

EE 6383 Nonlinear Control Systems

Random-Input Describing Function Methods

Dr. James H. Taylor
Department of Electrical and Computer Engineering
University of New Brunswick
Fredericton, NB CANADA E3B 5A3
email: jtaylor@unb.ca
website: www.ece.unb.ca/jtaylor

27 May 2019

Module Outline

- Introduction
- Problem Definition
- Covariance Analysis of Linear Systems
- Covariance Analysis of Nonlinear Systems
 - Exact / Unsolvable Propagation Equations
 - Gaussian Assumption and Random-Input Describing Functions (RIDFs)
 - Comparison with Monte Carlo Methods
- Examples

References:

1. D. P. Atherton, *Nonlinear Control Engineering*, Van Nostrand, 1975 (Reprinted as Student Edition, 1982).
2. A. Gelb & W. VanderVelde, *Multiple-Input DF's and Nonlinear System Design*, McGraw-Hill, 1968.
3. J. H. Taylor *et al.*, "Covariance Analysis of Nonlinear Stochastic Systems via Statistical Linearization", Chapter 7 in *Nonlinear System Analysis and Synthesis, Vol. II - Applications*, Ed. by R.V. Ramnath, J.K. Hedrick and H.M. Paynter, ASME, February 1981.
4. M. Landau and C. T. Leondes, "Volterra Series Synthesis of Nonlinear Stochastic Tracking Systems", *Trans. on Aerospace and Electronics Systems, Vol. AES-11*, No. 2, March 1975.
5. J. H. Taylor, "Comments on 'Volterra Series Synthesis of Nonlinear Stochastic Tracking Systems' ", *IEEE Trans. on Aerospace and Electronics Systems, Vol. AES-14*, No. 2, March 1978.
6. J. H. Taylor, "Random-Input Describing Functions for Multi-Input Nonlinearities", *International Journal of Control*, Vol. 23, No. 2, February 1976.

Motivation

- Many systems must deal with random inputs (e.g., flying drones are subject to wind gusts, radar pointing and tracking systems must deal with randomly-moving targets, autonomous vehicles must navigate rough roadways ...).
- Systems may also be perturbed by other random effects:
 - Random initial conditions (e.g., air-to-air missile launch conditions)
 - Random measurement noise (e.g., radar scintillation)

Other examples: manufacturing processes and systems; biological processes and ecosystems; social systems (e.g., economies) ...

Such systems must be analyzed and designed, taking their nonlinear characteristics and the statistical nature of their behavior into account. RIDF methods provide effective approximate techniques for the nonlinear case.

System Models

General stochastic model form:

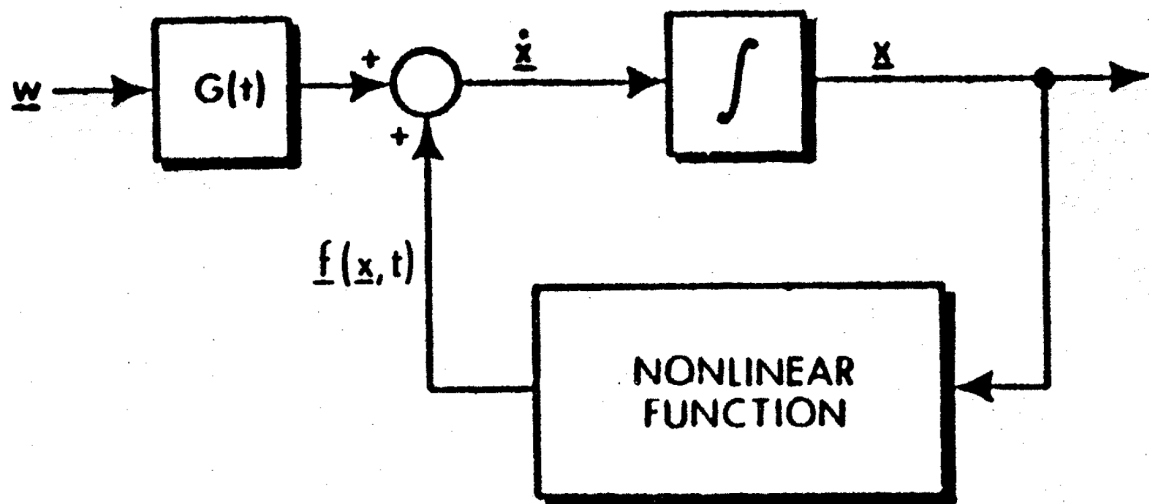
$$\dot{z} = g(z, y, t)$$

$$y = \text{random input}$$

where z is the state variable and y , the input, **cannot** be white noise¹; y is *usually* taken to be a first-order Markov (FOM) process whose input w is white noise plus possibly a deterministic signal such as a constant, ramp, sinusoid, exponential, ...

Equivalent model form if y is a FOM – our **standard model**:

$$\dot{x} = f(x, t) + G(t)w \quad (1)$$



¹The mathematics underlying stochastic differential equations becomes very technical, beyond the scope of this discussion.

State Variable Statistics

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

- Recall the definition of **Ensemble Average**:

$$E[g(x)] = \int_{-\infty}^{\infty} g(x)f(x)dx \quad (2)$$

where $f(x)$ is the **Probability Density Function** (PDF); $g(x)$ may be x , yielding the **mean**, x^2 yielding the **mean square value**, etc.

- Random input characterization:

$$\begin{aligned} b(t) &= E[w(t)] \\ u &= w - b \end{aligned} \quad (3)$$

$$Q(t)\delta(t - \tau) = E[(u(t)u^T(\tau))]$$

where $b(t)$ = **deterministic part** of $w(t)$ and $Q(t)$ = white noise **spectral density** of $u(t)$, a measure of the FOM random component power

- Initial condition statistics:

$$\begin{aligned} m_0 &= E[x(0)] \\ P_0 &= E[(x(0) - m_0)(x(0) - m_0)^T] \end{aligned} \quad (4)$$

where m_0 = mean initial condition, P_0 = initial condition of the covariance matrix.

Linear Covariance Analysis

Note that **covariance analysis** gives an **exact** solution for the statistical behavior of a **linear system** with random inputs; given

$$\dot{x} = A(t)x + B(t)u \quad \text{with} \quad u = w(t) + b(t) \quad (5)$$

where $b(t)$ is a **deterministic signal** and $w(t)$ is **white noise** we can simply solve the following differential equations:

$$\begin{aligned} \dot{m} &= A(t)m + B(t)b(t) \\ \dot{P} &= A(t)P + PA^T(t) + B(t)QB^T(t) \end{aligned} \quad (6)$$

One should **never** use the Monte Carlo Method for linear systems!

Random-Input DF Analysis

The problem definition (**model** and **goal**) is identical to the Monte Carlo case: Goal: we want to determine how the **state variable statistics** evolve with time.

