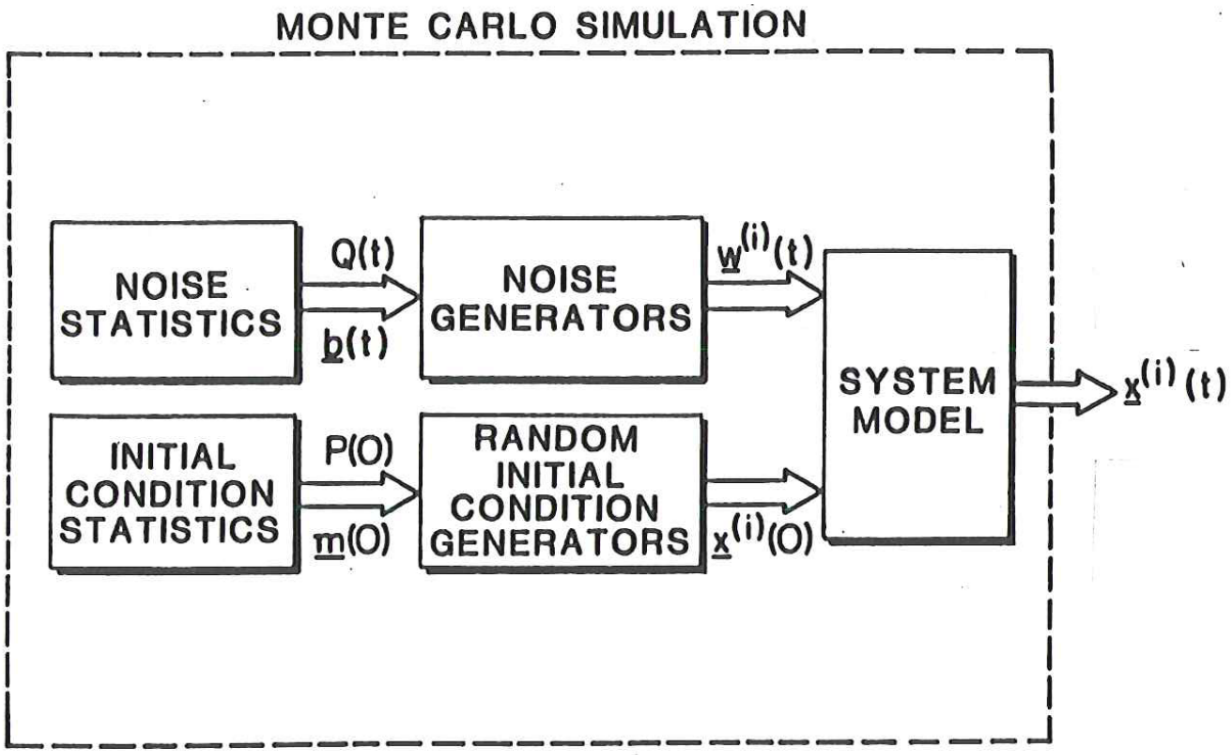
$$u(t) = u(kh)\,,\quad kh \le t < (k+1)h,\ k = 0,\ 1,\ 2,\ \ldots \tag{7}$$

where h is a small time increment ($1/h >>$ system bandwidth).

Digital simulation of a stochastic system with simulated "white noise" is no different from the deterministic case, **except the state derivatives are always discontinuous,** so must never use higher-order predictor/corrrector methods or a variable-step runge-kutta method. In accordance with Eqn. (6) and the $1/h >>$ system bandwidth constraint, **you must use a fixed-step algorithm with step** h. Since $1/h >>$ system bandwidth a **first-order Euler algorithm will generally perform well.**

[1]Random number generators **do not (cannot)** produce truly random numbers; hence these numbers are called **pseudorandom**.

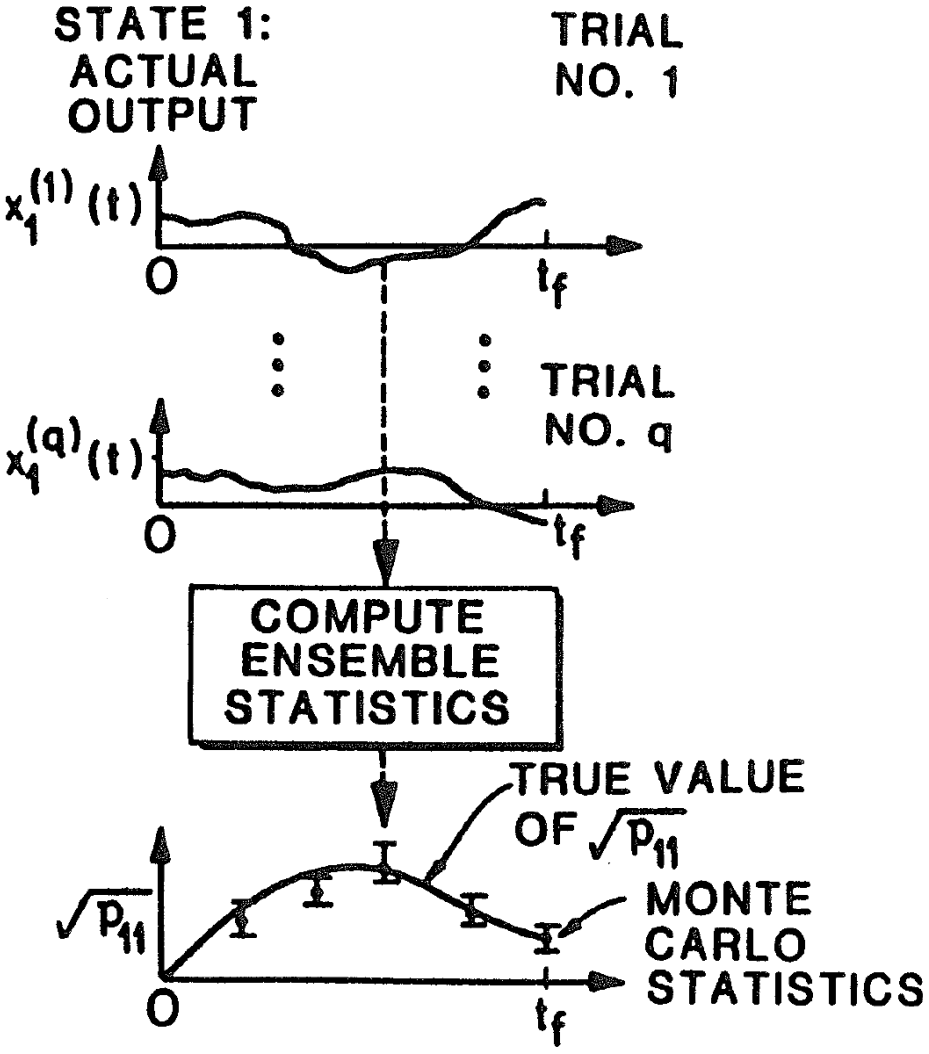
The Essence of the Monte Carlo Method



(a) RANDOM ENSEMBLE GENERATOR

The solutions $x^{(i)(t_k)}$ are saved for post-processing to determine the ensemble statistics.

The Essence of the Monte Carlo Method (Cont'd)



(b) ENSEMBLE STATISTICAL ANALYSIS

As shown above, the sample mean and standard deviation ($\sqrt{(}p_{ii}$) are plotted versus time, with the **confidence bands** indicated by I-bars; more on this soon.

The Essence of the Monte Carlo Method (Cont'd)

The following listings and plots illustrate a very simple test case for generating simulation ensembles and Monte Carlo statistics:

- First, `eurand.m`, a fixed-step Euler integrator **extended for generating random inputs within the model**
- Then a simple **tracking system model**
- Then a simple script for running q trial runs to create an ensemble
- Finally, a script that gathers statistics and plots them

