RELEASABLE

PHASE I OF THE DEVELOPMENT OF THE DEMODULATOR PORTION OF A JTIDS RECEIVE TERMINAL

FINAL REPORT

VOLUME 2 OF 2

MCS MILLER COMMUNICATIONS SYSTEMS LTD.

PHASE I OF THE DEVELOPMENT OF THE DEMODULATOR PORTION OF A JTIDS RECEIVE TERMINAL

FINAL REPORT
VOLUME 2 OF 2

MCS File No. 8231
SSC Contract No. OST81-00142
SSC File No. 21ST.36001-1-1401
Date: April 21, 1982

Prepared by: S. Crozier, B. Mazur
S. Crozier B. Mazur

Approved by: R Matyas
R. Matyas

SUBMITTED BY

MCS MILLER COMMUNICATIONS SYSTEMS LTD.
300 Legget Drive,
Kanata, Ontario,
Canada K2K 1Y5

APPENDIX A

INTERMEDIATE SIMULATION RESULTS FOR DMSK FILTER EVALUATION

The figure numbers have been labeled using the format A.X-Y where Y is the figure number in the category A.X, and the categories are as follows:

A.1 Receive Filter

A.2 Post-Demodulation Filter

A.3 Transmit filter

The parameters given with each figure are directly related to the simulation input parameters. The upper and lower curves in each figure refer to the performance of conventional DMSK and with SEC respectively.

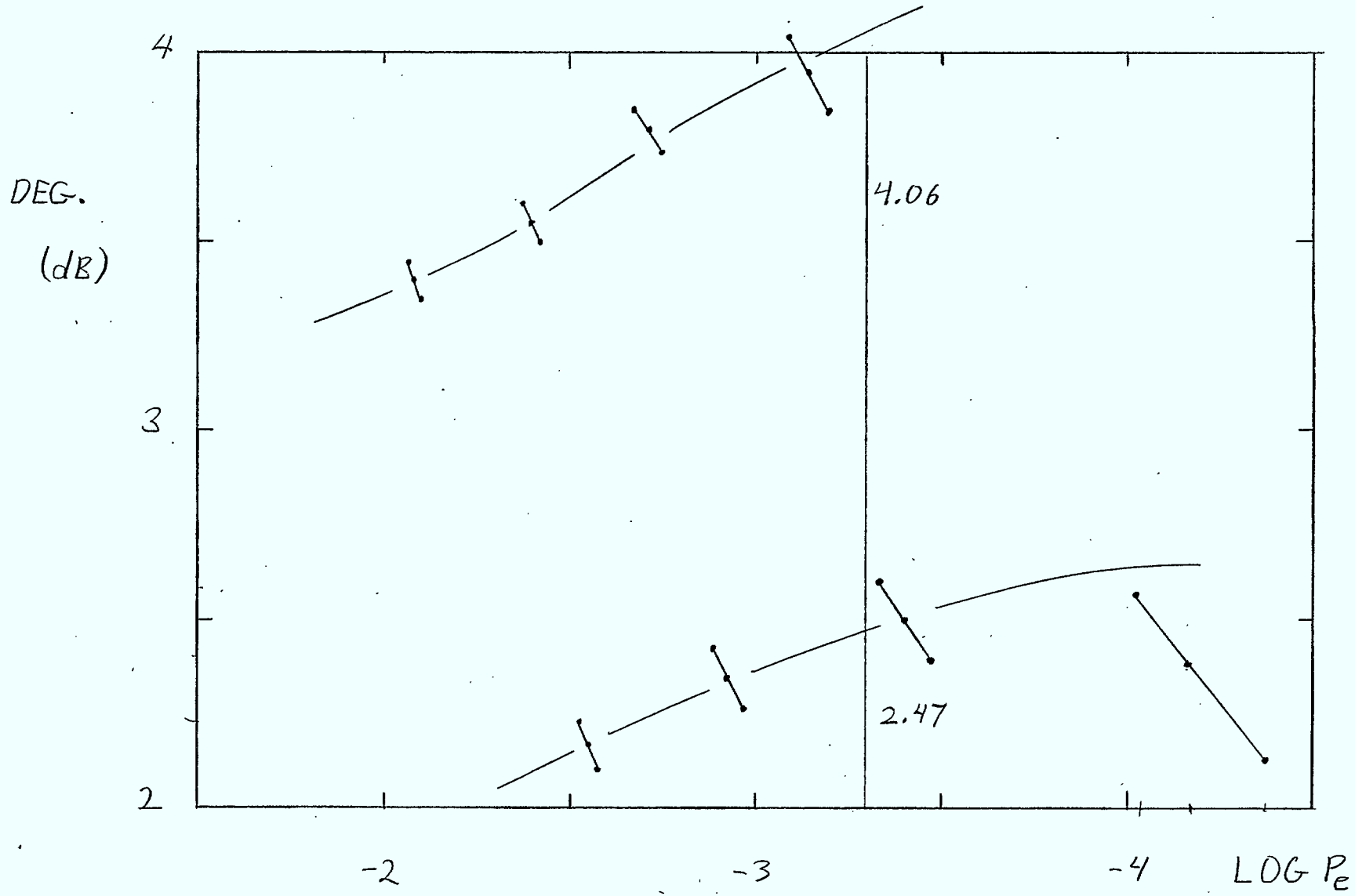

FIGURE A.1-1 Simulated DMSK With Gaussian Receive Filter (BT = 0.7)

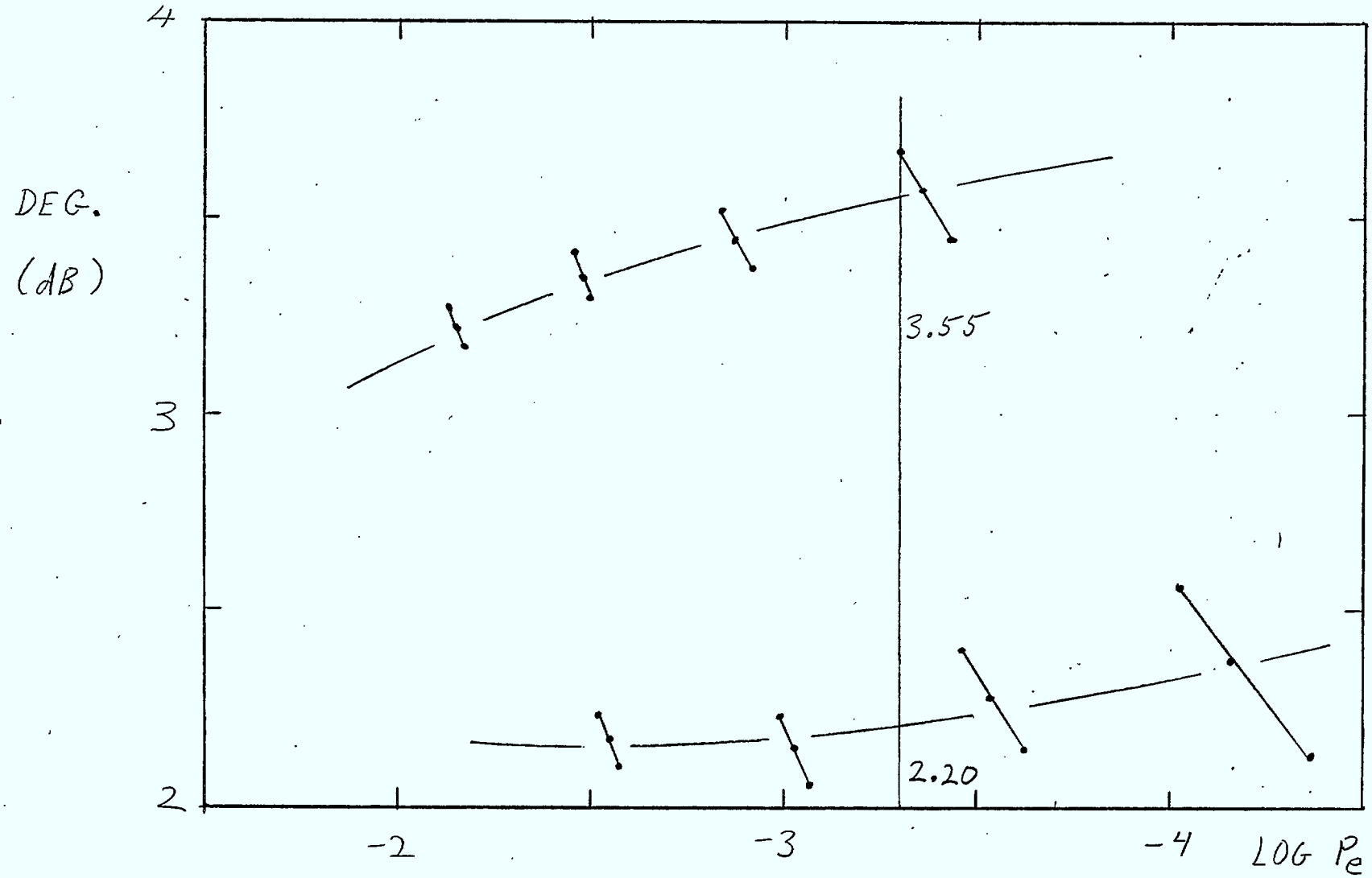

FIGURE A.1-2 Simulated DMSK With Gaussian Receiver Filter (BT=0.8)

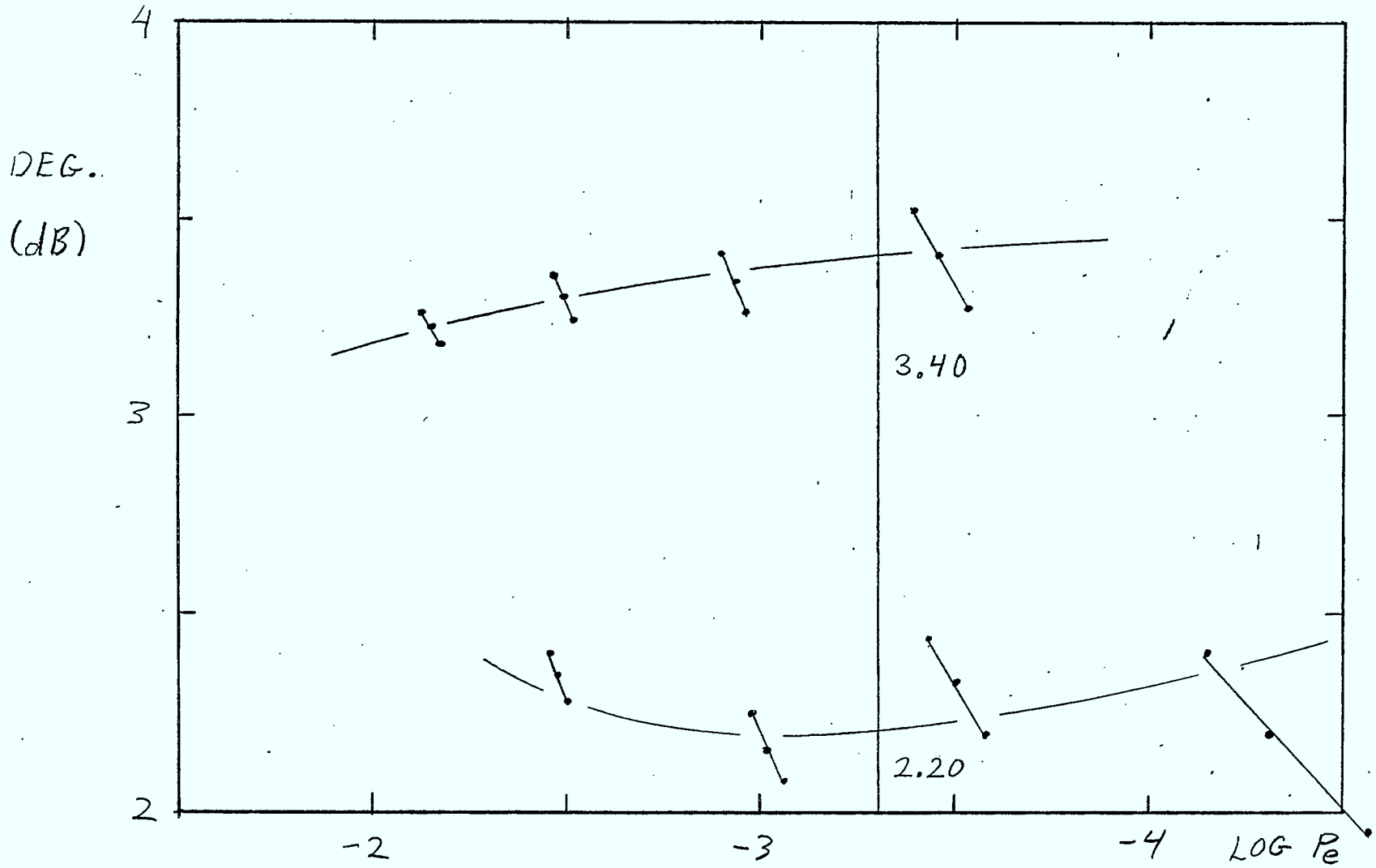

FIGURE A.1-3 Simulated Dmsk with Gaussian Receiver Filter (BT=0.9)

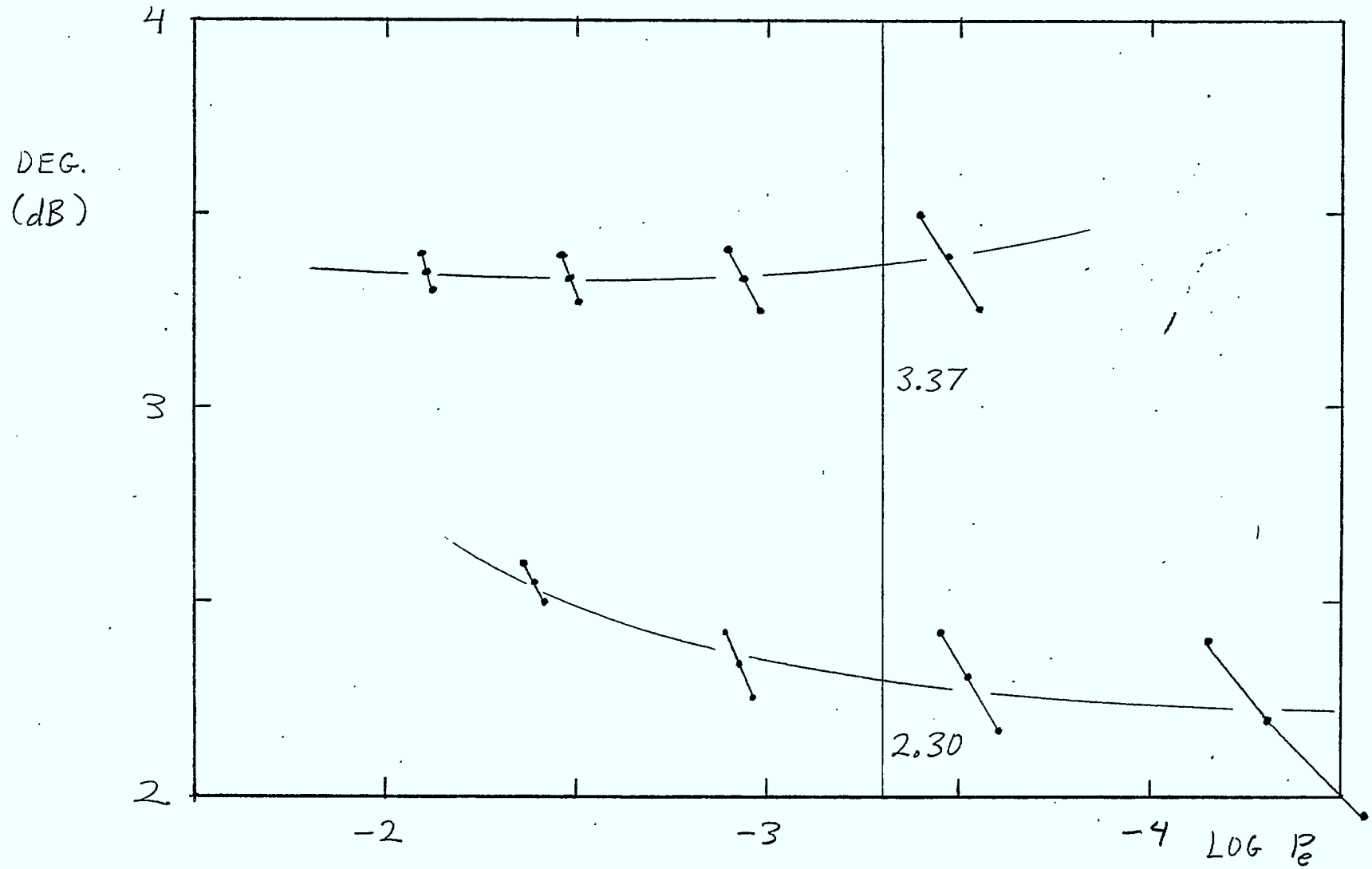

FIGURE H.1-4 Simulated DMSK With Gaussian Receiver Filter (BT=1.0)

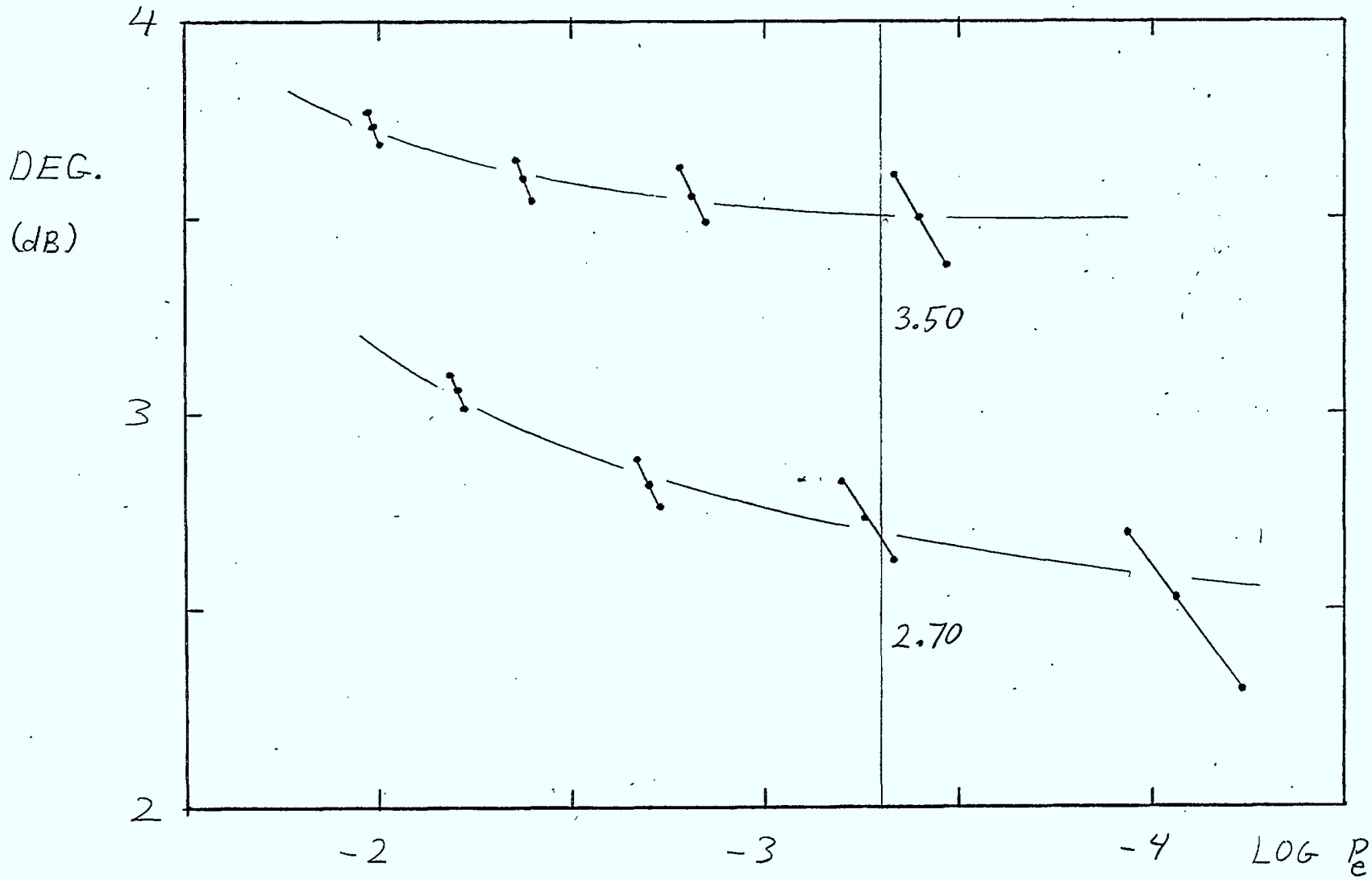

FIGURE A.1-5 Simulated DMSK with Gaussian Receiver Filter (BT= 1.2)

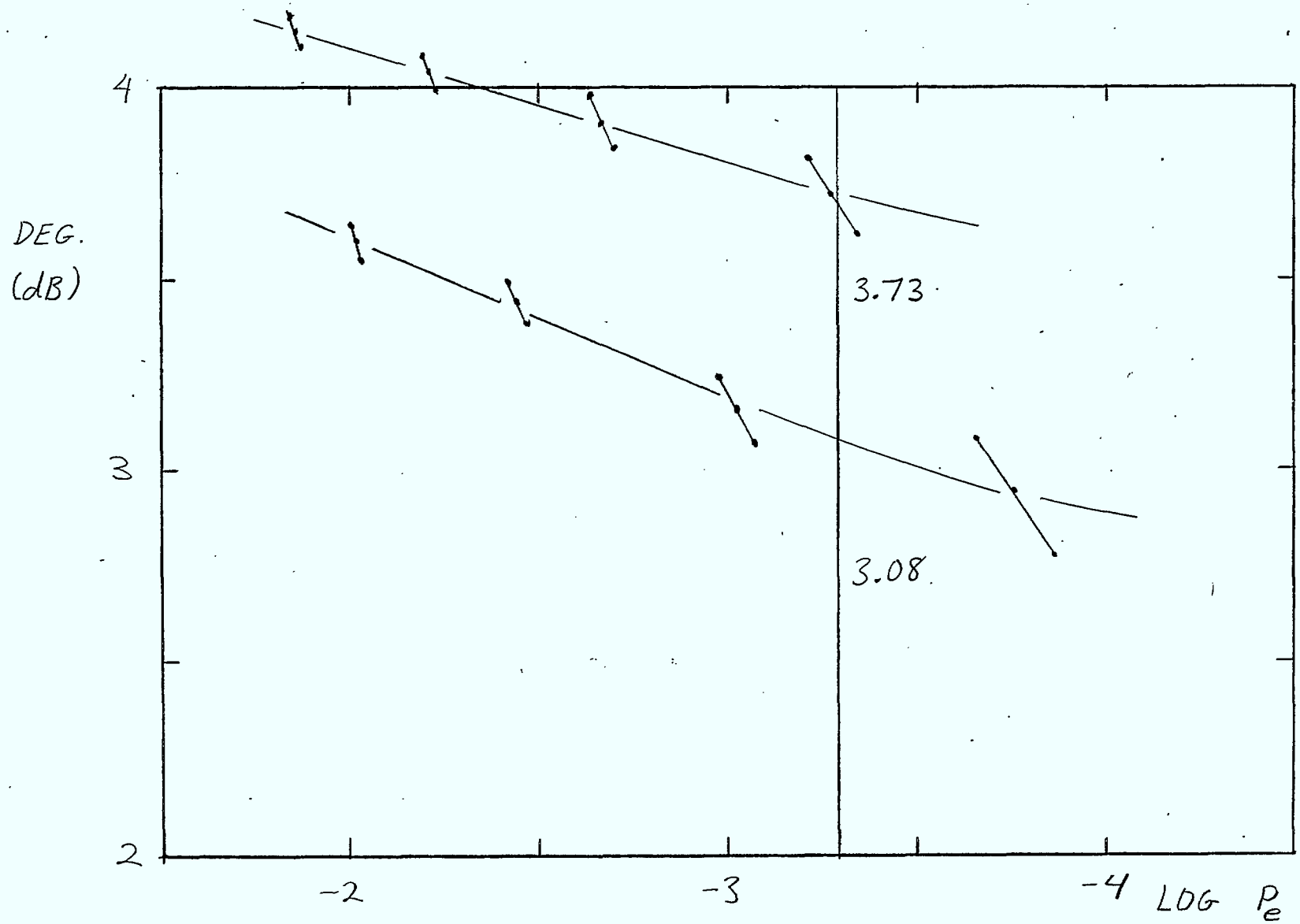

FIGURE A.1-6 Simulated DMSK with Gaussian Receiver Filter (BT=1.4)

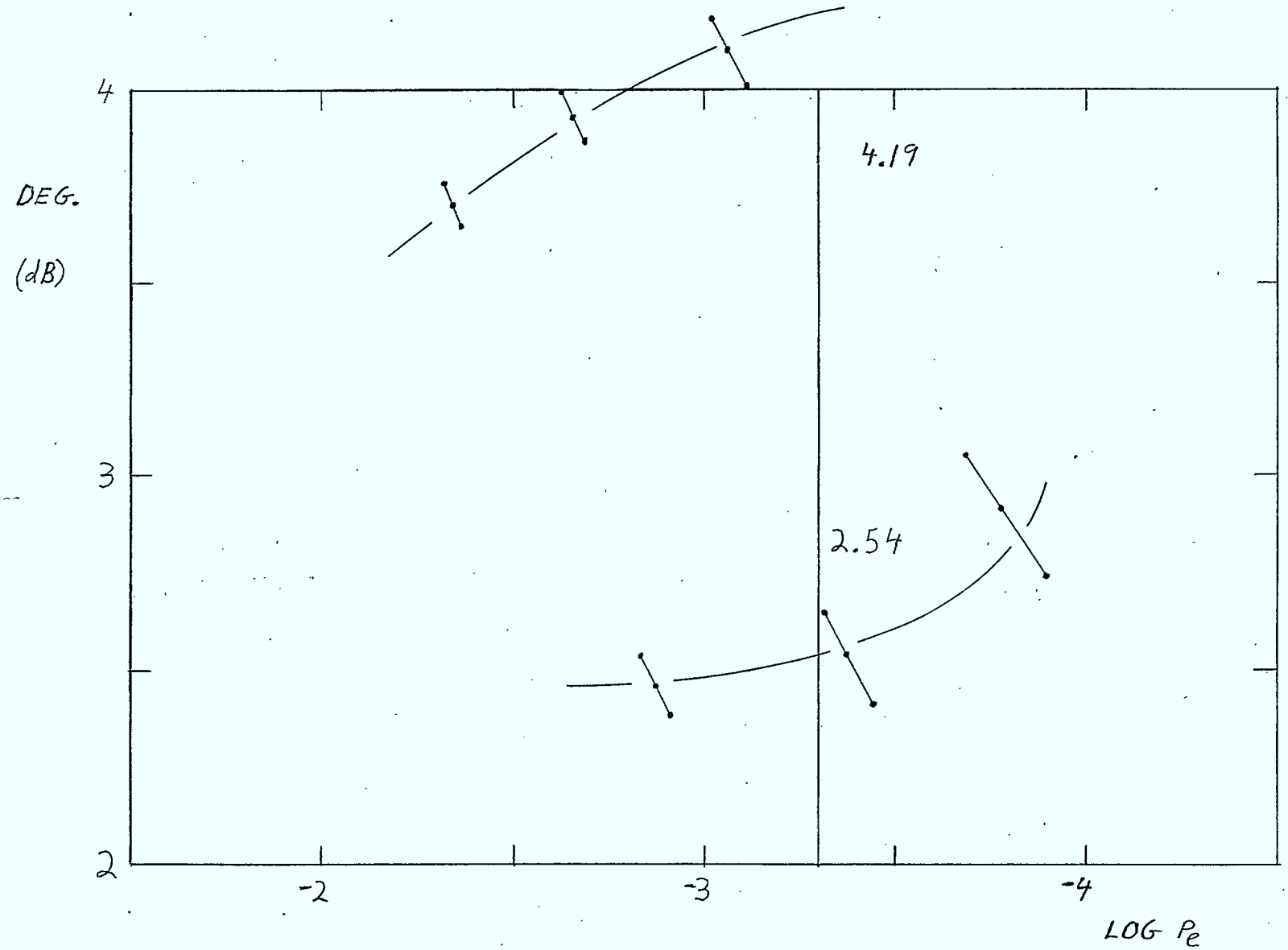

FIGURE A.1-7 Simulated DMSK With 2nd Order Butterworth Receiver Filter (BT = 0.8)

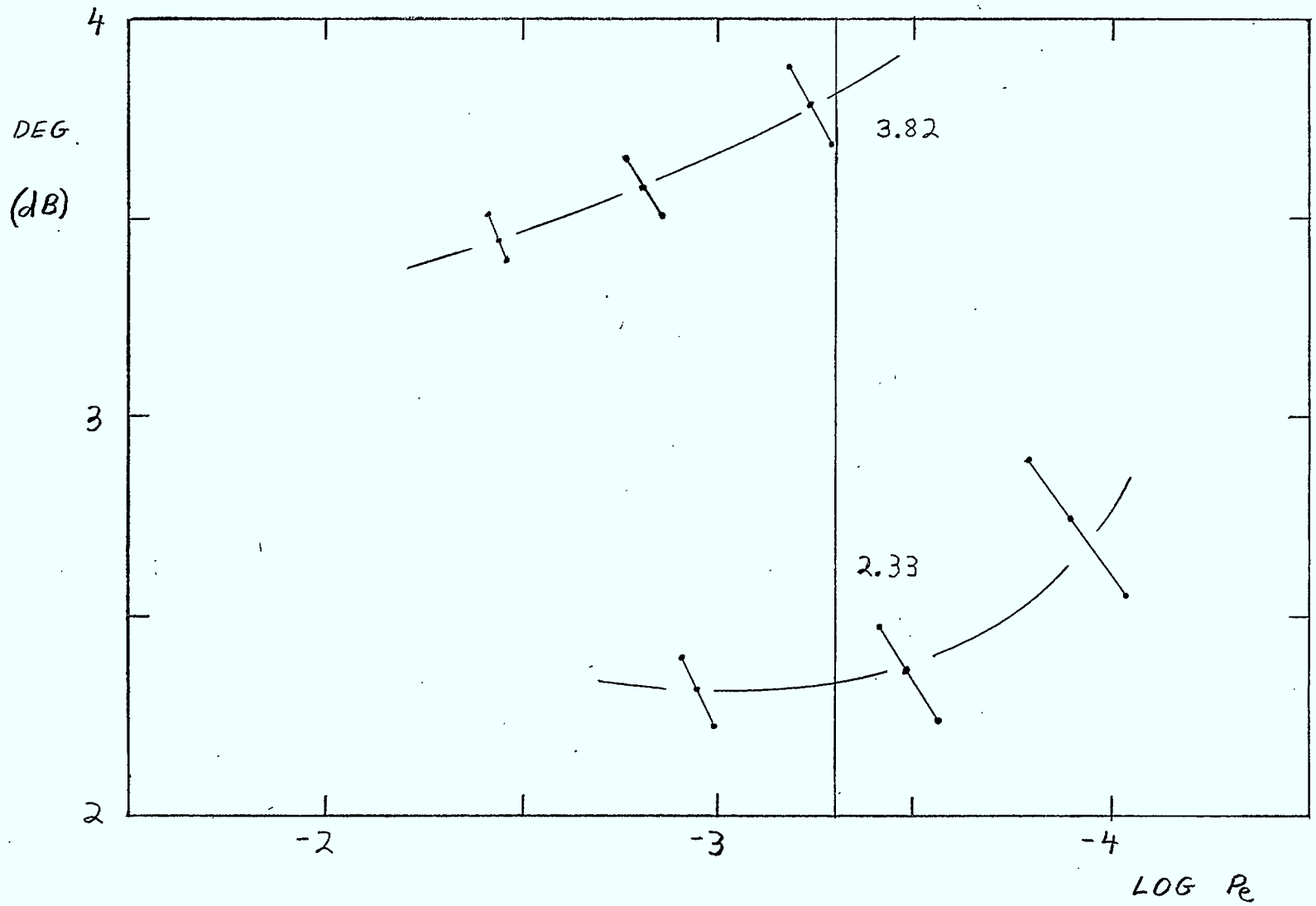

FIGURE A.1-8 Simulated DMSK With 2nd Order Butterworth Receiver Filter ($BT = 0.9$)

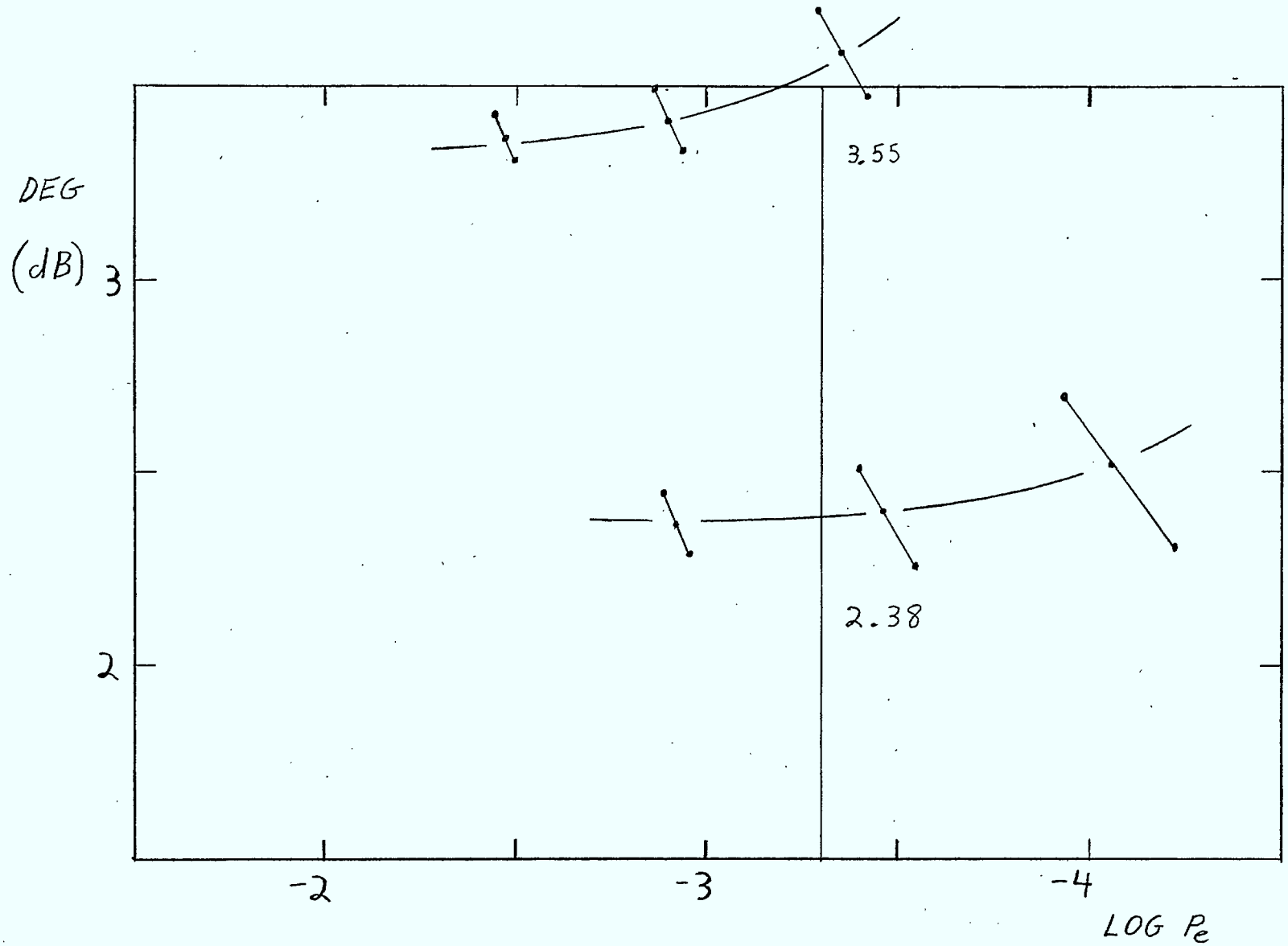

FIGURE A.1-9 Simulated DMSK With 2nd Order Butterworth Receiver Filter (BT = 1.0)