- The exact differential equations governing the propagation of $m(t)$ and $P(t)$ are [1]:

$$\dot{m} = E[f(x, t)] + G(t) b \quad (7)$$

$$\dot{P} = E[f r^T] + E[r f^T] + G(t) Q G^T(t)$$

However, evaluating $E[f(x, t)]$ and $E[f r^T]$ **requires knowing the joint PDF of the state variables ...**

- The equation for P can be expressed in a form analogous to the covariance equations for the linear case, by defining the auxiliary matrix N_R as:

$$N_R P \triangleq E[f(x, t) r^T] \quad (8)$$

- The RIDF matrix N_R is the direct vector / matrix extension of the scalar describing function definition [3, 4]. Then Eqn. 7 may be written as

$$\dot{m} = \hat{f} + G(t) b \quad (9)$$

$$\dot{P} = N_R P + P N_R^T + G(t) Q G^T(t)$$

where $\hat{f} \triangleq E[f(x, t)]$. Furthermore, the following is a very convenient relation² between \hat{f} and N_R :

$$N_R = \frac{\partial \hat{f}}{\partial m} \quad (10)$$

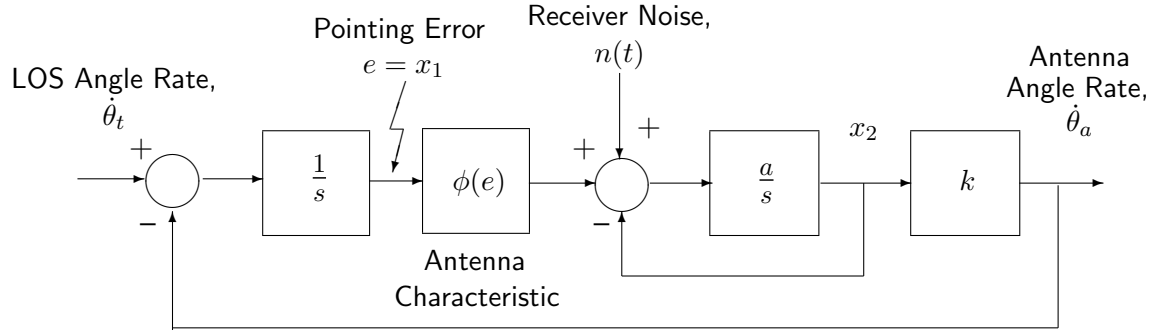
²R. J. Phaneuf, *Approximate Nonlinear Estimation*, PhD Thesis, Massachusetts Institute of Technology, May, 1968.

Random-Input DF Analysis (Cont'd)

- The quantities \hat{f} and N_R must be determined before one can solve Eqn. 10; **this requires knowledge of the joint PDF of the state variables.** While it is possible, in principle, to evolve the joint PDF $f(x, t)$ by solving a set of partial differential equations known as the **Fokker-Planck equations** [1], this is difficult except for simple, low-order systems – so, this procedure is generally not practically feasible.
- An approximate solution to Eqn. (10) may be obtained by **assuming the form of the joint PDF**; the assumption that the state variables are jointly normal is both reasonable and convenient \Rightarrow RIDFs depend only upon m and P . This was pioneered by the company I worked for in the 1970s; we called it CADET (Covariance Analysis DEscribing function Technique).
- The RIDF “machinery” is very different from SIDF approaches – here we use RIDFs to quazilinearize the differential equations for m and P so they can be solved numerically to determine how $m(t)$ and $P(t)$ evolve over time.
- This procedure will be illustrated by a simple case study.

Case Study: Antenna Pointing and Tracking Problem

A. The Model:



This schematic is equivalent to

$$\dot{x} = f(x) + w \quad (11)$$

where x_1 is the pointing error, e ; x_2 models the slewing of the antenna via a first-order lag, and

$$f(x) = \begin{bmatrix} -k x_2 \\ a(\phi(x_1) - x_2) \end{bmatrix} ; \quad w = \begin{bmatrix} \dot{\theta}_t \\ a n(t) \end{bmatrix} \quad (12)$$

Assume that θ_t is a deterministic ramp to be tracked,

$$\dot{\theta}_t = \Omega \quad (13)$$

where Ω is the slope of the ramp, in other words, the target is moving steadily across the sky with angular velocity Ω . The pointing error, $e = \theta_t - \theta_a$, is the input to a nonlinearity $\phi(e)$ which represents the **limited beamwidth of the antenna**, e.g.,

$$\phi(e) = e(1 - k_a e^2) \quad (14)$$

Case Study (Cont'd)

B. Statistical Characterization:

The noise $n(t)$ injected into the receiver is a white noise process having zero mean and spectral density q . Thus the statistics of the input vector w are:

$$E[w] \triangleq b = \begin{bmatrix} \Omega \\ 0 \end{bmatrix} \quad (15)$$

$$E[(w(t) - b)(w(\tau) - b)^T] \triangleq Q \delta(t - \tau) = \begin{bmatrix} 0 & 0 \\ 0 & a^2 q \end{bmatrix} \delta(t - \tau) \quad (16)$$

The initial state variable statistics, assuming $x_2(0) = 0$, are

$$E[x(0)] \triangleq m_0 = \begin{bmatrix} m_{e0} \\ 0 \end{bmatrix} ; \quad E[(x(0) - m_0)(x(0) - m_0)^T] \triangleq \begin{bmatrix} \sigma_{e0}^2 & 0 \\ 0 & 0 \end{bmatrix} \quad (17)$$

where m_{e0} and σ_{e0} are the initial mean and standard deviation of the pointing error, respectively.

Case Study (Cont'd)

C. RIDF Evaluation for the Cubic Nonlinearity:

To keep notation simple, we focus on the single-input nonlinearity $\phi(e) = e^3$. By definition, $\hat{\phi} = E[e^3]$, which, under the gaussian assumption, is

$$\hat{\phi} = \int_{-\infty}^{\infty} e^3 f(e) de = \frac{1}{\sqrt{2\pi}\sigma} \int_{-\infty}^{\infty} e^3 \exp\left[-\frac{(e-m)^2}{2\sigma^2}\right] de \quad (18)$$

where m and σ are the second-order statistics of e . We simplify this by the change of variable $u = (e - m)/\sigma$ to obtain

$$\begin{aligned} \hat{\phi} &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} (\sigma u + m)^3 \exp(-u^2/2) du \\ &= \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\infty} [\sigma^3 u^3 + 3m\sigma^2 u^2 + 3m^2\sigma u + m^3] \exp(-u^2/2) du \end{aligned} \quad (19)$$

Note that only the **even** part of the integrand contributes to the final result, so we can eliminate the u and u^3 terms. Resorting to a table of integrals we find

$$\int_{-\infty}^{\infty} \exp(-u^2/2) du = \int_{-\infty}^{\infty} u^2 \exp(-u^2/2) du = 1 \quad (21)$$

so, very simply, $\hat{\phi} = 3m\sigma^2 + m^3$.

This is all we need for \dot{m} . We also require the corresponding entry for N_R , which is easily obtained:

$$N_{\phi} = \frac{\partial \hat{\phi}}{\partial m} = 3(\sigma^2 + m^2) \quad (22)$$

Case Study (Cont'd)

D. RIDF Relations for Pointing Error Statistics

The quasilinear RIDF representation for $\phi(e)$ in Eqn. 14 is [3, 4]

$$\begin{aligned}\phi(x_1) &\approx \hat{\phi} + N_\phi (x_1 - m_1); \\ \hat{\phi} &= (m_1 - k_a (m_1^2 + 3\sigma_1^2) m_1) \\ N_\phi &= (1 - 3k_a (m_1^2 + \sigma_1^2))\end{aligned}\tag{23}$$

where m_1 and σ_1^2 are elements of m and P , respectively. The solution is then obtained by solving Eqn. 10, which specializes to

$$\dot{m} = \begin{bmatrix} -k m_2 \\ a(\hat{\phi} - m_2) \end{bmatrix} + b\tag{24}$$