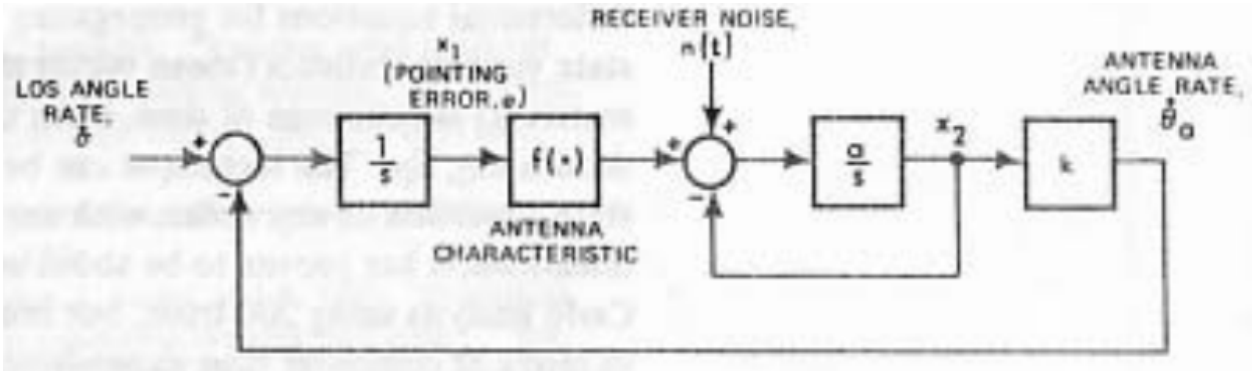
Fixed-step Integrator for the Monte Carlo Method

```
function[tout,yout] = eurand(dyfun,t0,tfinal,y0,step,Q,seed,trace)
%EURAND Solve ODEs, low order method, for stochastic processes.
% EURAND integrates a system of ordinary differential
% equations using the most elementary Euler algorithm.
%
% INPUT:
% dyfun - String containing name of user-supplied model.
%         Call: ydot = fun(t,y) where dyfun = 'fun'.
%         t     - Time (scalar).
%         y     - Solution column-vector.
%         ydot  - Returned derivative; ydot(i) = dy(i)/dt.
% t0    - Initial value of t.
% tfinal- Final value of t.
% y0    - Initial value column-vector.
%    step  - Integration step. (Default: step = 1.e-2).
% q     - white noise spectral density
% seed  - seed to start 'randn' noise generator
% trace - If nonzero, each step is printed. (Default: 0).
%
% OUTPUT:
% T  - Returned integration time points (column-vector).
% Y  - Returned solution, one col-vector per tout-value.
%
% The result can be displayed by: plot(tout, yout).
%% %%%%%%% Initialization
if nargin < 5, step = 1.e-2; end
if nargin < 6, Q = 1.0; end
if nargin < 7, seed = 0; end
if nargin < 8, trace = 0; end
t = t0;
h = step;
y = y0(:);
%% array sizes can be hard-wired for a fixed step:
chunk = ceil(tfinal/step);
```

```
if chunk > 1000, error('too many points!'); return; end
tout = zeros(chunk,1);
yout = zeros(chunk,length(y));
k = 1;
tout(k) = t;
yout(k,:) = y.';
if trace
   clc, t, h, y
end
%%%%%%% main integration loop
while (t < tfinal) & (t + h > t)
   if t + h > tfinal, h = tfinal - t; end
   % Compute the slope; provides for random number generator
   s1 = feval(dyfun, t, y, h, Q, seed); s1 = s1(:);
   % Update the solution (with no check on error)
   t = t + h;
   y = y + h*s1;
   k = k+1;
   tout(k) = t;
   yout(k,:) = y.';
   if trace
      home, t, h, y
   end
end
%%%%%%% error handling
if (t < tfinal)
   disp('Singularity likely.')
   t
end
tout = tout(1:k);
yout = yout(1:k,:);
```

Note: allowing `seed` to be specified permits running the same Monte Carlo trials again (with the **same** random inputs)

Tracker Model




```
function xdot = tracker(t,x,h,Q,seed)
%
% Tracker nonlinear model; Trans AES paper of March 1978
%
% JH Taylor, 15 February 1995
%
A = 50.0;       % servo lag
K = 10.0;       % servo gain
Kant = 0.4;     % antenna cubic term coeff.
global Omega
%% set in macro  Omega = 6.0;   % nominal target angle rate
% random input generator:
noise = sqrt(Q/h)*randn;
xdot(1) = Omega - K*x(2);
xdot(2) = A*(noise + x(1) - Kant*x(1)*x(1)*x(1) - x(2));
```

Note - we consider tracker performance for two LOS angular rates,
4 and 6 deg/sec

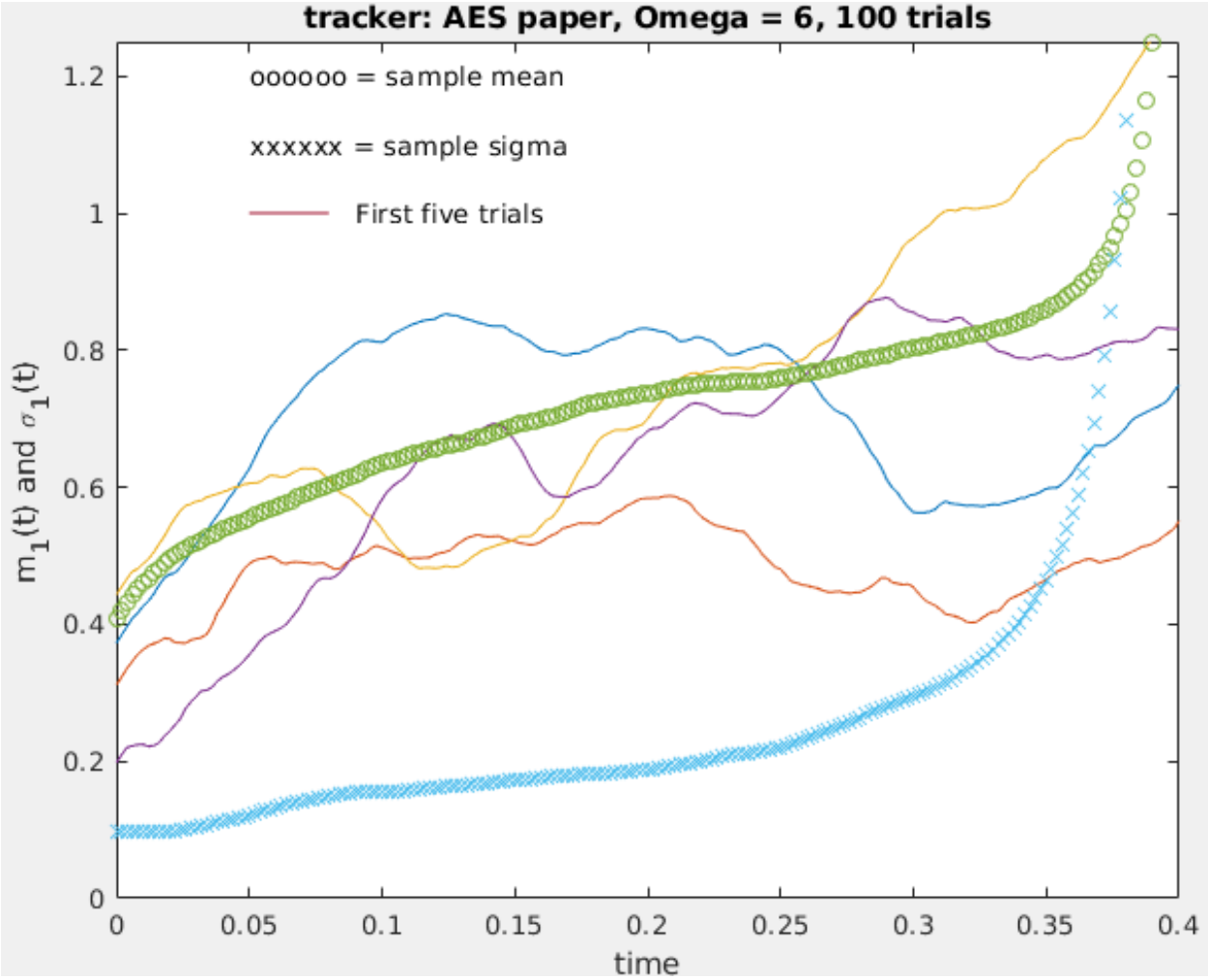
Run the Monte Carlo Trials

```
%  generate a Monte Carlo ensemble, gather and process statis
%  also save data for construction of a histogram
%
%  100 Monte Carlo Trials; step = dt = 0.002 sec probably equ
%
%  JH Taylor - 11 Feb 1995
%
%%  Initialize problem:
clear; close all;
global Omega
Omega = 6;
t0 = 0.0; tf = 0.3; x0 = [ 0.4 ; 0.0 ]; dt = 0.002;
m0 = 0.4; sig0=0.1; Q = 0.004; trace = 0;
Ntrial = 100;
%%  Allocate arrays:
Nsteps = 1+ceil(tf/dt);
Nstates = length(x0);
sum = zeros(Nsteps,Nstates);
sumsq = zeros(Nsteps,Nstates);
msamp = zeros(Nsteps,Nstates);
vsamp = zeros(Nsteps,Nstates);
sigma = zeros(Nsteps,Nstates);
%% Also gather stats for histogram at t = 0.25
x1p25 = zeros(Ntrial,1);
sumx1p25 = 0;
sumsqx1p25 = 0;
%%  Monte Carlo trial loop:
for j = 1:Ntrial
```

```
   x0 = [ m0 + sig0*randn ; 0.0 ]; % random IC for tracking e
   [t,x] = eurand('tracker_model',t0,tf,x0,dt,Q,0);  % 'seed'
   sum = sum + x;
   sumsq = sumsq + x.*x;
   %% catch value of x(0.25 sec) and do sums for statistics
   x1p25(j) = x(126,1); %% dt = 0.002 => t(126) = 0.25 sec
   sumx1p25 = sumx1p25 + x1p25(j);
   sumsqx1p25 = sumsqx1p25 + x1p25(j)^2;
   if j < 5
      plot(t,x(:,1)); % don't want to plot all states or all
   end
   if j == 1
      xlabel('time');
      ylabel('m_1(t) and \sigma_1(t)');
      title(['tracker: AES paper, Omega = ',num2str(Omega),',
      hold on
   end
   if trace
      j, randn('seed') % debug print
   end
end
%% end of trials -- process and plot statistics:
msamp = sum/Ntrial;
m1p25 = sumx1p25/Ntrial;
vsamp = sumsq/Ntrial - msamp.*msamp;
v1p25 = sumsqx1p25/Ntrial - m1p25^2;
debias = Ntrial/(Ntrial-1);
sigma = sqrt(debias*vsamp);
sig1p25 = sqrt(debias*v1p25);
%% Note: msamp and sigma may also be obtained using 'normfit'
plot(t,msamp(:,1),'o');
```

```
plot(t,sigma(:,1),'x');
axis([0 tf 0 1.25])
text(0.05,1.20,'oooooo = sample mean')
text(0.05,1.10,'xxxxxx = sample sigma')
plot([0.05 0.08],[1.0 1.0]), text(0.09,1.0,'First five trials
save x1p25data x1p25 m1p25 sig1p25 %% save data for histogram
%% end of script tracker_MCSim.m
```