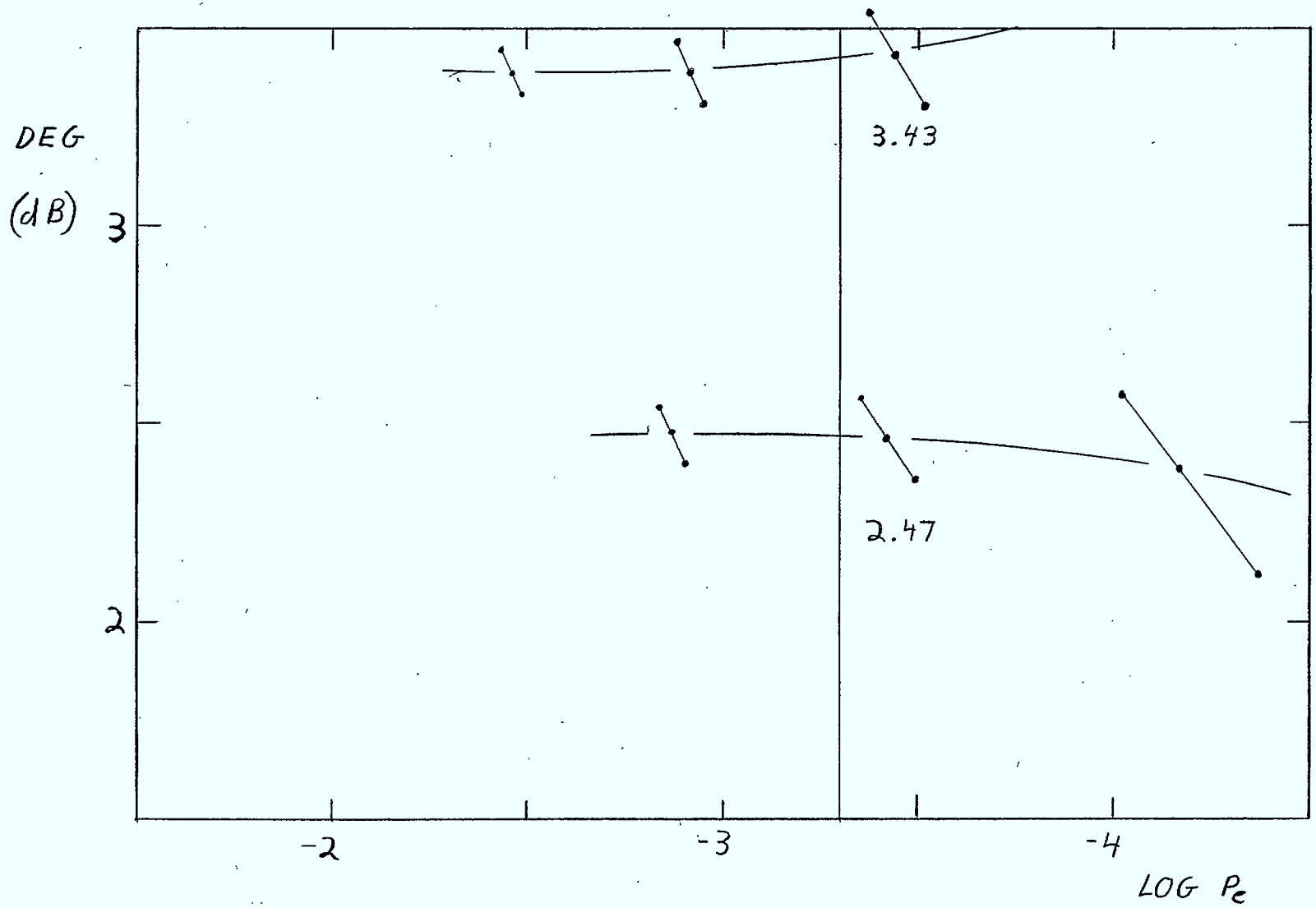

FIGURE A.1-10 Simulated DMSK with 2nd Order Butterworth Receiver Filter (BT = 1.1)

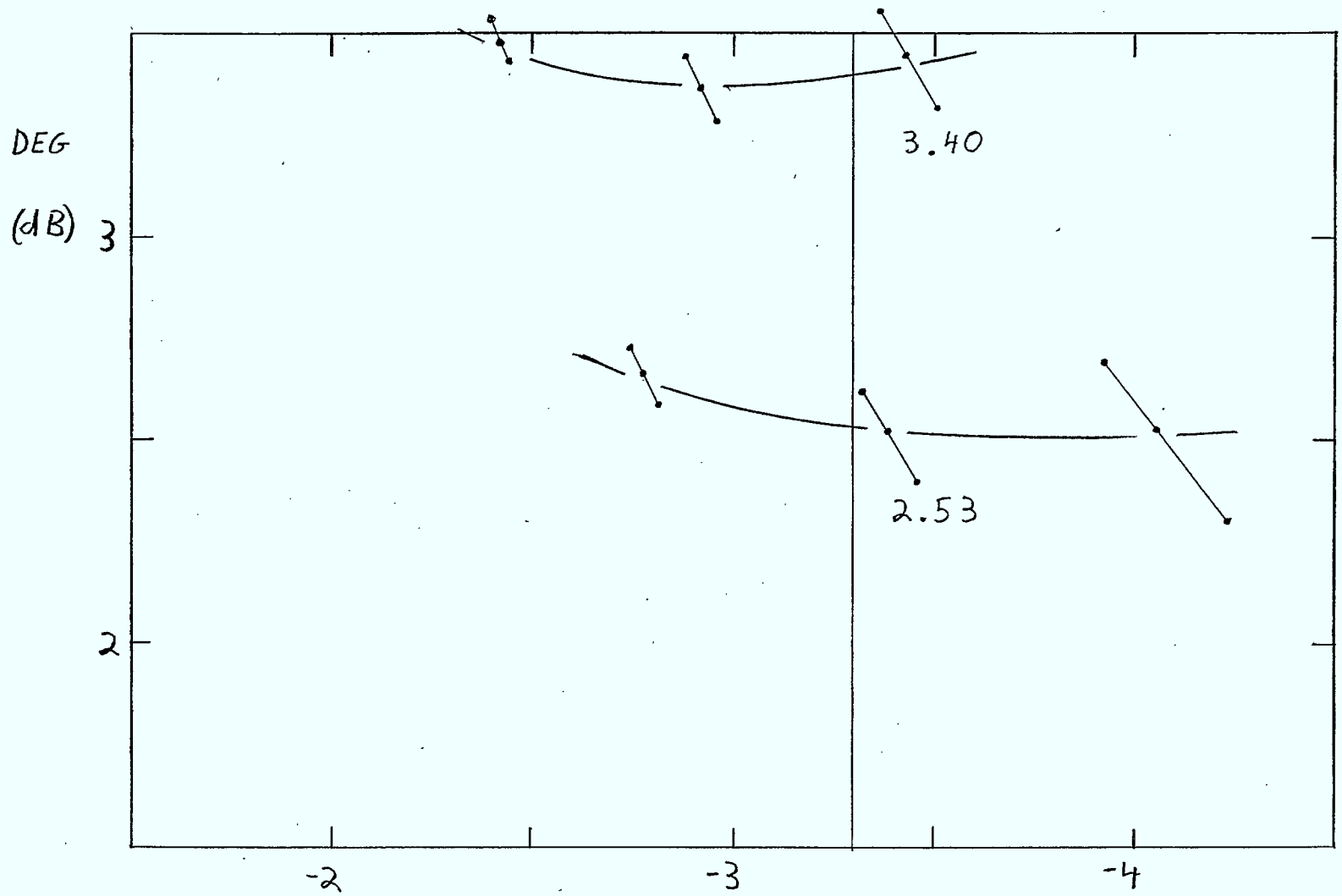

FIGURE A.1-11 Simulated DMSK With 2nd Order Butterworth Receiver Filter (BT = 1.2)

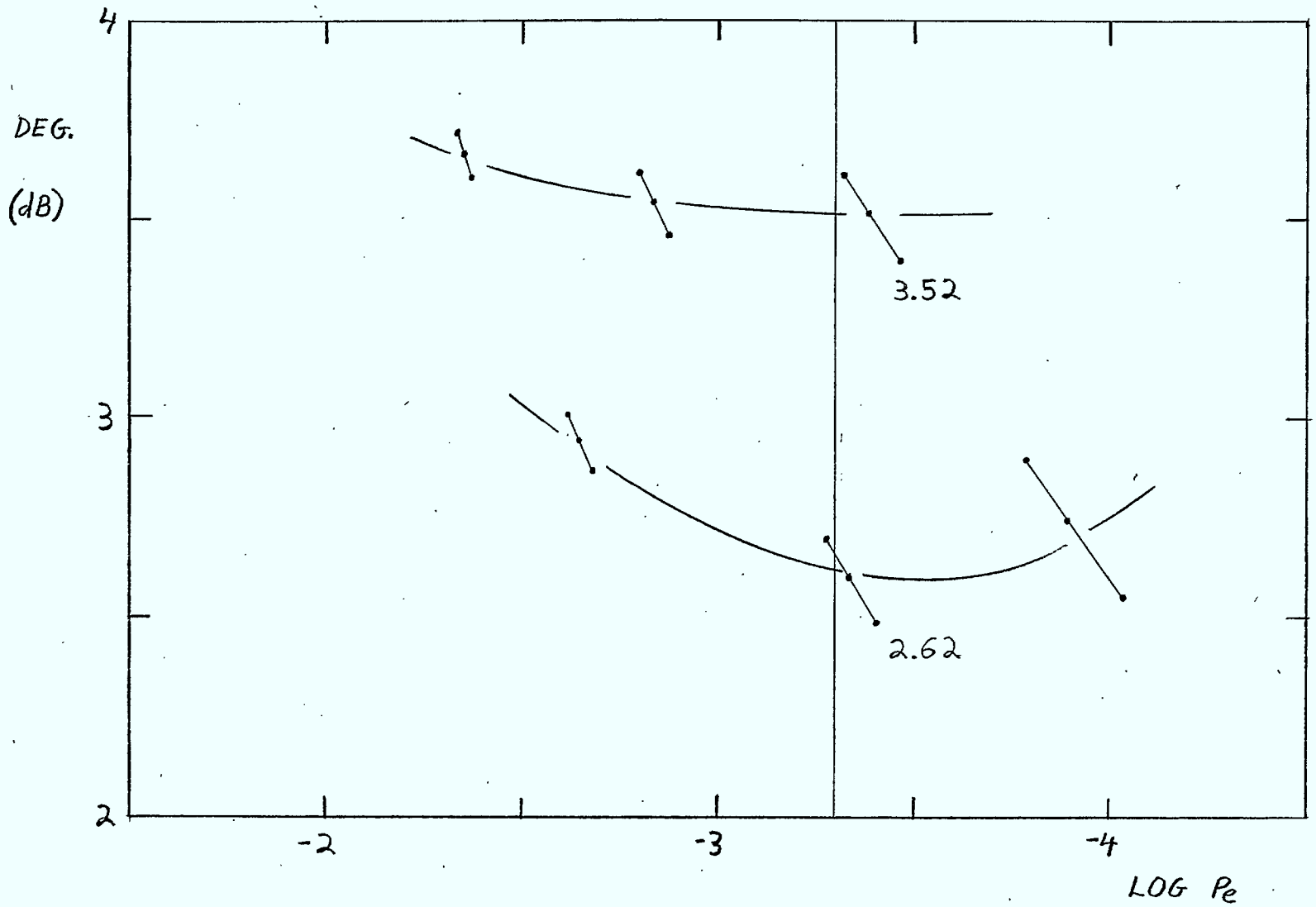

FIGURE A.1-12 Simulated DMSK With 2nd Order Butterworth Receiver Filter (BT= 1.3)

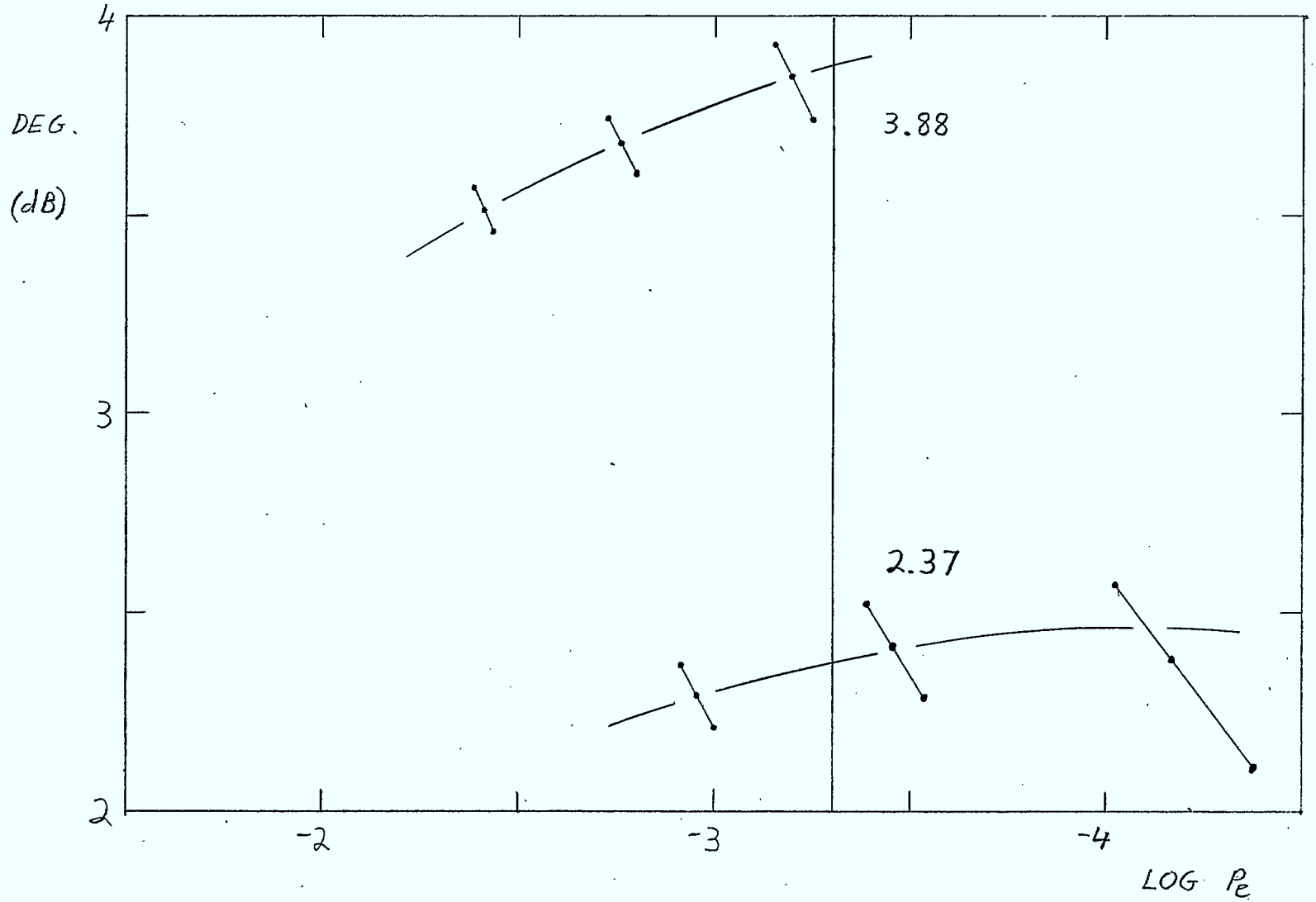

FIGURE A.1-13 Simulated DMSK With 2nd Order Butterworth Equalized Receiver Filter (BT = 0.8)

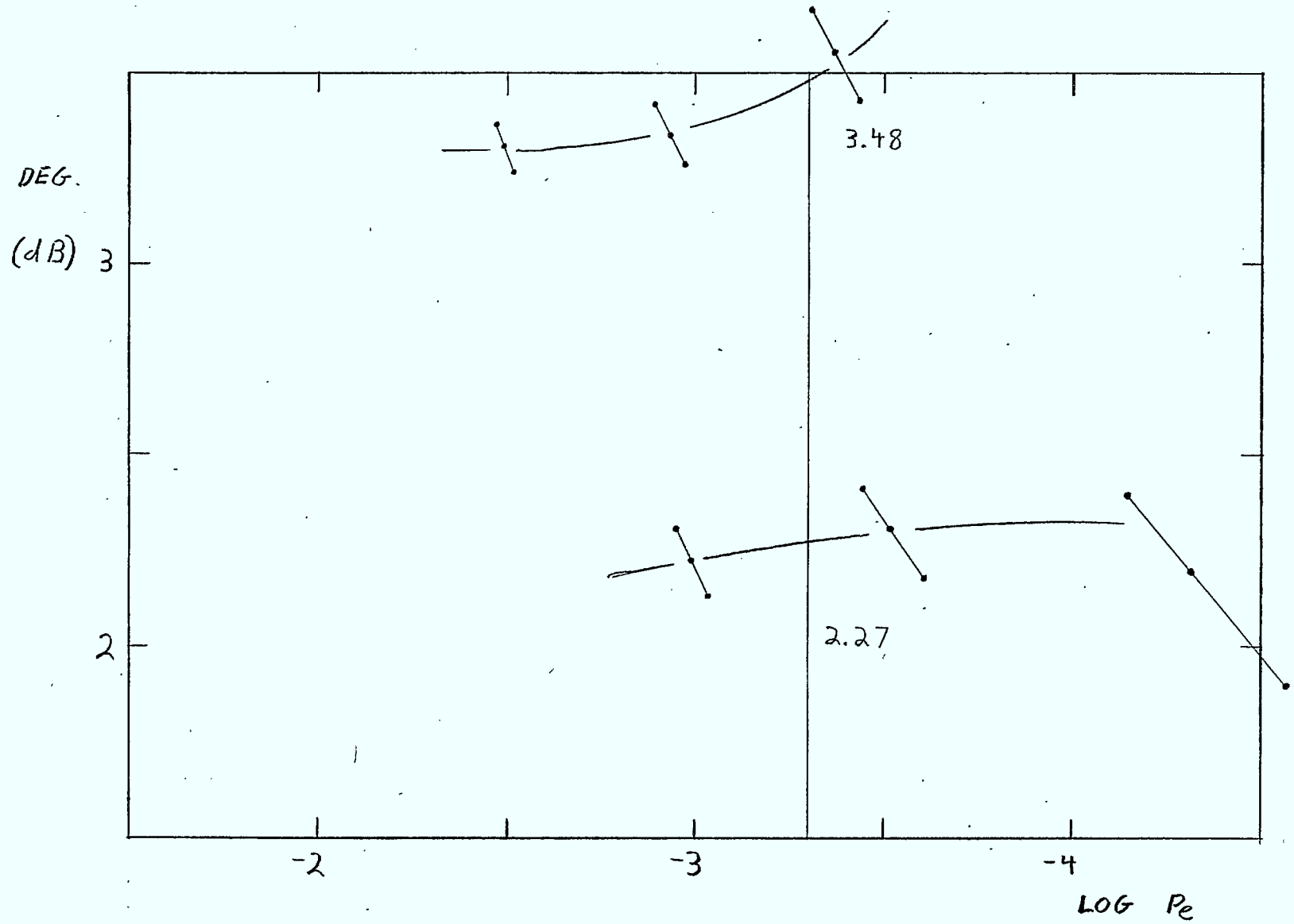

FIGURE A.1-14 Simulated DMSK With 2nd Order Butterworth Equalized Receiver Filter (BT = 0.9)

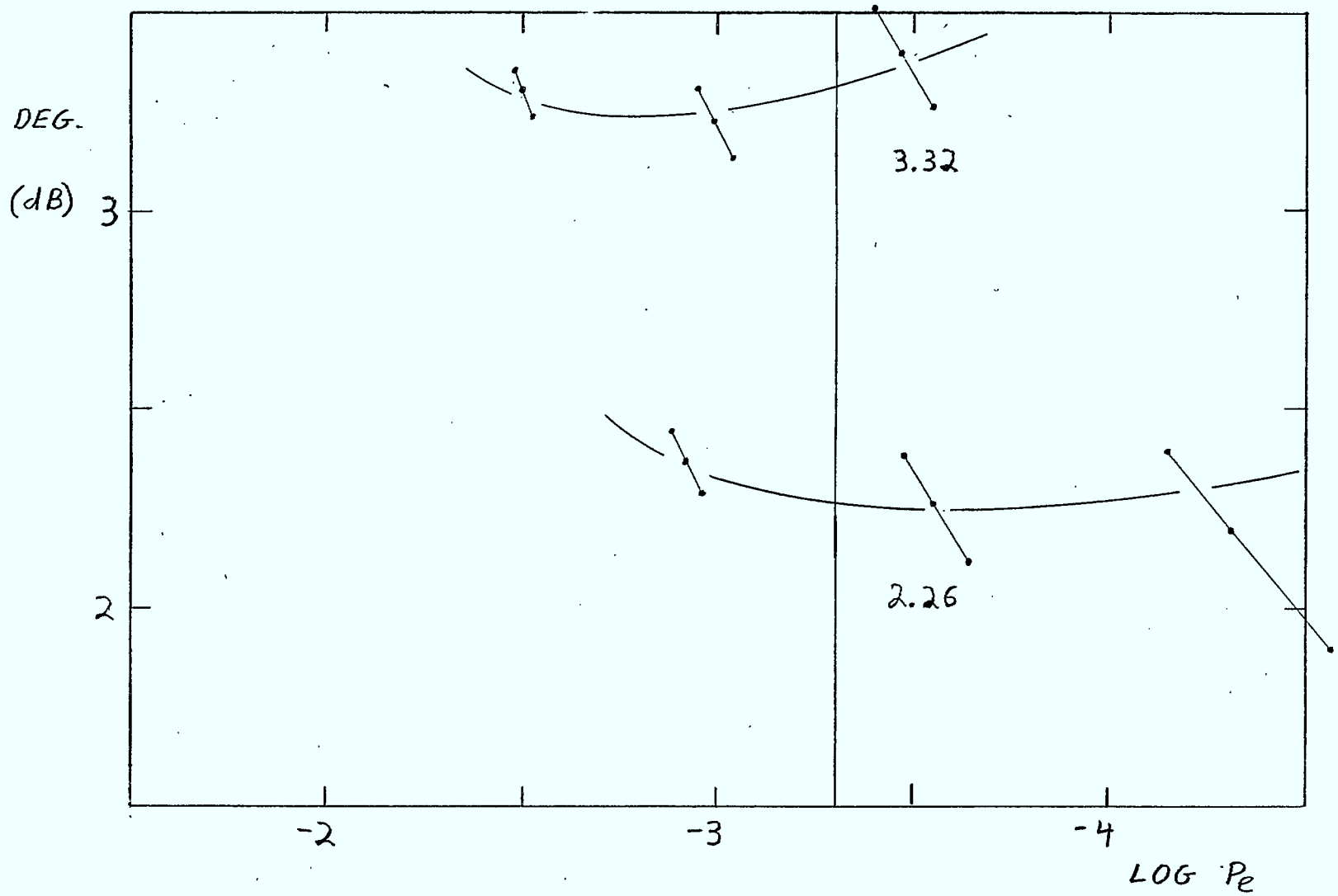

FIGURE A.1-15 Simulated DMSK With 2nd Order Butterworth Equalized Receiver Filter (BT=1.0)

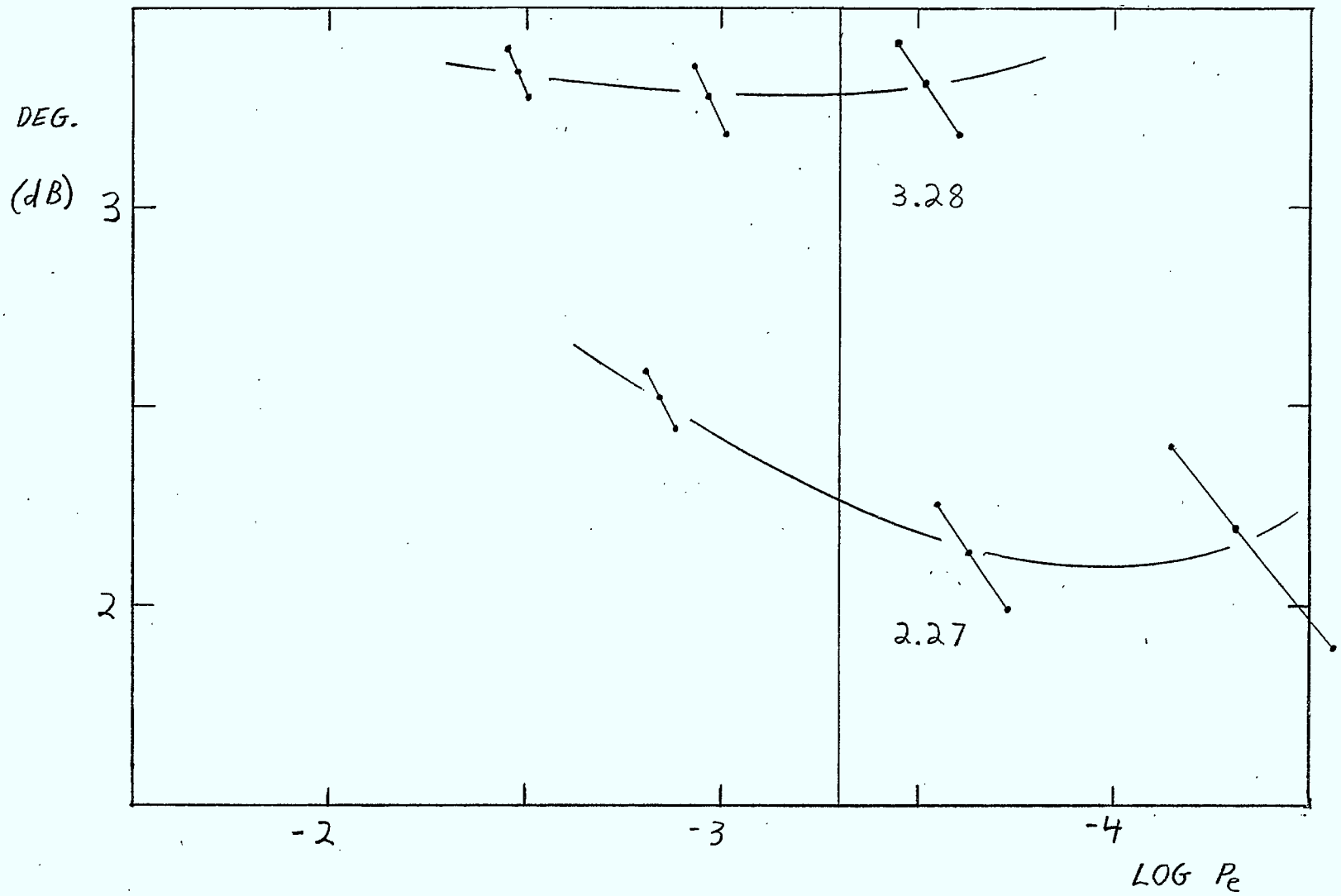

FIGURE A.1-16 Simulated DMSK With 2nd Order Butterworth Equalized Receiver Filter ($BT = 1.1$)

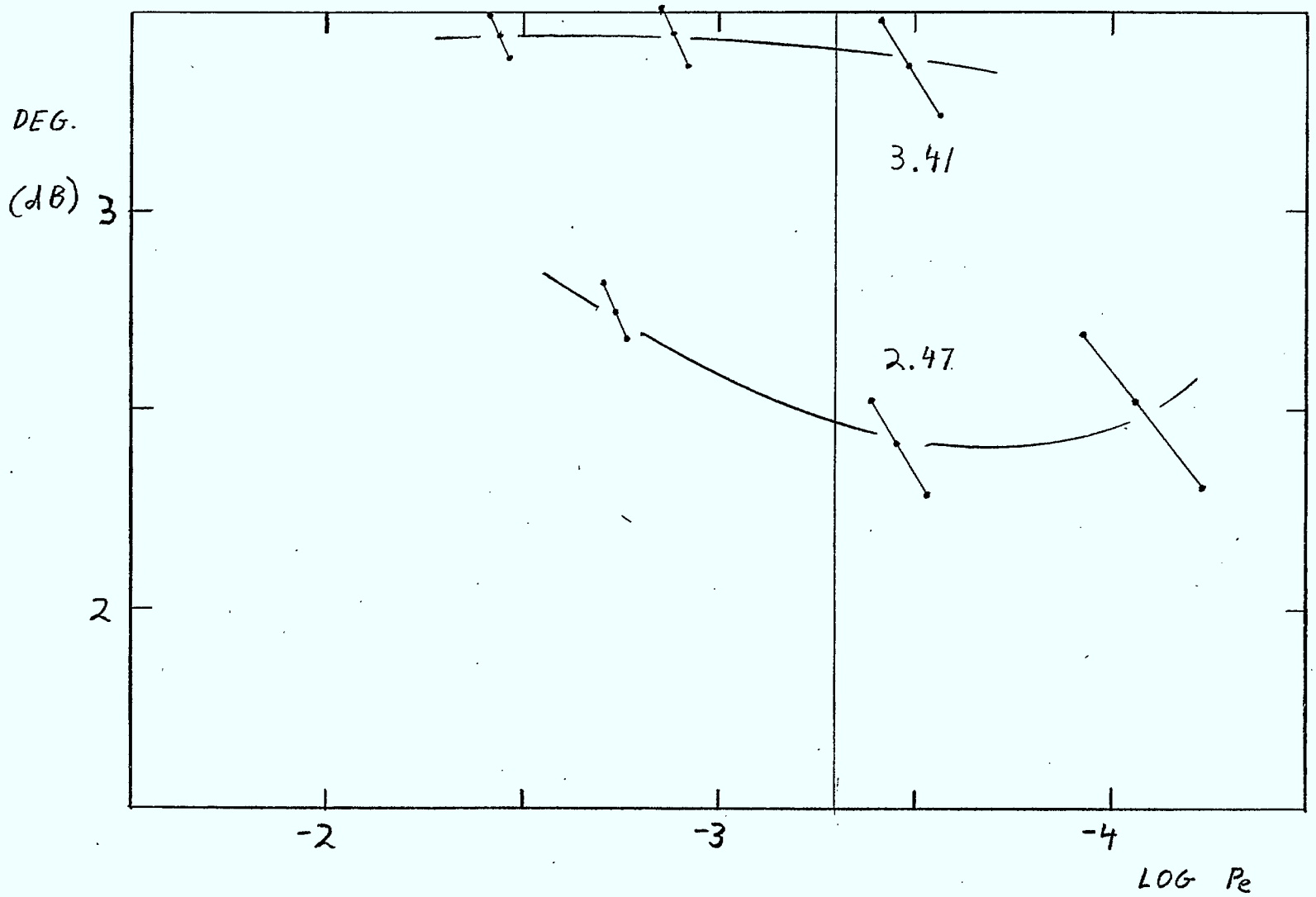

FIGURE A.1-17 Simulated DMSK With 2nd Order Butterworth Equalized Receiver Filter (BT = 1.2)

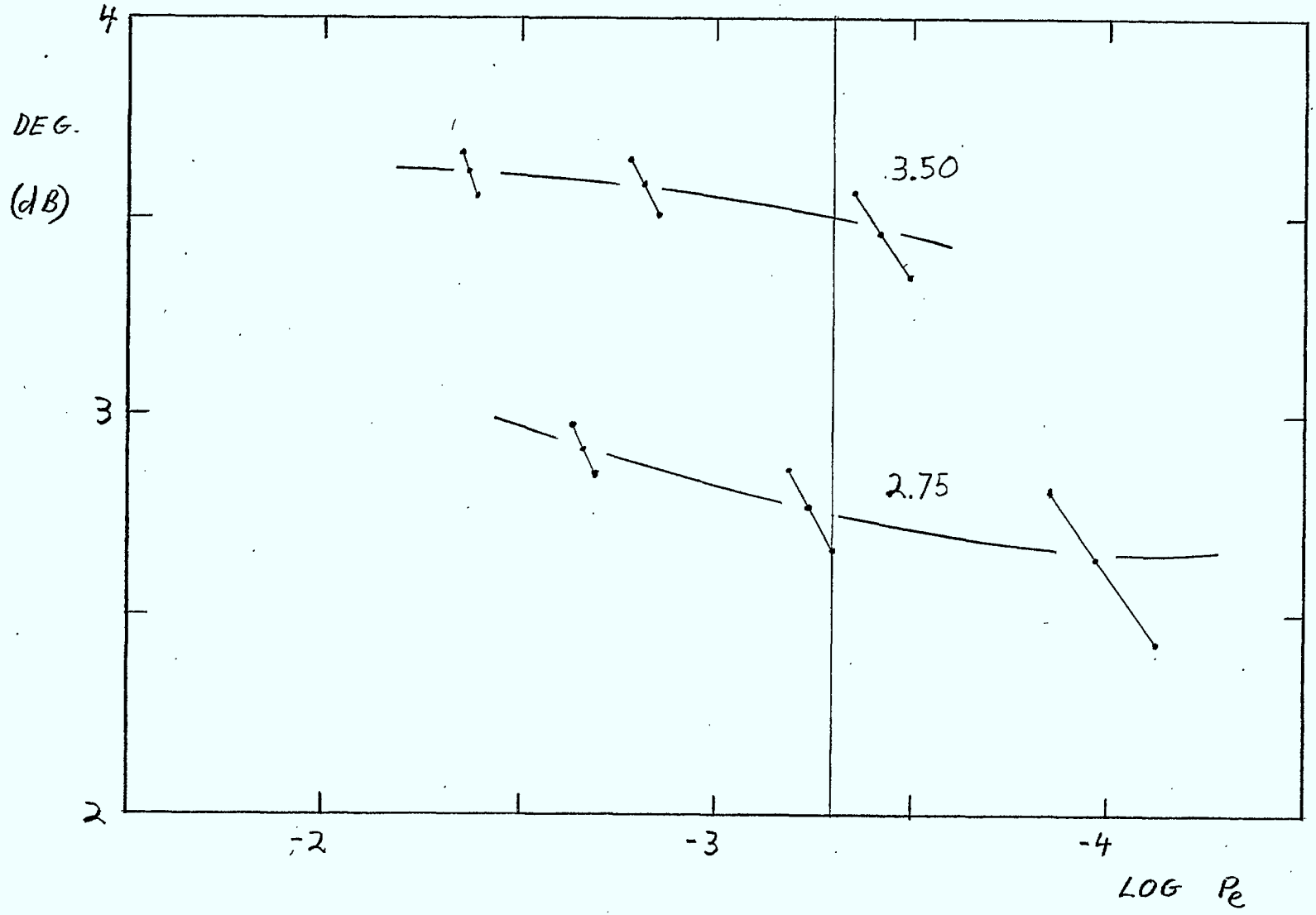

FIGURE A.1-18 simulated DMSK With 2nd Order Butterworth Equalized Receiver Filter (BT = 1.3)

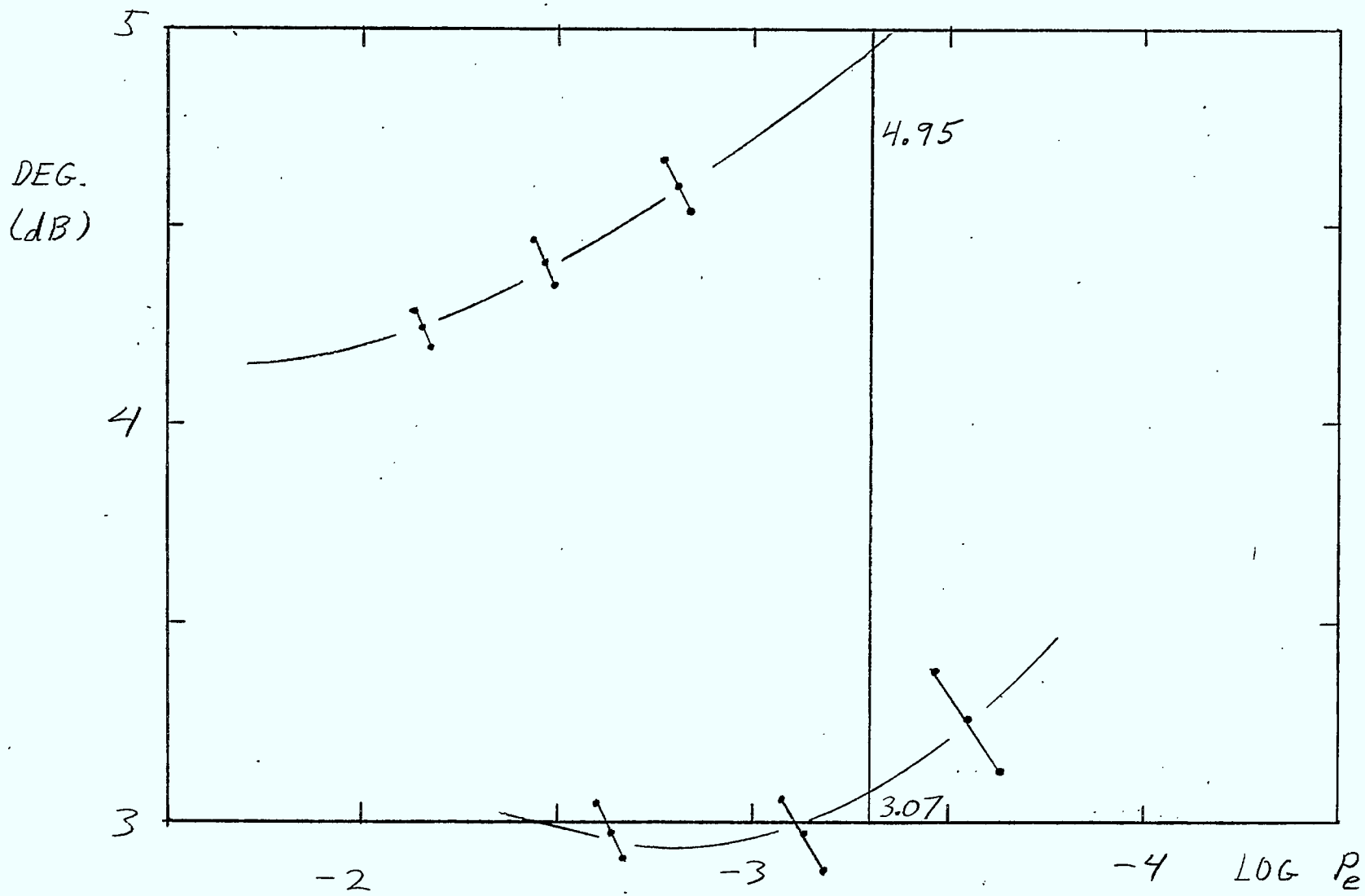

FIGURE A.1-19 Simulated DMSK With 4th Order Butterworth Receiver Filter (BT = 0.9)