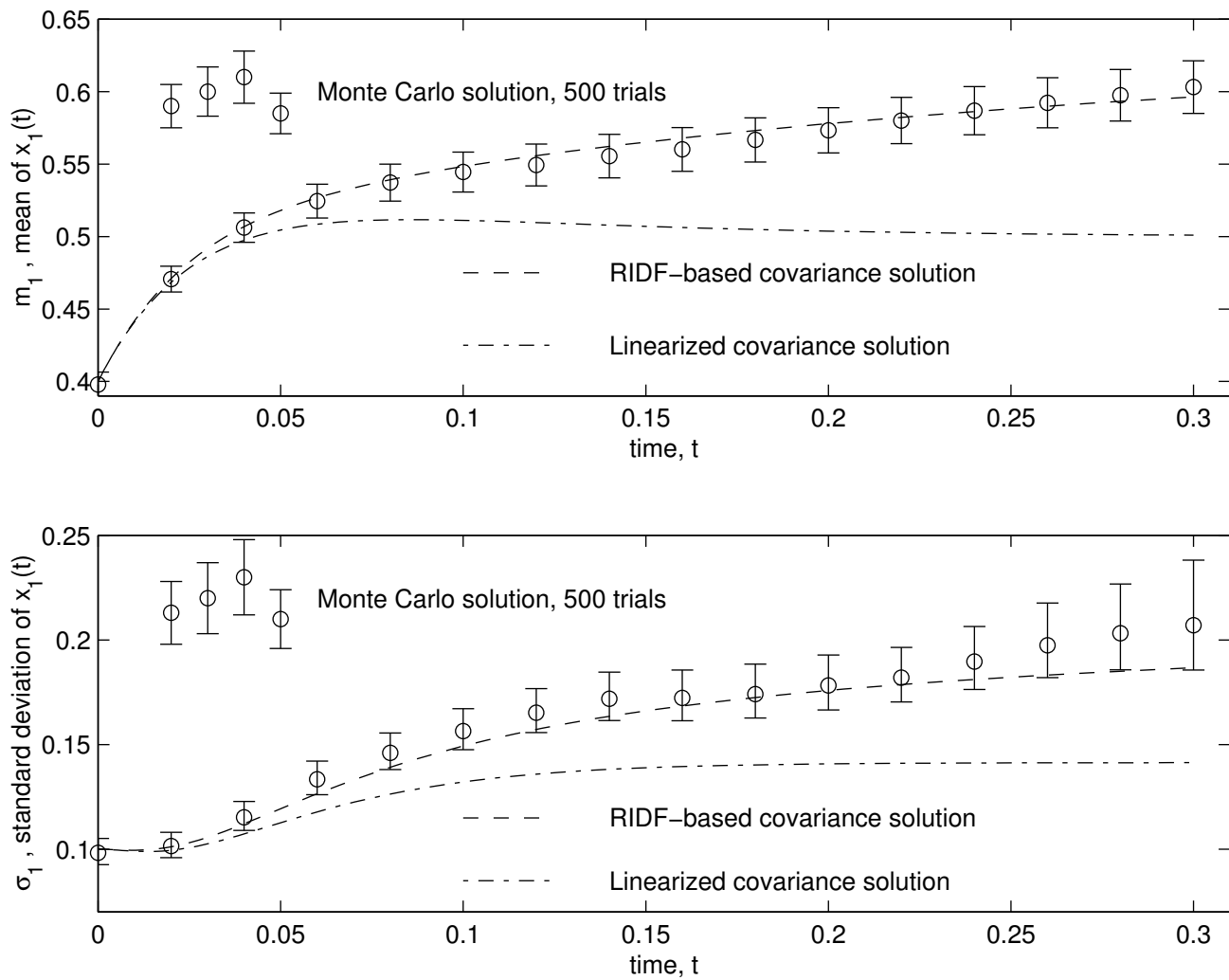
$$\dot{P} = N_R P + P N_R^T + Q ; \quad N_R = \begin{bmatrix} 0 & -k \\ a N_\phi & -a \end{bmatrix}\tag{25}$$

subject to the initial conditions in Eqn. 17. Note that $\phi(e)$ is linear plus cubic, $\phi(e) = e(1 - k_a e^2)$, not cubic alone.

Antenna Pointing and Tracking Problem (Cont'd)

D. Evolution of Pointing Error Statistics

Tracking capability for $\Omega = 5$ deg/sec is shown. The system parameters are: $a = 50 \text{ sec}^{-1}$, $k = 10 \text{ sec}^{-1}$, $k_a = 0.4 \text{ deg}^{-2}$; the pointing error initial condition statistics are $m_{e0} = 0.4 \text{ deg}$, $\sigma_{e0} = 0.1 \text{ deg}$; and the noise spectral density is $q = 0.004 \text{ deg}^2$.



Antenna Tracking MATLAB Code

First, here is the routine for calculating \dot{m} and \dot{P} :

```
function xdot = tracker_ridf(t,x)
% RIDF m-dot P-dot model for antenna tracking problem
% "states" are m1, m2, p11, p12, p22
% JH Taylor - 27 May 2019
A = 50;           % sec-1
K = 10;           % sec-1
global Kant       % make Kant settable in script
Omega = 5;        % deg/sec LOS angle rate
q = .004;         % deg2
% note: case in paper -> x(0) = [ 0.4 0 0.01 0 0 ]';
fhat = x(1) - Kant*x(1)*(x(1)*x(1) + 3*x(3));
Nr = 1 - 3*Kant*(x(1)*x(1) + x(3));
xdot(1) = - K*x(2) + Omega;
xdot(2) = A*(fhat - x(2));
P = [ x(3) x(4) ; x(4) x(5) ];
% test to be sure P stays positive semi-definite
T1 = P(1,1) ; T2 = det(P);
if T1 < 0, error('P(1,1) negative - bummer!'); return; end
if T2 < 0, error('P negative-def. - bummer!'); return; end
NR = [ 0 -K ; A*Nr -A ];
Q = [ 0 0 ; 0 A*A*q ];
Pdot = NR*P + P*(NR') + Q;
xdot(3) = Pdot(1,1);
xdot(4) = (Pdot(1,2) + Pdot(2,1))/2; %% Ensure P symmetric
xdot(5) = Pdot(2,2);
xdot = xdot(:);
```

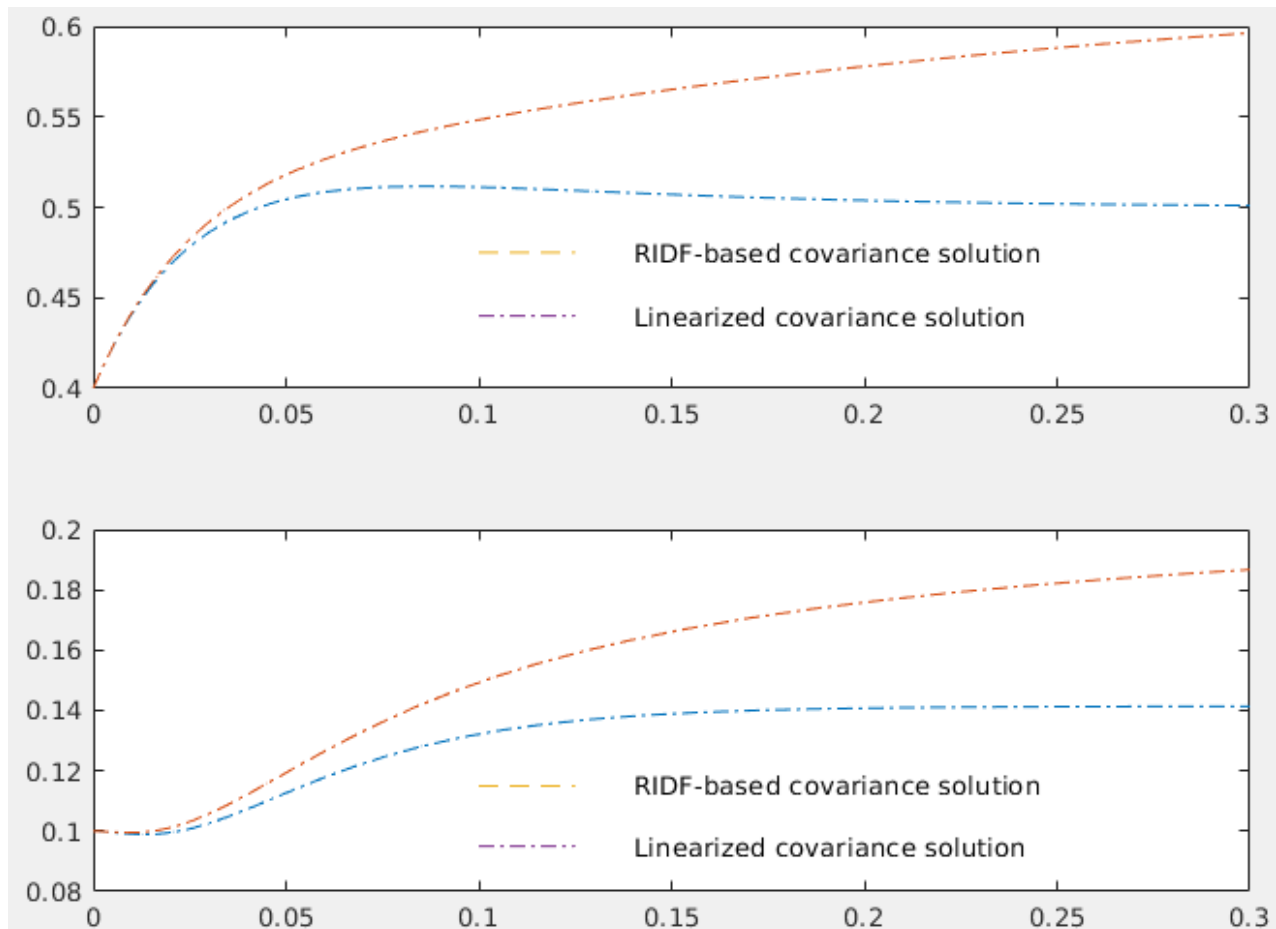
Antenna Tracking MATLAB Code (Cont'd)