Running this last last script does all the work!



Note that the tracker cannot follow a target moving at 6 deg/sec, the antenna beamwidth is too small (it "loses sight" of the target)

Statistics of Estimated Statistics

- Given y, a single random variable, with **true statistics**

$$m = E[y]\,, \quad p = E[(y-m)^2] \tag{8}$$

  where the scalar y could be a system output at a particular time t_k

- We obtain $\hat{m}$, $\hat{p}$ by performing q Monte Carlo trials, i.e., simulating Eqn. (3) with random inputs q times;

$$\hat{m} = \frac{1}{q}\sum_{i=1}^{q} y_i\,, \quad \hat{p} = \frac{1}{q-1}\sum_{i=1}^{q}(y_i - \hat{m})^2 \tag{9}$$

- Thus, $\hat{m}$ and $\hat{p}$ are random variables: a different set of trials will produce different results. For large q, the random variables $\hat{m}$ and $\hat{p}$ are asymptotically Gaussian, with statistics given by

$$E[\hat{m}] = m \quad , \quad E[\hat{p}] = p \tag{10}$$

$$E[(\hat{m}-m)^2] \;=\; \sigma_{\hat{m}}^2 = \frac{p}{q} \tag{11}$$

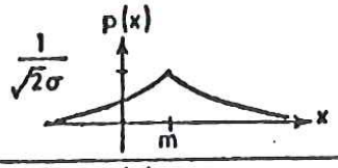
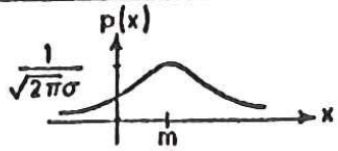
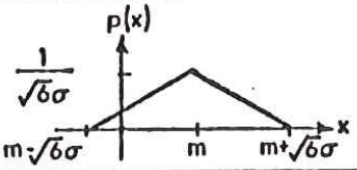
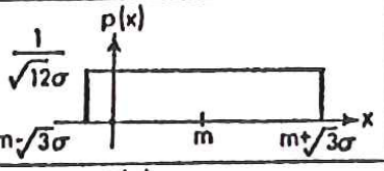
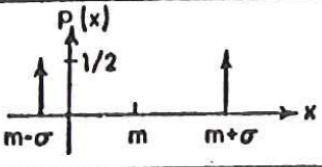
$$E[(\hat{p}-p)^2] \;=\; \sigma_{\hat{p}}^2 = \frac{\mu_4 - p^2}{q} \tag{12}$$

  where μ_4 is the fourth central moment; $\mu_4 = E[(y-m)^4]$.

We will use $\sigma_{\hat{m}}$ and $\sigma_{\hat{p}}$ to judge the **confidence** we have in the accuracy of our estimates of $\hat{m}$ and $\hat{p}$; this will produce **confidence limits** which can be plotted with $\hat{m}$ and $\hat{p}$ to portray that confidence.

Kurtosis

So, we need to consider the fourth central moment of y in order to judge our confidence in $\hat{p}$. For many commom probability density functions[2] $\mu_4 = \lambda p^2$; λ is called the distribution's **kurtosis**.

DESIGNATION	FUNCTIONAL REPRESENTATION*	GRAPHICAL REPRESENTATION	λ
EXPONENTIAL	$\frac{1}{\sqrt{2}\sigma}\ \exp\left(-\frac{\sqrt{2}}{\sigma}\lvert x-m\rvert\right),$ $-\infty < x < +\infty$		6
NORMAL	$\frac{1}{\sqrt{2\pi}\sigma}\ \exp\left(-\frac{1}{2}\left(\frac{x-m}{\sigma}\right)^2\right),$ $-\infty < x < +\infty$		3
TRIANGULAR	$\frac{1}{\sqrt{6}\sigma}\left(1-\frac{\lvert x-m\rvert}{\sqrt{6}\sigma}\right),$ $m-\sqrt{6}\sigma \le x \le m+\sqrt{6}\sigma$		2.4
UNIFORM	$\frac{1}{\sqrt{12}\sigma},$ $m-\sqrt{3}\sigma \le x \le m+\sqrt{3}\sigma$		1.8
BIPOLAR (Discrete)	$\frac{1}{2}\delta(x-m-\sigma)$ $+\frac{1}{2}\delta(x-m+\sigma)$		1.0

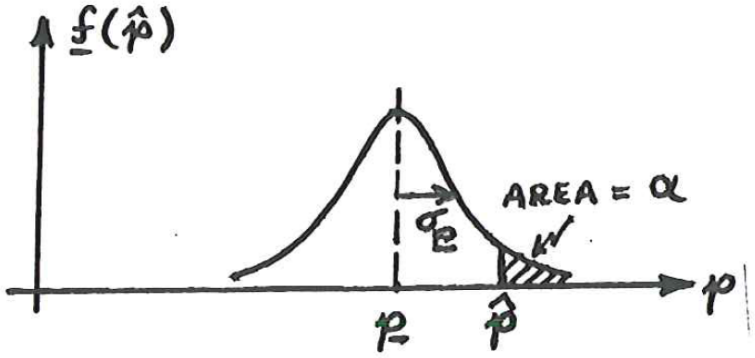
*Formulated to have mean m and standard deviation σ

We observe that kurtosis is, in a sense, a measure of the "tails" of the random variable's probability density function – the tails of the exponential distribution approach zero more slowly than the other cases, while the bipolar density function has no tails.

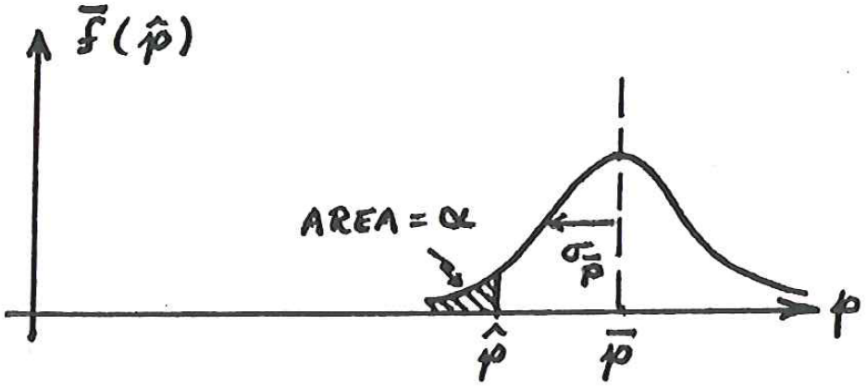
[2]Exceptions: binomial, poisson, cauchy, gamma distributions

Confidence Limits for Estimated Variance

- For $\hat{p}$ given, the lower confidence level $\underline{p}$ is defined by:
  Prob $[p < \underline{p} \mid \hat{p}] = \alpha$; α = small probability, e.g. 0.025



- For $\hat{p}$ given, the upper confidence limit $\overline{p}$ is defined by:
  Prob $[p > \overline{p} \mid \hat{p}] = \alpha$



- For $\alpha = 0.025$ we obtain 95 % confidence limits: Prob $[\underline{p} < p < \overline{p}] = 0.95$. Other confidence levels can be used, but 95 % is quite standard.