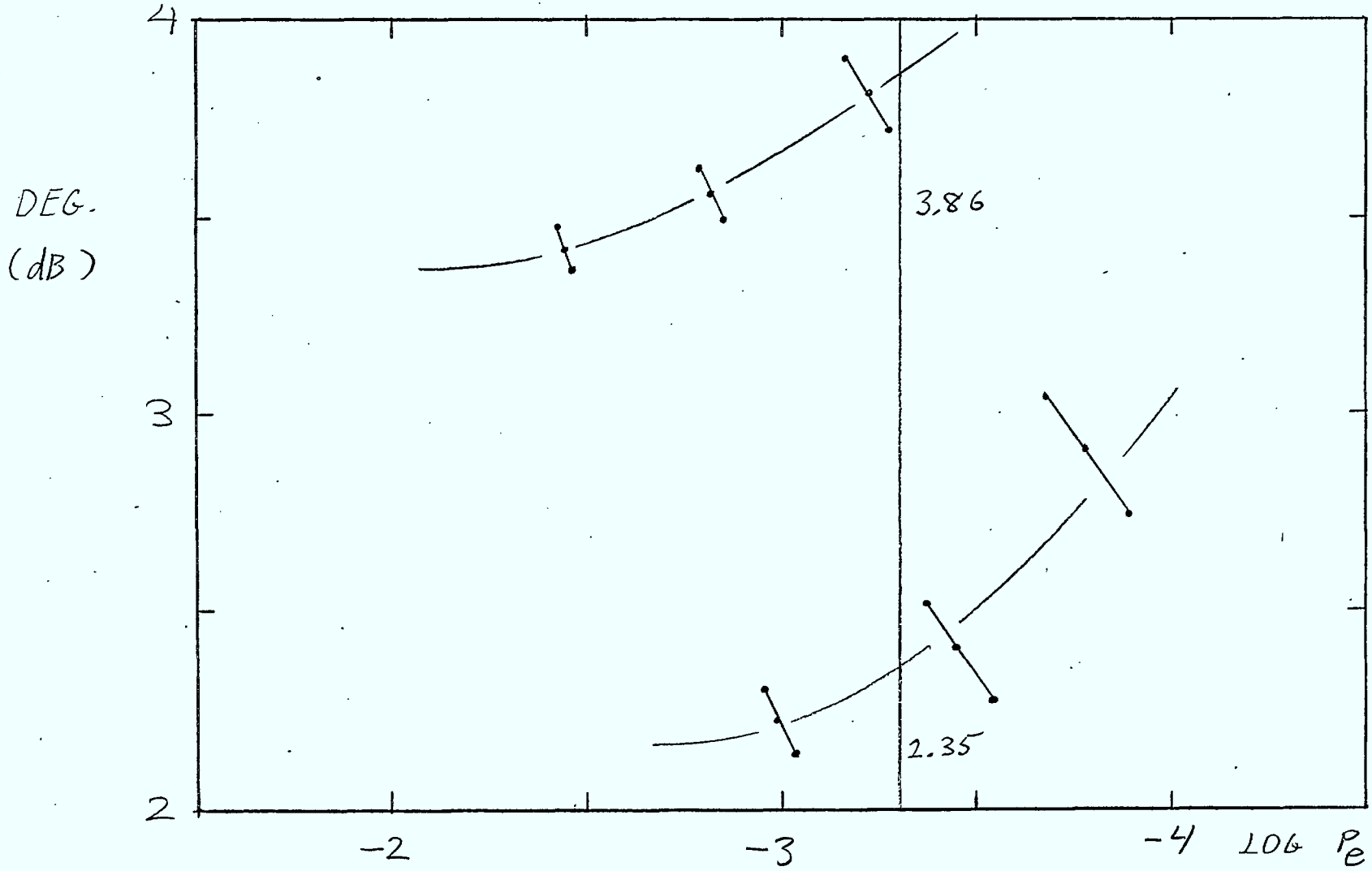

FIGURE A.1-20 Simulated DMSK with 4th Order Butterworth Receiver Filter (BT = 1.0)

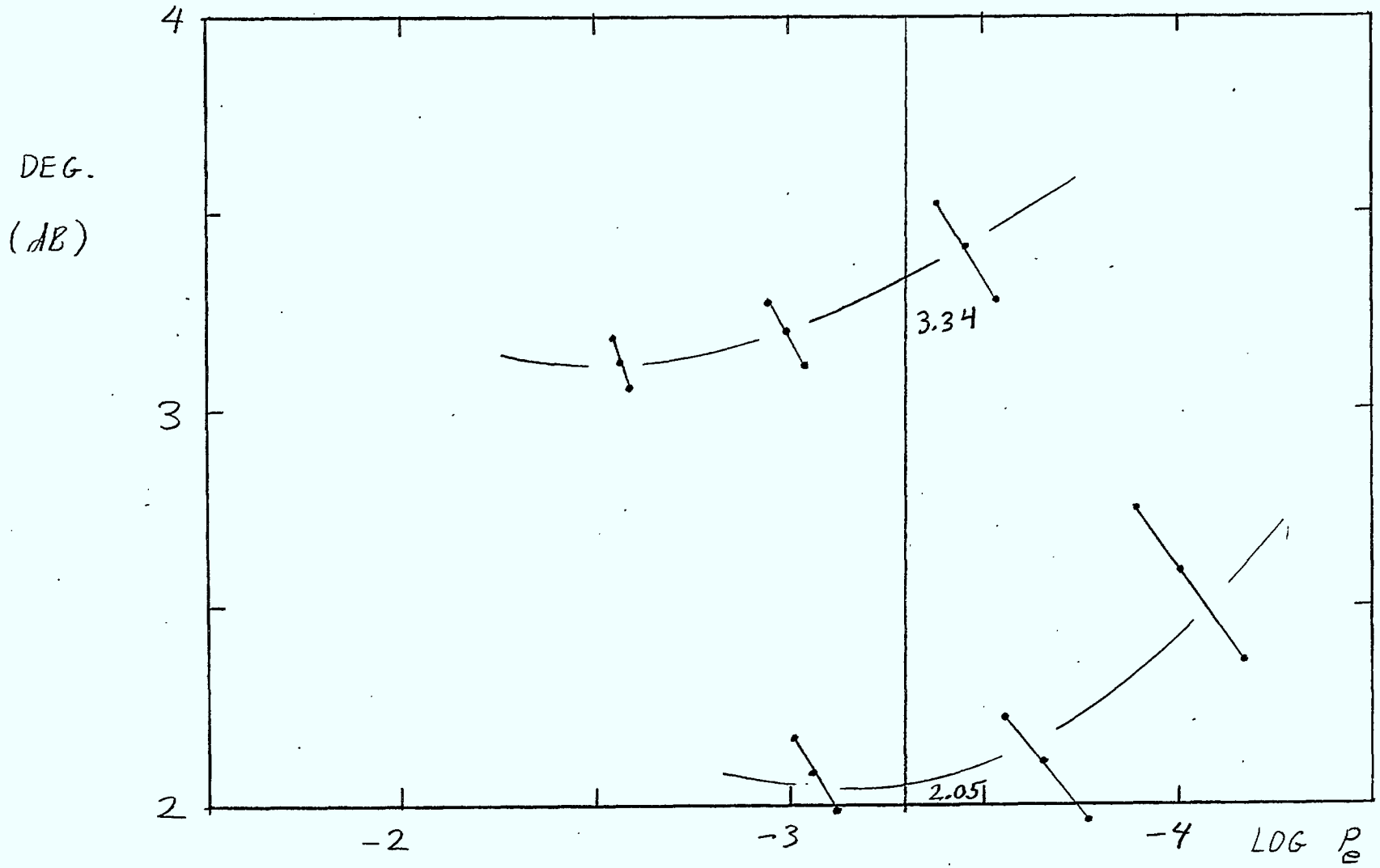

FIGURE A.1-21 Simulated DMSK with 4th Order Butterworth Receiver Filter (BT = 1.1)

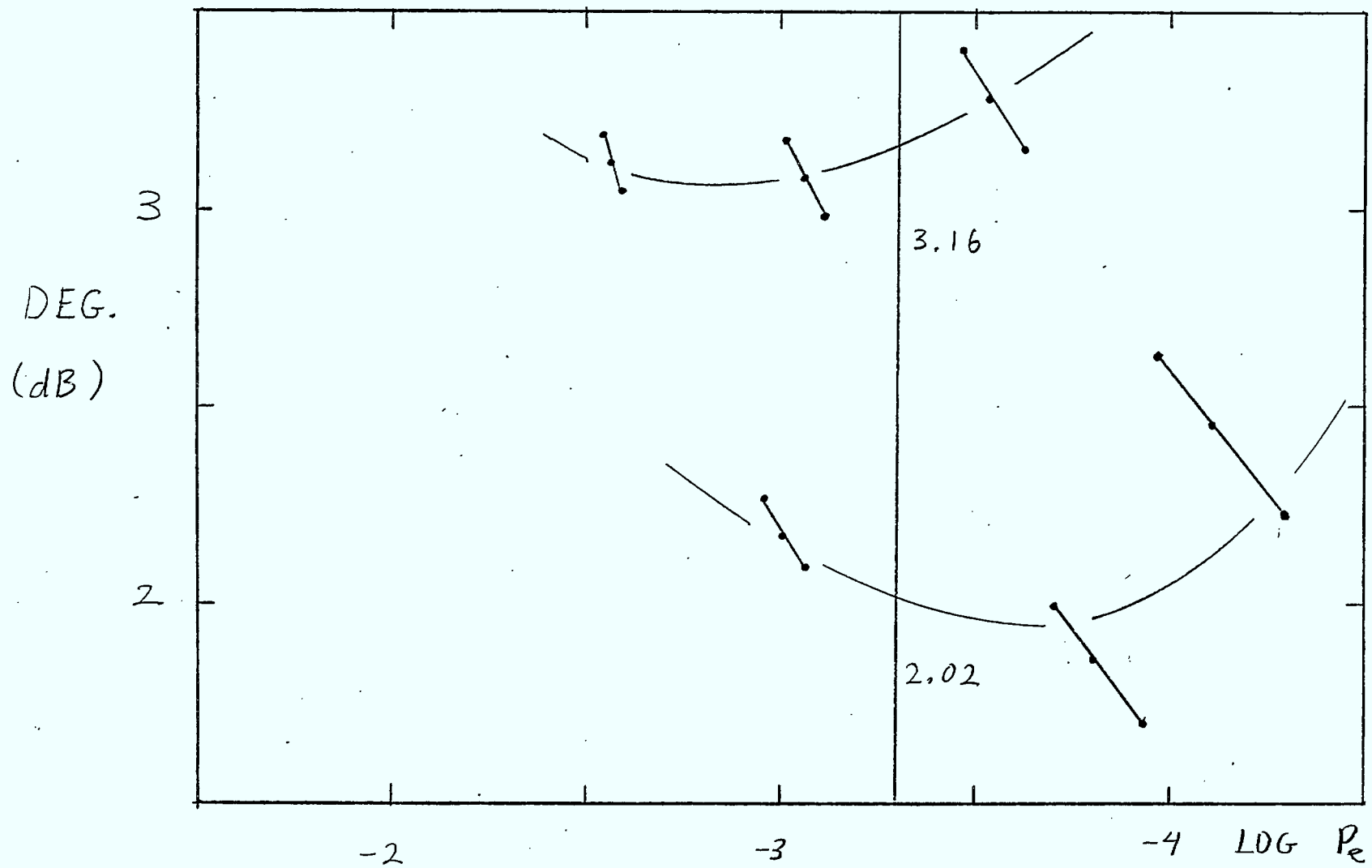

FIGURE A.1-22 Simulated DMSK with 4th Order Butterworth Receive Filter (BT = 1.2)

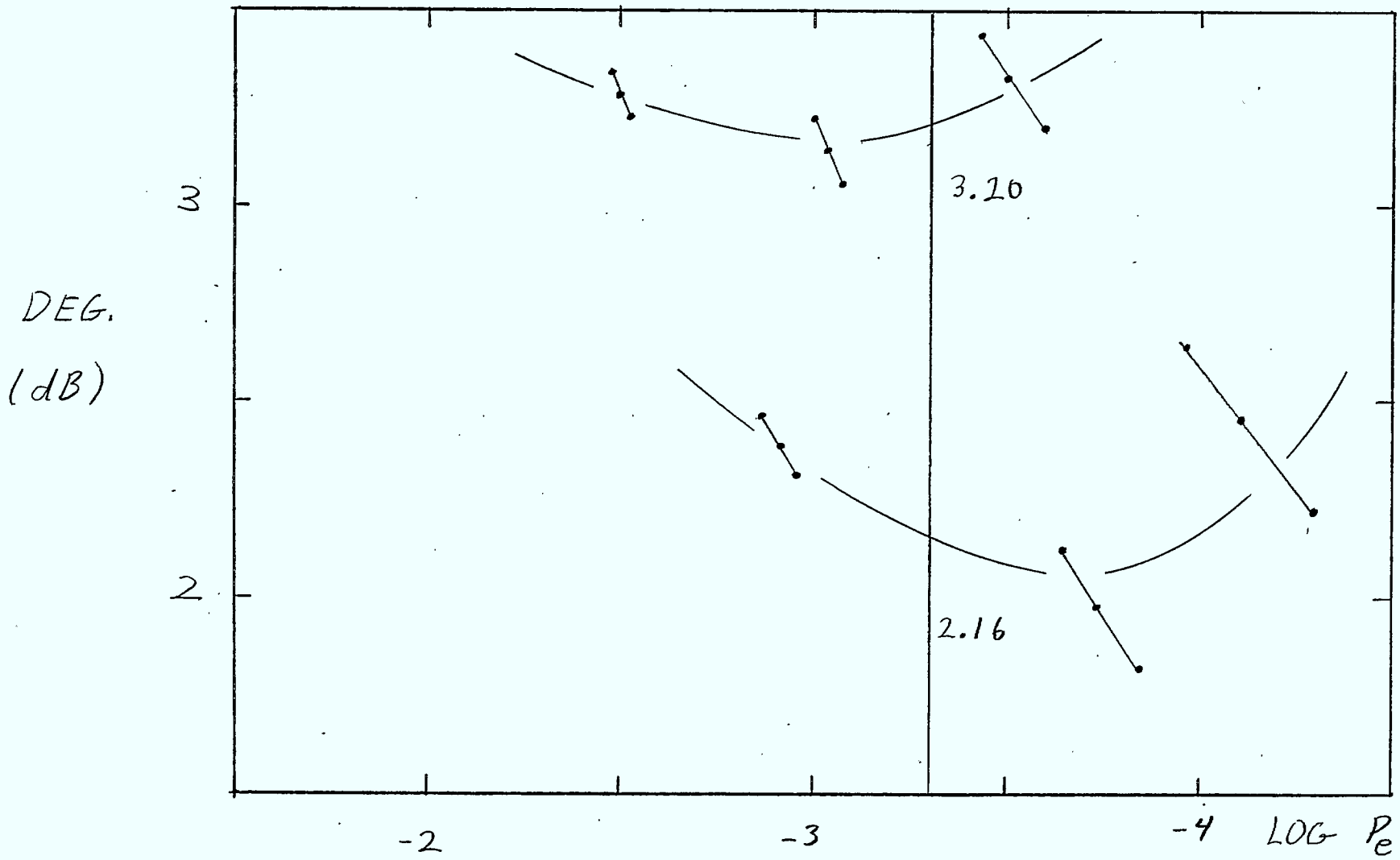

FIGURE A.1-23 Simulated DMSK with 4th Order Butterworth Receiver Filter (BT = 1.3)

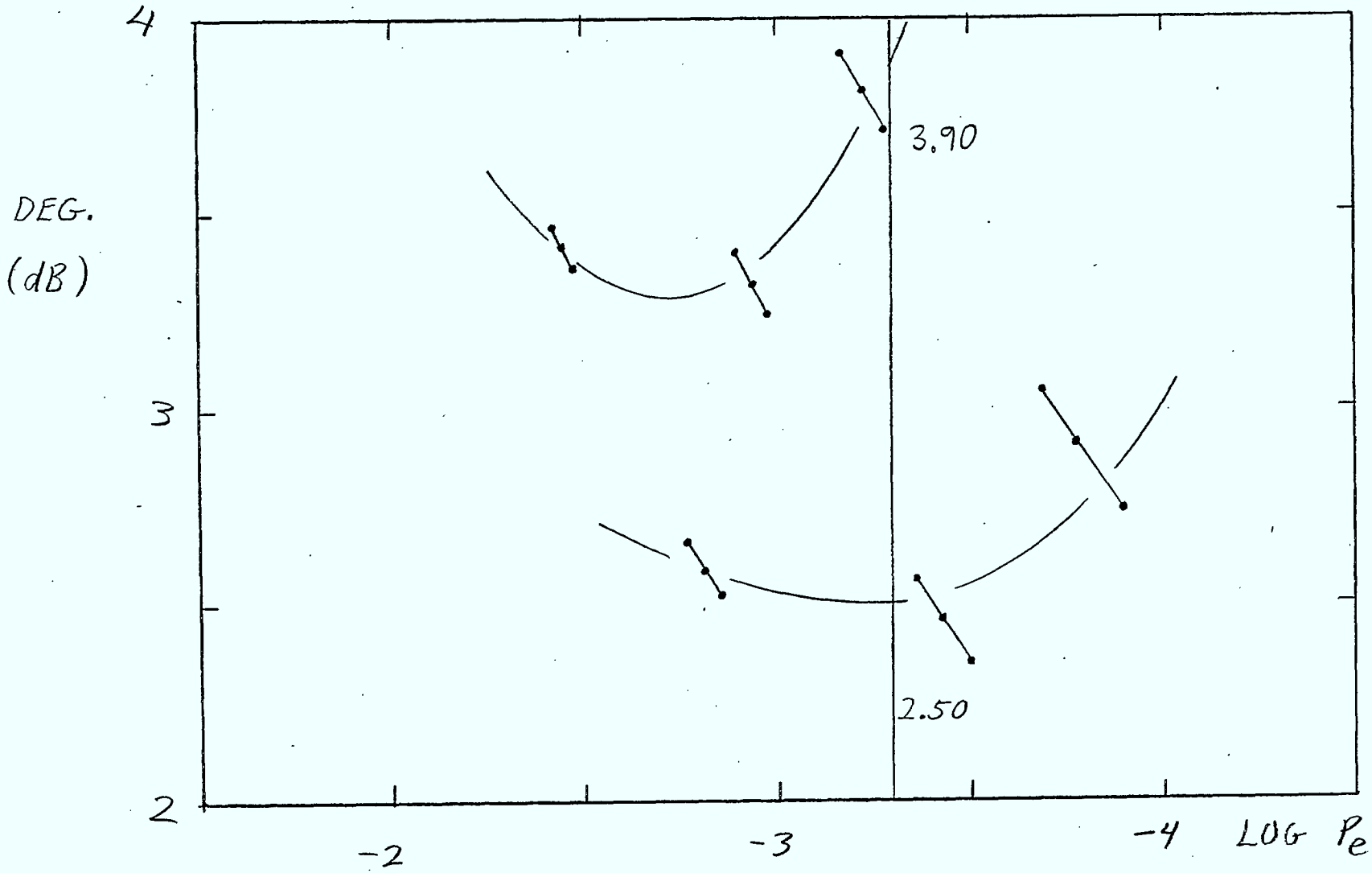

FIGURE A.1-24 Simulated DMSK with 4th Order Butterworth Receiver Filter (BT = 1.4)

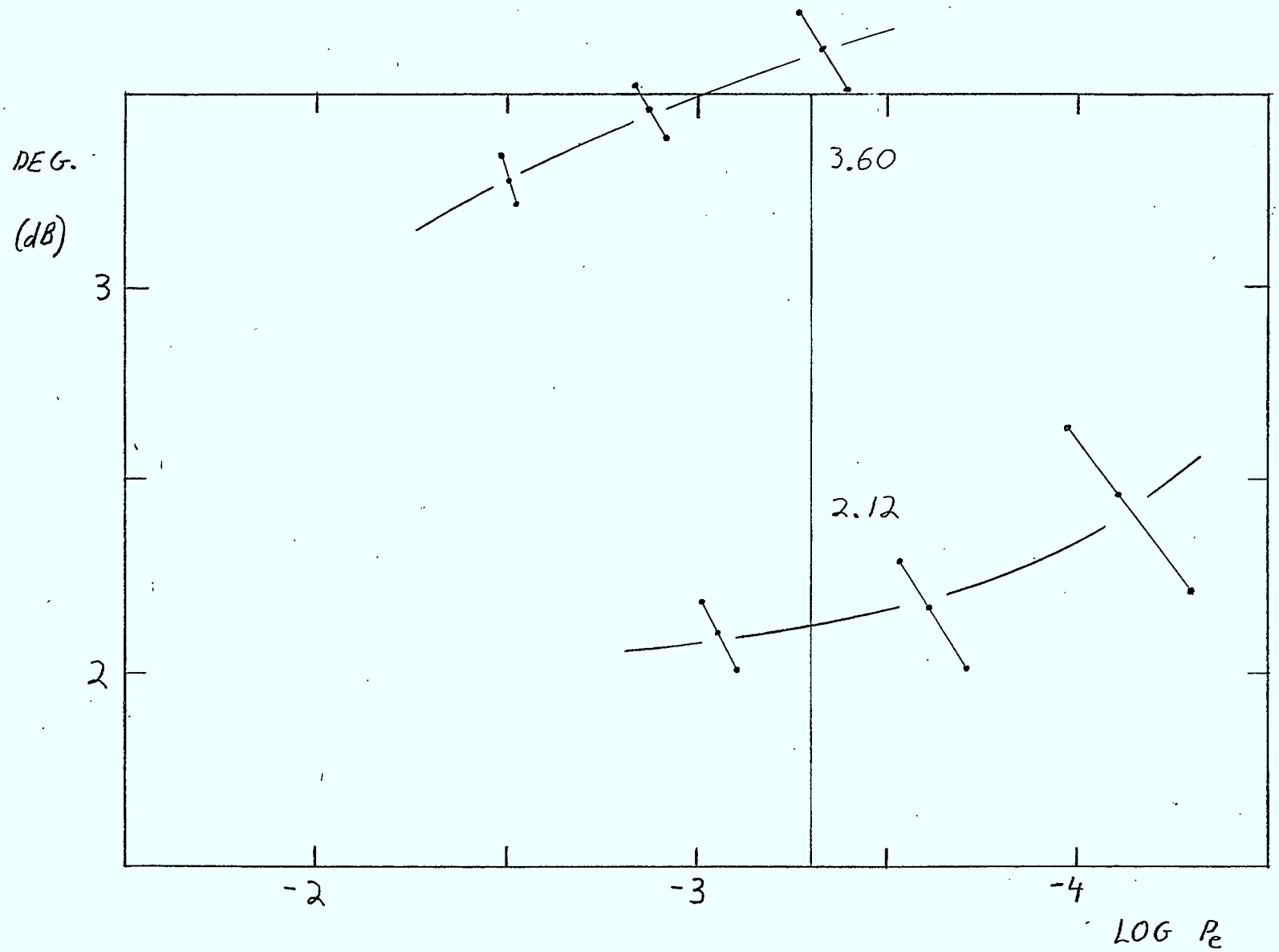

FIGURE A.1-25 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT = 0.9)

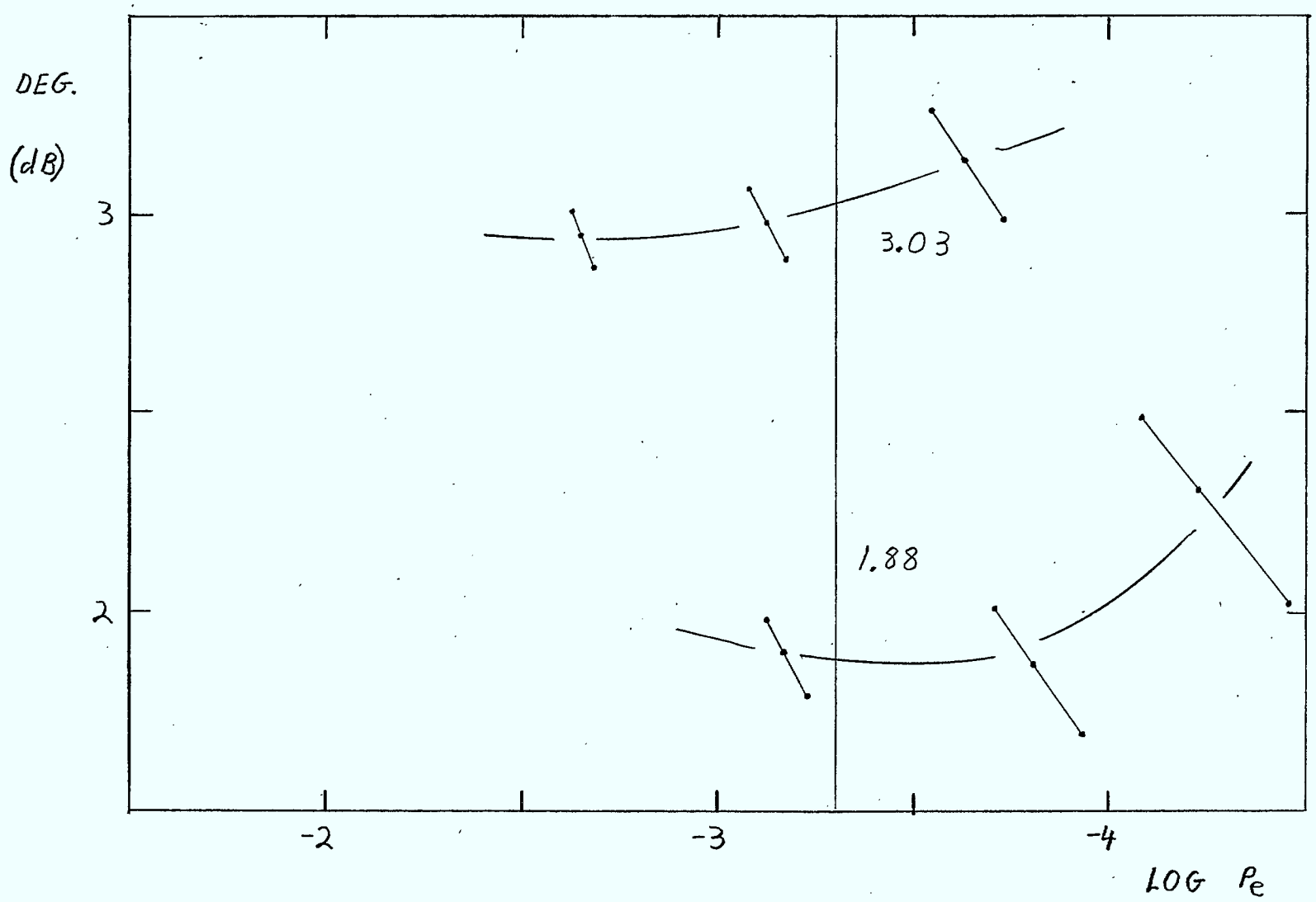

FIGURE A.1-26 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT= 1.0)

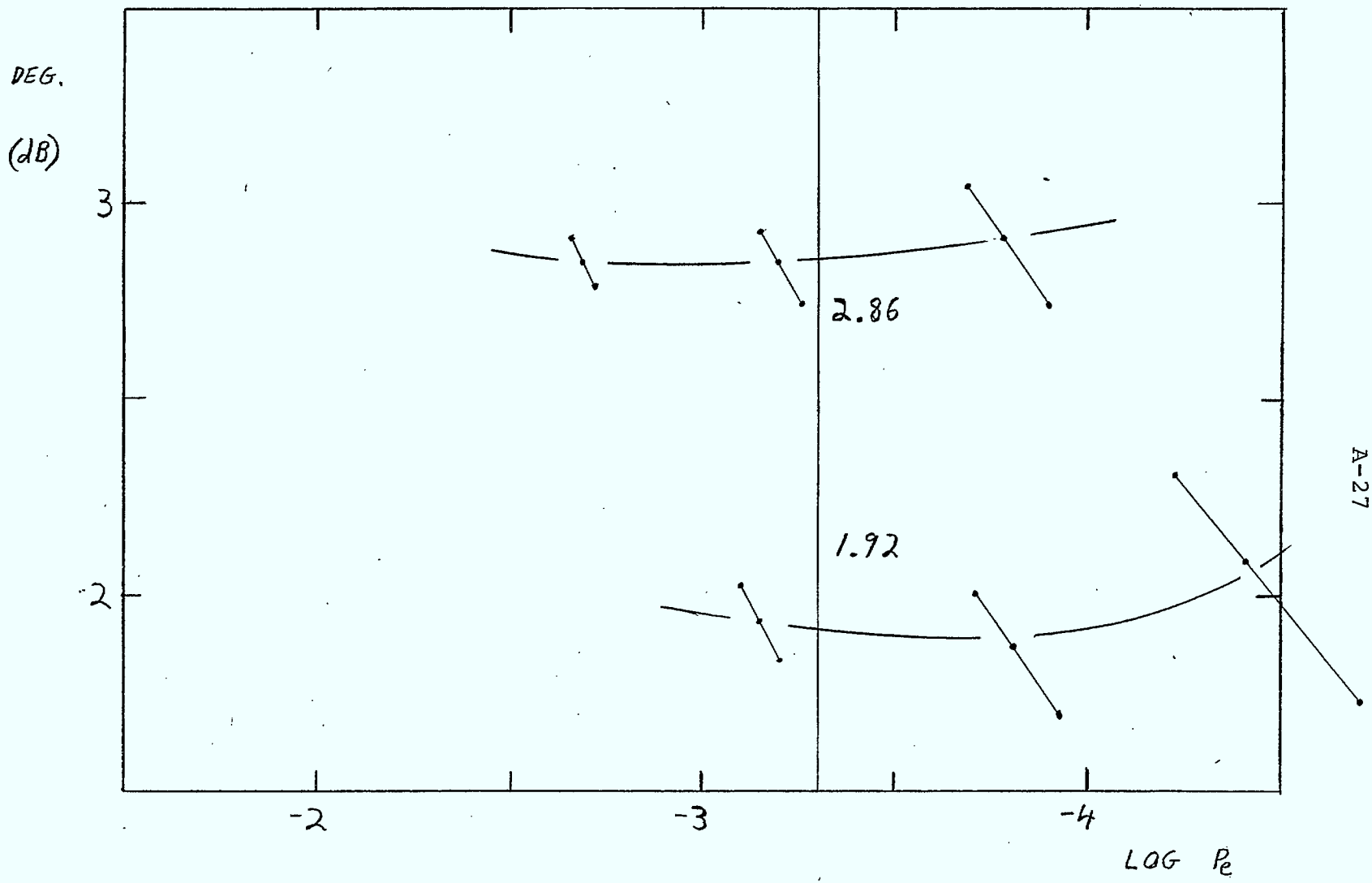

FIGURE A.1-27 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT = 1.1)

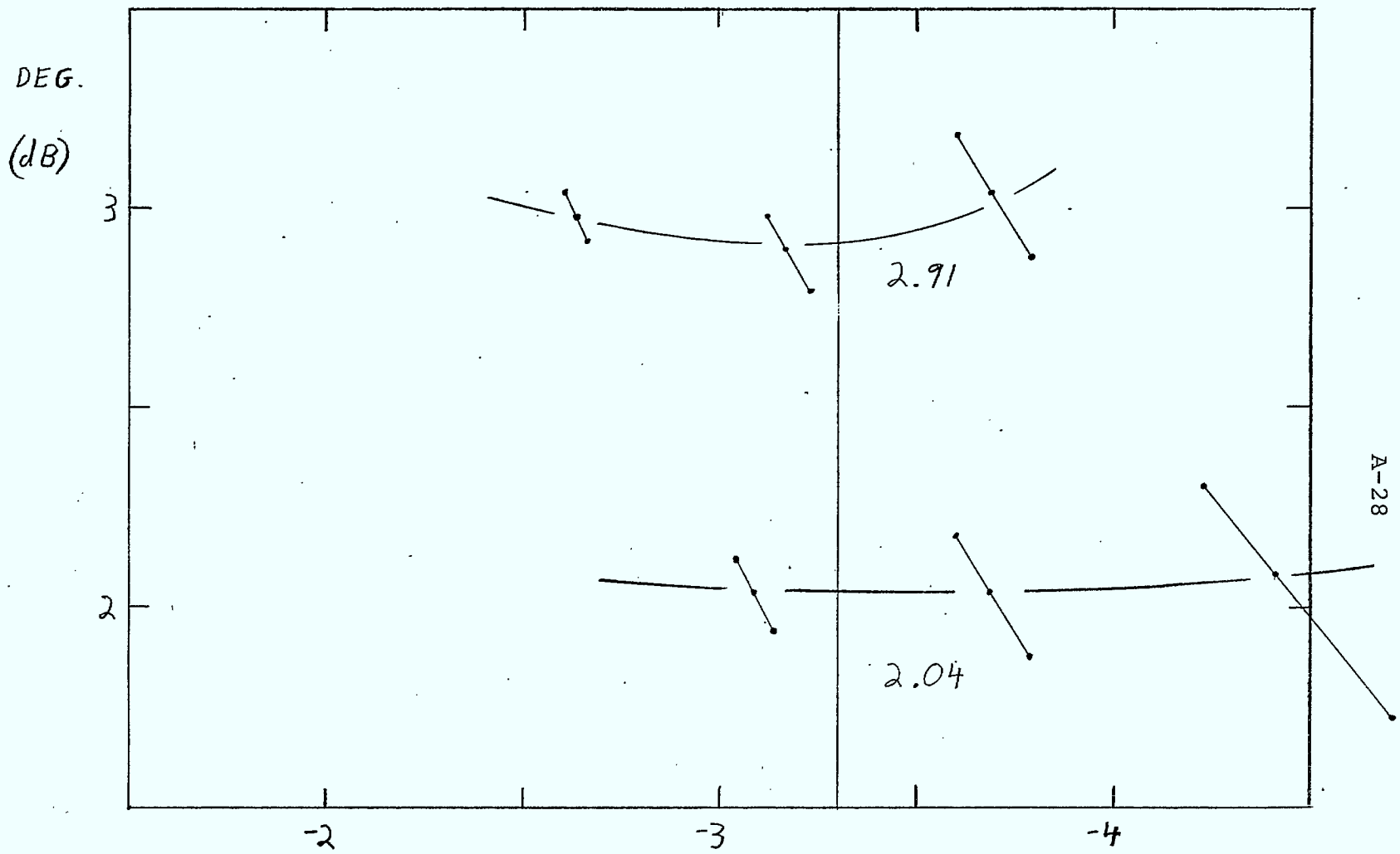

FIGURE A.1-28 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT= 1.2)