So, all we need is a nice script file:

```
% script to run RIDF covariance analysis of antenna tracking
% problem, then plot results, WITHOUT Monte Carlo stuff
% JH Taylor - 27 May 2019
clear; close all
global Kant
tspan = [ 0 .3]; x0 = [ 0.4 0 0.01 0 0 ]; x0 = x0(:);
Kant = 0; %% run the linear case first:
[t1,x1] = ode45('track_ridf',tspan,x0);
subplot(211); plot(t1,x1(:,1),'-'); hold on;
subplot(212); plot(t1,sqrt(x1(:,3)),'-'); hold on;
Kant = 0.4; %% now, run the nonlinear case:
[tn1,xn1] = ode45('track_ridf',tspan,x0);
subplot(211); plot(tn1,xn1(:,1),'-'); hold on;
%% annotate
plot([0.10 0.125],[.475 .475],'--');
text(0.14,0.475,'RIDF-based covariance solution')
plot([0.10 0.125],[.440 .440],'-');
text(0.14,0.440,'Linearized covariance solution')
subplot(212); plot(tn1,sqrt(xn1(:,3)),'-'); hold on;
axis([0 .3 0.08 0.2])
%% annotate
plot([0.10 0.125],[.115 .115],'--');
text(0.14,0.115,'RIDF-based covariance solution')
plot([0.10 0.125],[.095 .095],'-');
text(0.14,0.095,'Linearized covariance solution')
```

Antenna Tracking MATLAB Code (Cont'd)

This MATLAB code gave the expected result (see slide 13), which is sometimes a pleasant surprise, confirming a result obtained by very different means about 45 years ago ...



Antenna Pointing and Tracking Assessment

The preceding plots show the antenna pointing and tracking results obtained via the RIDF approach together with ensemble statistics from a 500-trial Monte Carlo simulation, along with the corresponding 95 % confidence error bars. Also, the results of a linearized covariance analysis are shown, based on assuming that the antenna characteristic is linear ($k_a = 0$).

The linearized analysis indicates that the pointing error statistics reach steady-state values at about $t = 0.2$ sec. However, it is evident from the two nonlinear analyses that this not the case: in fact, the tracking error can become so large that the antenna characteristic effectively becomes a negative gain, producing unstable solutions (the **antenna loses track**).

The RIDF approach **is approximate**, but Monte Carlos analysis shows it provides a **good approximation**, requiring much less computation time than 500 Monte Carlo simulations. Maybe this isn't a huge issue with modern computer, but then again, you don't know how good the Monte Carlo result may be unless you look at the histogram ...

Describing Function Development

Using the same notation as on slide 11 we consider a **scalar function** $\phi(v)$ of a **variable** v whose statistics are m and σ . Once we have the DF results for ϕ , i.e., $\hat{\phi}$ and N_ϕ , we insert them into the **complete problem** as follows:

Assume ϕ occurs in the i^{th} state derivative, i.e., $f_i(x)$, and involves the j^{th} state, i.e., $v = x_j$. Then $\hat{\phi}$ is inserted in \dot{m}_i and N_ϕ is inserted into the i, j element of N_R . This is exactly what was done for the antenna tracking problem.

DF evaluation is quite straight forward for polynomials and other smooth functions (e.g., $\sin(v)$) – just use a **table of integrals** to evaluate the expectations. However, there are several auxiliary functions that must be used in the DFs for **piece-wise linear** characteristics: Given the Gaussian PDF,

$$f(v) = \frac{1}{\sqrt{2\pi} \sigma} \exp \left[-\frac{(v - m)^2}{2\sigma^2} \right] \quad (26)$$

Then the required auxiliary functions are:

$$\begin{aligned} F(v) &= \int_{-\infty}^v f(\zeta) d\zeta \\ G(v) &= \int_{-\infty}^v F(\zeta) d\zeta \\ &= vF(v) + f(v) \end{aligned} \quad (27)$$

Note that $dG/dv = F(v)$ and $dF(v)/dv = f(v)$; recall that $F(v)$ is the Probability **Distribution** Function; $G(v)$ has no formal name.

Describing Function Development (Cont'd)

In terms of these auxiliary functions we have the following for an **ideal relay**,

$$\phi(v) = \begin{cases} -D, & v < 0 \\ D, & v > 0 \end{cases} \quad (28)$$

$$\hat{\phi} = \frac{D}{m} \left[2F\left(-\frac{m}{\sigma}\right) - 1 \right] \quad (29)$$

Given $\hat{\phi}$, N_ϕ is merely $\partial\hat{\phi}/\partial m$, or

$$N_\phi = \frac{2D}{\sigma} f\left(\frac{m}{\sigma}\right) \quad (30)$$

But we're not ready for MATLAB yet ...

- We must express F and G in terms of “erf(x)”

$$\begin{aligned} \text{erf}(x) &= \frac{1}{\sqrt{\pi}} \int_{-x}^x e^{-\zeta^2} d\zeta \\ \Rightarrow F(v) &= \frac{1}{2} \left(\text{erf}\left(\frac{v}{\sqrt{2}}\right) + 1 \right) \end{aligned} \quad (31)$$

$$\text{and } G(v) = vF(v) + \frac{1}{\sqrt{2\pi}} e^{-v^2/2} \quad (32)$$

- These results allow us to treat systems with relays, limiters, deadzones, etc., directly in MATLAB

Describing Function Development (Cont'd)

Another common piece-wise linear characteristic is the **ideal limiter** (saturation),

$$\phi(v) = \begin{cases} v, & |v| \leq \delta \\ \delta \operatorname{sign}(v), & |v| > \delta \end{cases} \quad (33)$$

$$\hat{\phi} = \sigma \left[G\left(\frac{\delta + m}{\sigma}\right) - G\left(\frac{\delta - m}{\sigma}\right) \right] - m \quad (34)$$

Again, given $\hat{\phi}$, N_{ϕ} is just $\partial \hat{\phi} / \partial m$, or

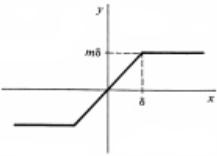
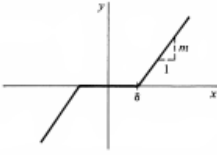
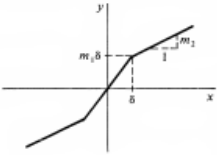
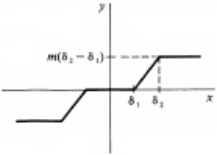
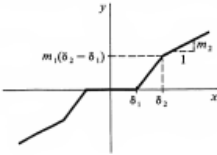
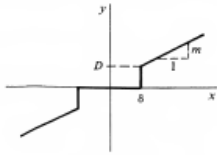
$$N_{\phi} = F\left(\frac{\delta + m}{\sigma}\right) - F\left(\frac{\delta - m}{\sigma}\right) - 1 \quad (35)$$

These two results allow us to treat the relay and limiter; for more complicated piece-wise linear functions it's advisable to consult the RIDF tables in Refs. 1 or 2.