- Since $\hat{p}$ is asymptotically Gaussian, $\alpha = 0.025 \Rightarrow \hat{p} - \underline{p} = 1.96\sigma_{\underline{p}}$, and similarly for $\overline{p} - \hat{p}$.

Confidence Limits for Estimated Variance (Cont'd)

- Express $\hat{p} - \underline{p}$ and $\hat{p} - \overline{p}$ in terms of σ_p: since $\hat{p}$ is asymptotically normal, $\alpha = 0.025 \rightarrow$ we need $\hat{p} - \underline{p} = 1.96\,\sigma_p$

$$\hat{p} - \underline{p} = 1.96\sqrt{\frac{\lambda - 1}{q}}\;\underline{p} \;\rightarrow\; \hat{p} = \underline{p}(1 + 1.96\sqrt{\frac{\lambda - 1}{q}}) \quad (13)$$

$$\overline{p} - \hat{p} = 1.96\sqrt{\frac{\lambda - 1}{q}}\;\overline{p} \;\rightarrow\; \hat{p} = \overline{p}(1 - 1.96\sqrt{\frac{\lambda - 1}{q}}) \quad (14)$$

- therefore

$$\underline{p} = \frac{\hat{p}}{1 + 1.96\sqrt{\frac{\lambda-1}{q}}}, \quad \overline{p} = \frac{\hat{p}}{1 - 1.96\sqrt{\frac{\lambda-1}{q}}}, \quad (15)$$

$$\Rightarrow \text{Prob}\,[\,\underline{p} < p < \overline{p}\,] = 1 - 2\alpha = 0.95 \quad (16)$$

- For other confidence levels (we say $\underline{p}$ and $\overline{p}$ defined above are the "95 percent confidence limits") we use different values for the number of standard deviations in $\hat{p} - \underline{p}$ and $\overline{p} - \hat{p}$:

Confidence	α	n_σ
90 %	0.05	1.645
95 %	0.025	1.960
99 %	0.005	2.576

Corresponding Confidence Limits for σ:

- We can express the confidence limits on $\hat{\sigma}$ equivalently,

$$\text{Prob}\left[\underline{\sigma} \triangleq \underline{\rho}\hat{\sigma} \leq \sigma \leq \overline{\rho}\hat{\sigma} \triangleq \overline{\sigma}\right] = 1 - 2\alpha \tag{17}$$

  where

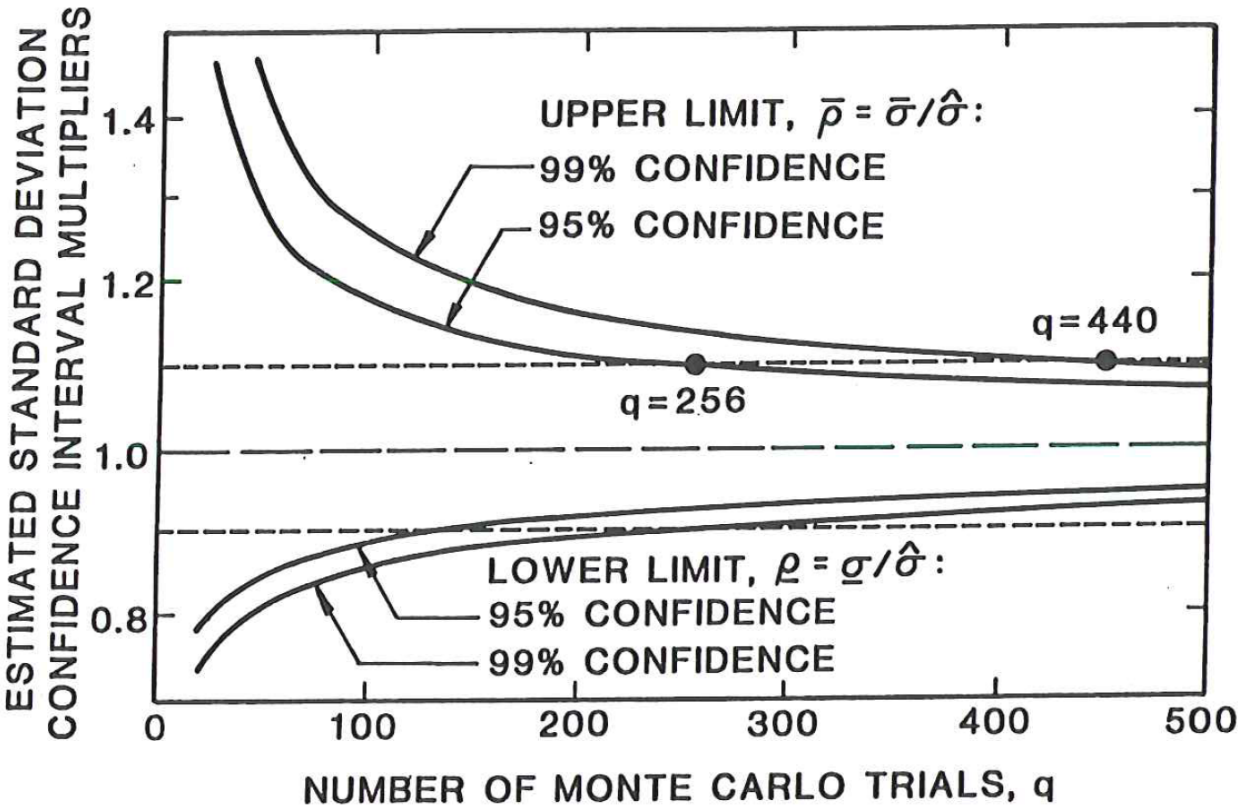
$$\begin{aligned} \underline{\rho} &= \sqrt{\frac{1}{1 + n_\sigma \sqrt{\frac{\lambda - 1}{q}}}} \\ \overline{\rho} &= \sqrt{\frac{1}{1 - n_\sigma \sqrt{\frac{\lambda - 1}{q}}}} \end{aligned} \tag{18}$$

- This is meaningful provided q is reasonably large; as a rule of thumb,

$$n_\sigma \sqrt{\frac{\lambda - 1}{q}} << 1 \tag{19}$$

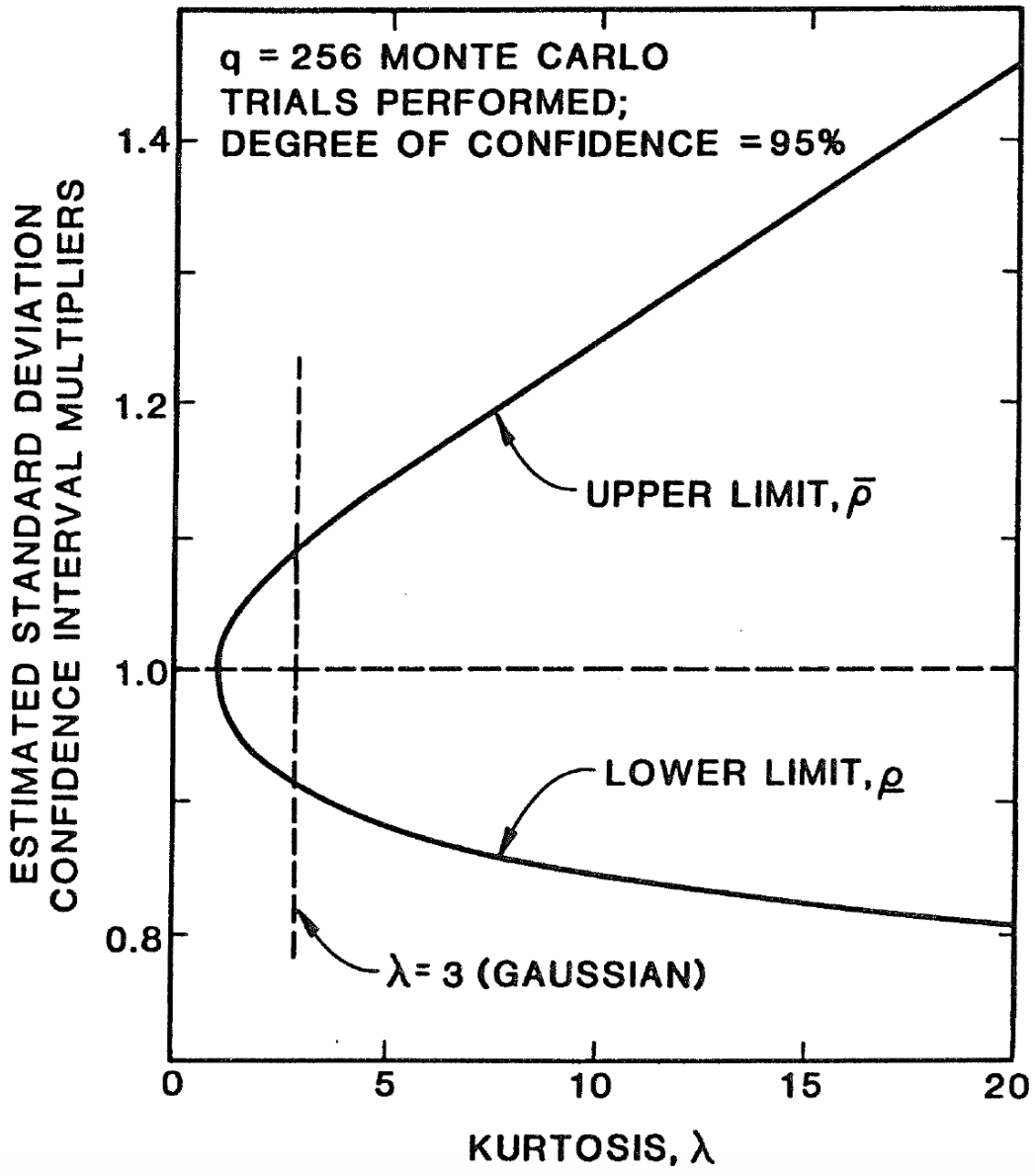
The quantities $\underline{\rho}$, $\overline{\rho}$ are called **confidence limit multipliers**, and are measures of the reliability of $\hat{\sigma}$ ($\underline{\rho}\,,\ \overline{\rho} \cong 1 \Rightarrow \hat{\sigma}$ is a good estimate of σ).