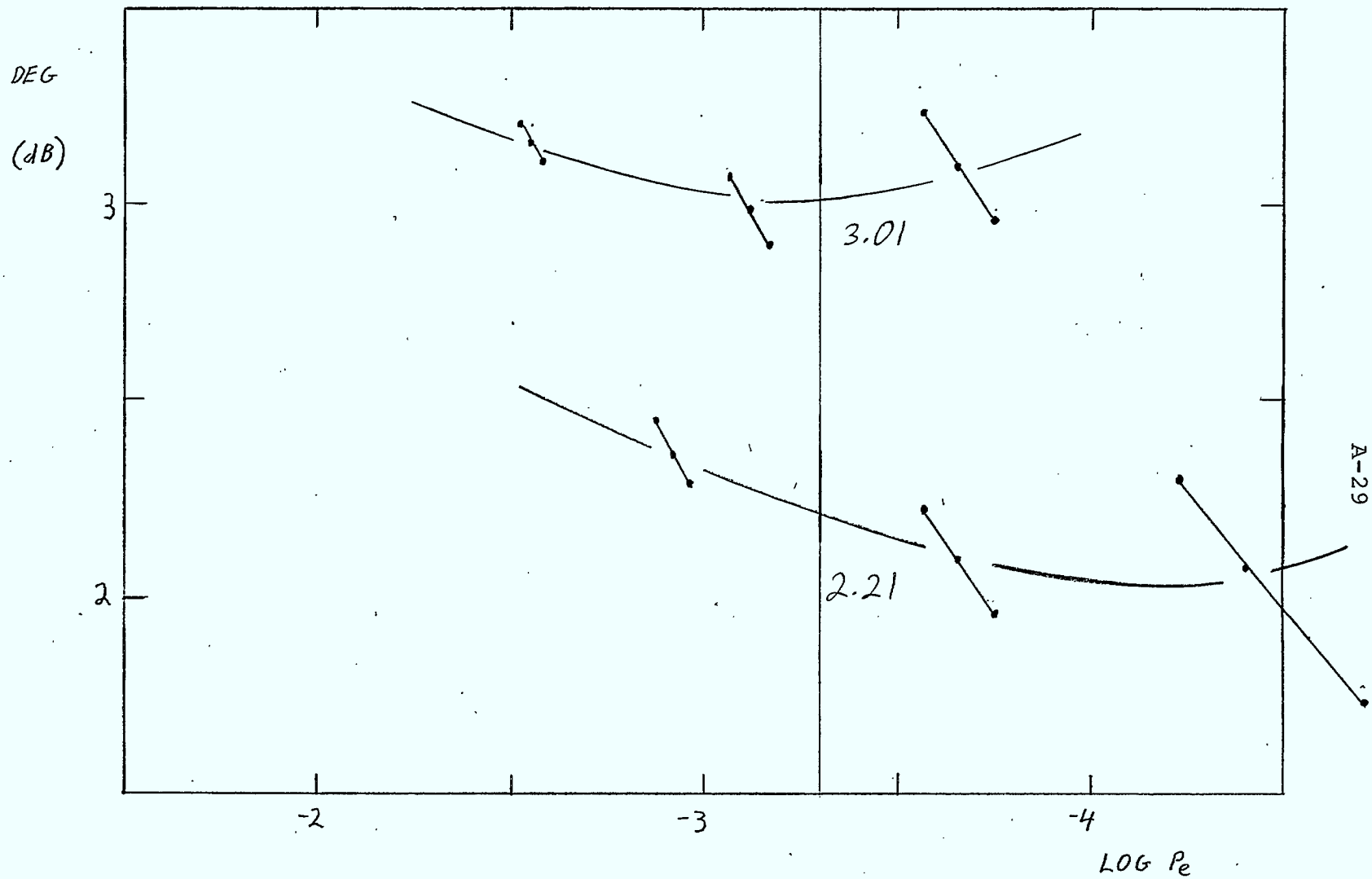

FIGURE A.1-29, Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT = 1.3)

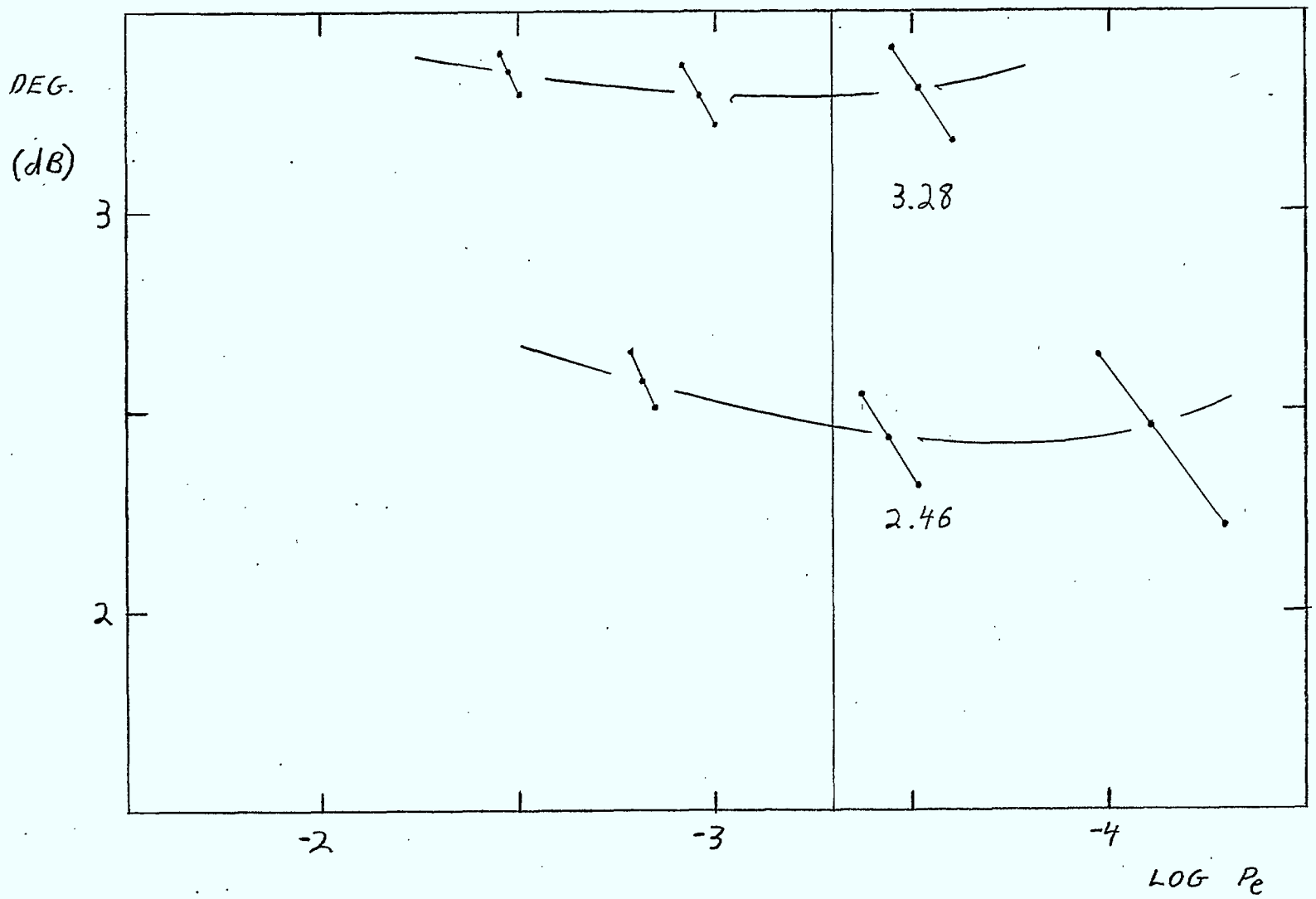

FIGURE A.1-30 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT= 1.4)

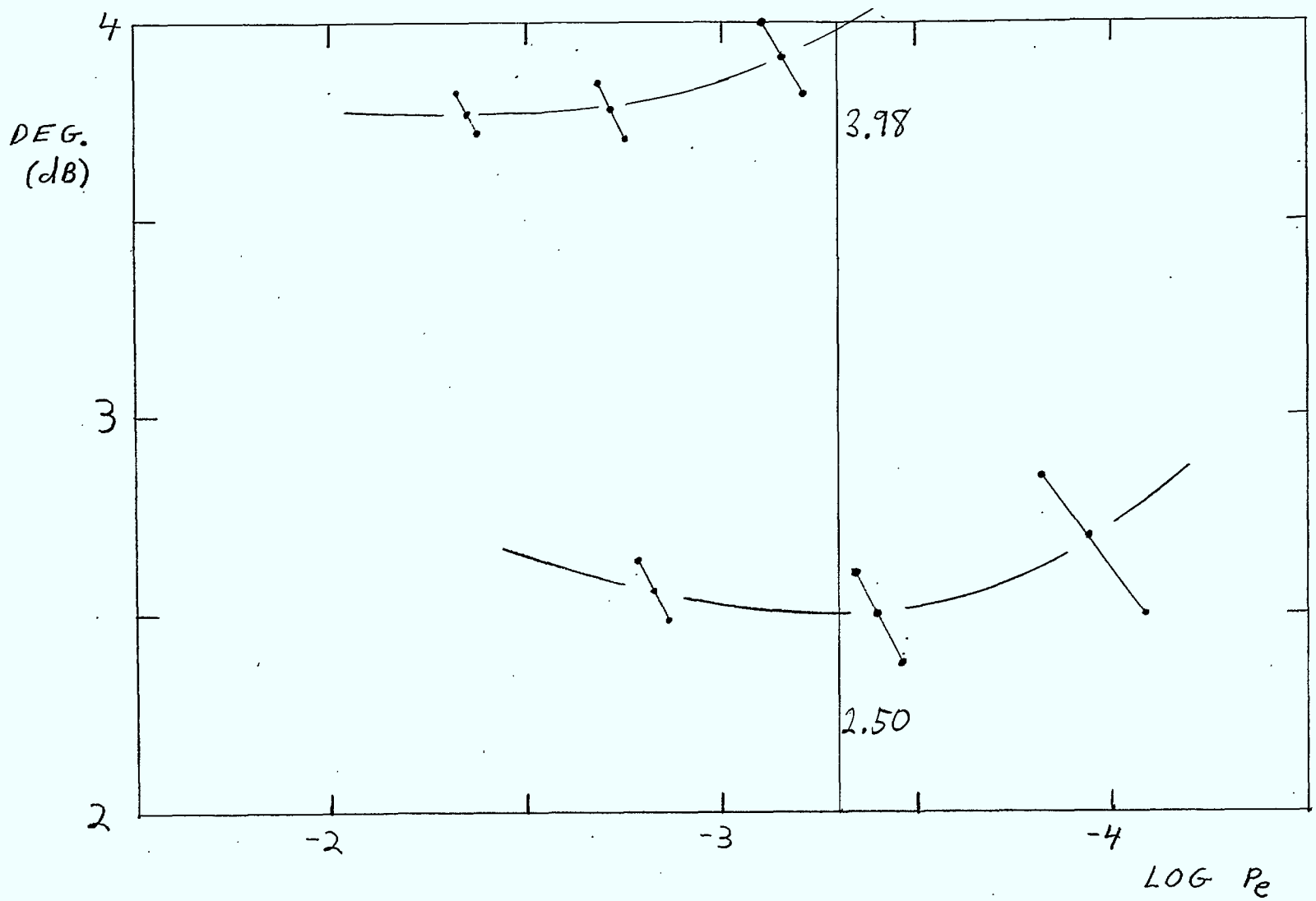

FIGURE A.1-3/ Simulated DMSK With Ideal Band-Pass Filter (BT = 1.0)

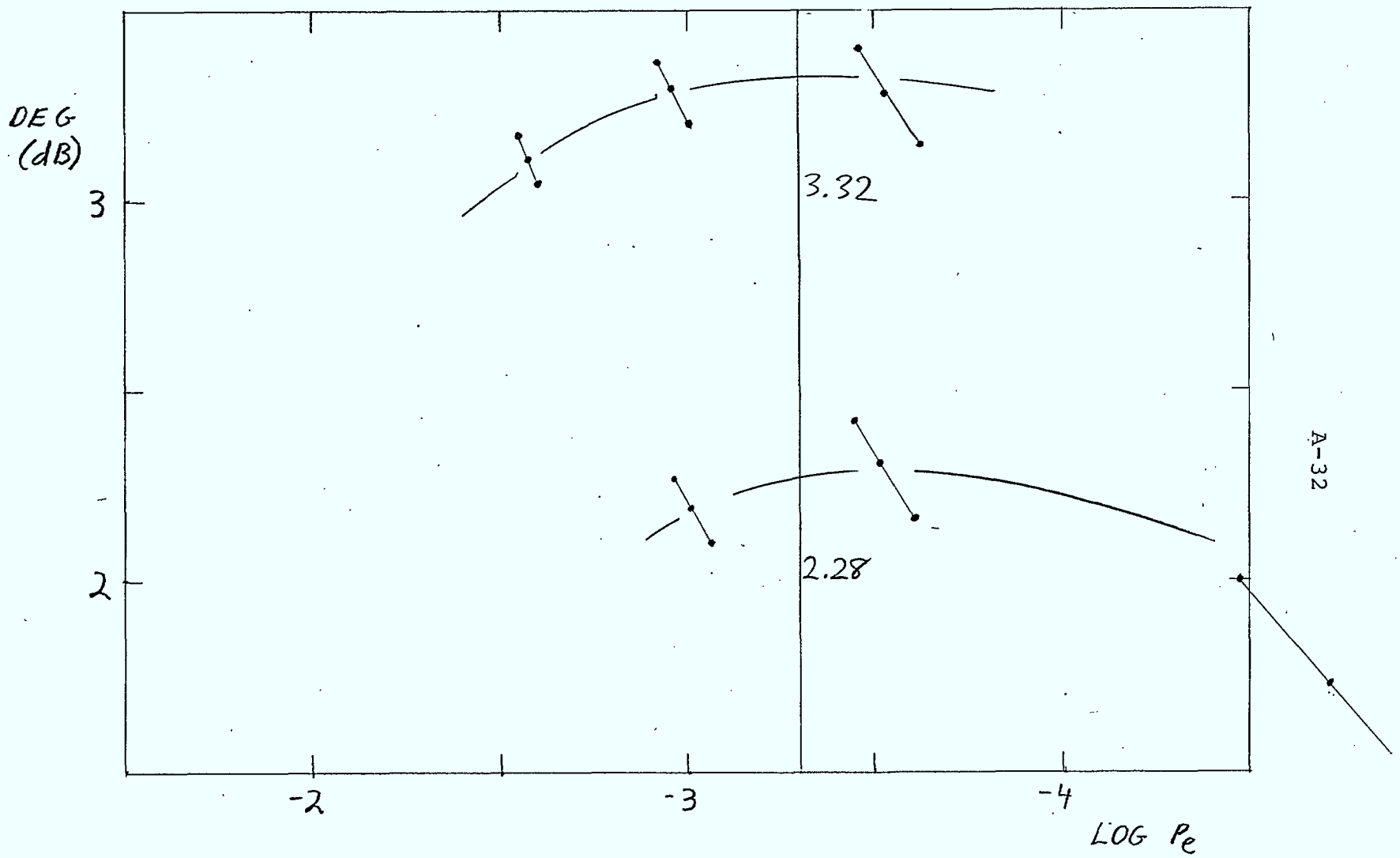

FIGURE A.1-32 Simulated DMSK With Ideal Band-Pass Filter ($BT = 1.1$)

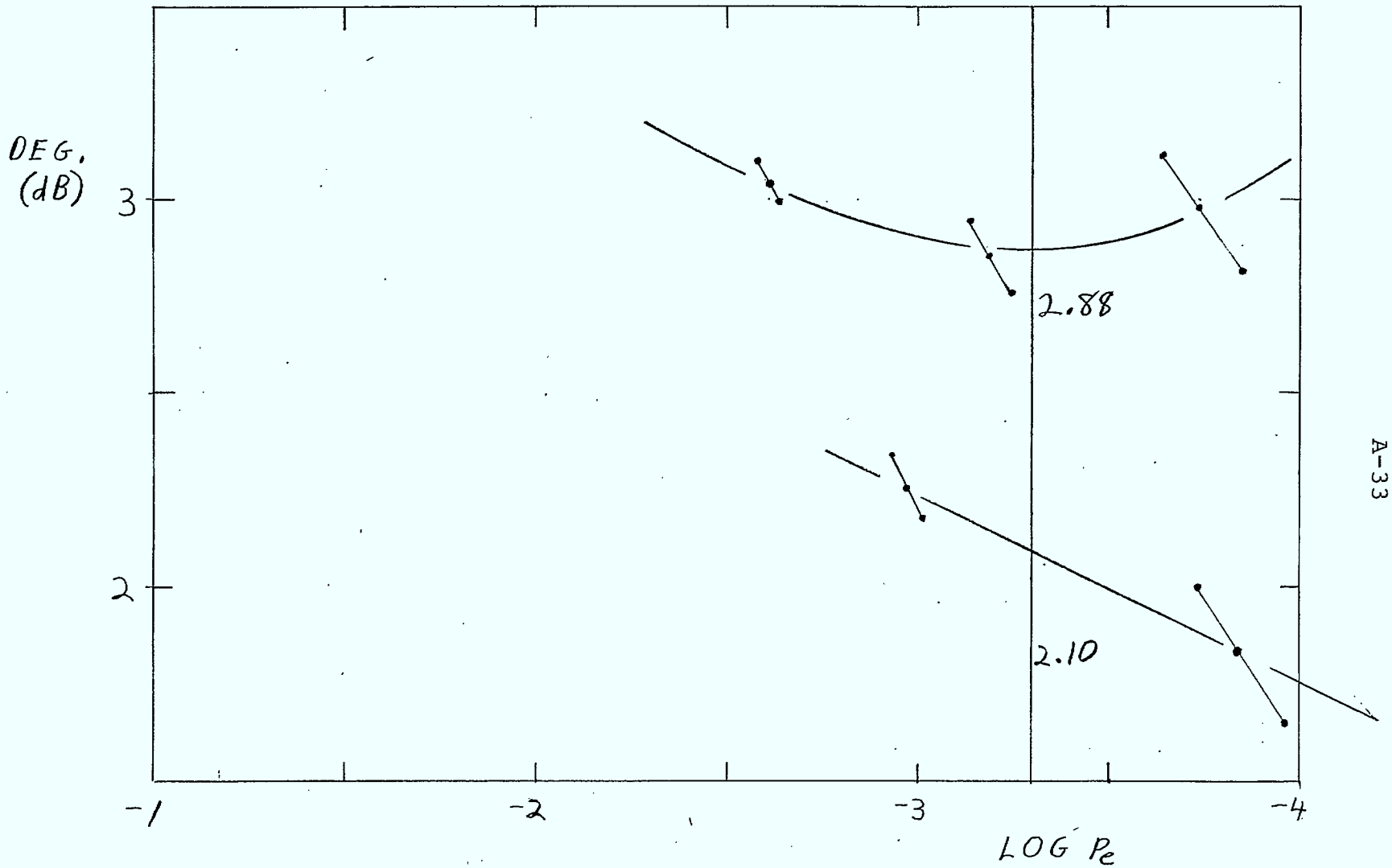

FIGURE A.1-33 Simulated DMSK With Ideal Band-Pass Filter (BT = 1.2)

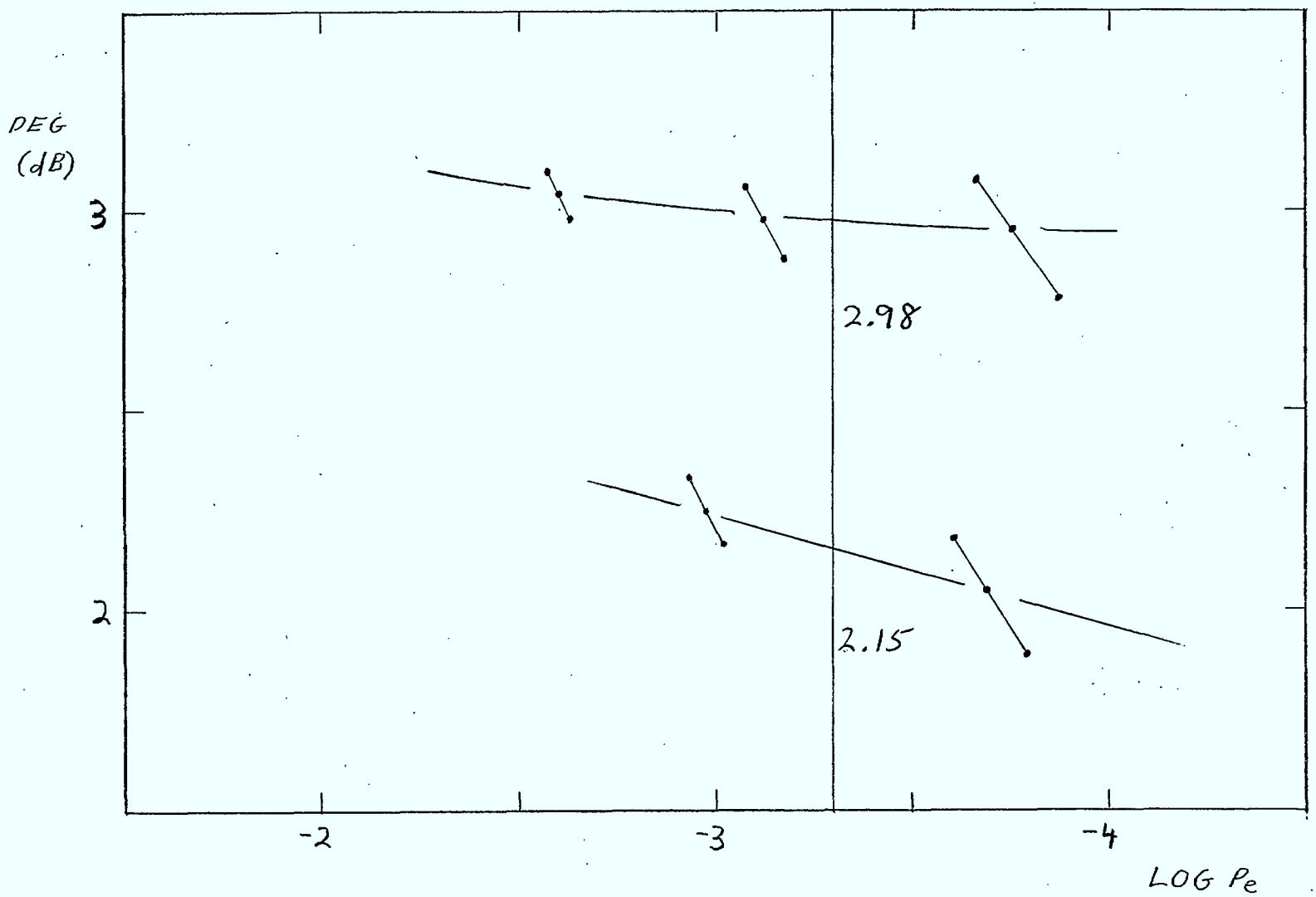

FIGURE A.1-34 Simulated DMSK with Ideal Band Pass Filter (BT = 1.3)

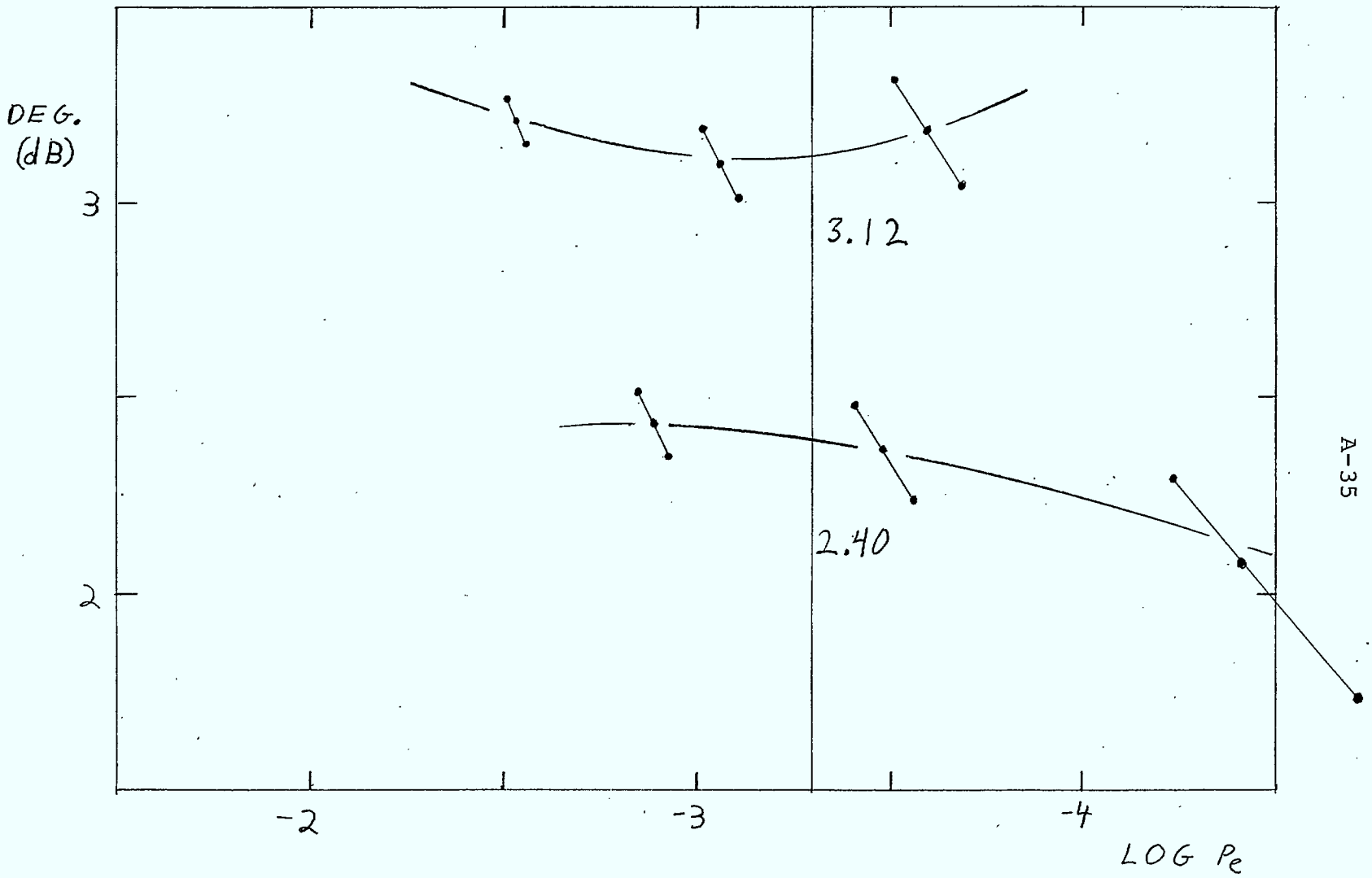

FIGURE A.1-35 - Simulated DMSK with Ideal Band-Pass Filter (BT= 1.4)

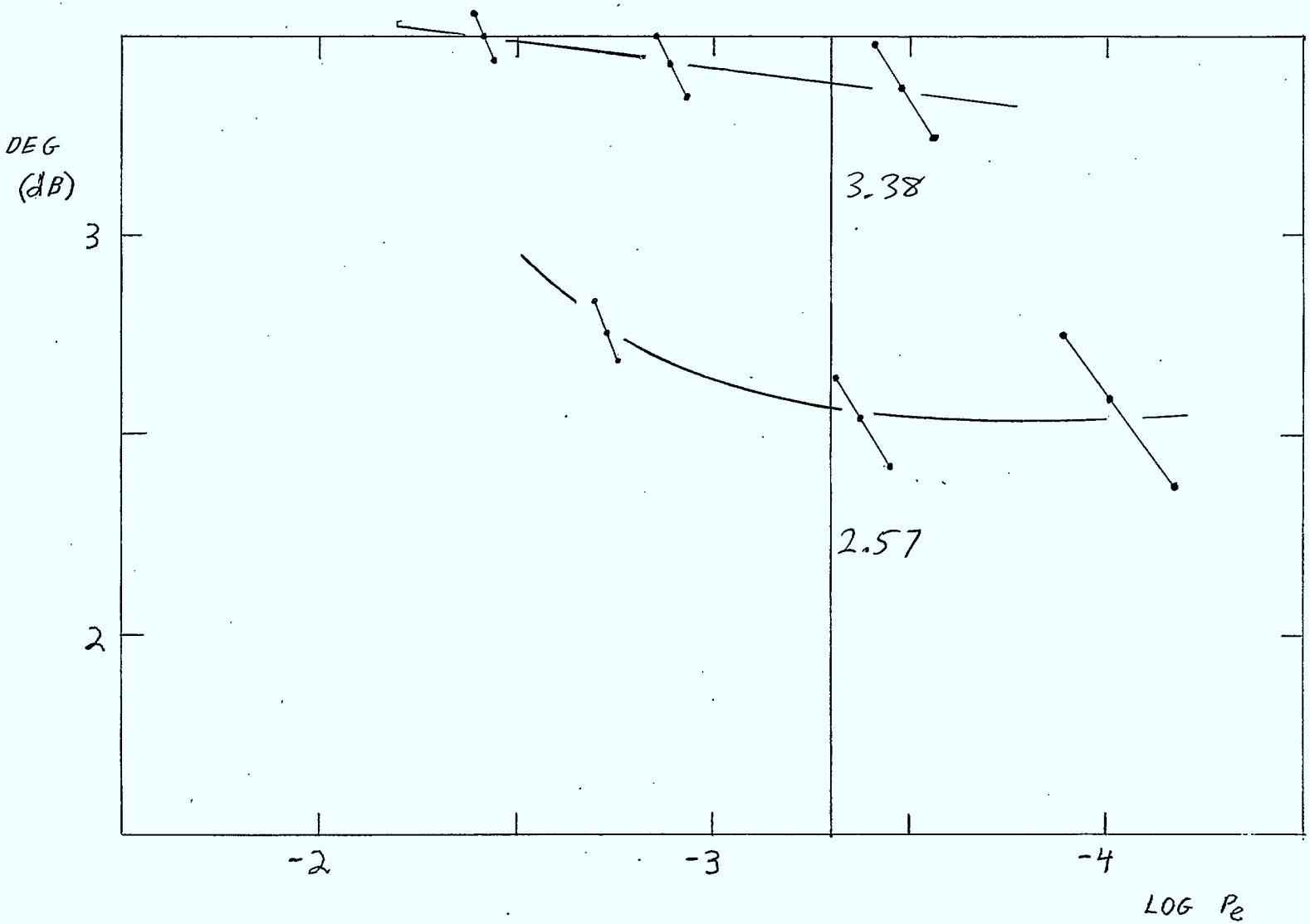

FIGURE A.1-36 Simulated DMSK With Ideal Band-Pass Filters (BT = 1.5)

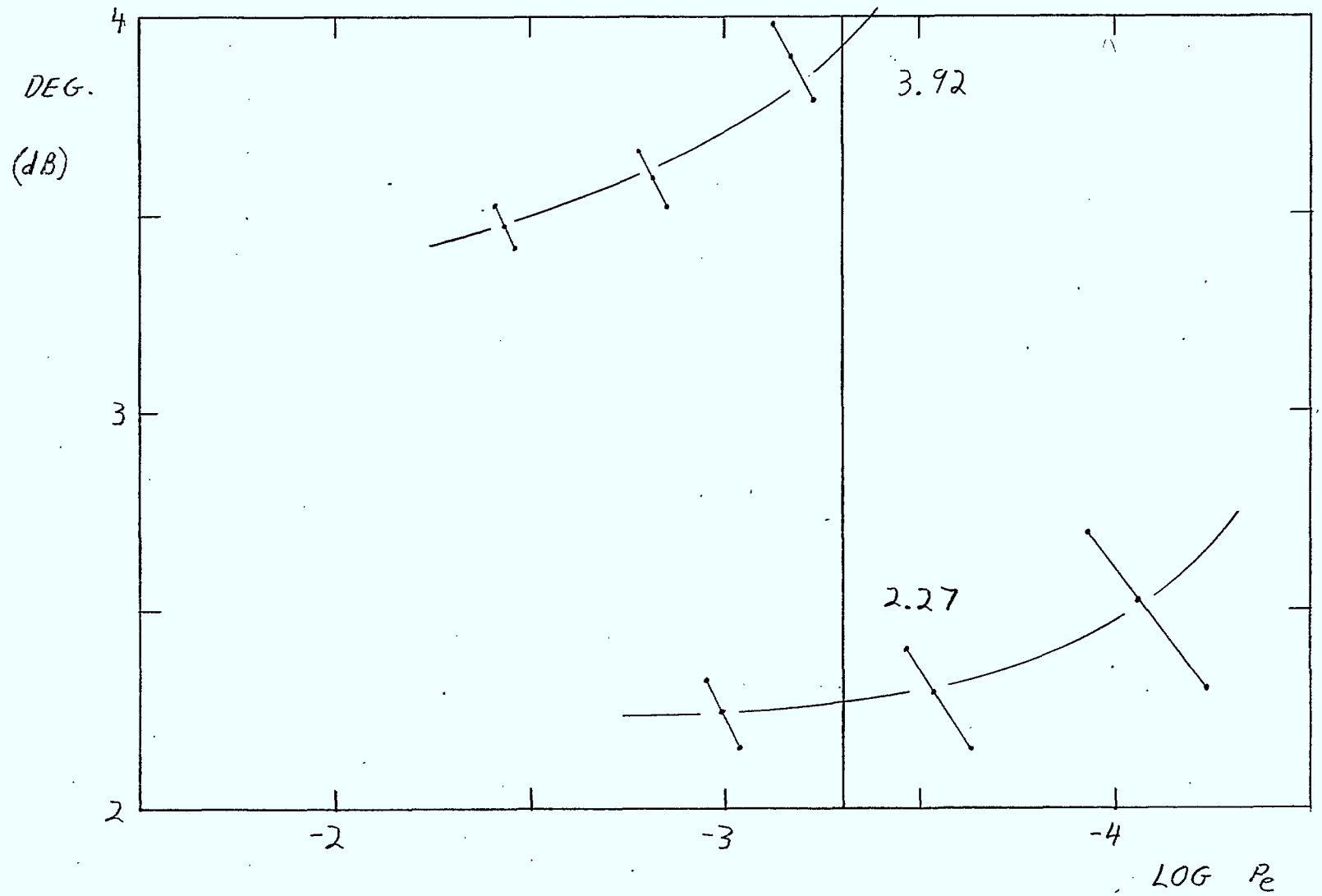

FIGURE A.2-1 Simulated DMSK With: TX = Ideal Band Pass Filter (BT = 4.0)
RX = 4th Order Butterworth Equalized (BT = 1.1)
DEM = 4th Order Butterworth Equalized (BT = 1.0)