Describing Function Development (Cont'd)

From Ref. 2; notation: $m \rightarrow B$, $\hat{\phi} \rightarrow N_B$, $N_{\hat{\phi}} \rightarrow N_R$, $F \rightarrow PI$

TABLE OF RANDOM-INPUT DESCRIBING FUNCTIONS (RIDFs) (Continued)

Nonlinearity	Comments	$N_R(\sigma, B)$ and $N_B(\sigma, B)$
 <p>7. Sharp saturation or limiter</p>		$N_R = m \left[PI \left(\frac{\delta + B}{\sigma} \right) + PI \left(\frac{\delta - B}{\sigma} \right) - 1 \right]$ $N_B = m \left(\frac{\sigma}{B} \left[G \left(\frac{\delta + B}{\sigma} \right) - G \left(\frac{\delta - B}{\sigma} \right) \right] - 1 \right)$
 <p>8. Dead zone or threshold</p>		$N_R = m \left[2 - PI \left(\frac{\delta + B}{\sigma} \right) - PI \left(\frac{\delta - B}{\sigma} \right) \right]$ $N_B = m \left(2 - \frac{\sigma}{B} \left[G \left(\frac{\delta + B}{\sigma} \right) - G \left(\frac{\delta - B}{\sigma} \right) \right] \right)$
 <p>9. Gain-changing nonlinearity</p>		$N_R = m_1 + (m_2 - m_1) \left[2 - PI \left(\frac{\delta + B}{\sigma} \right) - PI \left(\frac{\delta - B}{\sigma} \right) \right]$ $N_B = m_1 + (m_2 - m_1) \left(2 - \frac{\sigma}{B} \left[G \left(\frac{\delta + B}{\sigma} \right) - G \left(\frac{\delta - B}{\sigma} \right) \right] \right)$
 <p>10. Limiter with dead zone</p>		$N_R = m \left[PI \left(\frac{\delta_2 + B}{\sigma} \right) + PI \left(\frac{\delta_2 - B}{\sigma} \right) - PI \left(\frac{\delta_1 + B}{\sigma} \right) - PI \left(\frac{\delta_1 - B}{\sigma} \right) \right]$ $N_B = m \frac{\sigma}{B} \left[G \left(\frac{\delta_2 + B}{\sigma} \right) - G \left(\frac{\delta_2 - B}{\sigma} \right) - G \left(\frac{\delta_1 + B}{\sigma} \right) + G \left(\frac{\delta_1 - B}{\sigma} \right) \right]$
 <p>11. Gain-changing nonlinearity with dead zone</p>		$N_R = 2m_2 - m_1 \left[PI \left(\frac{\delta_2 + B}{\sigma} \right) + PI \left(\frac{\delta_2 - B}{\sigma} \right) \right] +$ $(m_1 - m_2) \left[PI \left(\frac{\delta_1 + B}{\sigma} \right) + PI \left(\frac{\delta_1 - B}{\sigma} \right) \right]$ $N_B = 2m_2 + \frac{\sigma}{B} \left[-m_1 \left[G \left(\frac{\delta_2 + B}{\sigma} \right) - G \left(\frac{\delta_2 - B}{\sigma} \right) \right] + \right.$ $(m_1 - m_2) \left[G \left(\frac{\delta_1 + B}{\sigma} \right) - G \left(\frac{\delta_1 - B}{\sigma} \right) \right] \left. \right]$
		$N_R = m \left[2 - PI \left(\frac{\delta + B}{\sigma} \right) - PI \left(\frac{\delta - B}{\sigma} \right) \right] + \frac{D}{\sigma} \left[PF \left(\frac{\delta + B}{\sigma} \right) + PF \left(\frac{\delta - B}{\sigma} \right) \right]$ $N_B = m \left(2 - \frac{\sigma}{B} \left[G \left(\frac{\delta + B}{\sigma} \right) - G \left(\frac{\delta - B}{\sigma} \right) \right] \right) + \frac{D}{B} \left[PI \left(\frac{\delta + B}{\sigma} \right) - PI \left(\frac{\delta - B}{\sigma} \right) \right]$

Describing Function Development (Cont'd)

Interestingly, certain multiple-input nonlinearities are also “easy” to handle. Results have been obtained for $f_1(v) = v_1g(v_2)$, $f_2(v) = v_1g(v_2, v_3)$ and $f_3(v) = v_1^k g(v_2)$, $k = 2, 3, \dots$, **assuming that the RIDFs for the g functions are available**. Details are given in Ref. 6.

The general form for the RIDF approximation of multiple-input nonlinearities is

$$f(v_1, v_2, \dots, v_k) \cong \hat{f} + N_1 r_1 + N_2 r_2 + \dots + N_k r_k \quad (36)$$

I use v as a generality; each v_i may be a linear combination of states, $v_i = h_i^T x$. In principle we derive \hat{f} and N_i by integrating over the multivariable Gaussian PDF, given in the Monte Carlo slide 5; however, we still can use the simple approach for N_i , $N^T = \partial \hat{f} / \partial m$. So, we only need to focus on \hat{f} .

- For $f_1(v) = v_1g(v_2)$ we have

$$\hat{f}_1 = m_1 \hat{g} + \rho \sigma_1 \sigma_2 \frac{\partial \hat{g}}{\partial m_2} = m_1 \hat{g} + p_{12} N_g \quad (37)$$

- For $f_2(v) = v_1g(v_2, v_3)$ we have

$$\hat{f}_2 = m_1 \hat{g} + p_{12} \frac{\partial \hat{g}}{\partial m_2} + p_{13} \frac{\partial \hat{g}}{\partial m_3} \quad (38)$$

- For $f_3(v) = v_1^k g(v_2)$, $k = 2$ (and so on by induction) we have

$$\hat{f}_3 = (m_1^2 + \sigma_1^2) \hat{g} + 2m_1 p_{12} N_g + p_{12}^2 \frac{\partial N_g}{\partial m_2} \quad (39)$$

Again, given \hat{f}_i we take partials with respect to m_i to get the N_i .