Typical Confidence Interval Multipliers for Estimated Standard Deviation of a Gaussian Random Variable ($\lambda = 3$)



Many monte carlo snalyses use 256 trials to achieve +10 % / -9.2 % limits for 95 % confidence on $\hat{\sigma}$; for 99 % confidence, 440 trials are required. But remember . . . this holds **only** if y is a gaussian random variable.

Effect of Kurtosis on Confidence Interval Limits



The width of the confidence band for $q = 256$ increases rapidly as kurtosis increases; for example, for a random variable with an exponential density function, $\lambda = 6$, the confidence limits are (0.886,1.174) rather than (0.92, 1.10) for the Gaussian case.

Confidence Limits for Estimated Mean

Fortunately, the situation is much simpler for the sample mean:

- We saw that $\sigma_{\hat{m}} = \sqrt{\frac{p}{q}}$; p is unknown, so we use the conservative value $\overline{p}$ instead: $\sigma_{\hat{m}} \cong \sqrt{\frac{\overline{p}}{q}}$
- Then directly, since $\sigma_{\hat{m}}$ is not a function of m,

$$\underline{m} \triangleq \hat{m} - n_\sigma \frac{\overline{\sigma}}{\sqrt{q}} \tag{20}$$

$$\overline{m} \triangleq \hat{m} + n_\sigma \frac{\overline{\sigma}}{\sqrt{q}} \tag{21}$$

$$\Rightarrow \text{Prob}\,[\,\underline{m} \leq m \leq \overline{m}\,] = 1 - 2\alpha \tag{22}$$

where α and n_σ values are as given before . . . easy, for once

Compensating Confidence Limit Tables For Variations in Kurtosis

Given $\hat{\sigma}$, obtained from q trials, for a random variable with kurtosis $\hat{\lambda}$ (known or estimated) different from $\lambda_{Gauss} = 3$:

- The confidence limit multipliers $\underline{\rho}$, $\overline{\rho}$ are equal for any pair (λ, q) such that $(\lambda - 1)/q = \text{constant}$ (see Eqn. (18))
- Thus the Equivalent Number of Trials required to estimate the σ of a Gaussian random variable with the same reliability (confidence limit multipliers) satisfies

$$\frac{\lambda_{Gauss} - 1}{q_{eq}} = \frac{\hat{\lambda} - 1}{q} \tag{23}$$

or, since $\lambda_{Gauss} = 3$,

$$q_{eq} = \frac{2q}{\hat{\lambda} - 1} \tag{24}$$

For example, if y is characterized by the exponential distribution, $\lambda = 6$, and 256 trials are performed then we only obtain a confidence band comparable to $q_{eq} = 102$ trials for a Gaussian random variable. Conversely, if we want a confidence band comparable to that of a Gaussian random variable we must perform 640 trials.

Examples of the Significance of Kurtosis

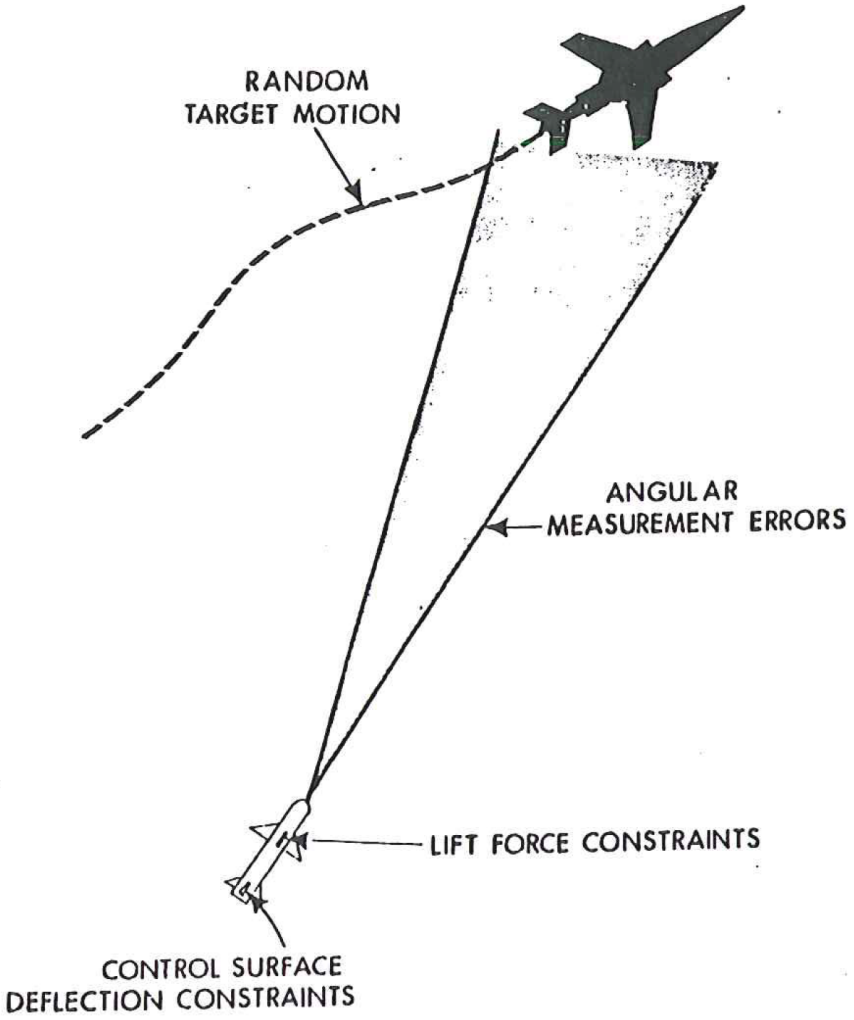
- **Small kurtosis:** Given a true coin, typified by the bipolar density with $m = 0$, $\sigma =$ face value, we noted before that $\lambda = 1$. Observe that $\underline{\rho} = \overline{\rho} = 1$ for any q! One toss is sufficient for perfect knowledge of σ (25 cents, say).

- **Large kurtosis:** Miss distance in the missile-target intercept problem has been observed to be quite non-Gaussian. An extensive study (500 trials) of a relatively benign case has led to $\hat{\lambda} \cong 15$. Thus σ_{miss} obtained in this study is as reliable as the estimated σ of a Gaussian random variable obtained with