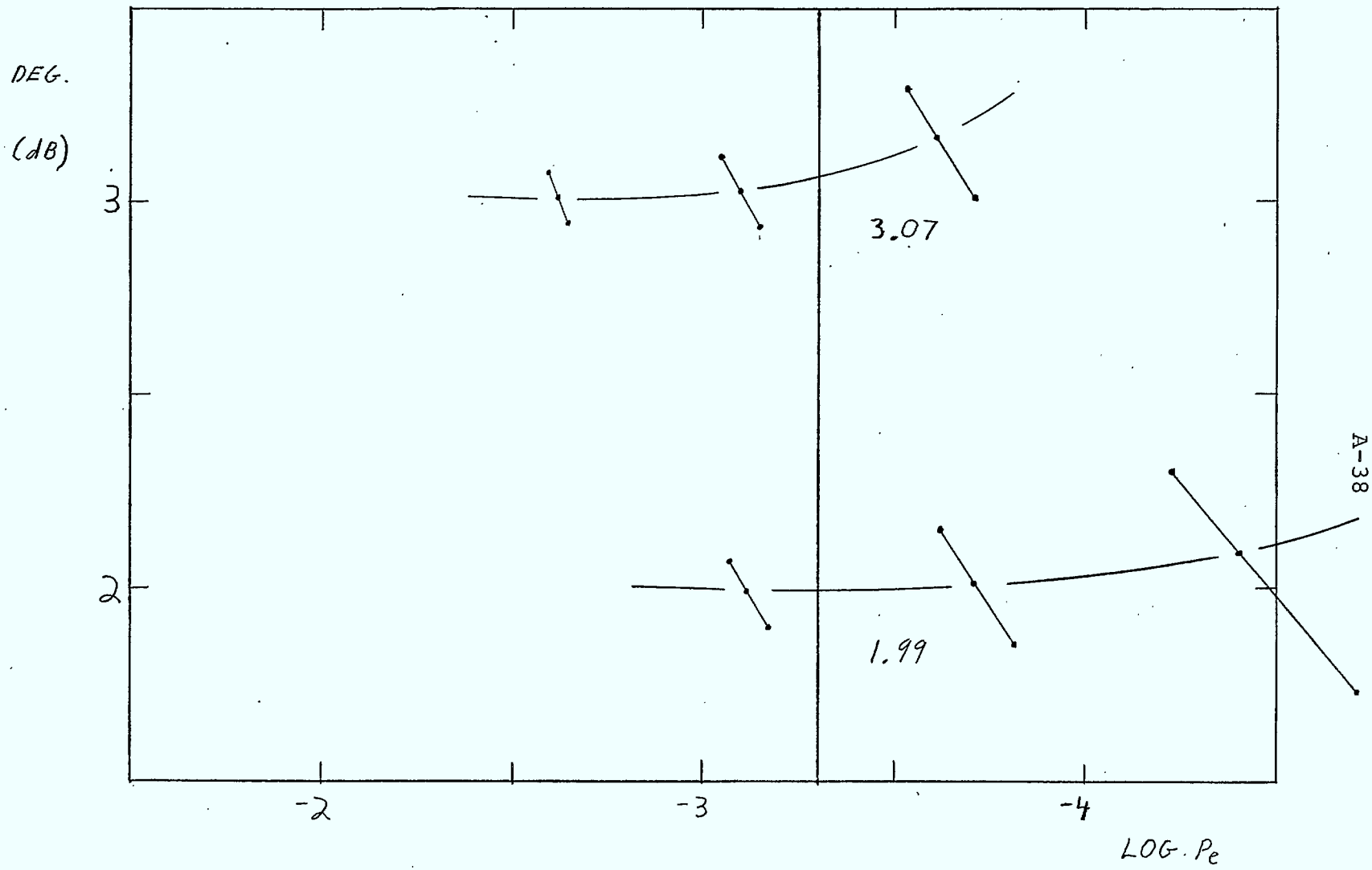

FIGURE A.2-2 Simulated DMSK With: TX= Ideal Band Pass Filter (BT=4.0)
RX= 4th Order Butterworth Equalized (BT=1.1)
DEM= 4th Order Butterworth Equalized (BT=1.4)

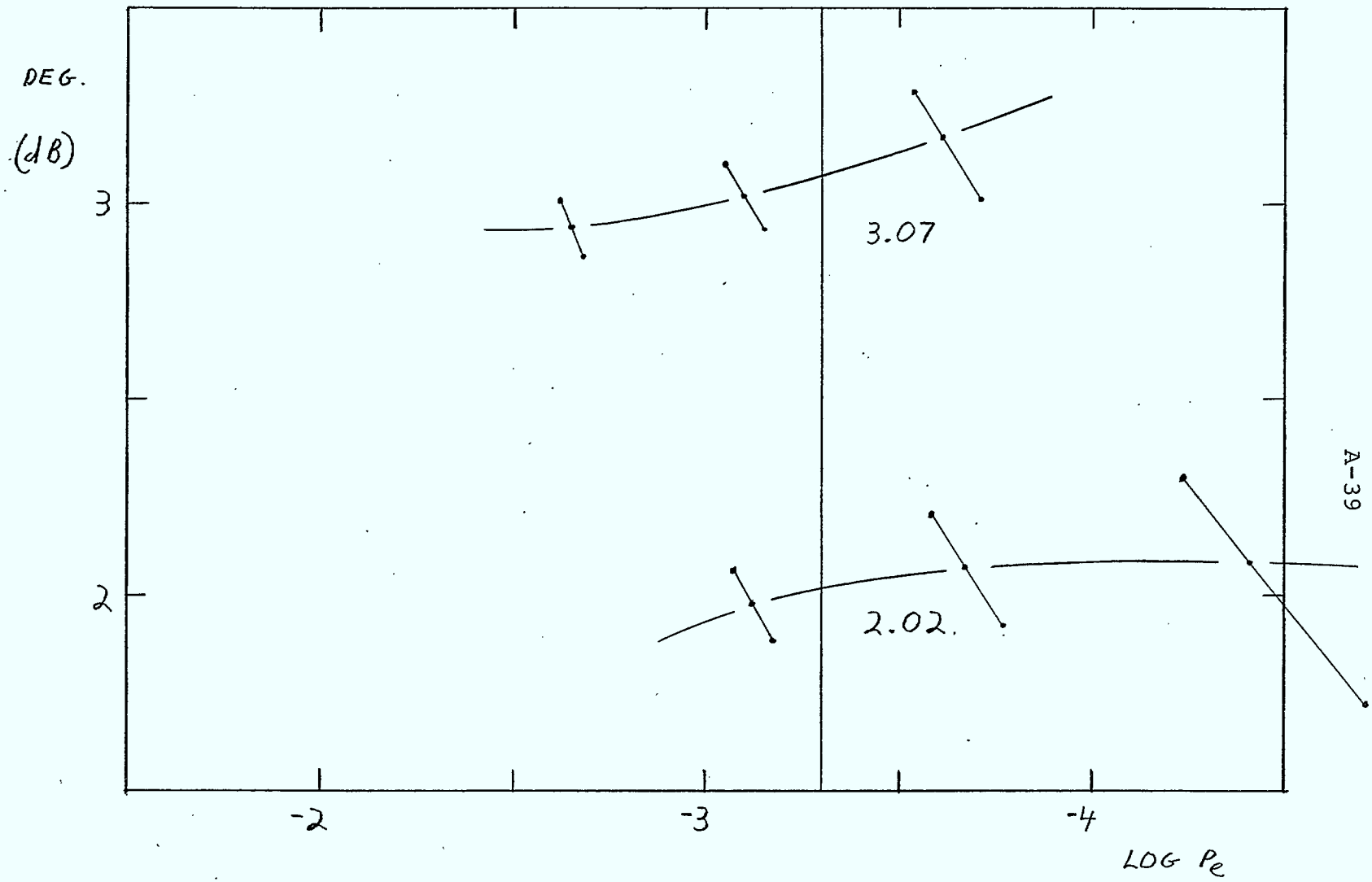

FIGURE A.2-3 Simulated DMSK With: TX = Ideal Band Pass Filter (BT = 4.0)
RX = 4th Order Butterworth Equalized (BT = 1.1)
DEM = 4th Order Butterworth Equalized (BT = 1.8)

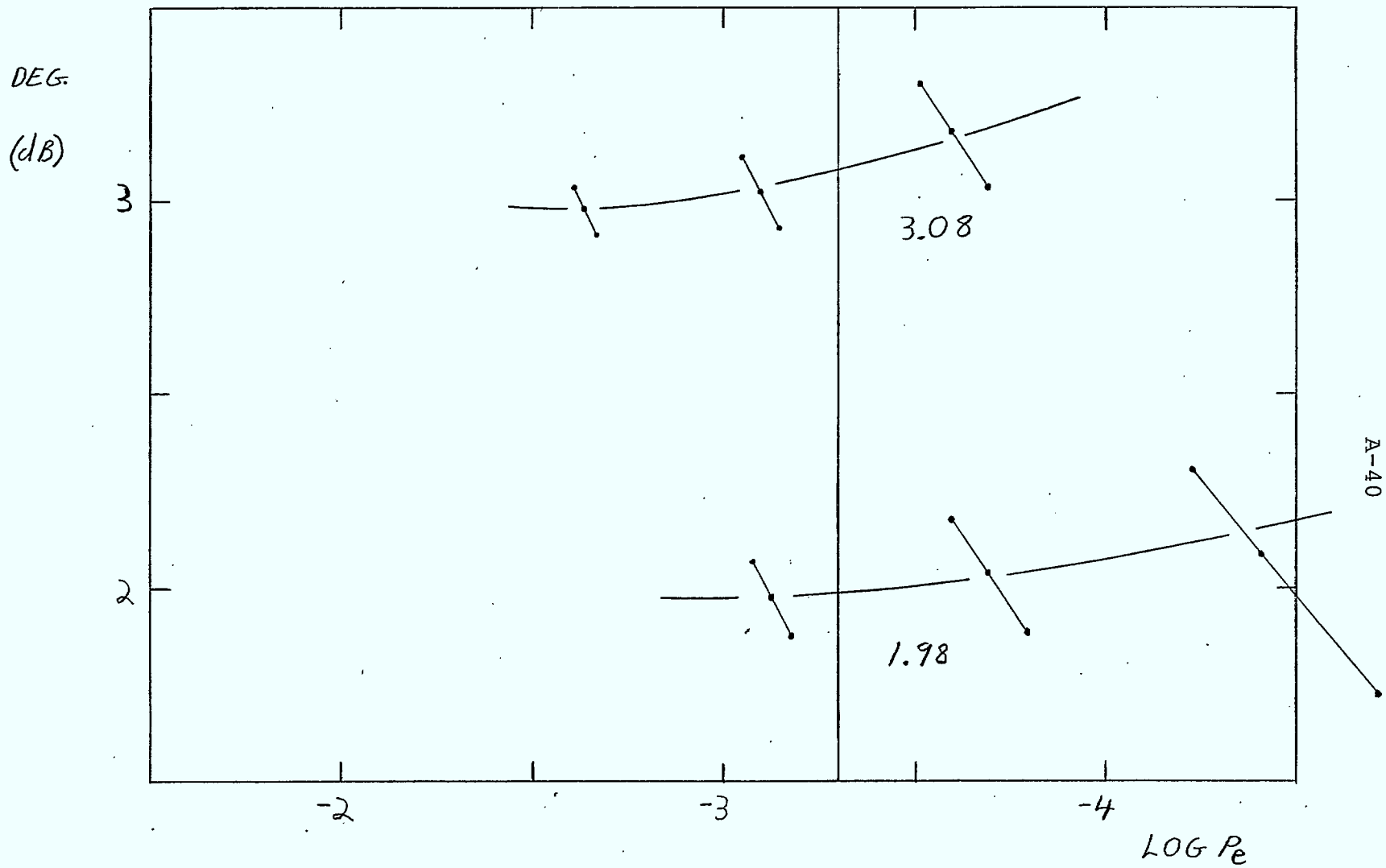

FIGURE A.2-4 Simulated DMSK with: TX = Ideal Band Pass Filter (BT = 4.0)
RX = 4th Order Butterworth Equalized (BT = 1.1)
DEM = 4th Order Butterworth Equalized (BT = 2.0)

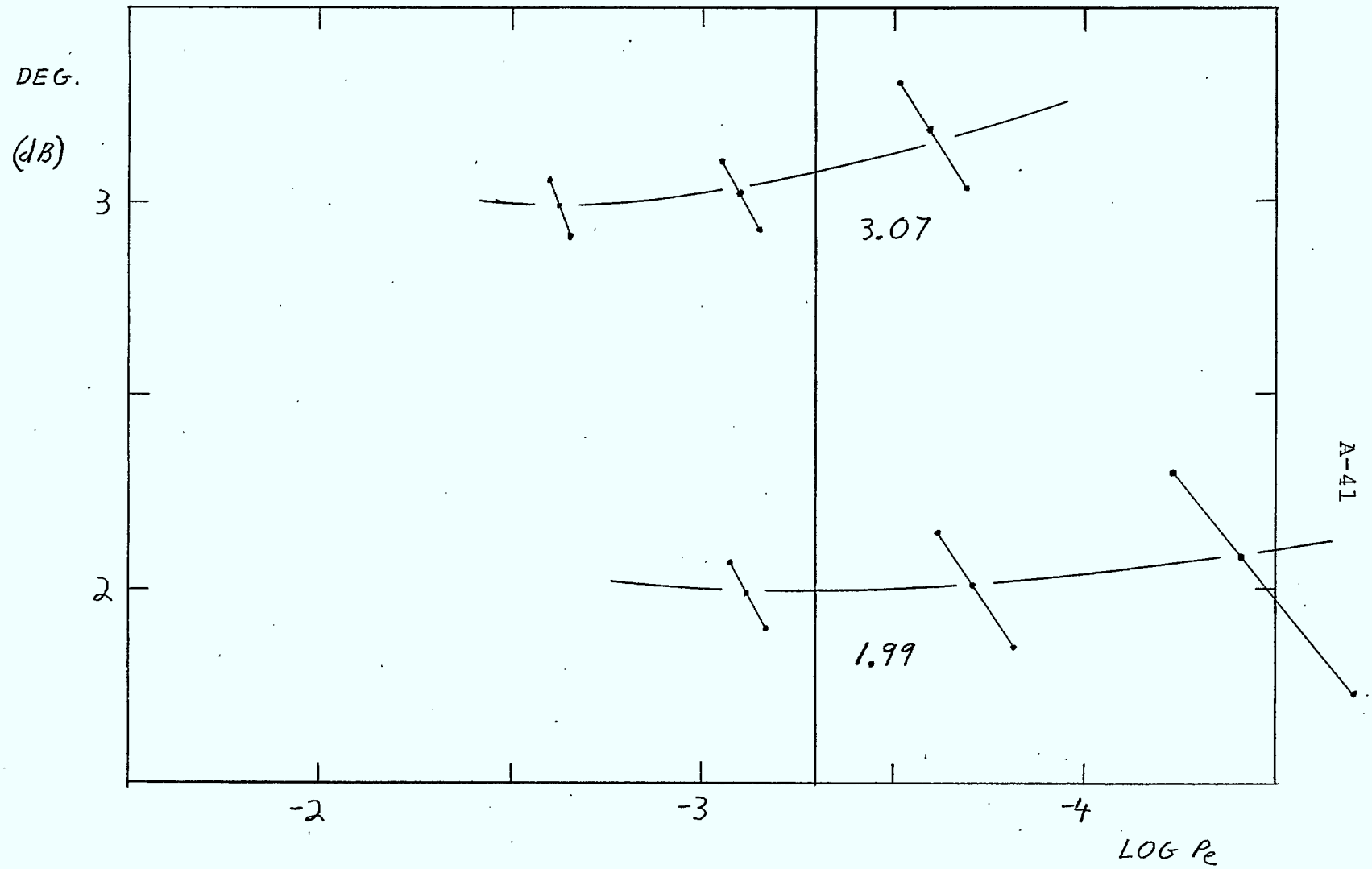

FIGURE A.2-5 Simulated DMSK With: TX = Ideal Band Pass Filter (BT= 4.0)
RX = 4th Order Butterworth Equalized (BT= 1.1)
DEM = 4th Order Butterworth Equalized (BT=2.2)

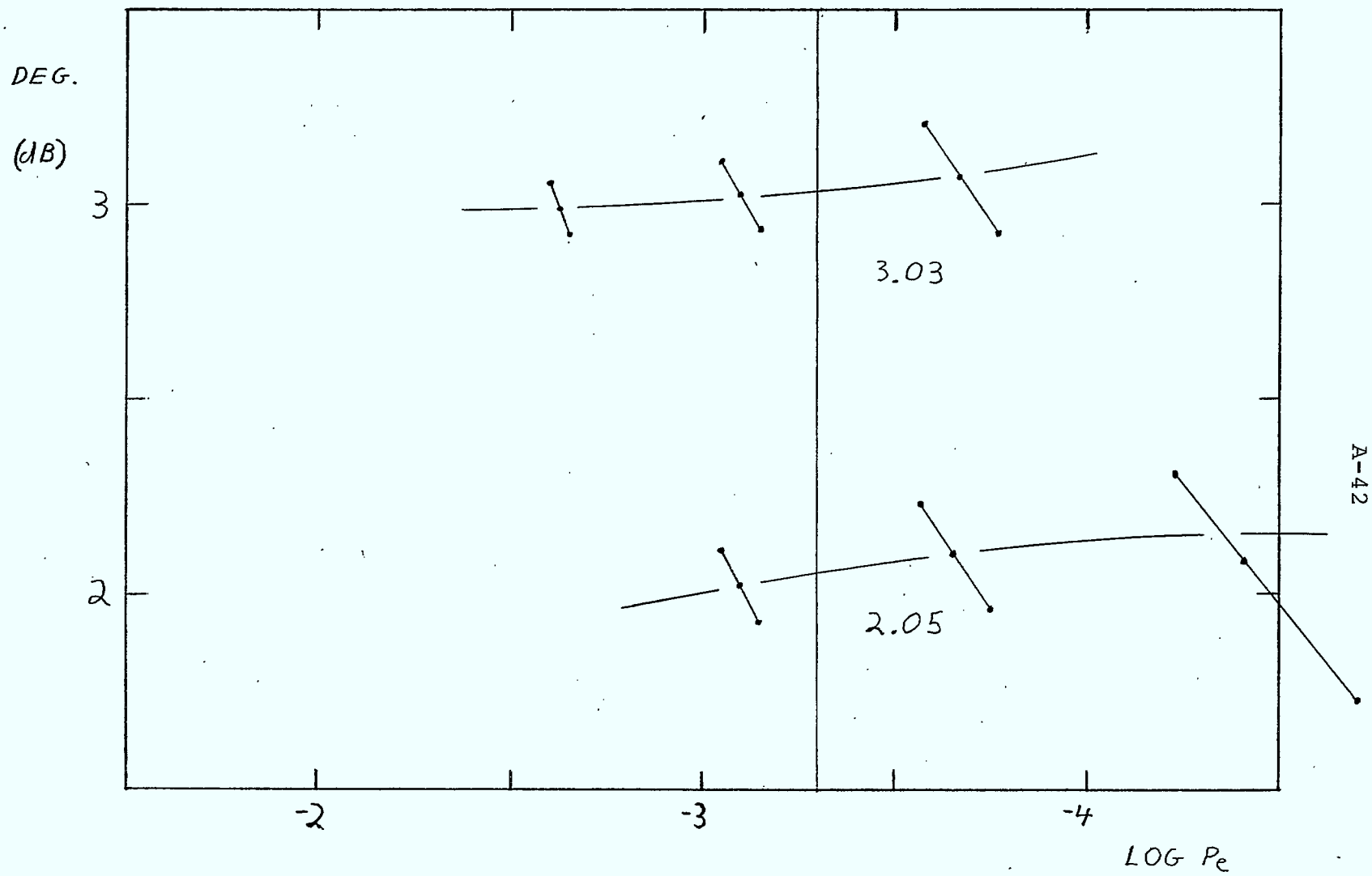

FIGURE A2-6 Simulated DMSK With: TX = Ideal Band Pass Filter (BT= 4.0)
RX = 4th Order Butterworth Equalized (BT= 1.2)
DEM= 4th Order Butterworth Equalized (BT= 1.8)

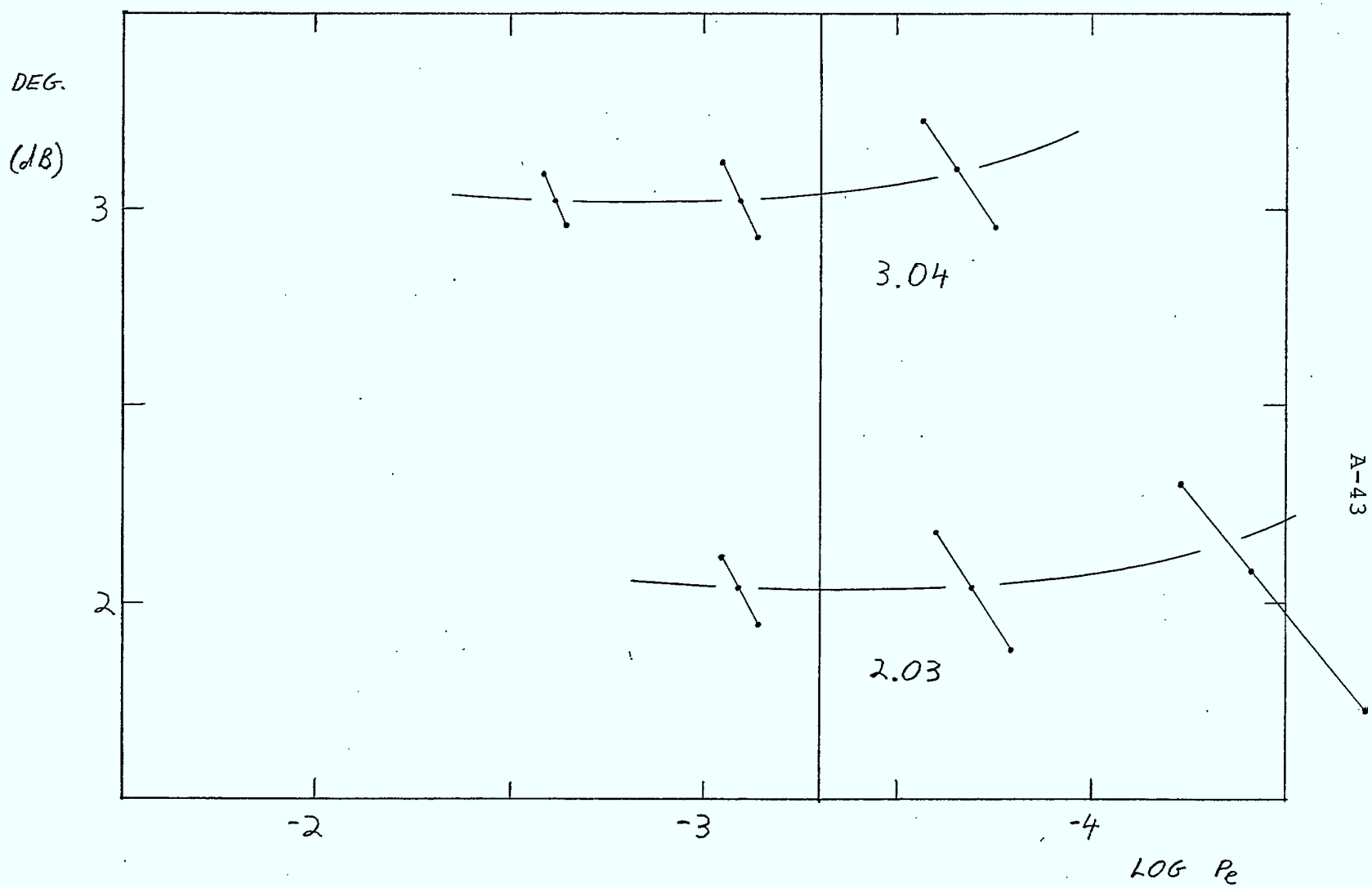

FIGURE A2-7 Simulated DMSK With: TX = Ideal Band Pass Filter (BT = 4.0)
RX = 4th Order Butterworth Equalized (BT = 1.2)
DEM = 4th Order Butterworth Equalized (BT = 2.0)

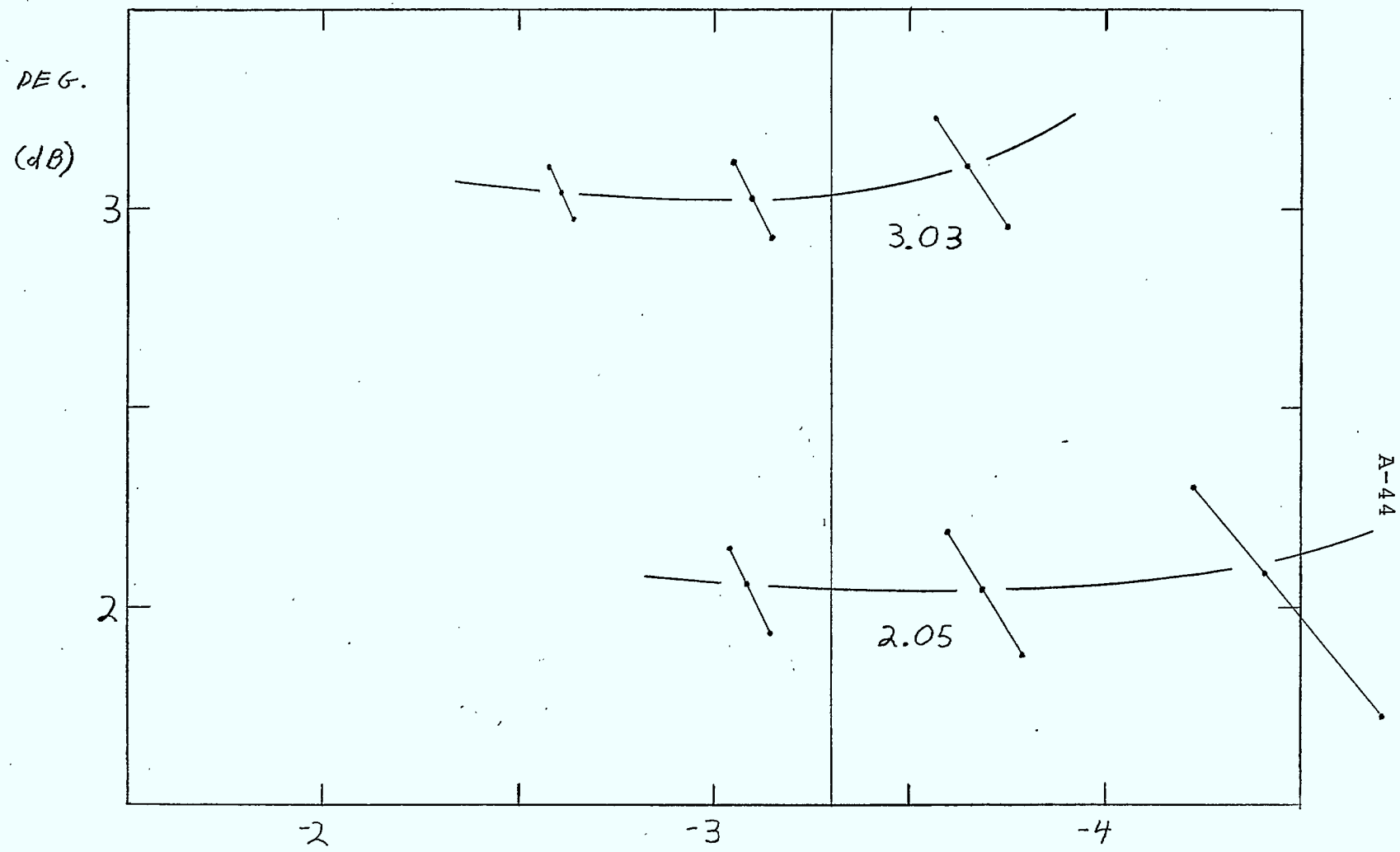

FIGURE A.2-8 Simulated DMSK With: TX: Ideal Band Pass Filter (BT = 4.0)
RX: 4th Order Butterworth Equalized (BT = 1.2)
DEM: 4th Order Butterworth Equalized (BT = 2.2)

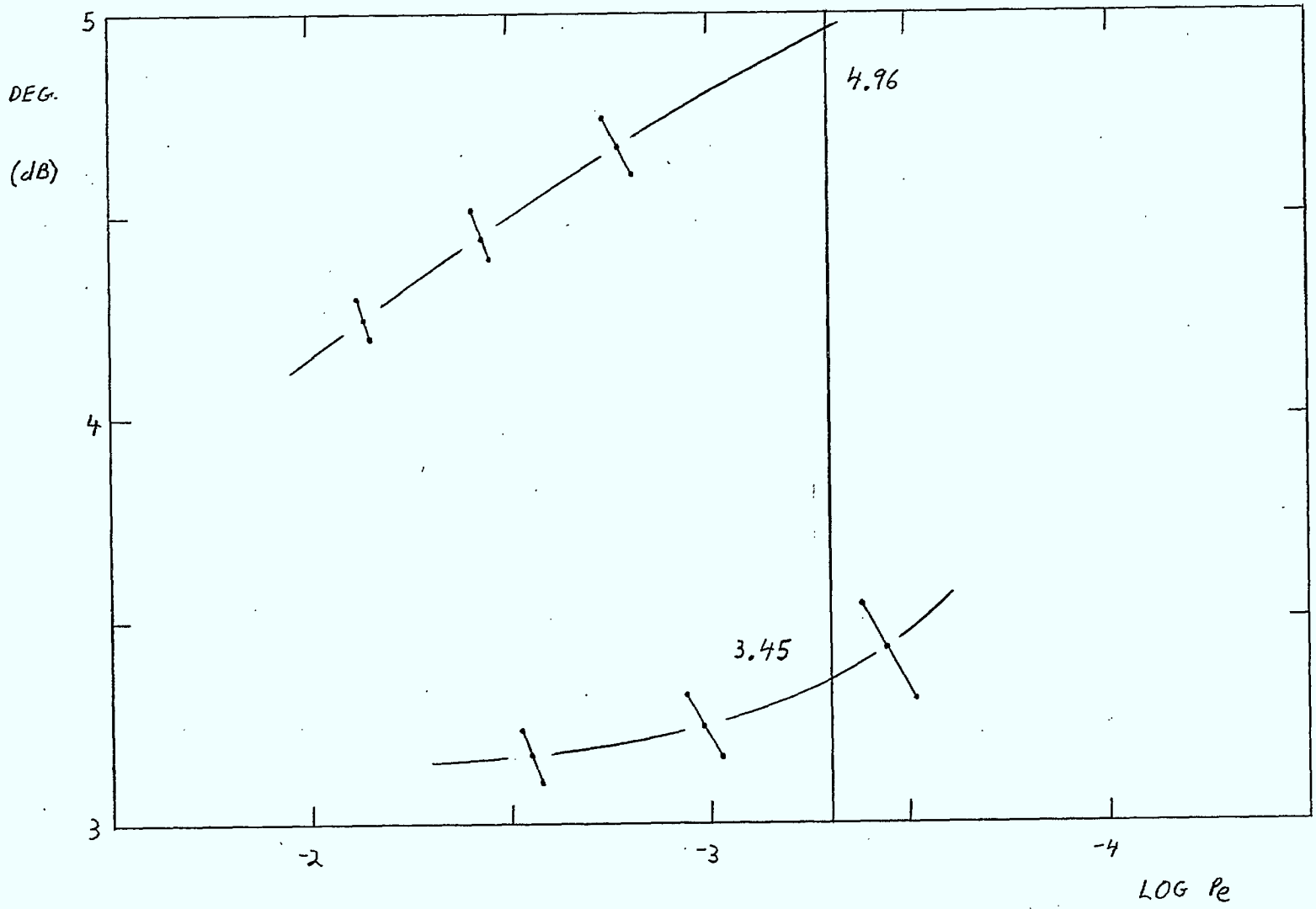

FIGURE A.3-1 Transmit Filter Analysis With:
Rx = BW4E (BT = 1.1)
Tx = BW4E (BT = 1.0)

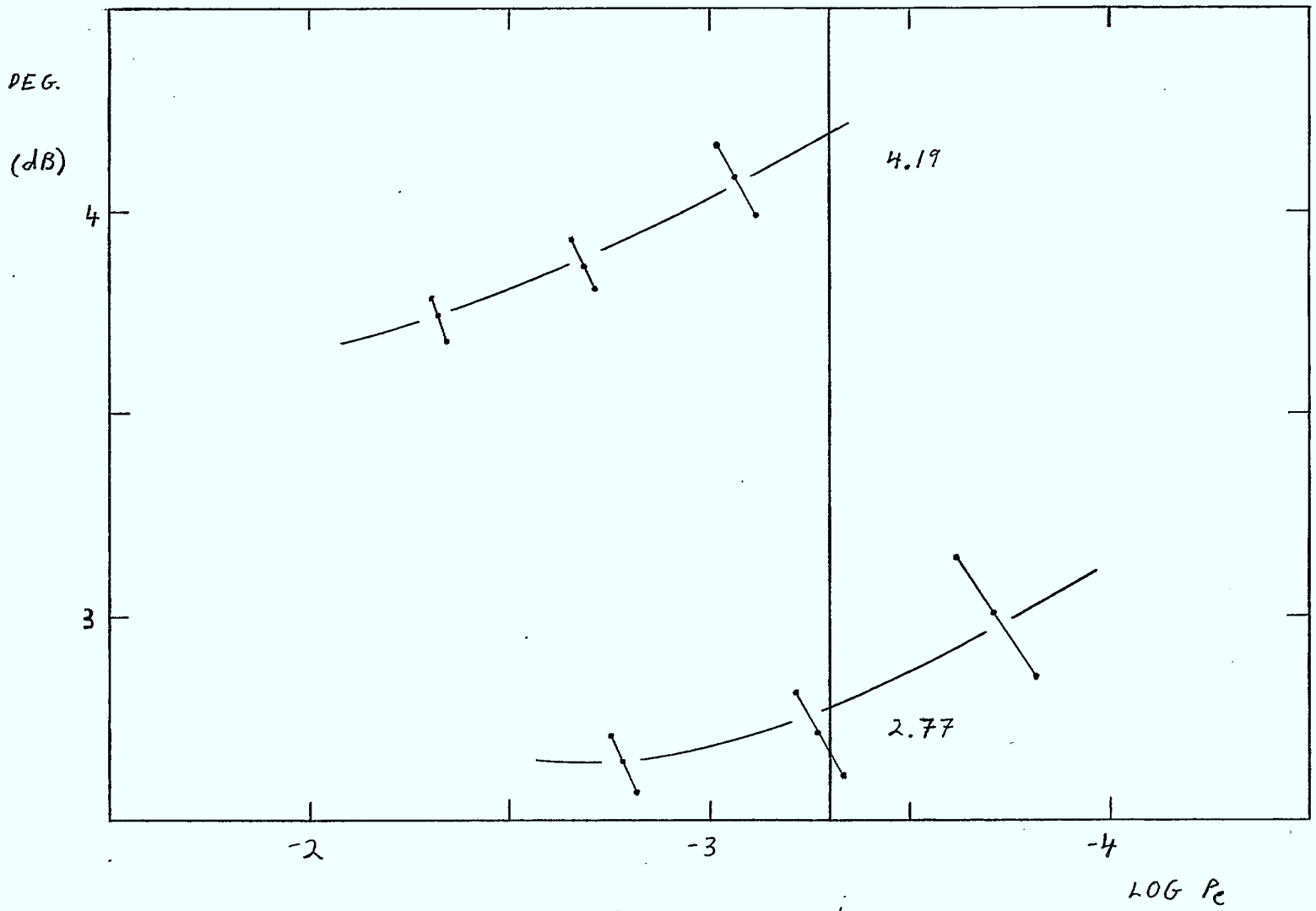

FIGURE A.3-2 Transmit Filter Analysis With:
Rx = BW4E (BT=1.1)
Tx = BW4E (BT=1.1)

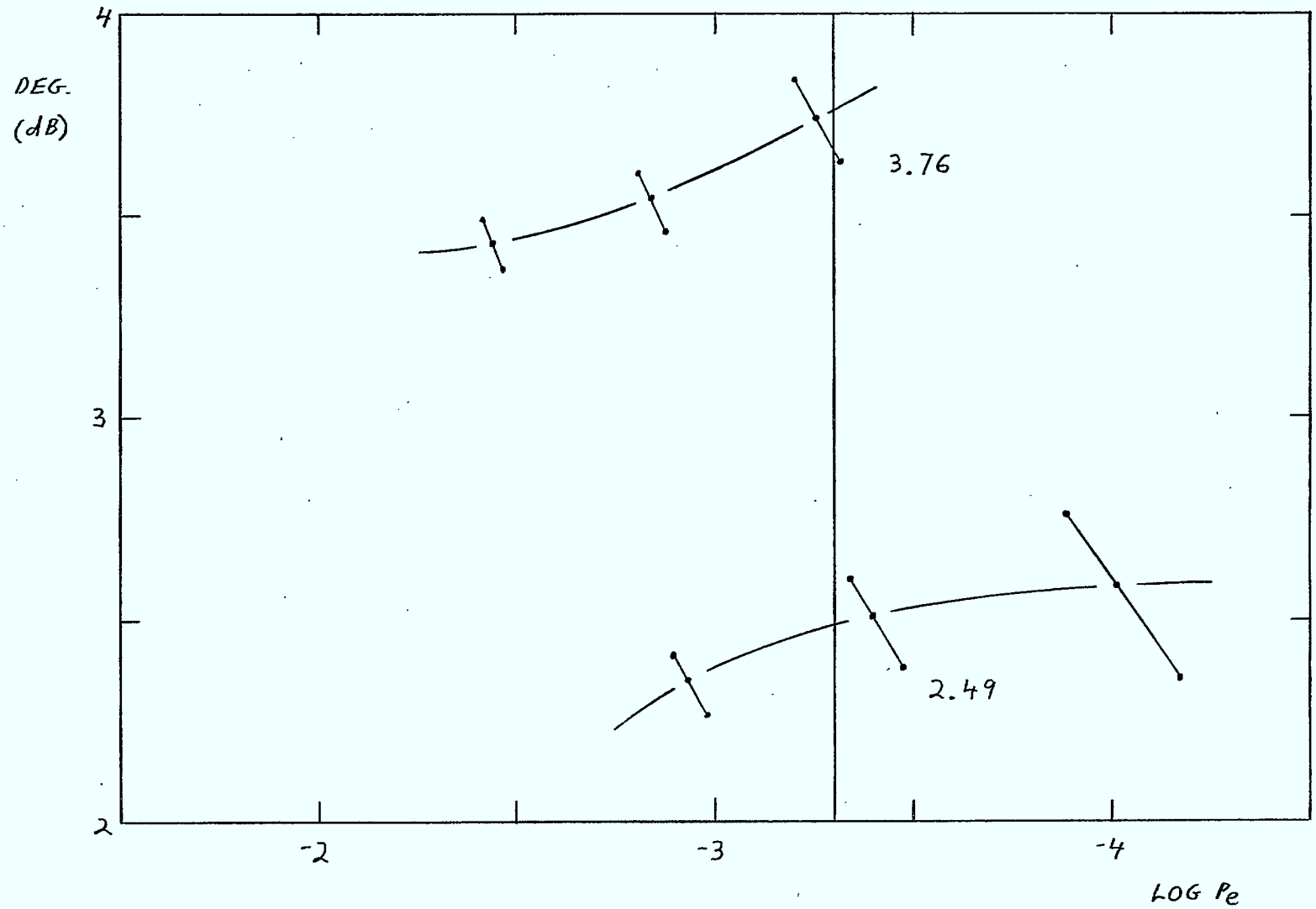

FIGURE A.3-3 Transmit Filter Analysis With:
Rx = BW4E (BT = 1.1)
Tx = BW4E (BT = 1.2)