Accuracy of Random-Input DF Analysis

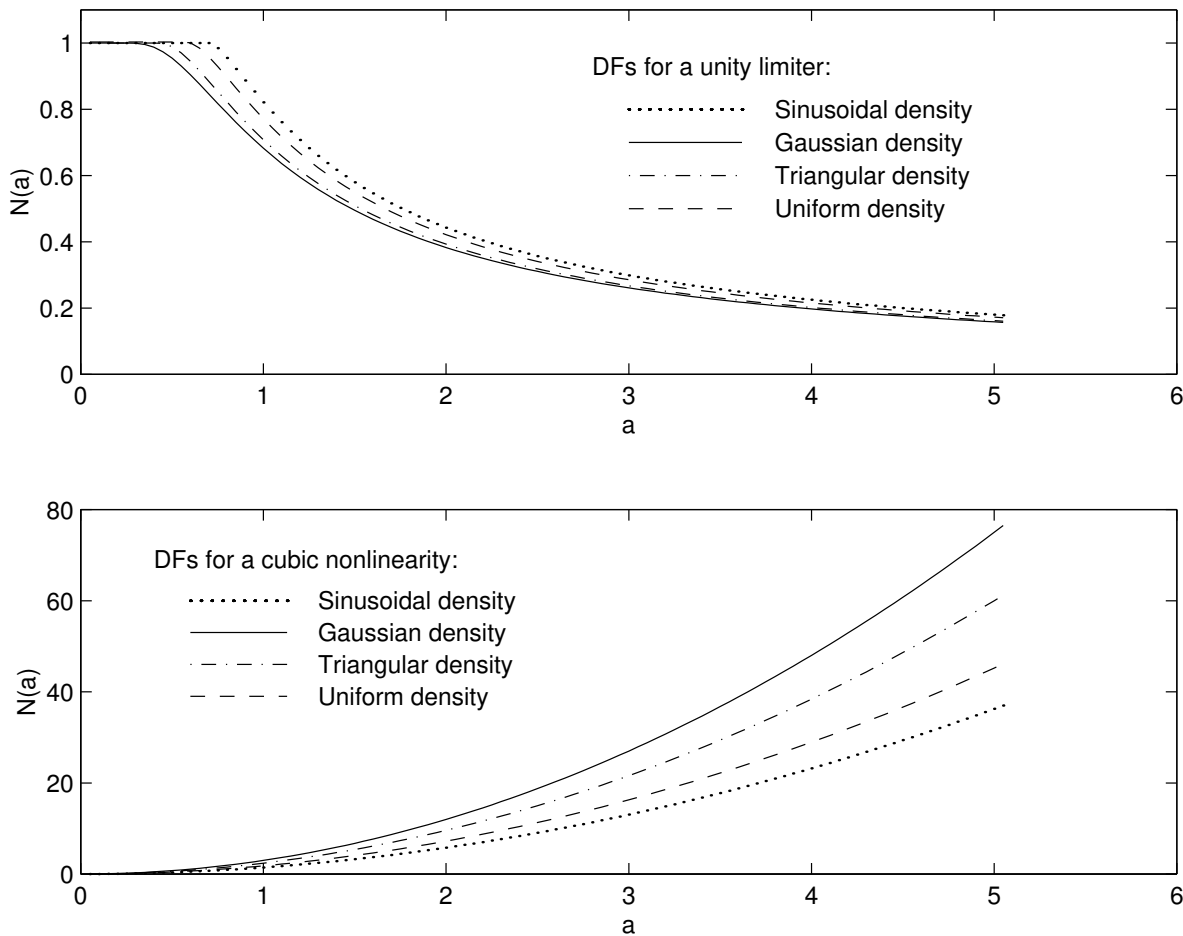
In SIDF analysis the assumption is made that nonlinear function inputs are basically sinusoidal (i.e, higher harmonics are small / negligible). The ability of SIDF analysis to explain nonlinear phenomena such as the existence and nature of limit cycles and the effective “frequency response” of a nonlinear system has been shown to be powerful as long as the analyst recognizes that the method is approximate and some cross-checking is needed, e.g., simulation guided by limit cycle predictions, to verify the validity of SIDF predictions.

From another viewpoint, the SIDF is **always** superior to small-signal linearization, which totally fails to capture the amplitude-sensitivity of a nonlinearity and thereby produces unrealistic performance predictions (except for “small signals”).

The same observations hold for RIDF analysis. The amplitude dependence of RIDFs of a limiter and cubic nonlinearity for Gaussian inputs and for other probability density functions are depicted below (next page); although there is gain variation for different assumed PDFs they certainly provide a better characterization of the nonlinear effect than small-signal linearization.

Accuracy of RIDF Analysis (Cont'd)

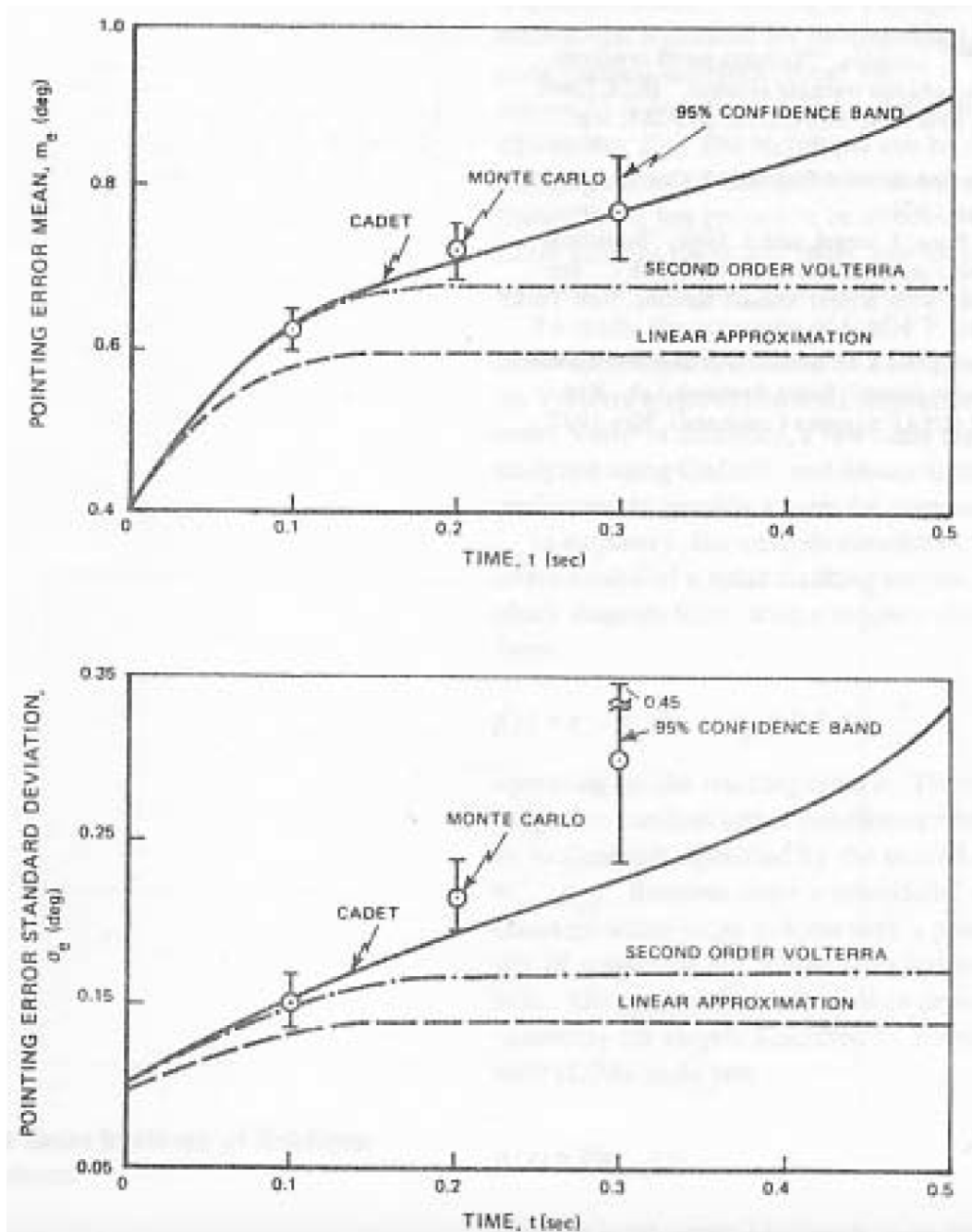
Finally, we consider the influence of amplitude density function variation on RIDF evaluation; we note, as in the SIDF case, the amplitude-dependent gain of two common nonlinearities does depend on the PDF. However, again, RIDFs provide a better characterization of the nonlinear effect than small-signal linearization:



For **piece-wise linear characteristics** the difference near the break-point is typically small and transitory; for **power-law characteristics** the differences are larger.