$$q_{eq} = \frac{2(500)}{15 - 1} \cong 70 \text{ trials} \tag{25}$$

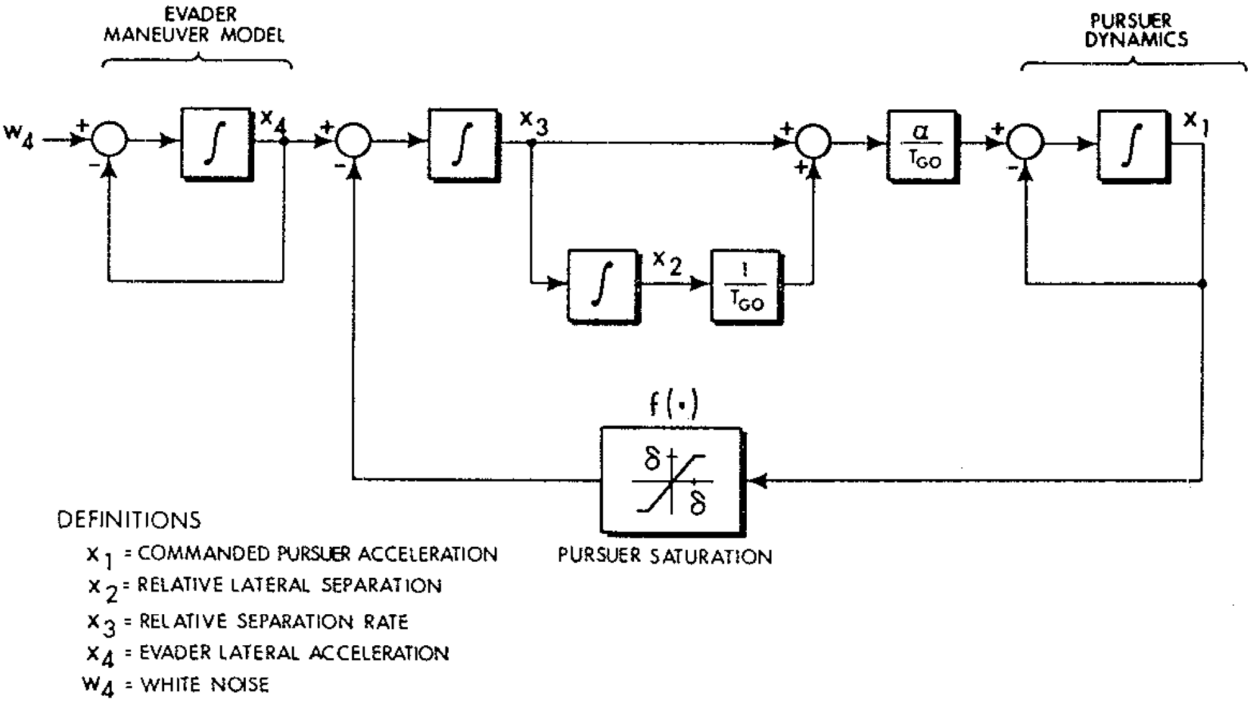
Case Study: The Estimation of RMS Lateral Separation in the Missile-Target Intercept Problem

A. The Scenario:



Missile-Target Intercept Problem (Cont'd)

B. Block Diagram and State-space Model

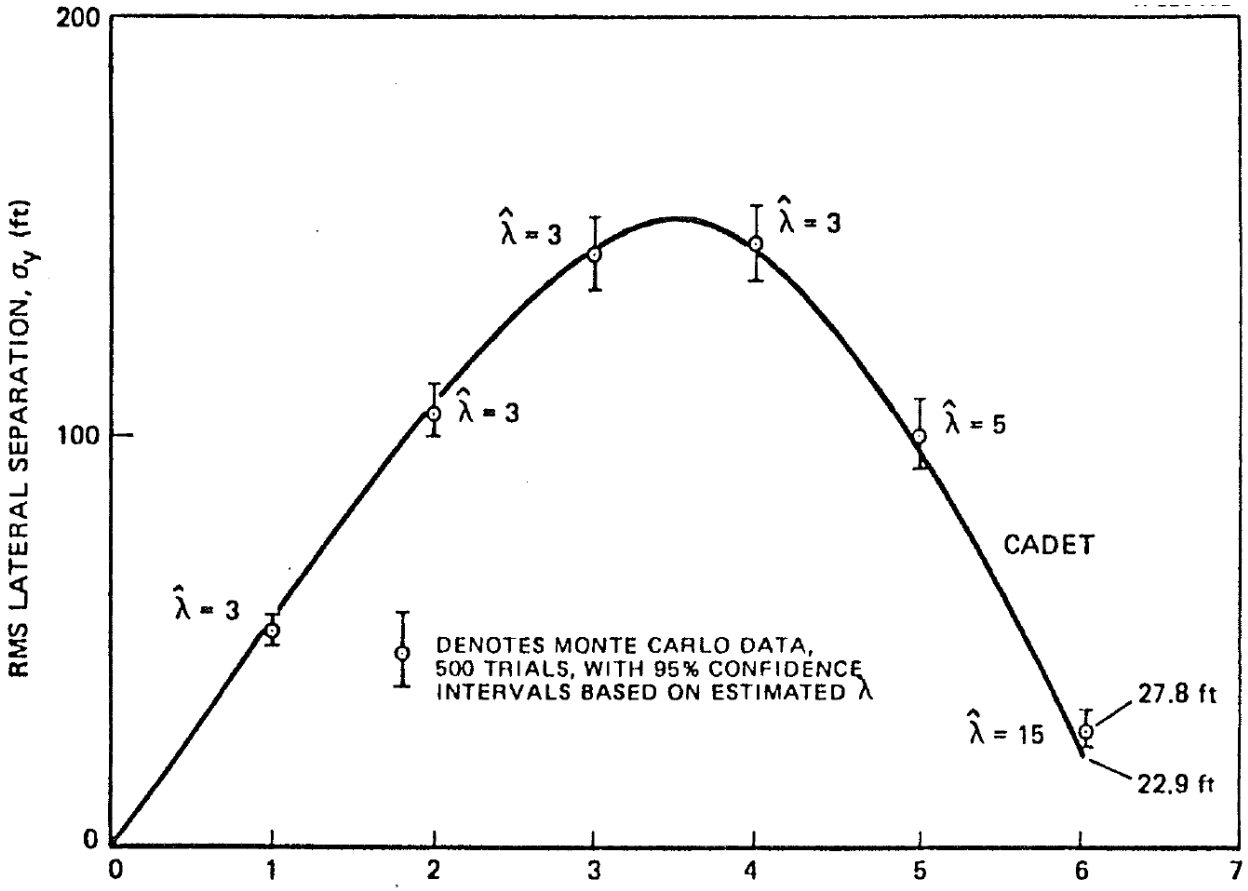


The control algorithm is called "proportional navigation", i.e., the lateral acceleration guidance command is proportional to the line-of-sight angulaar rate;

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \\ \dot{x}_3 \\ \dot{x}_4 \end{bmatrix} = \begin{bmatrix} -1 & \frac{\alpha}{T_{GO}^2} & \frac{\alpha}{T_{GO}} & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ -f(x_1) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ w_4 \end{bmatrix}$$

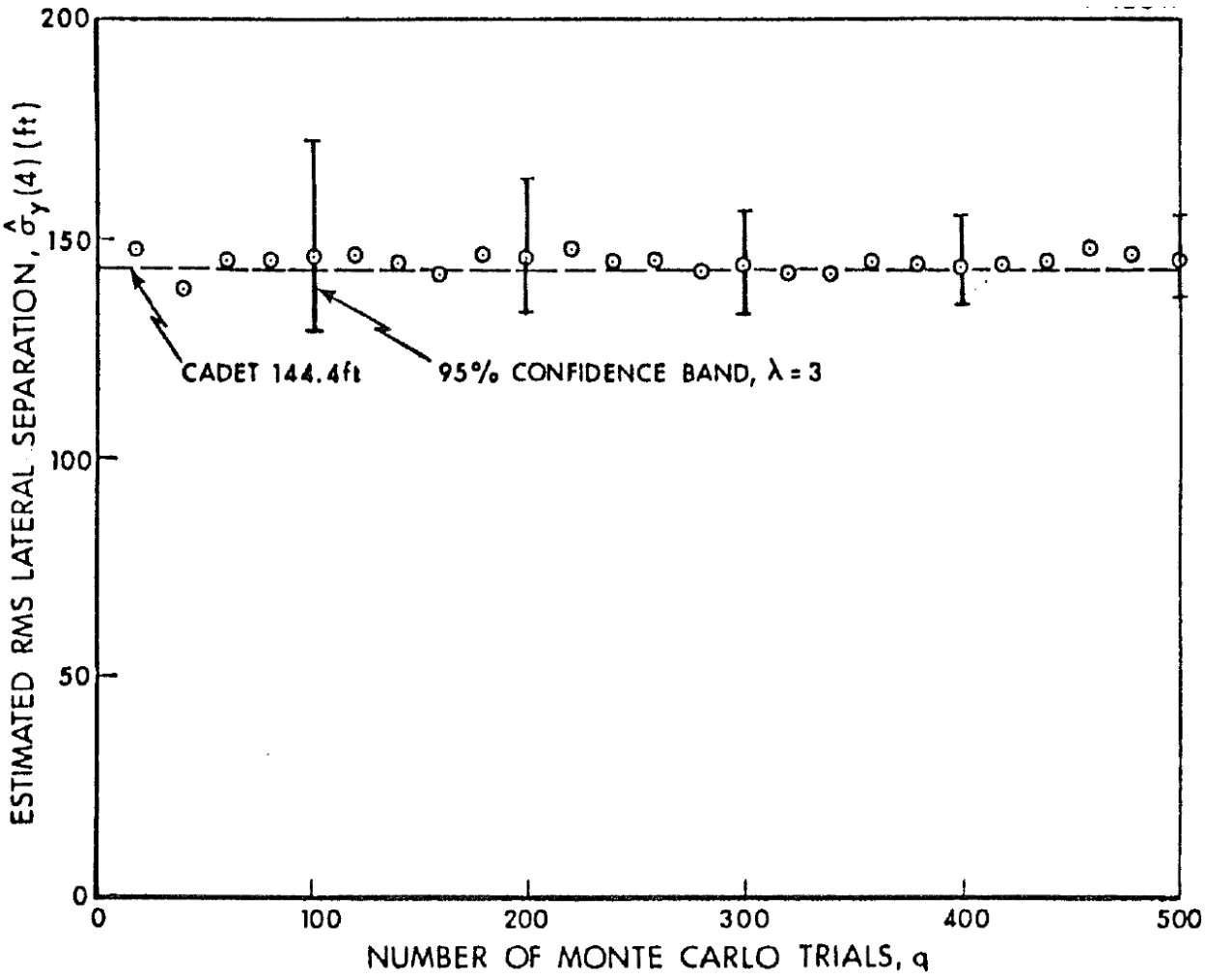
Missile-Target Intercept Problem (Cont'd)