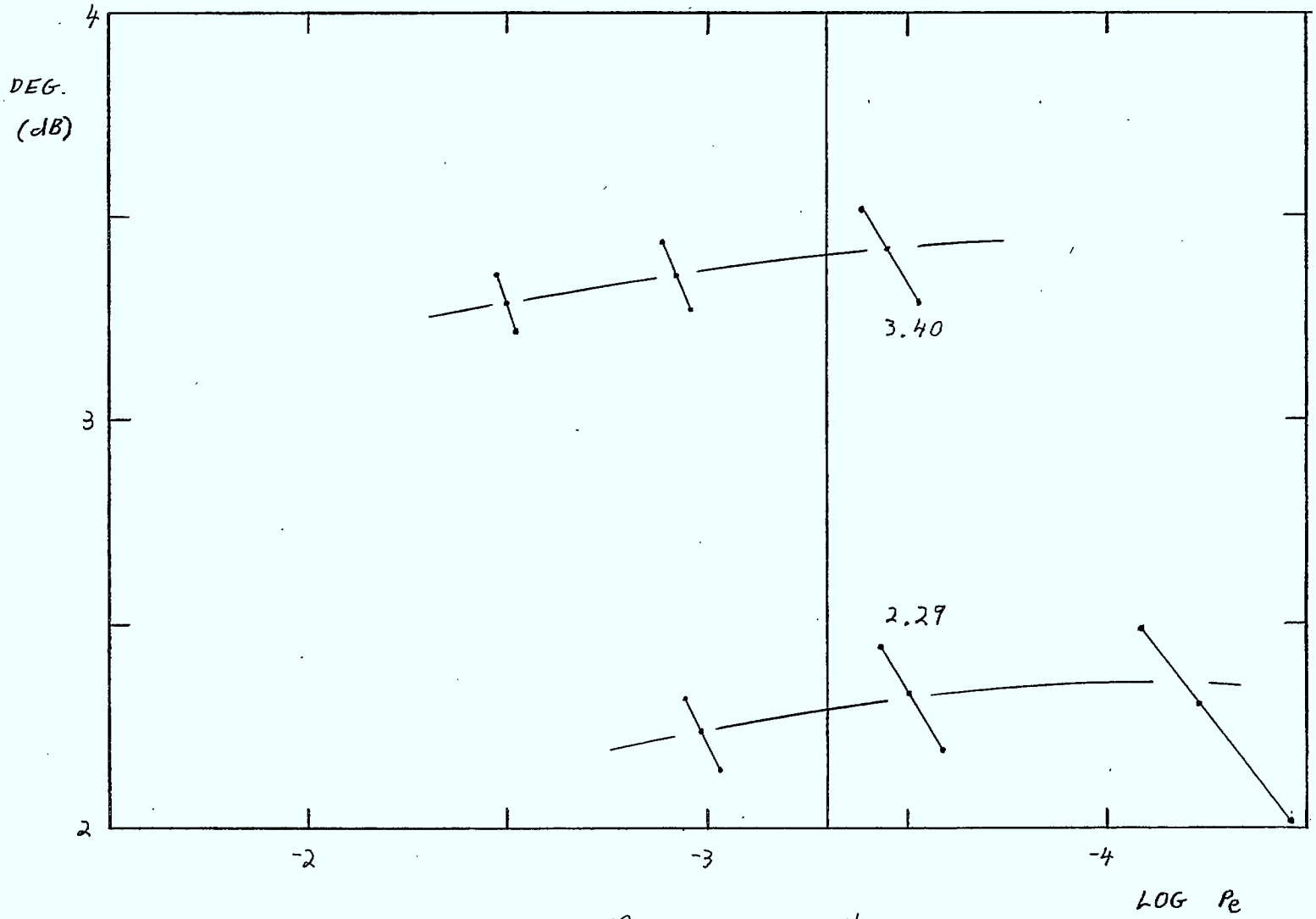

FIGURE A.3-4. Transmit Filter Analysis With:
Rx = BW4E (BT=1.1)
Tx = BW4E (BT=1.3)

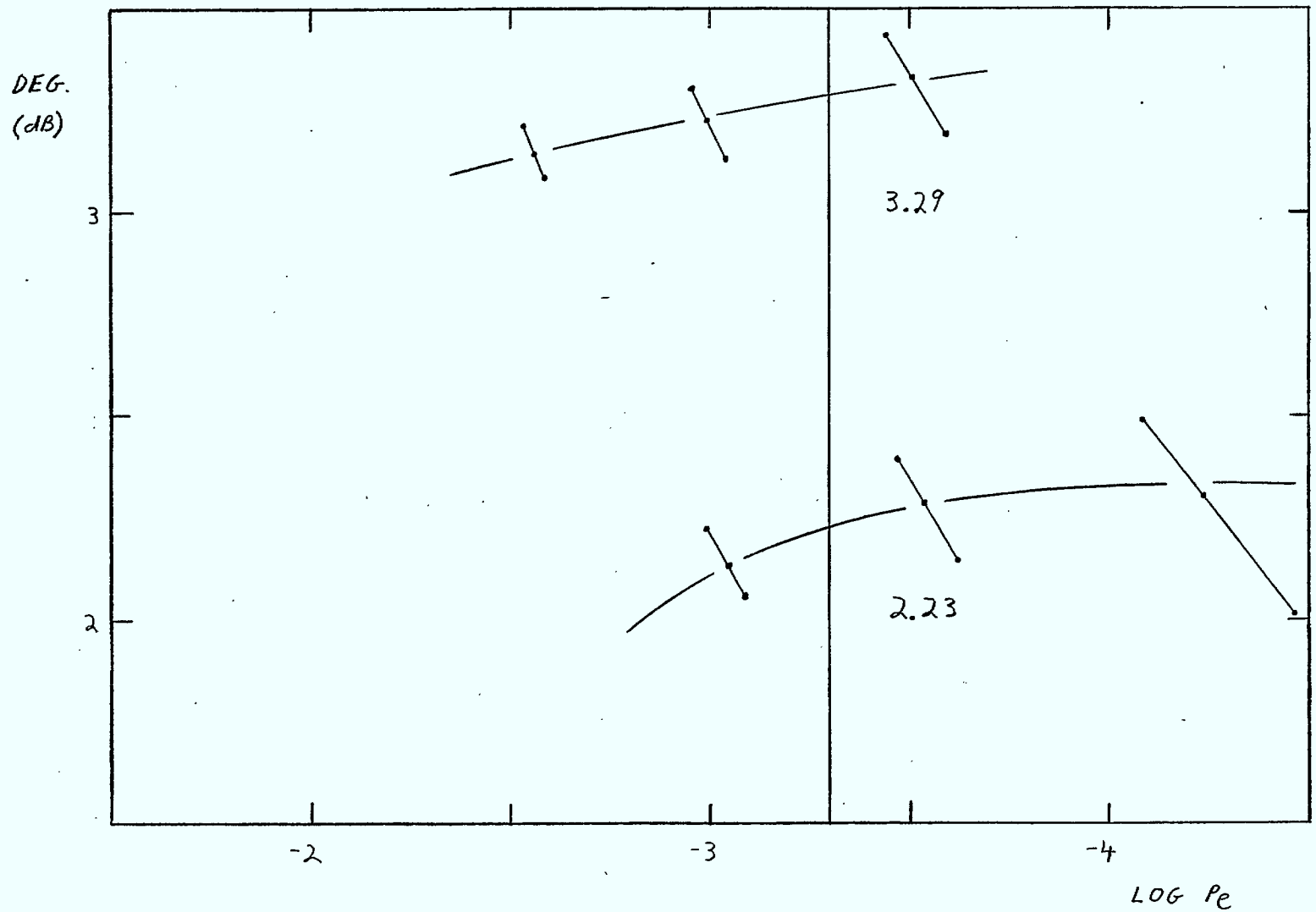

FIGURE A.3-5 Transmit Filter Analysis With:
Rx = BW4E (BT = 1.1)
Tx = BW4E (BT = 1.4)

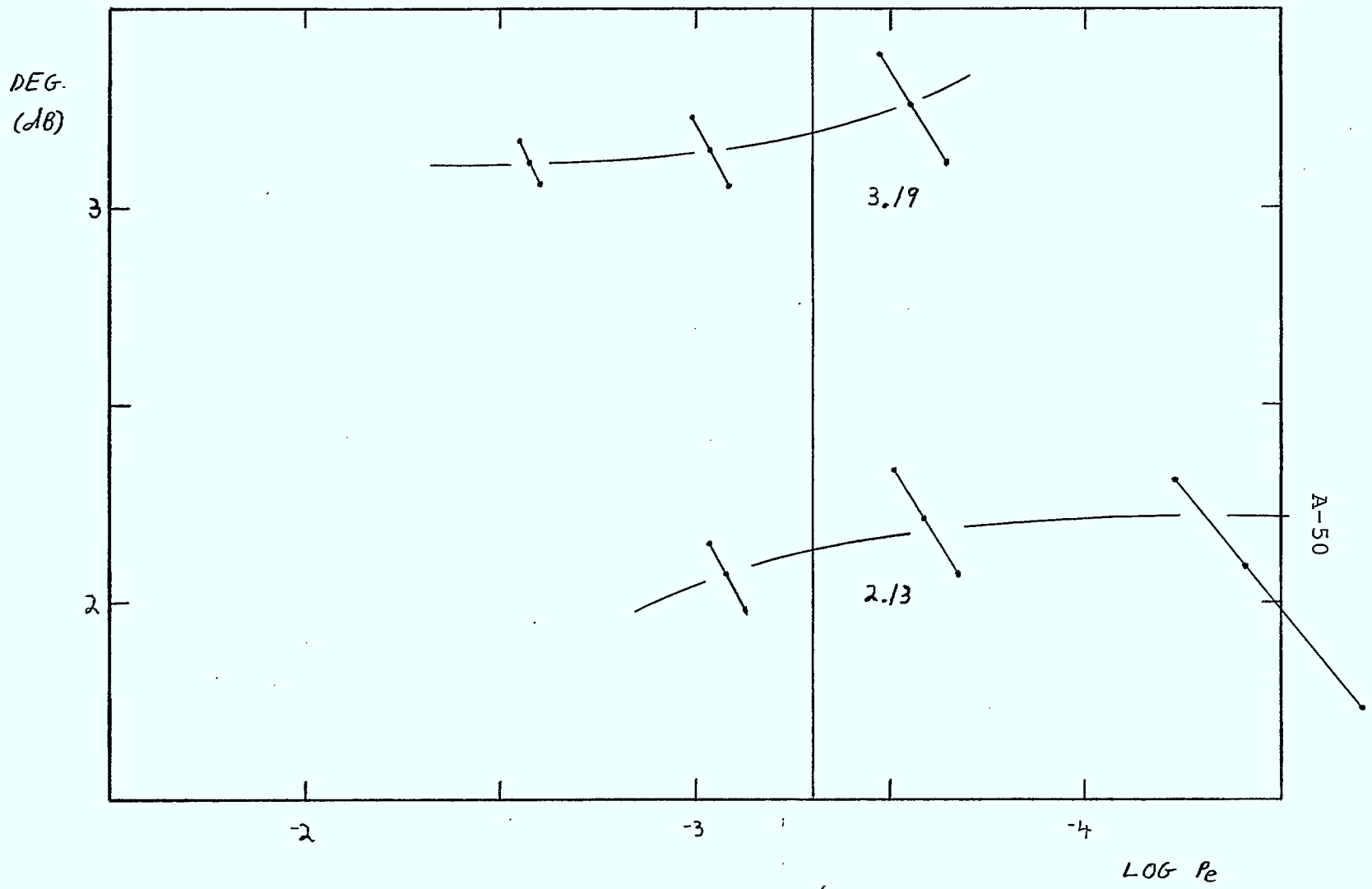

FIGURE A.3-6 Transmit Filter Analysis With:
Rx = BW4E (BT=1.1)
Tx = BW4E (BT=1.5)

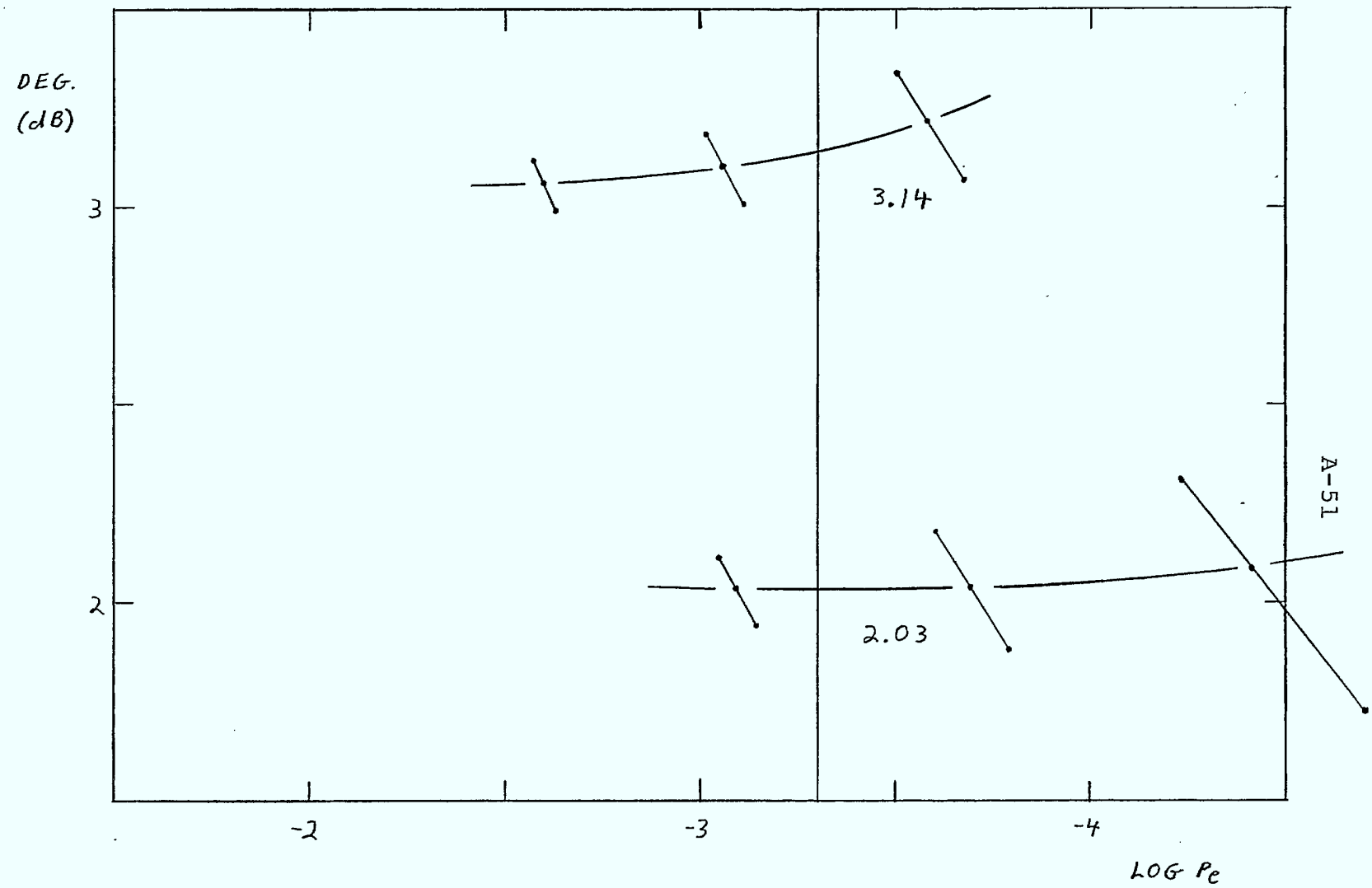

FIGURE A.3-7 Transmit Filter Analysis With:
Rx = BW4E (BT = 1.1)
Tx = BW4E (BT = 1.6)

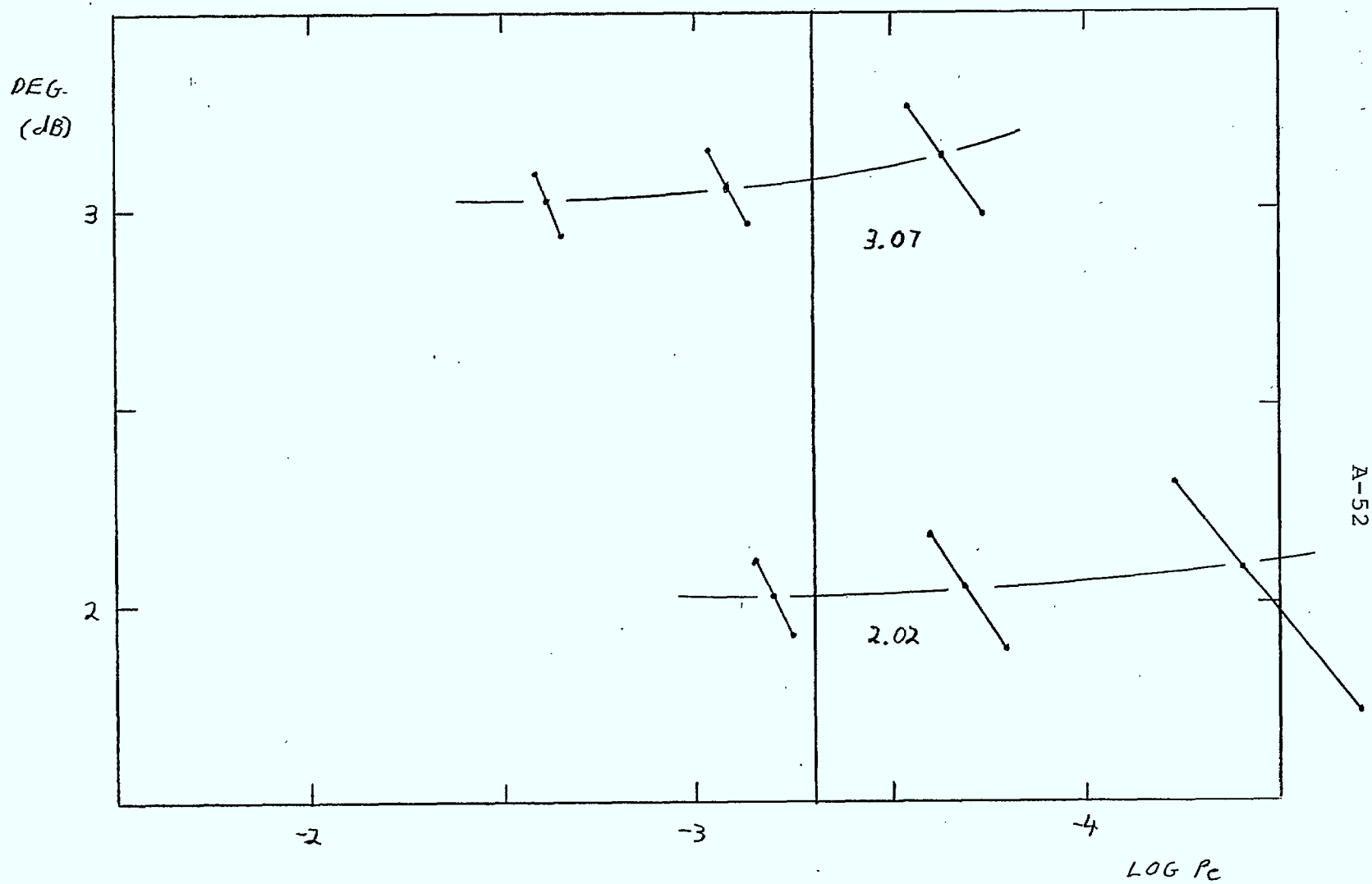

FIGURE A-3-8 Transmit Filter Analysis With:
Rx = BW4E (BT = 1.1)
Tx = BW4E (BT = 1.8)

APPENDIX B

INTERMEDIATE SIMULATION RESULTS FOR SENSITIVITY EVALUATION

The figure numbers have been labeled using the format B.X-Y where Y is the figure number in the category B.X, and the categories are as follows:

B.1 Bit Timing Errors

B.2 Threshold Errors

B.3 Delay Errors

B.4 Phase Shift Errors

B.5 Carrier Frequency Offset

B.6 Non-Constant Group Delay

The parameters given with each figure are directly related to the simulation input parameters. The upper and lower curves in each figure refer to the performance of conventional DMSK and with SEC respectively.

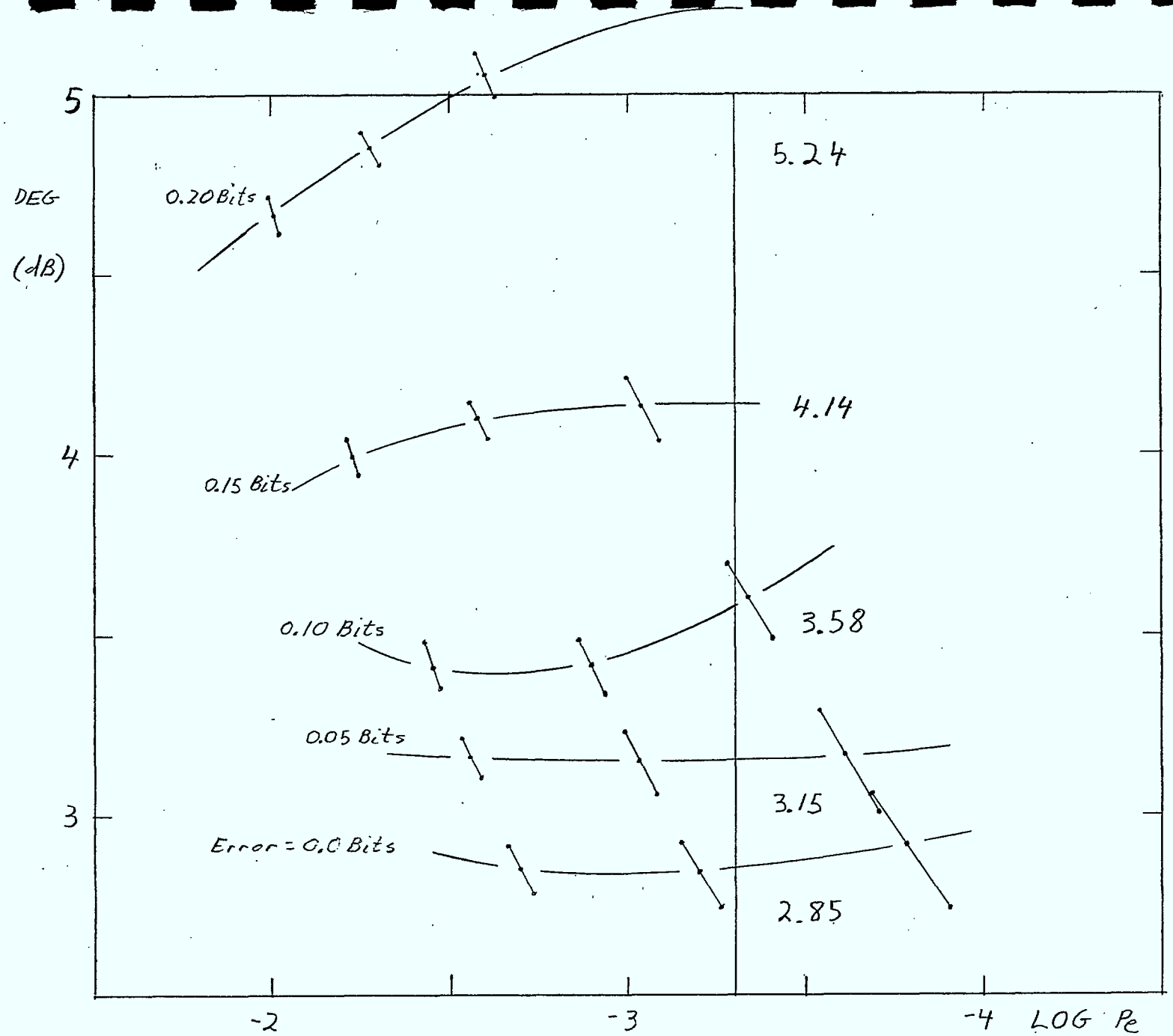

FIGURE B.1-1 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) For Various Bit Timing Errors

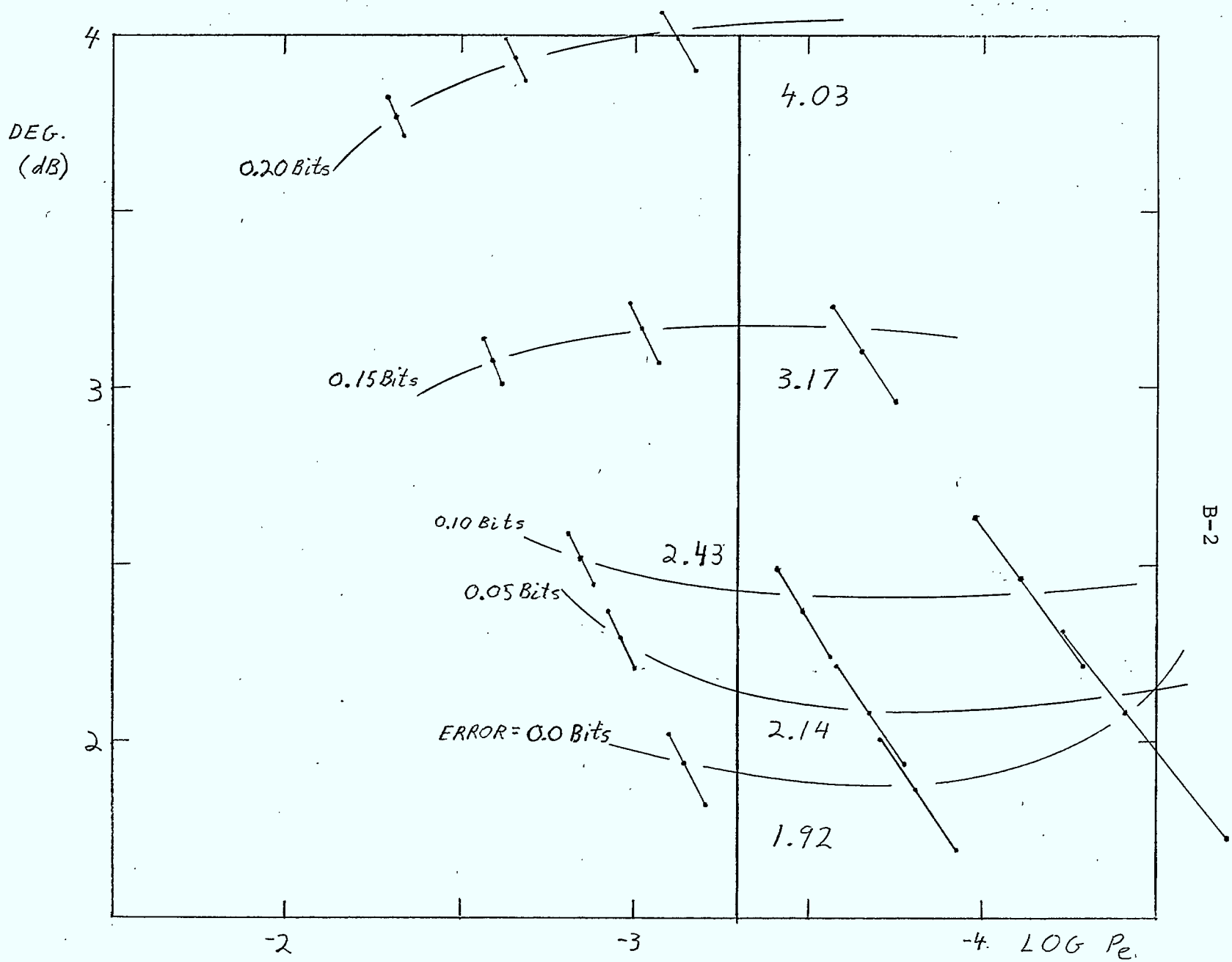

FIGURE B.1-2 Simulated DMSK With SEC For A 4th Order Butterworth Equalized Receiver Filter (BT = 1.1) For Various Bit Timing Errors

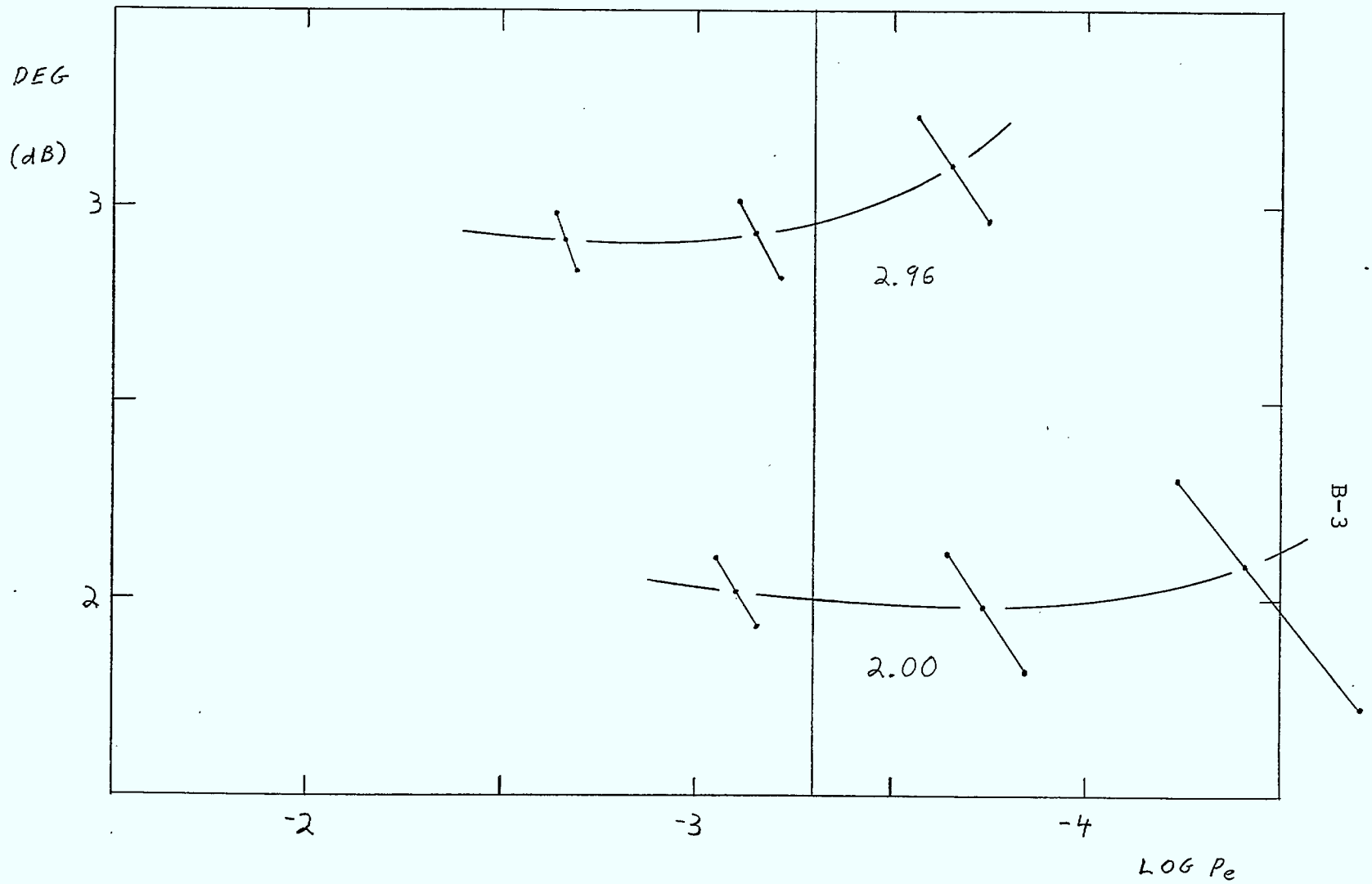

FIGURE B.2-1 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Threshold Level = 0.05

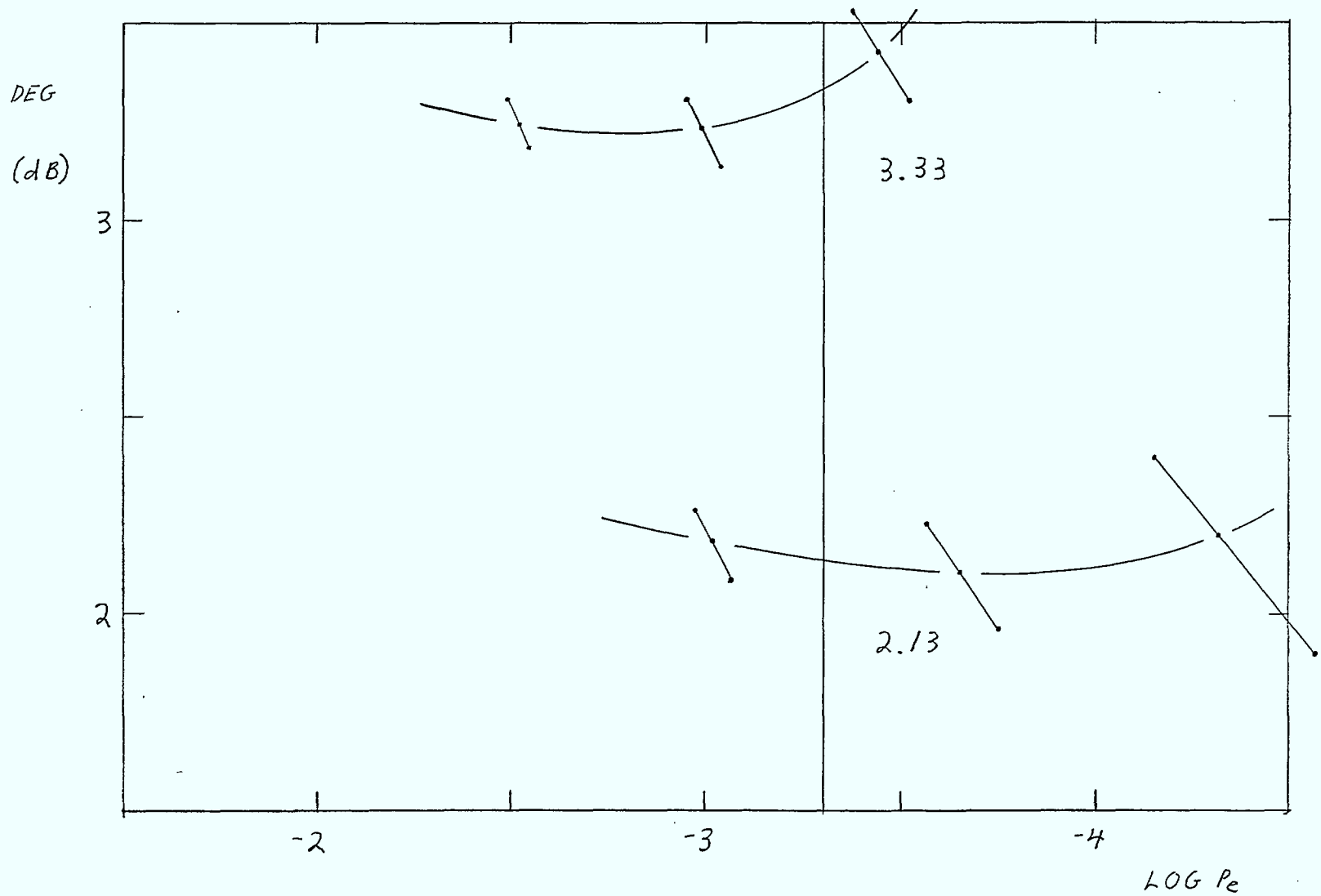

FIGURE B.2-2 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And. Threshold Level = 0.10