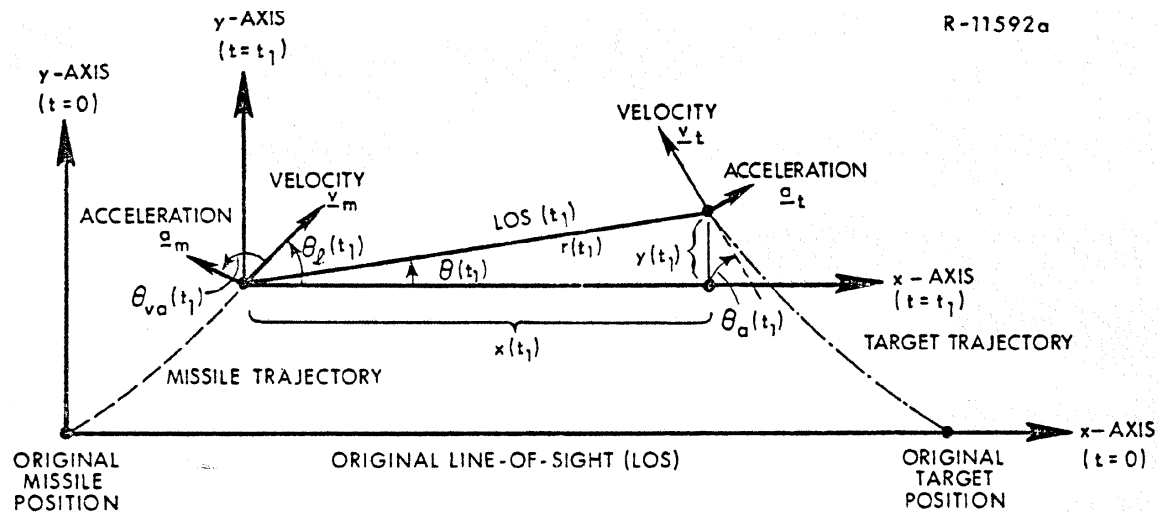
Accuracy of Random-Input DF Analysis

Finally, just to brag a little bit, here is a comparison of the antenna tracking problem solution for a more rapidly moving target, $\Omega = 6$ deg/sec., using the Monte Carlo Method, RIDFs, a Volterra series approach from Ref. 4, and small-signal linearization.

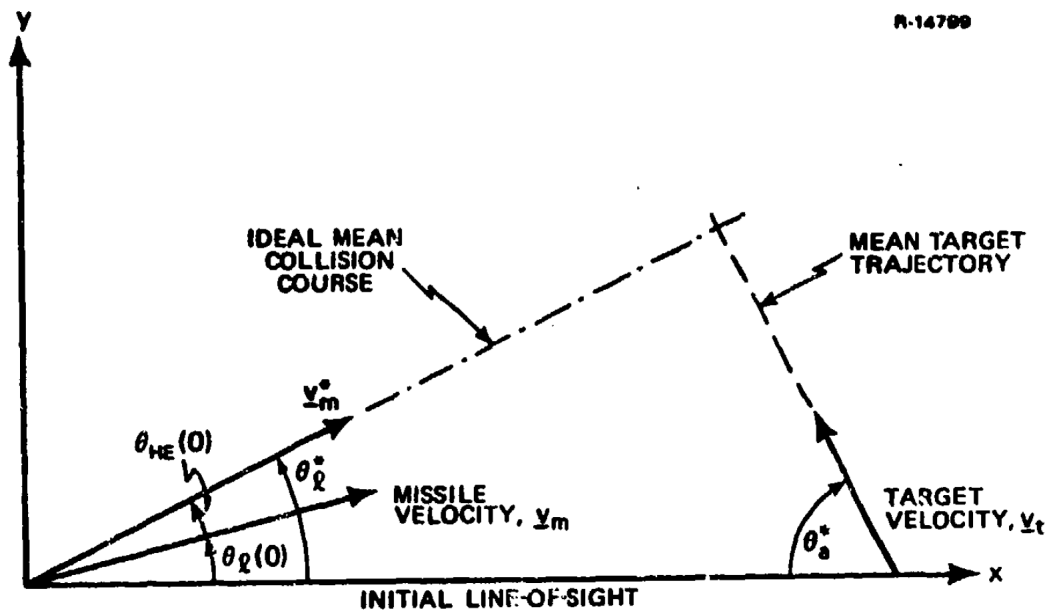


Missile-Target Intercept Problem (Cont'd)

B. Intercept Geometry

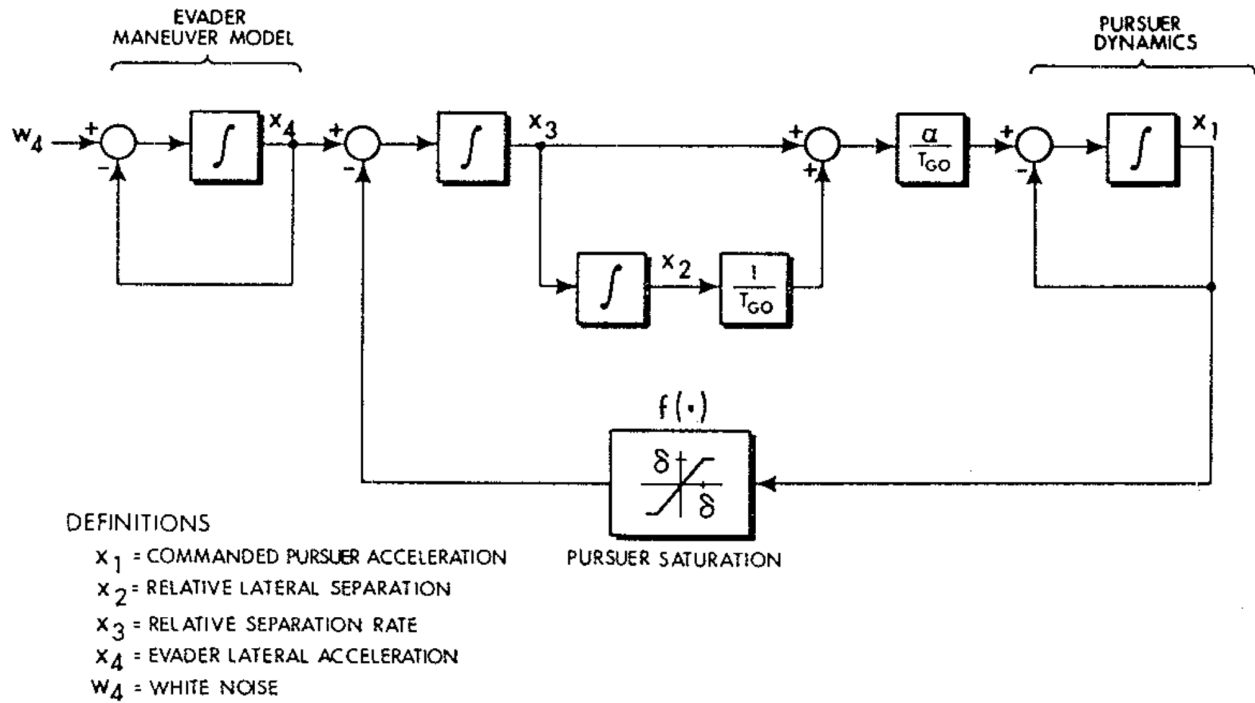


The basic strategy for interception is to estimate the target velocity vector and then direct your velocity vector to complete a “collision-course triangle”; to succeed the “line-of-sight angle” must be constant; the **guidance law** adjusts as the target maneuvers.