C. Evolution of σ_y and λ_y vs Time



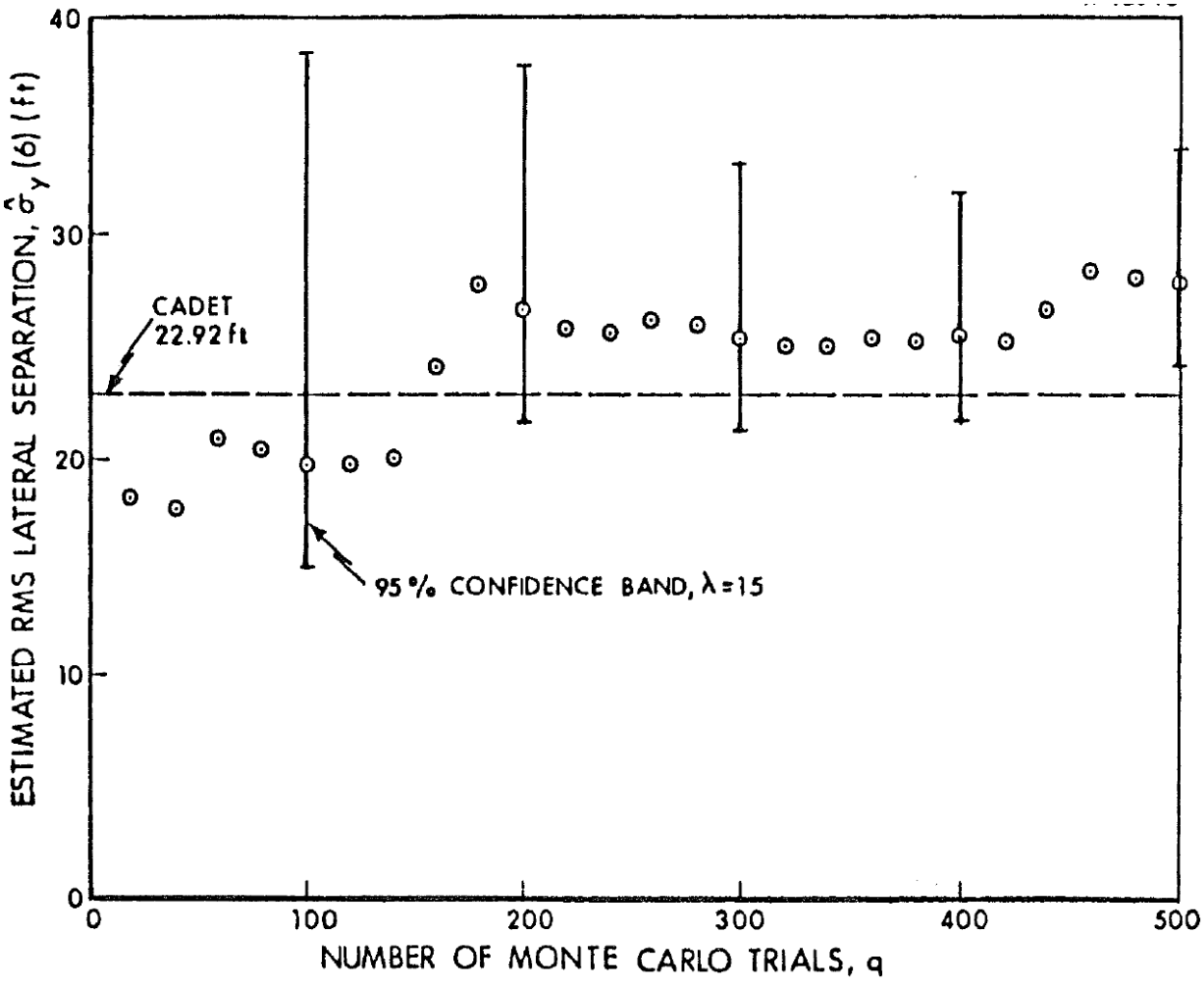
Missile-Target Intercept Problem (Cont'd)

D. "Convergence" of σ_y with increasing q when y is reasonably Gaussian ($t = 3$ sec, $\lambda \cong 3$)



Missile-Target Intercept Problem (Cont'd)

E. "Convergence" of σ_y with increasing q when y is quite Non-Gaussian ($t = 6$ sec, $\lambda \cong 15$)



The "Kurtosis Dilemma"

The 500-trial study shown above is the aggregation of 5 simulation sets of 100 trials each. Individual set results are:

Study No.	$\hat{\sigma}_y$ (ft)	$\hat{\lambda}$
1	19.72	4
2	32.1	15
3	22.3	6
4	25.7	4
5	35.9	23
Aggregate	27.8	15

- Clearly, despite the fact that 100 trials is often considered to be "respectable", the above 100-trial estimates of σ_y are highly questionable.
- 100-trial estimates of kurtosis are **almost meaningless** in this case. They distinguish "benign sets of trials" (sets 1, 3, 4) from the "nonbenign" ones, but do not distinguish "good" or "meaningful" sets of trials from"bad" ones.
- In treating nonlinear systems with random inputs, $\lambda(t)$ is generally **unknown**
- Estimating higher-order moments (recall $\lambda = \mu_4/p^2$) reliably via the Monte Carlo method becomes **successively more difficult** (requiring still more trials).

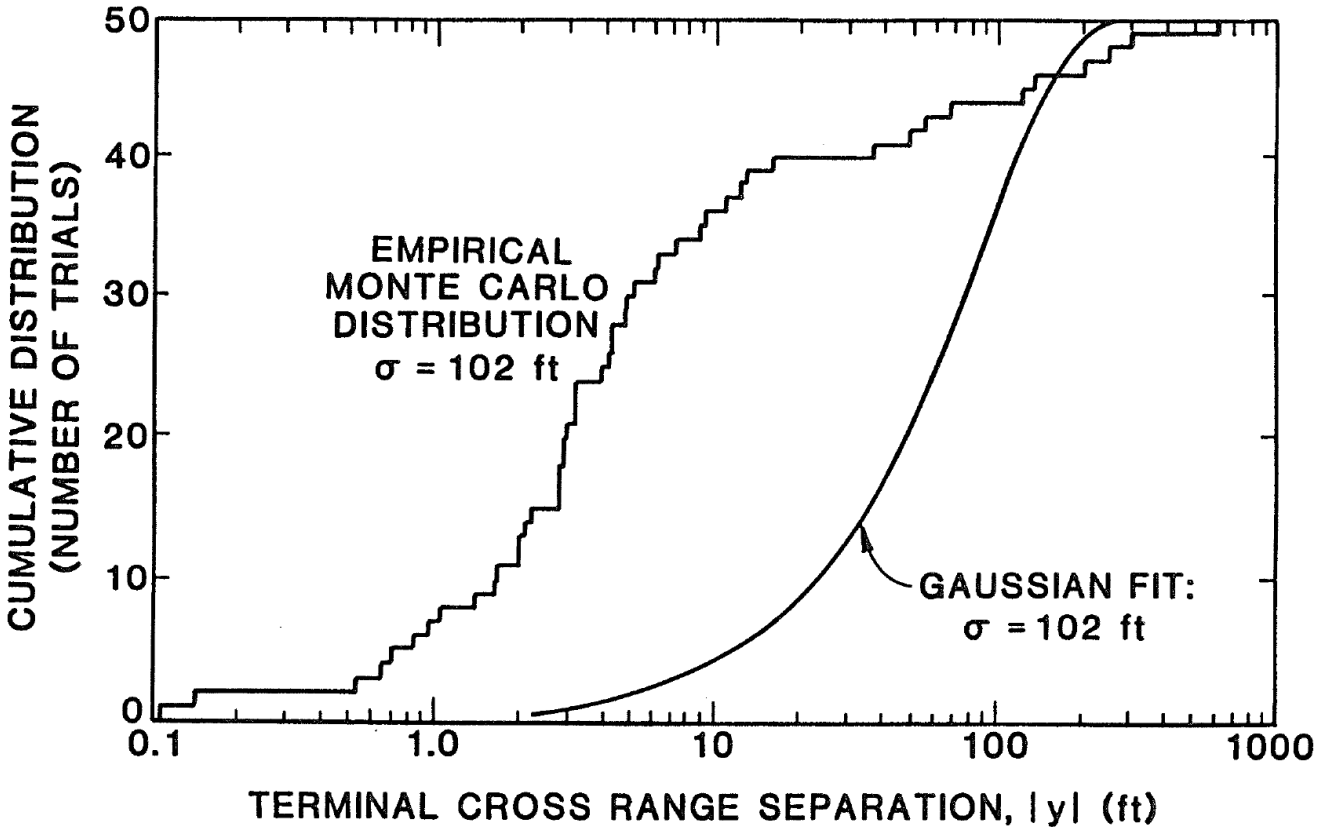
Conclusion:

It is very difficult to know how much credence to place in Monte Carlo estimates of standard deviation unless a quite conservatively large ensemble of trials is performed, or unless you have an **independent basis** for knowing kurtosis.

Dilemma Resolution:

- Look at the **convergence** of $\hat{\sigma}$ (as in previous plots) – this is more informative than considering the final estimate alone ($\hat{\sigma} = 27.8$ feet), but still not very helpful.
- Use the ensemble of trials to generate **histograms** (approximate probability distribution function plots) – this is the **only meaningful** way to assess the performance of a nonlinear system **whether or not** kurtosis is known!

Performance Assessment Histograms



- This is a different scenario from the one we just considered.
- Comparing the histograms with a Gaussian density clearly shows the "heavy tails" ($\lambda \cong 24$).
- The miss distance standard deviation, 102 ft, is meaningless, when viewed in Gaussian terms.
- The statement "80 % of the engagements resulted in miss distance less than 20 feet" **is** meaningful (assuming that an RMS miss distance of 20 feet is "good enough").

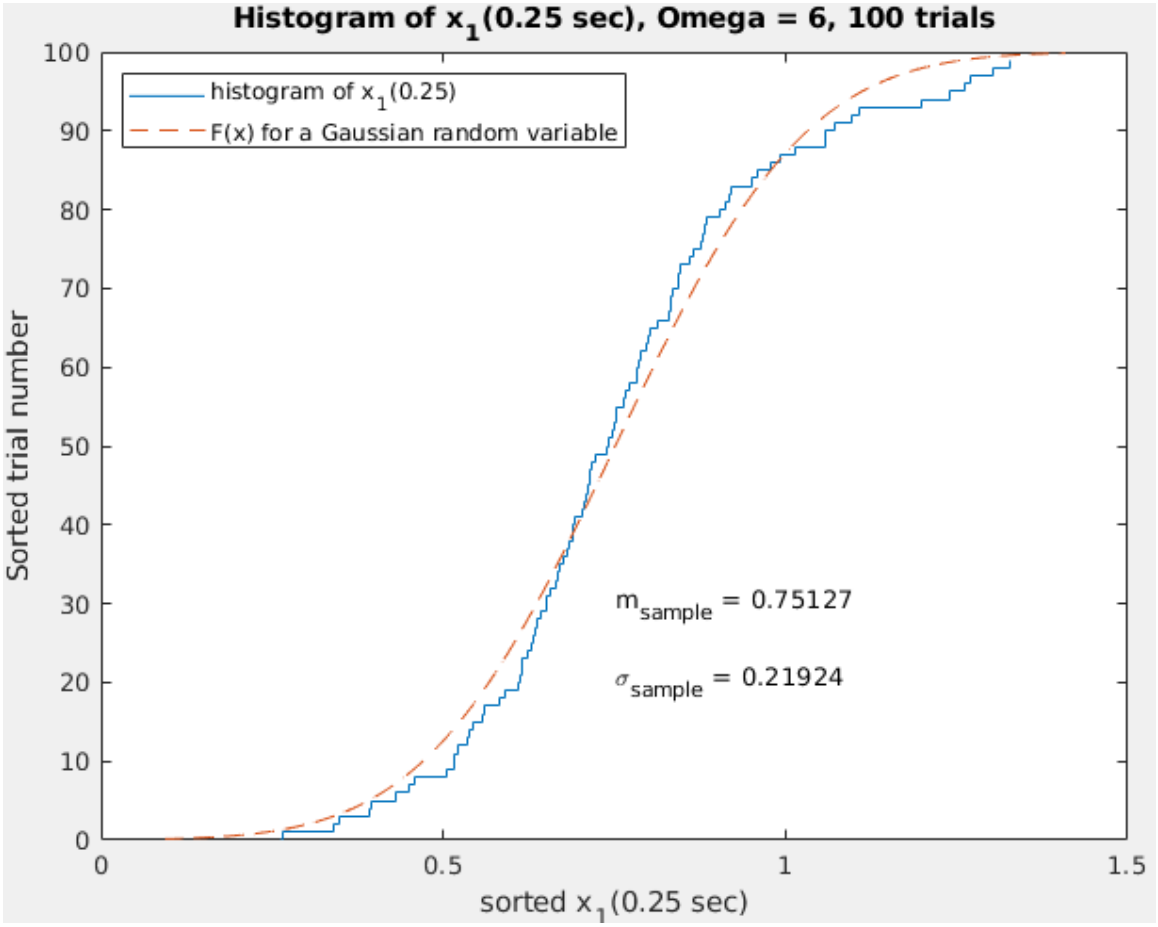
Creating a Histogram

- To illustrate, we return to the Tracker problem treated in slides 18 - 21
- Here is the MATLAB code to create and plot the histogram; note that the data was saved when we ran the Monte Carlo simulations, slide 20, namely `x1p25`, and the sample mean and sigma `m1p25, sig1p25` were also calculated at the end of that script:

```
%% script histo.m
%
% process Monte Carlo data to get a histogram
% run mc_tracker first!
%
% JH Taylor - 8 Feb 2011
%
load x1p25data % data stored during the MC run, Omega = 6
y1sort = sort(x1p25);
for i=1:100 xd(i) = i-1; end %% create the x-axis data
figure; stairs(y1sort,xd)
title('Histogram of x_1(0.25 sec), Omega = 6, 100 trials')
ylabel('Sorted trial number')
xlabel('sorted x_1(0.25 sec)')
hold on
%%  now produce the F(x) for a Gaussian RV with the same
%%  mean and sigma to overlay; generate 401 points, from
%%  m - 3*sigma to m + 3*sigma
sqrt2 = sqrt(2);
for i = 1:601
```

```
    vmult = (i-301)/100;  % ranges from -3 to +3
    Pdf(i) = vmult*sig1p25 + m1p25;
    Horiz(i) = 50*(1 + erf(vmult/sqrt2));
end
plot(Pdf,Horiz,'--')
legend('histogram of x_1(0.25)', ...
'F(x) for a Gaussian random variable' ...
,'location','northwest')
text(0.75,30,['m_{sample} = ',num2str(m1p25)])
text(0.75,20,['\sigma_{sample} = ',num2str(sig1p25)])
```

- And, here is the resulting Monte Carlo-based histogram or **approximate probability distribution**:



This variable appears to be quite Gaussian . . . OK!

Comparing Similar Situations via Monte Carlo Methods

- Comparing **similar systems** for **exactly the same scenarios** can be done quite confidently with fewer trials; you might want to do this to compare alternative system design concepts
- The critical point is **using exactly the same random initial conditions and noise sequences**
- This can be done by:
  - Having the **same states supplied with random initial conditions**, e.g., x_1, x_3, x_6 (the orders of the two models do not have to be the same) and the **same number of random inputs**;
  - Using the **same step size and final time** (recall that you have to use a fixed-step integration method)
  - Being sure **the random-number generator seed** is reset before each set of trials
- Care (e.g., scrutinizing your results) must still be taken.
- I am not aware of anything being **proven** about this (!) but colleagues believe this is valid

Summary and Conclusions

- Random initial conditions and inputs are a fact of life - **all** systems are affected randomly, and we must be able to assess their effect on system performance.
- The Monte Carlo Method is a powerful means of assessment **but** you must be careful to **run enough trials** to produce results with **high confidence**.
- Plotting **histograms** of key states is the best way to determine if you have run a sufficient number of trials.
- In the next module we will discuss an alternative way to assess the performance of stochastic systems – CADET (Covariance Analysis Describing function Technique).