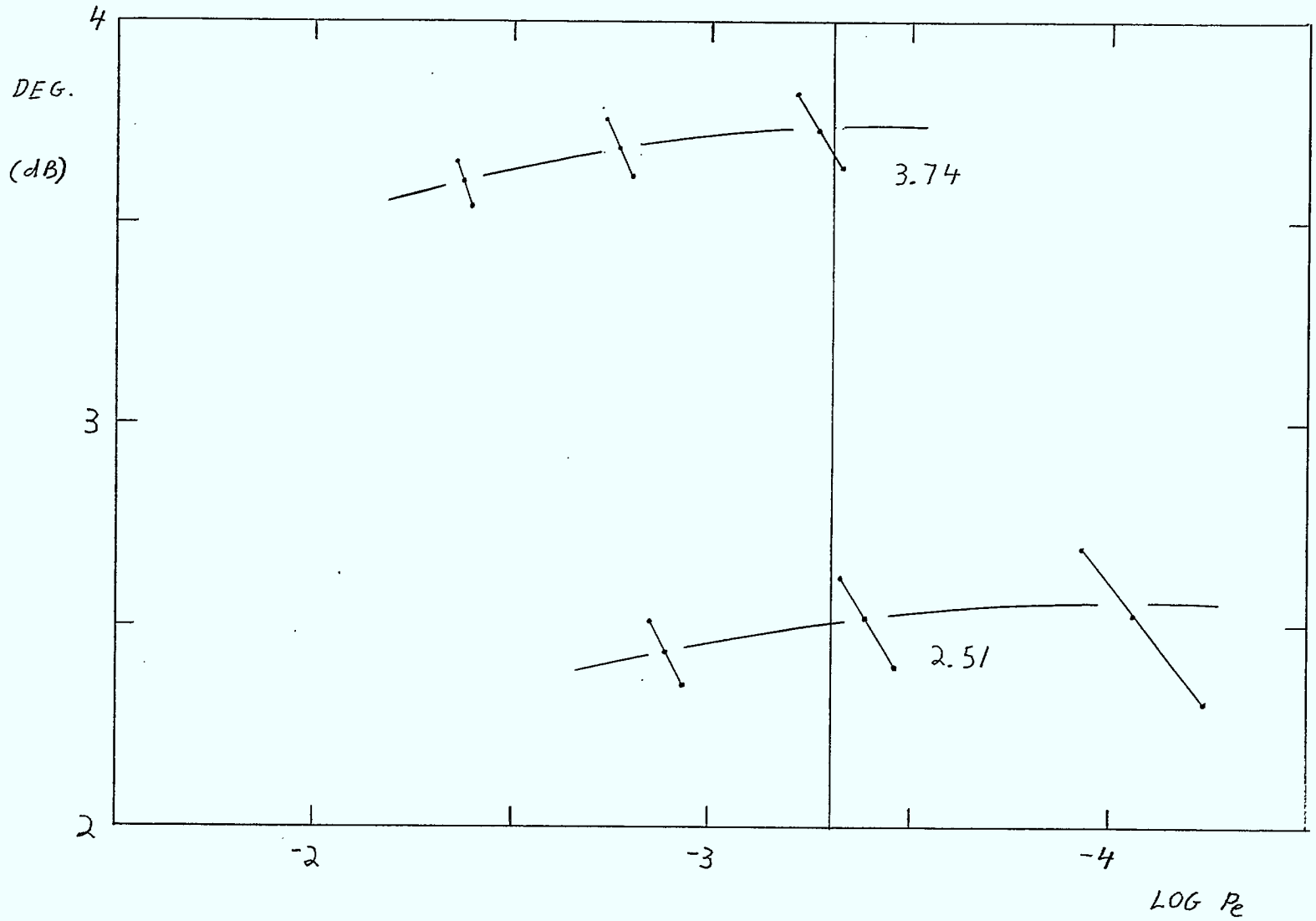

FIGURE B-2-3 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Threshold Level = 0.15

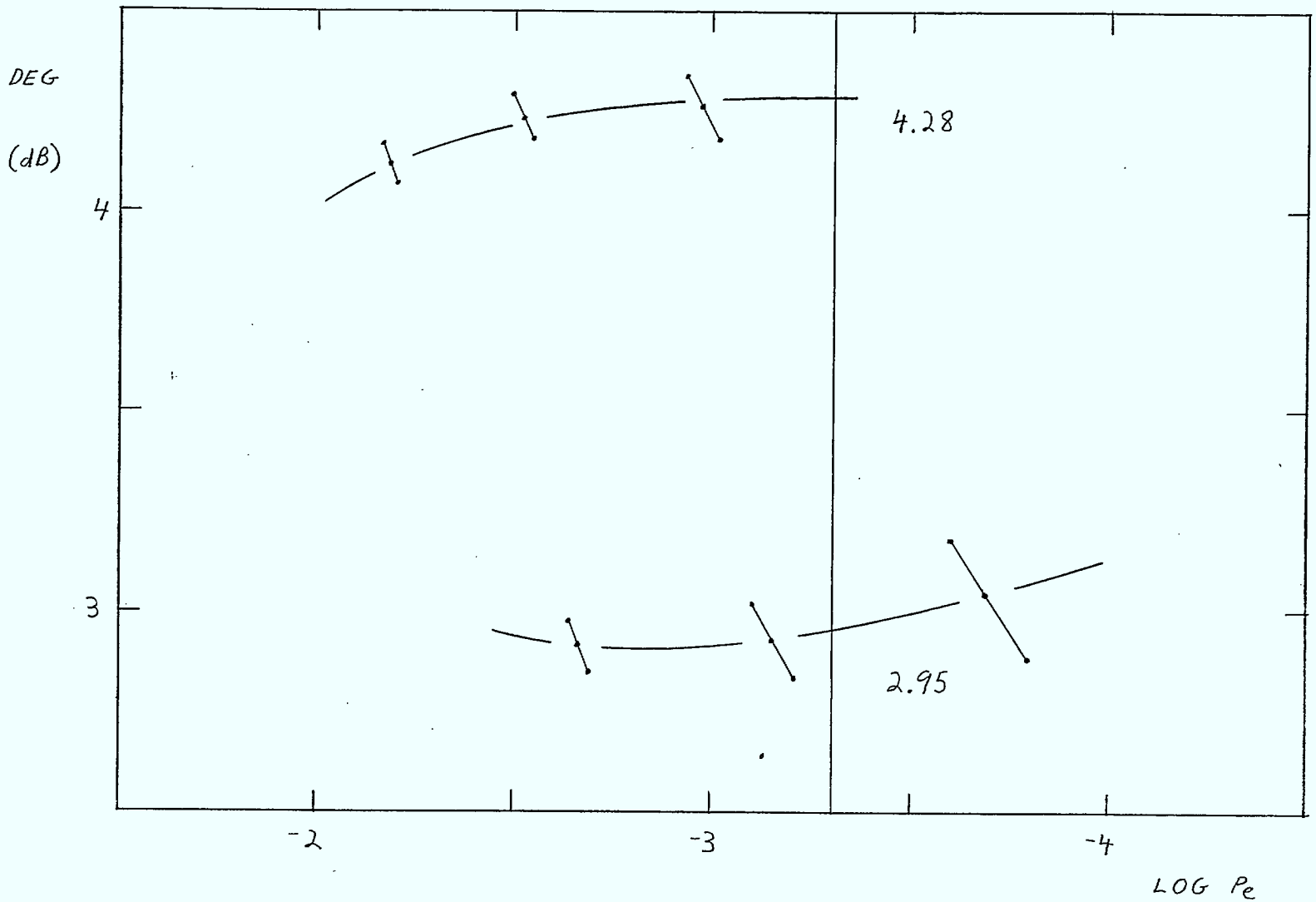

FIGURE B.2-4 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Threshold Level = 0.20

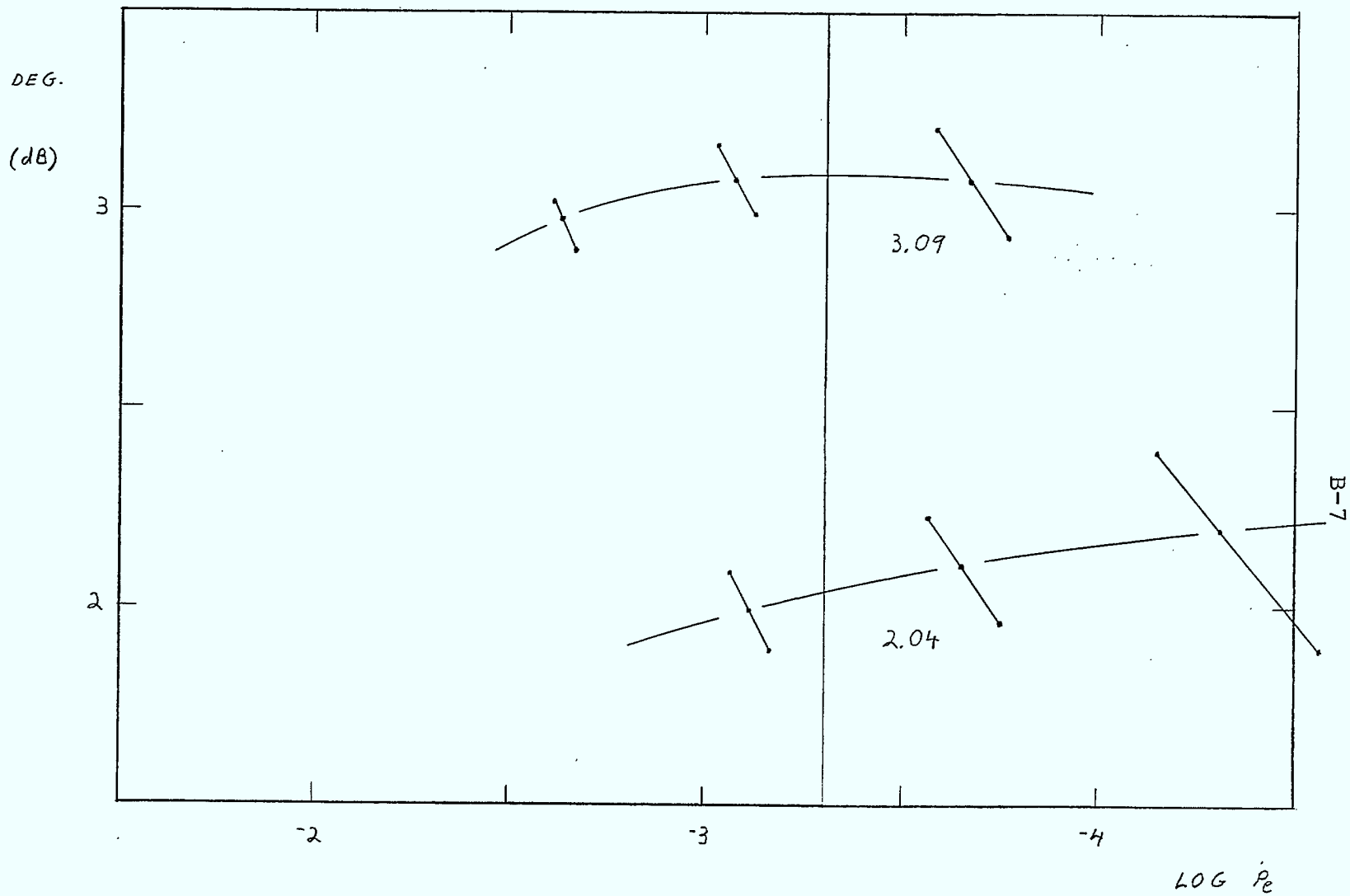

FIGURE B.3-1 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Delay Errors of 0.1%

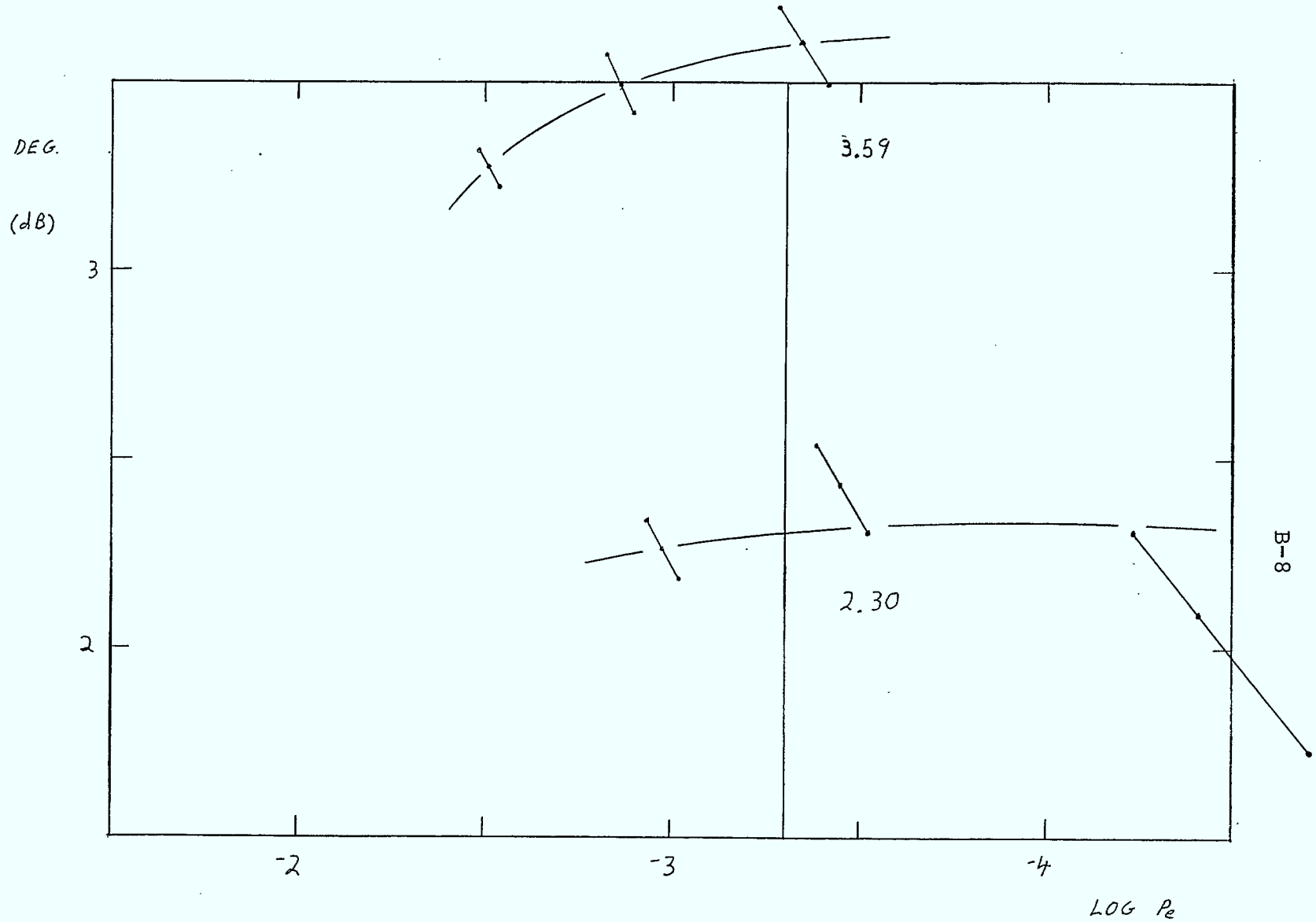

FIGURE B.3-2 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Delay Errors of 0.2%

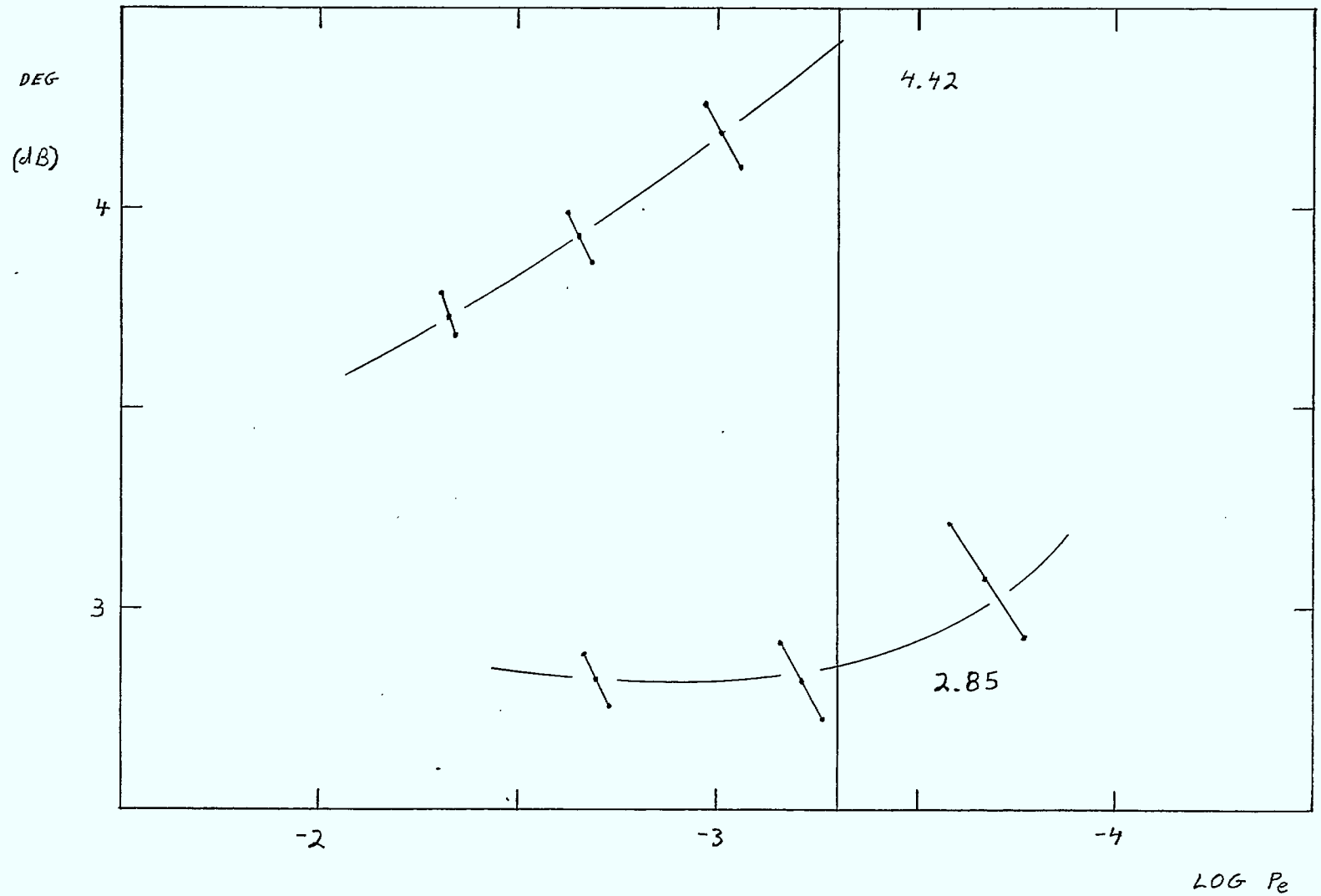

FIGURE B3-3 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Delay Errors of 0.3%

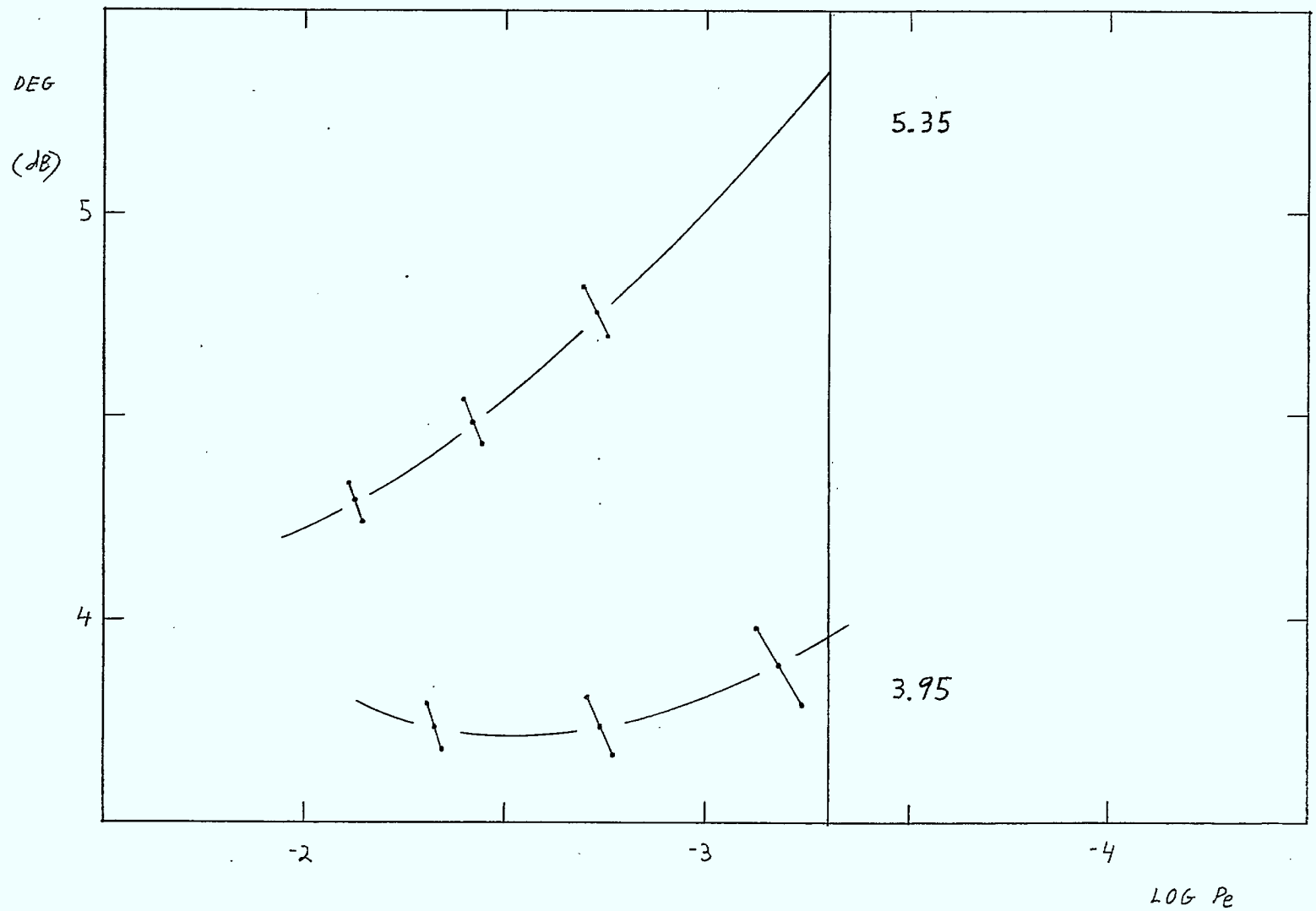

FIGURE B.3-4 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Delay Errors of 0.4 %

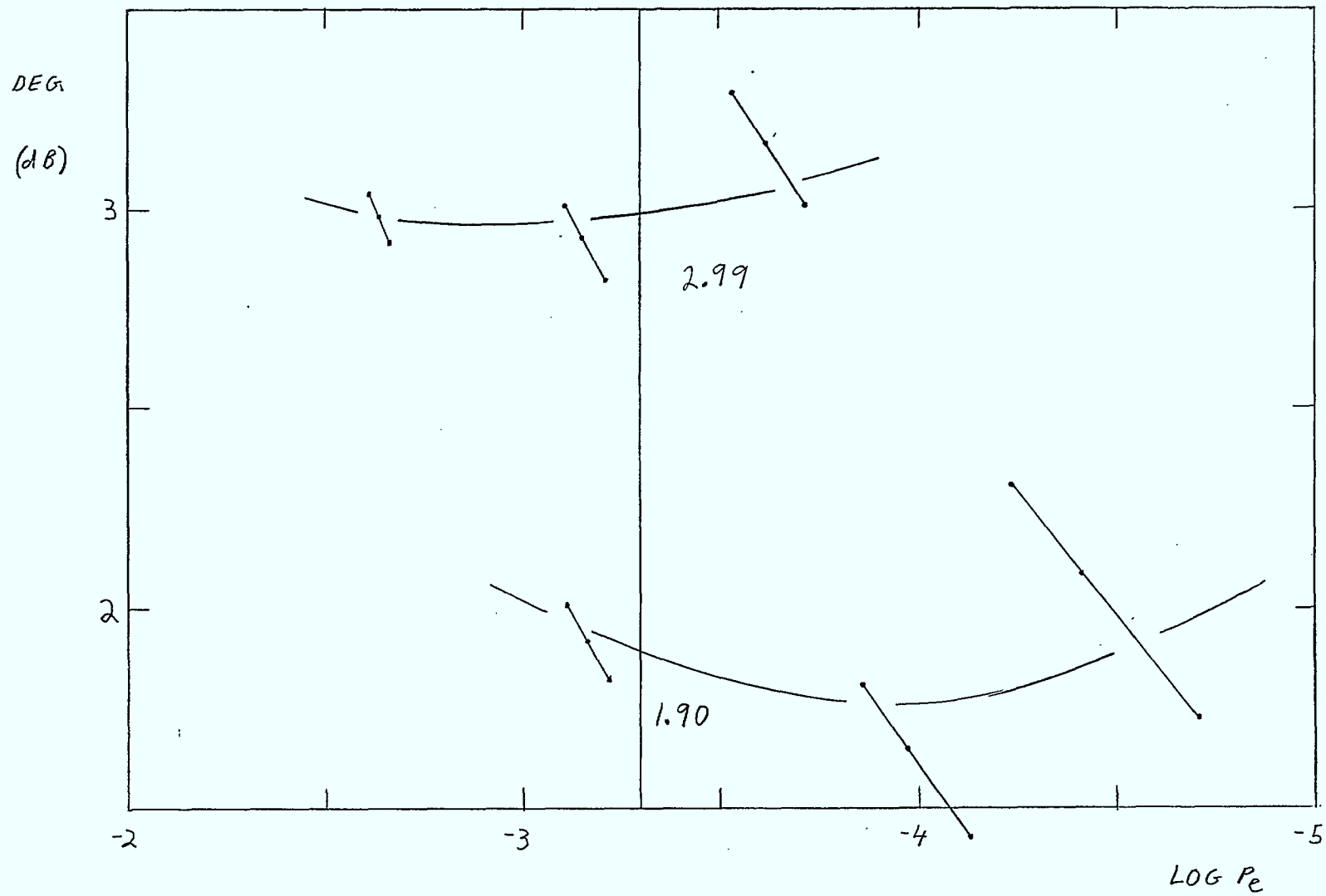

FIGURE B.4-1 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Phase Shift Error = 5°

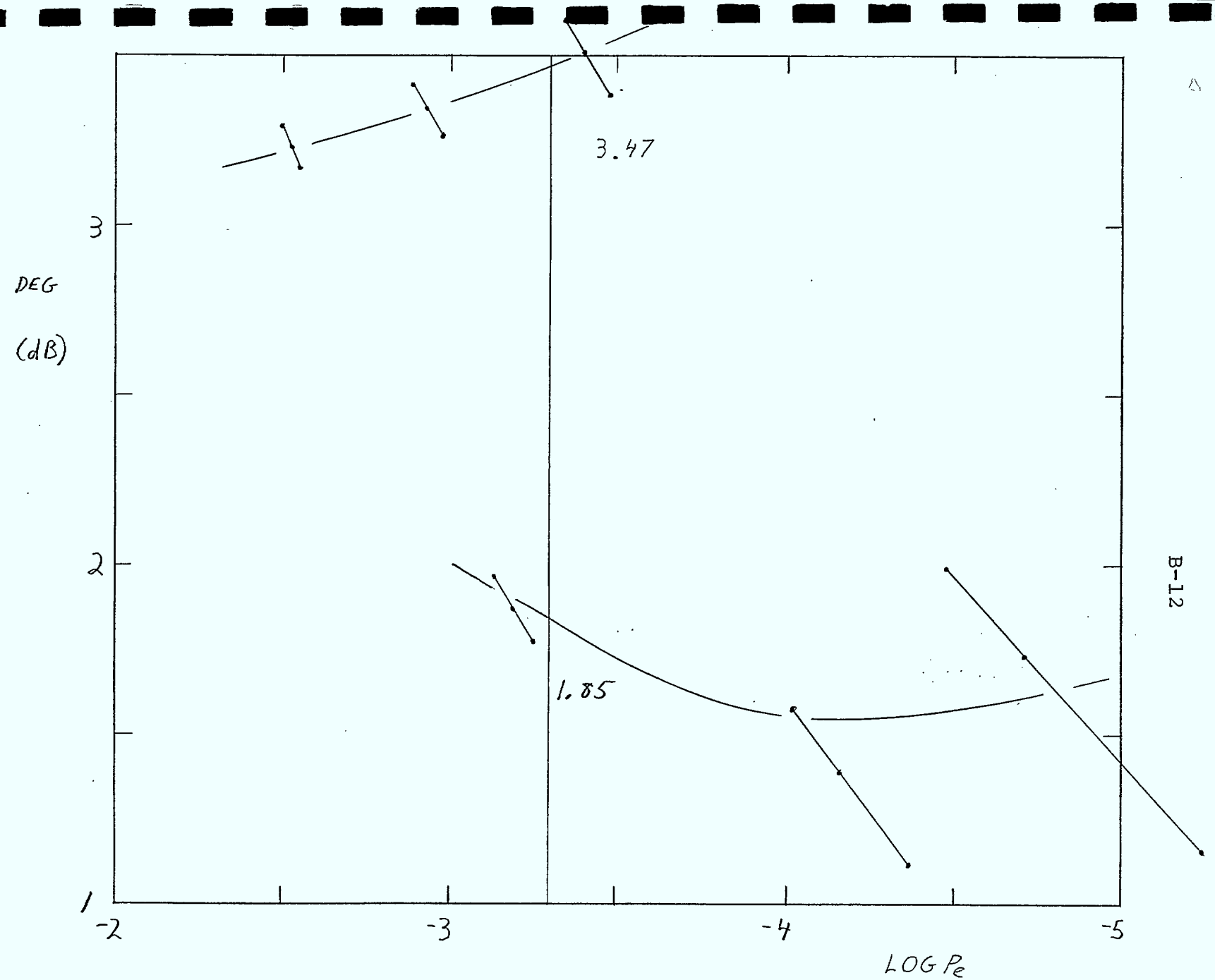

FIGURE B.4-2 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Phase Shift Error=10°

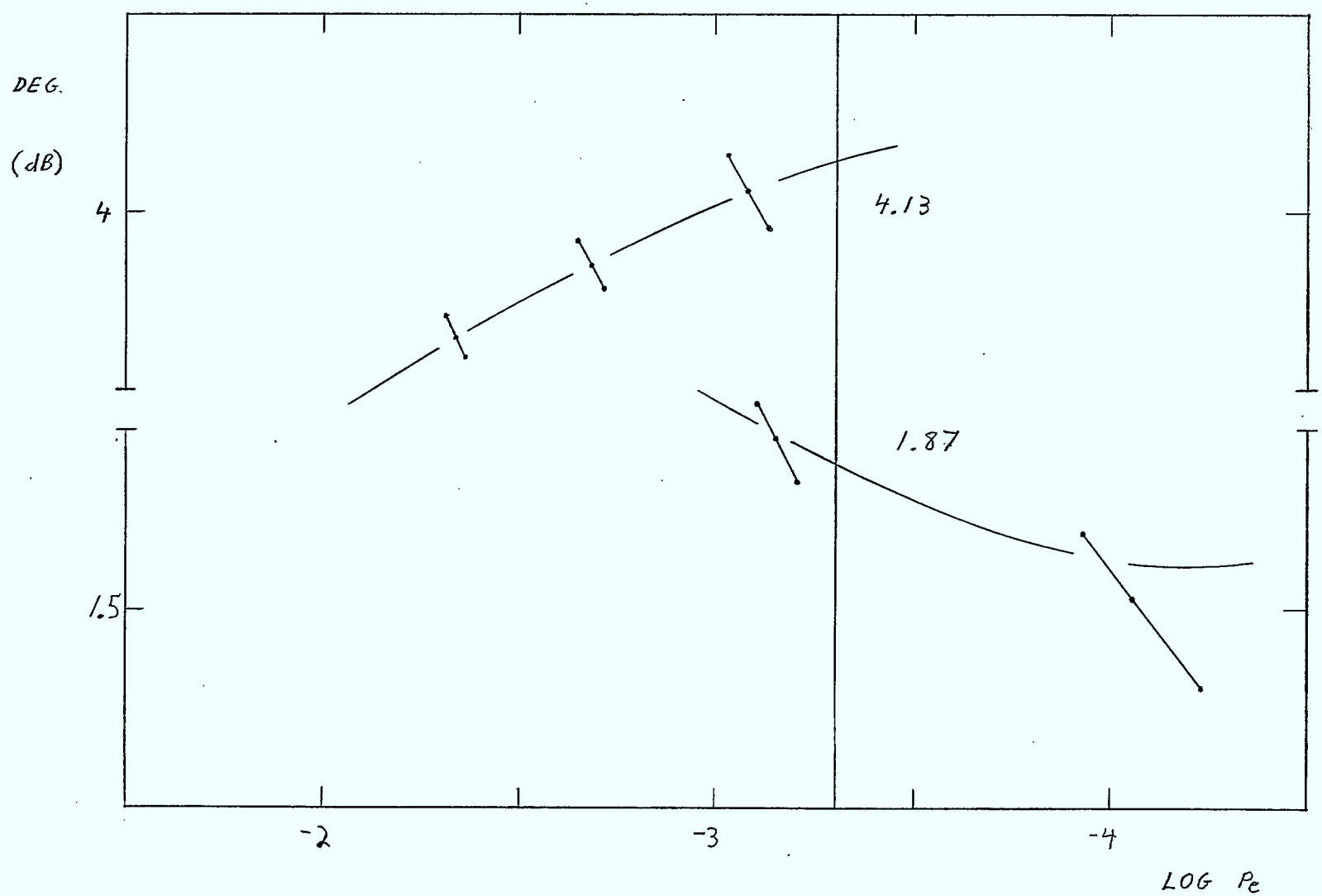

FIGURE B.4-3 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Phase Shift Error = 15°

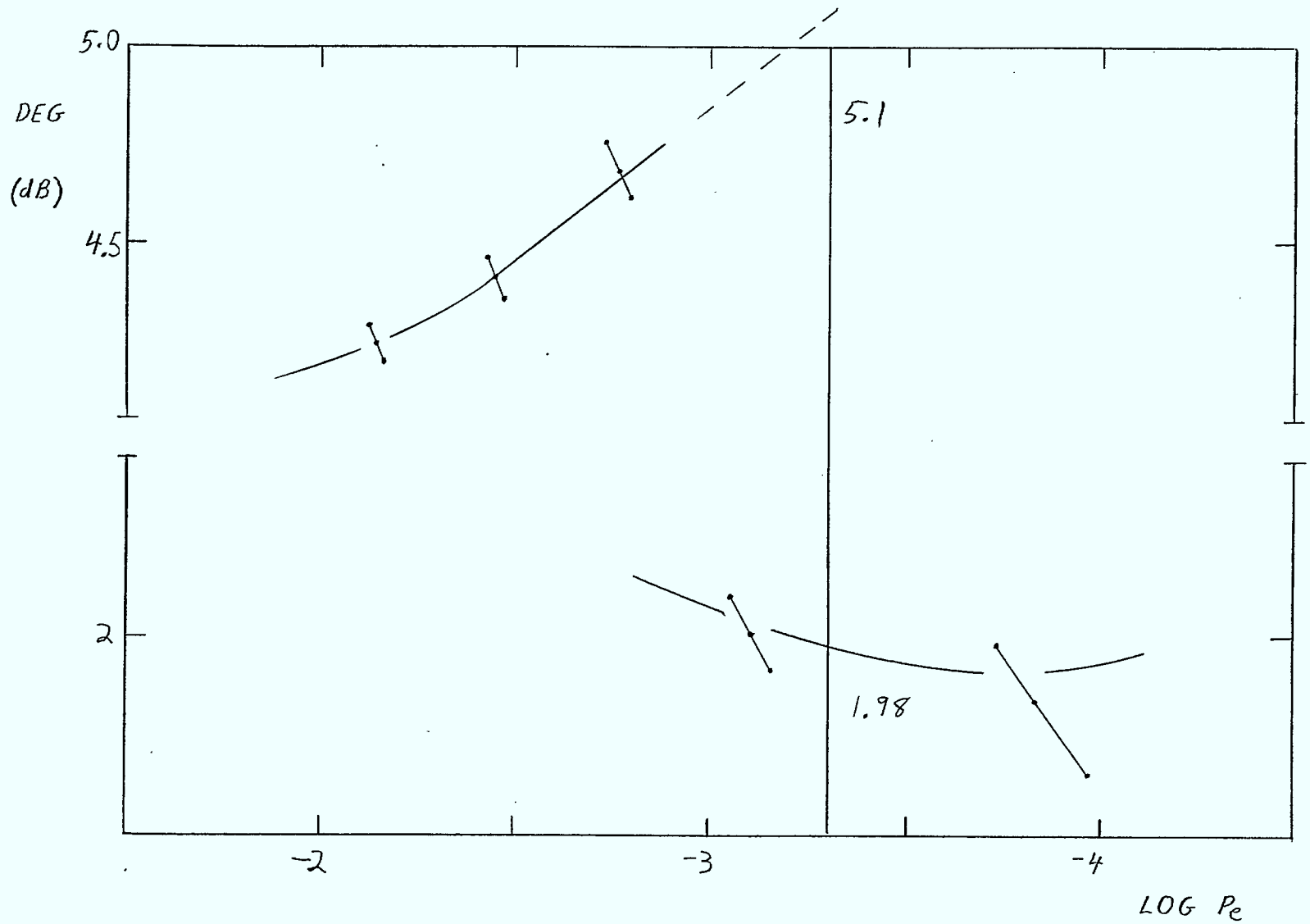

FIGURE B.4-4 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter ($BT = 1.1$) And Signal Phase Shift Error = 20°