Missile-Target Intercept Problem (Cont'd)

C. Block Diagram and State-space Model

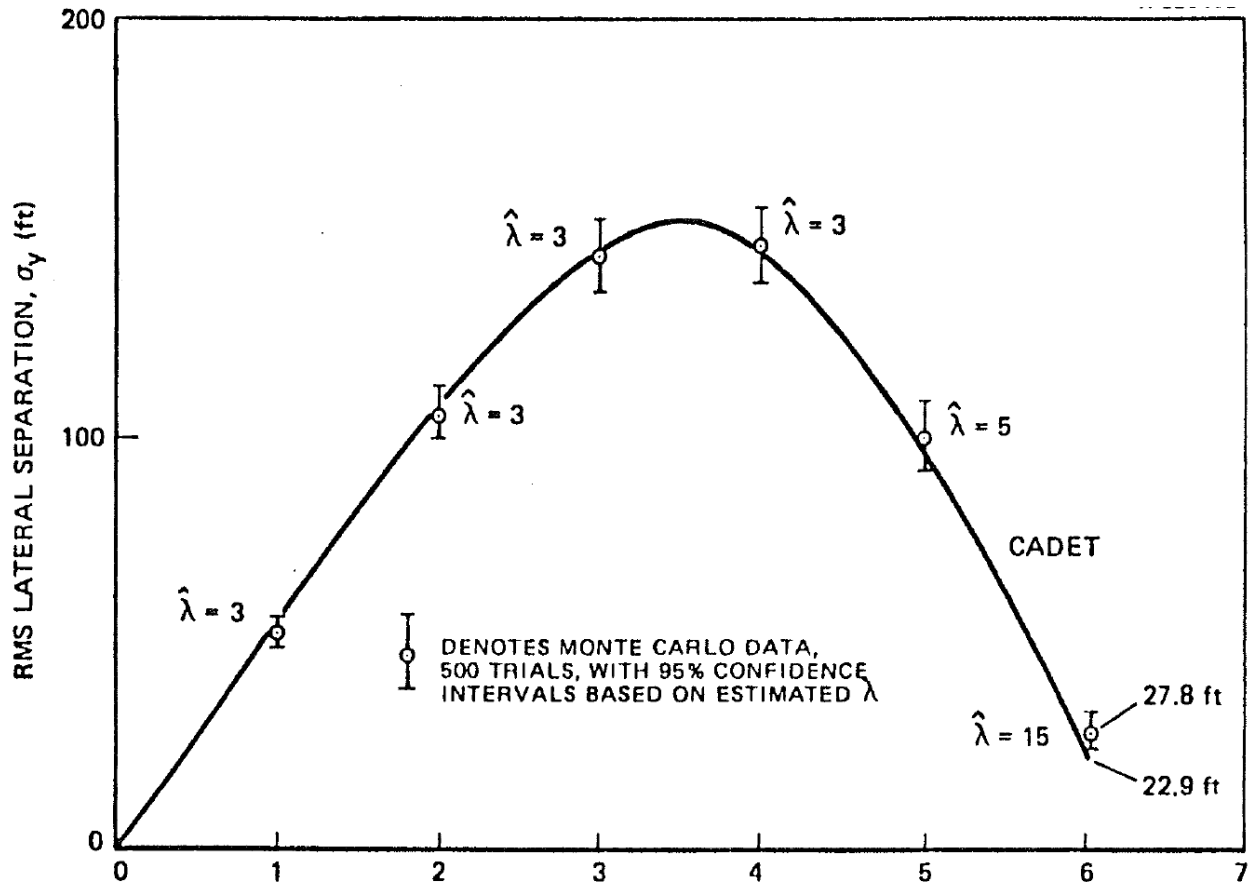


The control algorithm is called “proportional navigation”, i.e., the lateral acceleration guidance command is proportional to the line-of-sight angular rate; note that the guidance gains increase as the time-to-intercept (T_{GO}) decreases.

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

Missile-Target Intercept Problem (Cont'd)

D. Evolution of σ_y and λ_y vs Time

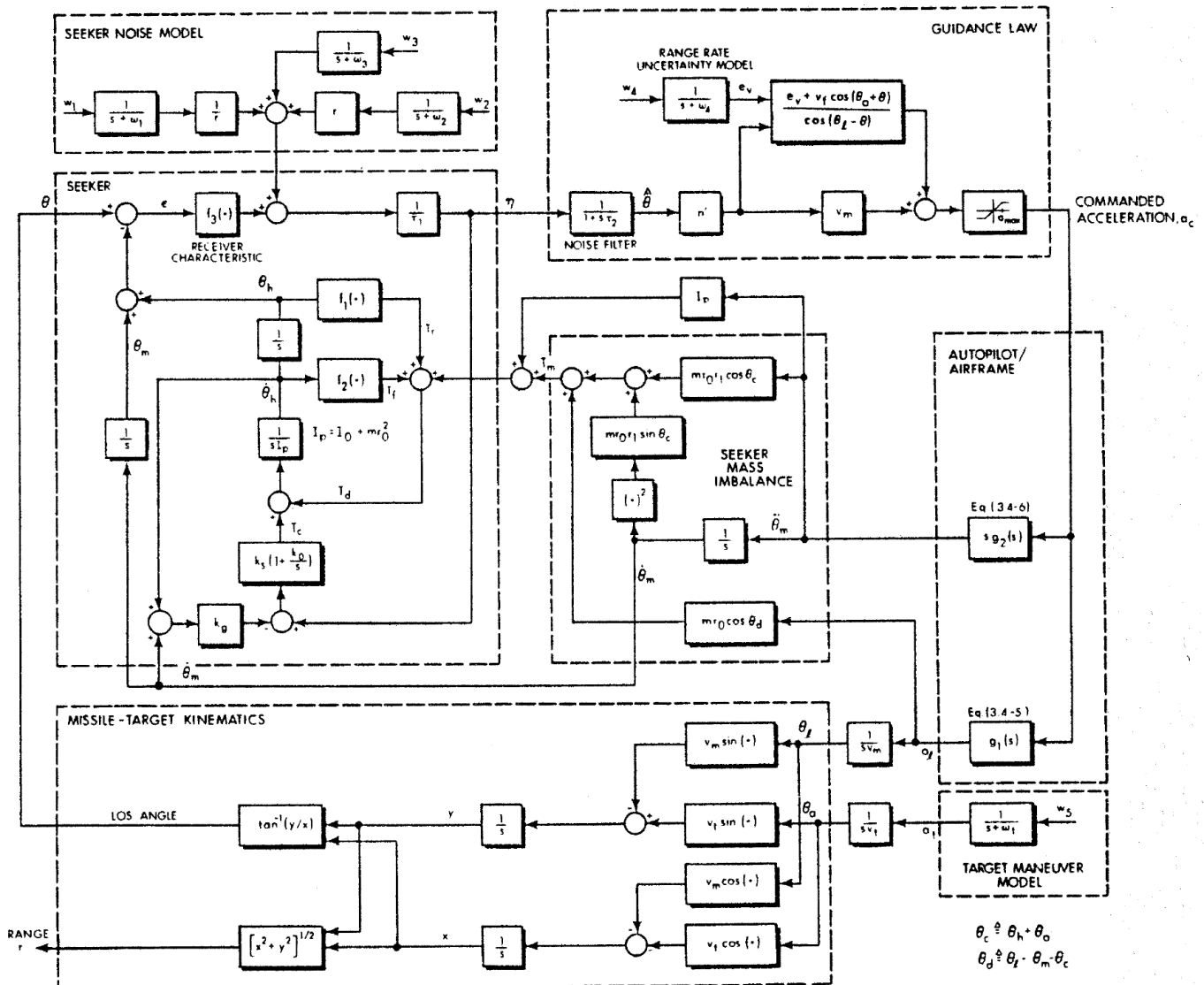


Clearly the RMS initial pointing error of the missile was not well aligned with the collision-course triangle; the missile often had to turn at its maximum rate until the RMS miss distance started to decrease. At the end of the six-second engagement the RMS miss distance is quite small (assuming the missile has a warhead...).

Missile-Target Intercept Problems

The US Navy sponsored a three-year effort to develop CADET for more realistic missile-target intercept problems; I investigated many aspects of missile guidance systems analysis and design, including nonlinearities in the “seeker”, guidance system, airframe, kinematics

...



In other words, a real-world problem (in many senses).

Appraisal of the RIDF Method

- The RIDF method is a powerful but approximate approach for evaluating the performance statistics of complex nonlinear systems.
- Monte Carlo analysis is also approximate, and we have seen that we must perform many trials and look at the state variable histograms before we can “trust” the results.
- The combined use of the two approaches is the most effective way to assess nonlinear system performance statistics.