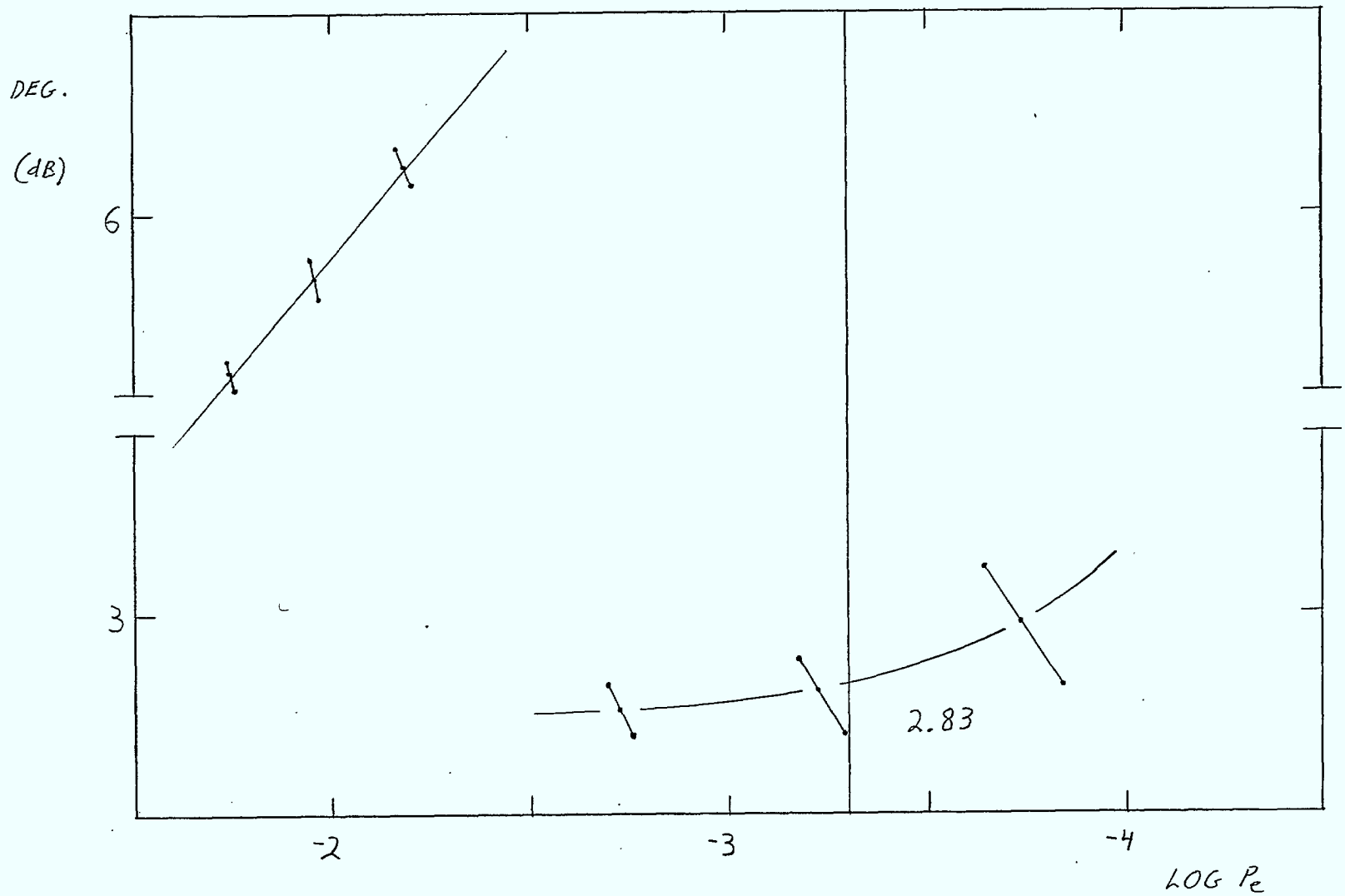

FIGURE B.4-5 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Phase Shift Error = 30°

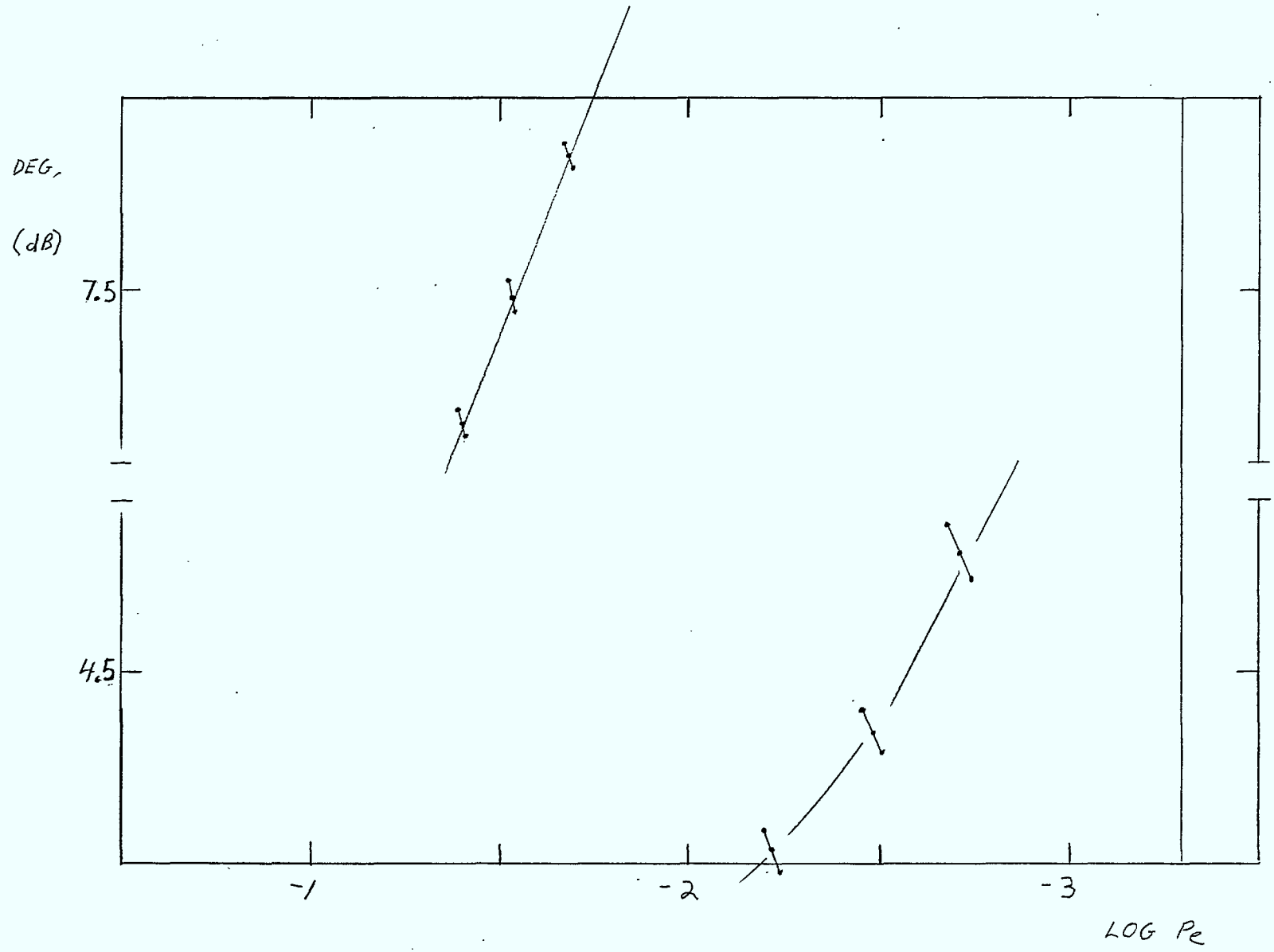

FIGURE B.4-6 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Signal Phase Shift Error = 40°

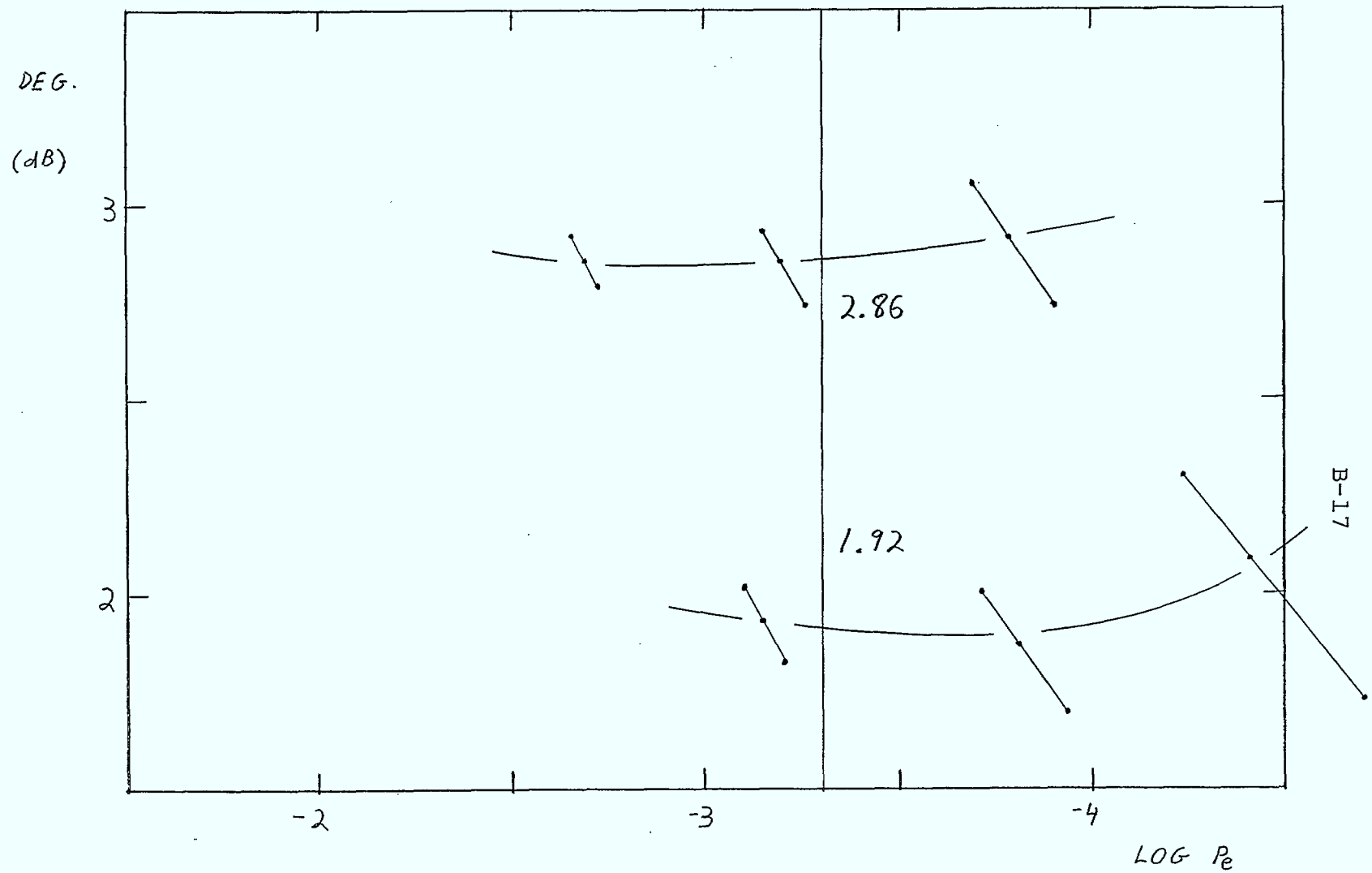

FIGURE B.5-1 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Carrier Frequency = 70.0 MHz.

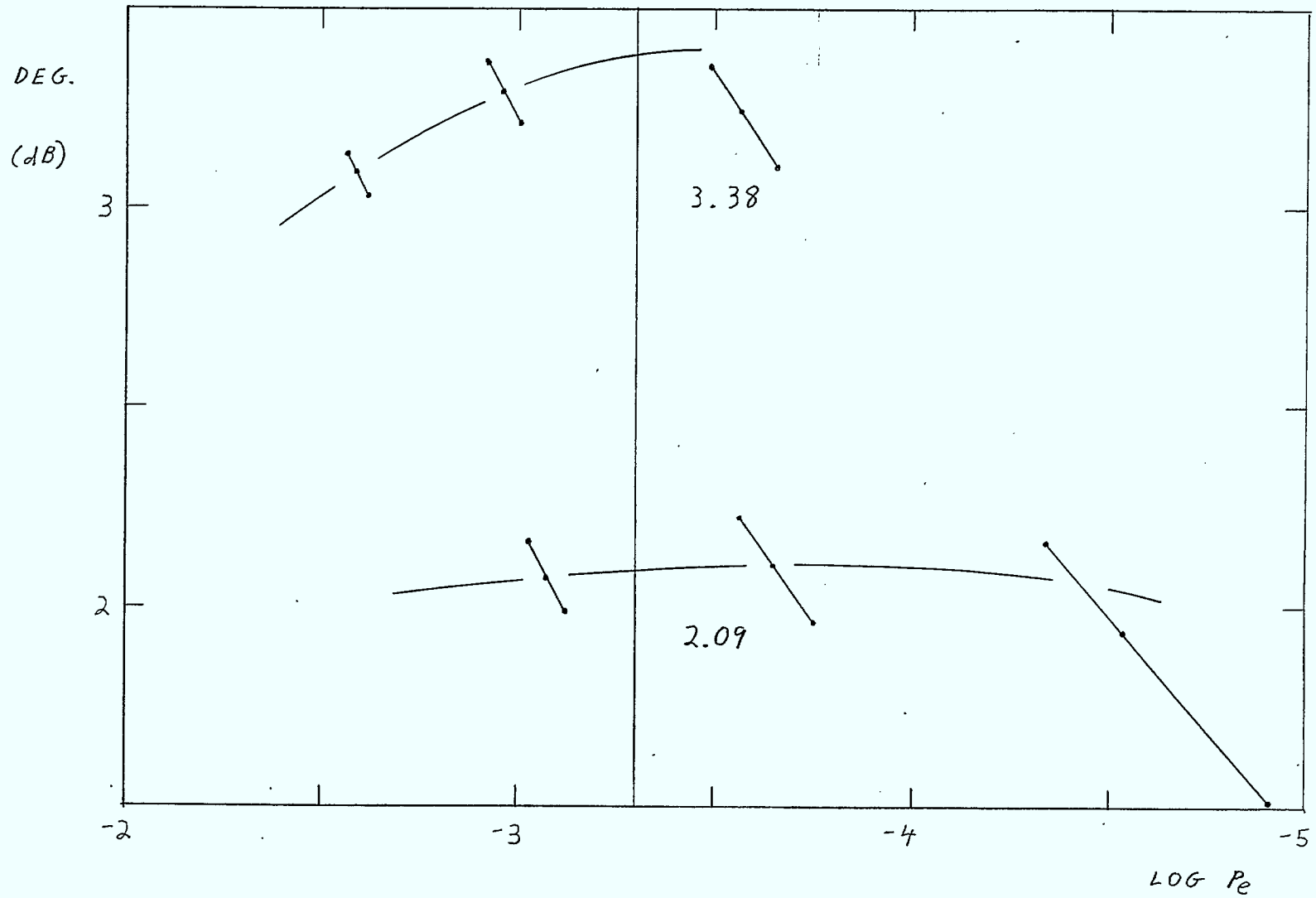

FIGURE B.5-2 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Carrier Frequency = 70.10 MHz

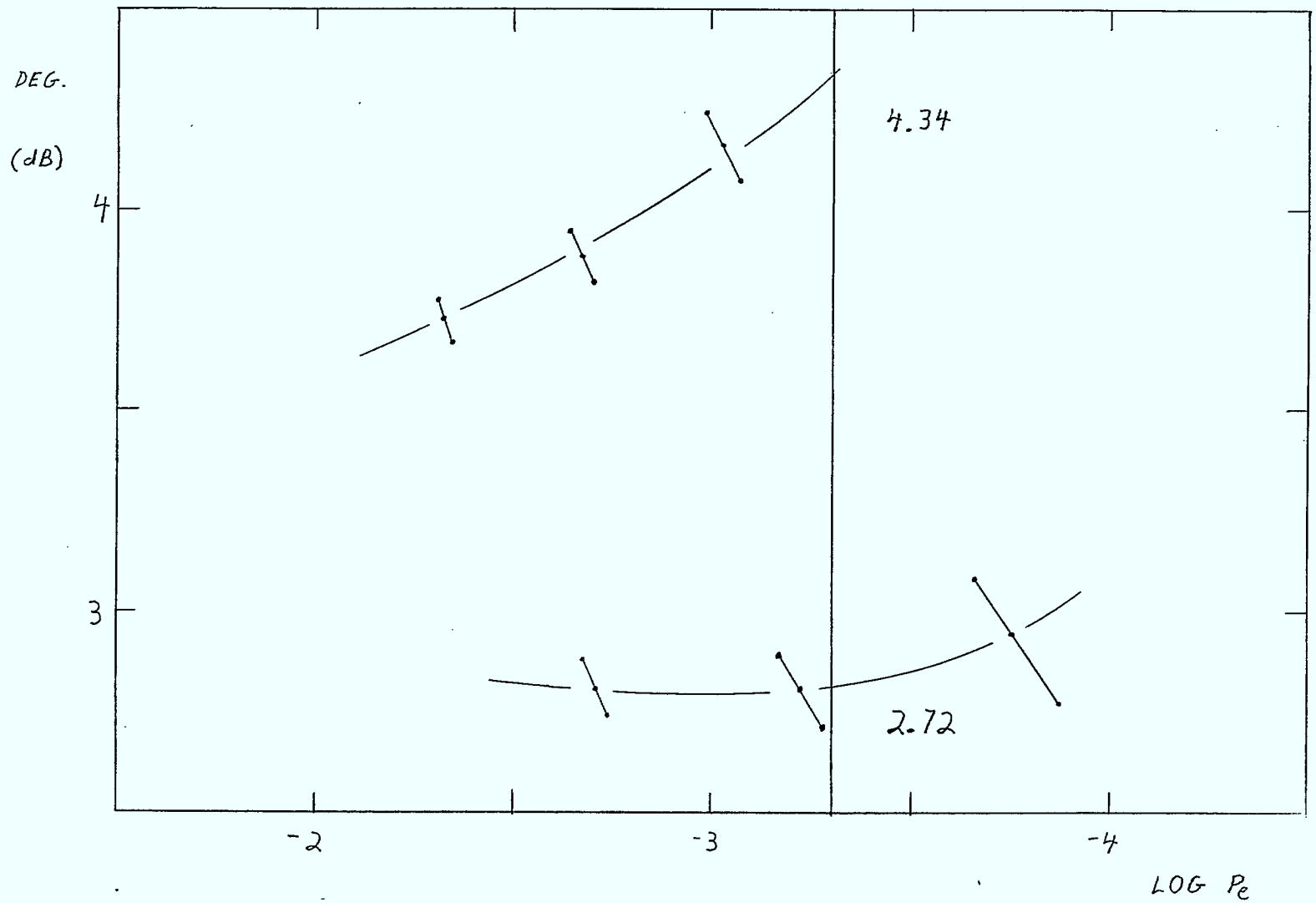

FIGURE B.5-3 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT=1.1) And Carrier Frequency = 70.2 MHz

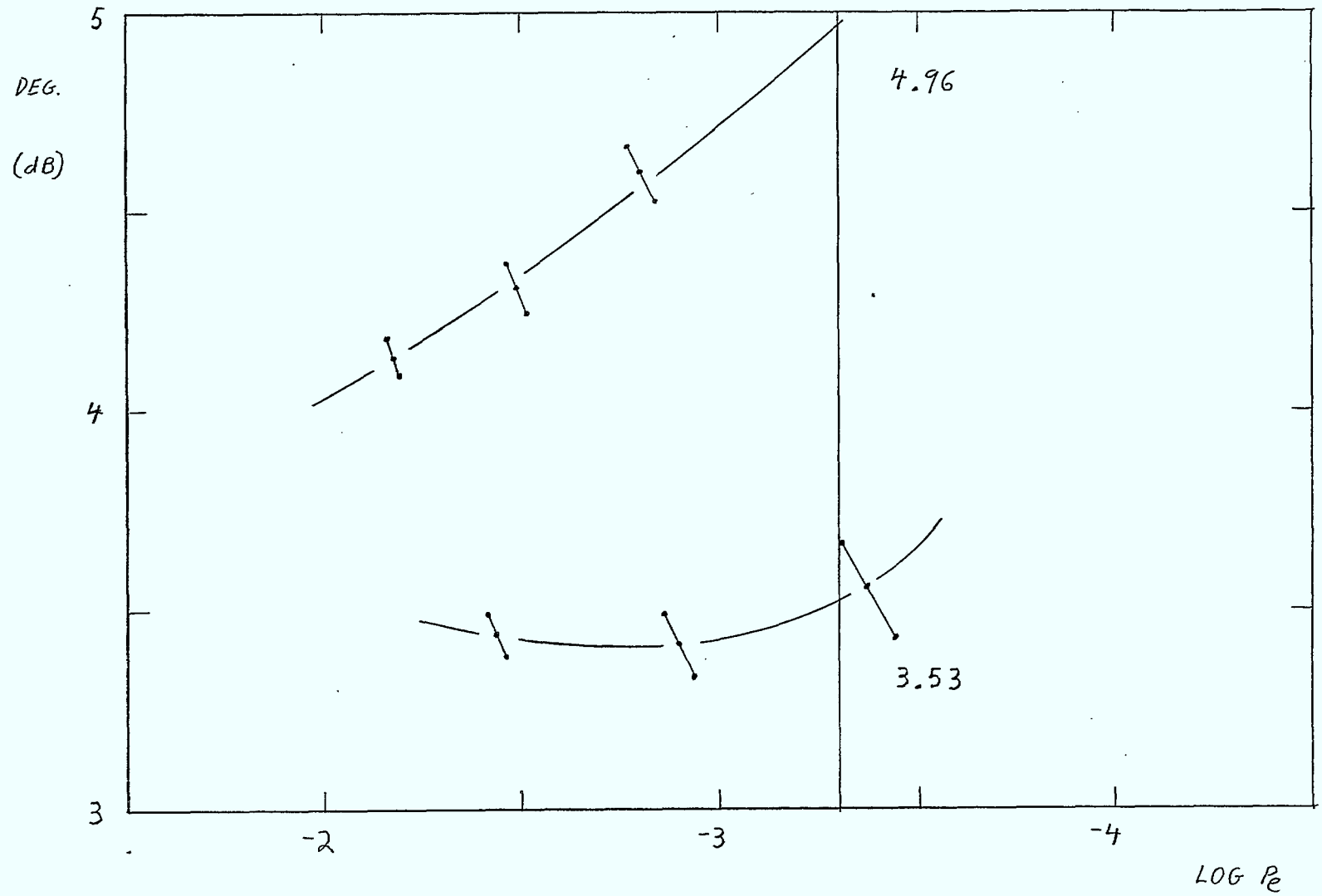

FIGURE B.5-4 Simulated DMSK With. 4th Order Butterworth Equalized Receiver Filter (BT = 1.1) And Carrier Frequency = 70.25 MHz

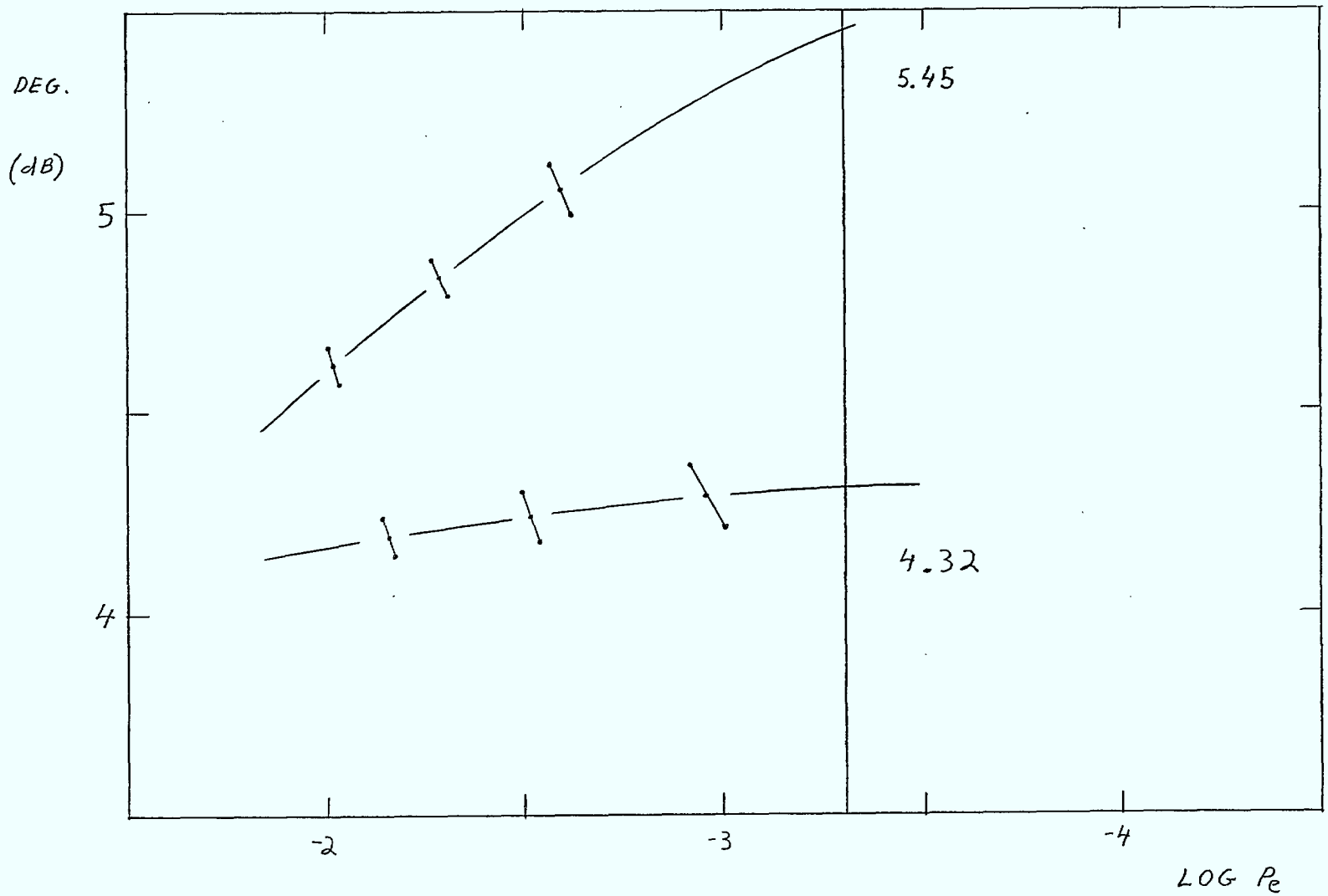

FIGURE B.5-5 Simulated DMSK With 4th Order Butterworth Equalized Receiver Filter (BT = 1.1) And Carrier Frequency = 70.30

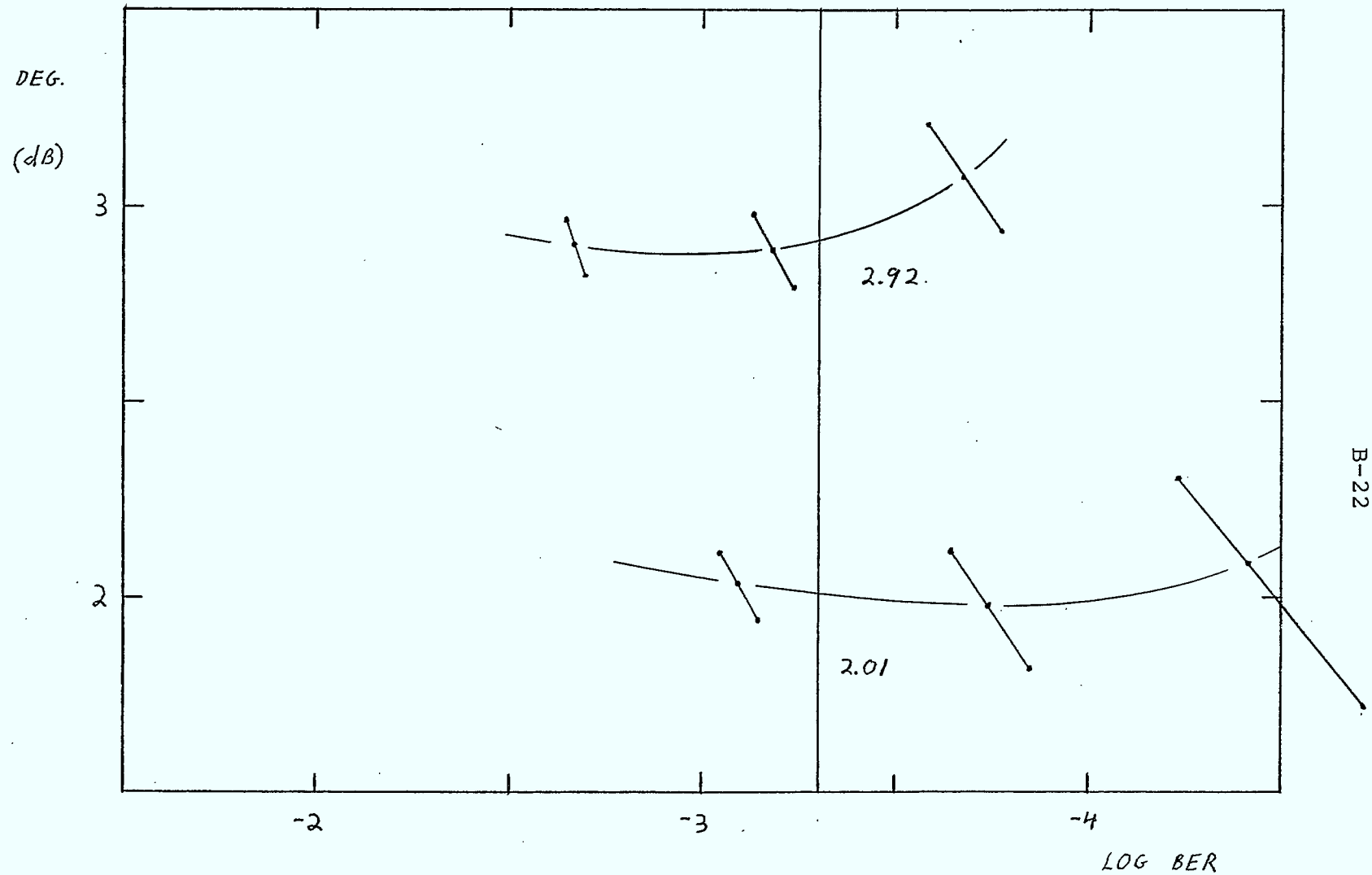

FIGURE B.6-1 Simulated DMSK With:
RX = 4th Order Butterworth Equalized (BT=1.1)
Group Delay (A=0.2)

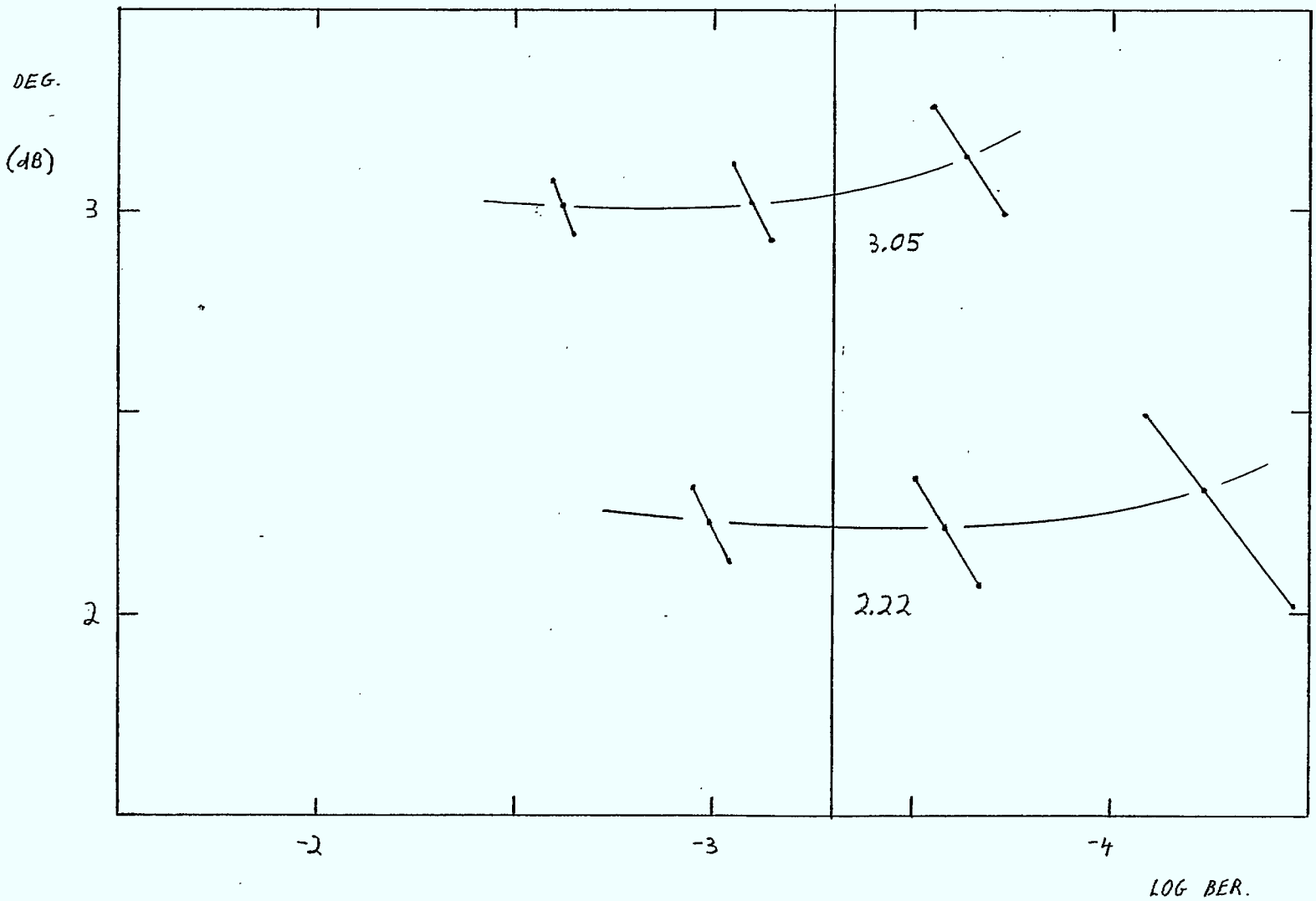

FIGURE B.6-2 Simulated DMSK With:

Rx = 4th Order Butterworth Equalized (BT=1.1)

Group Delay (A= 0.4)

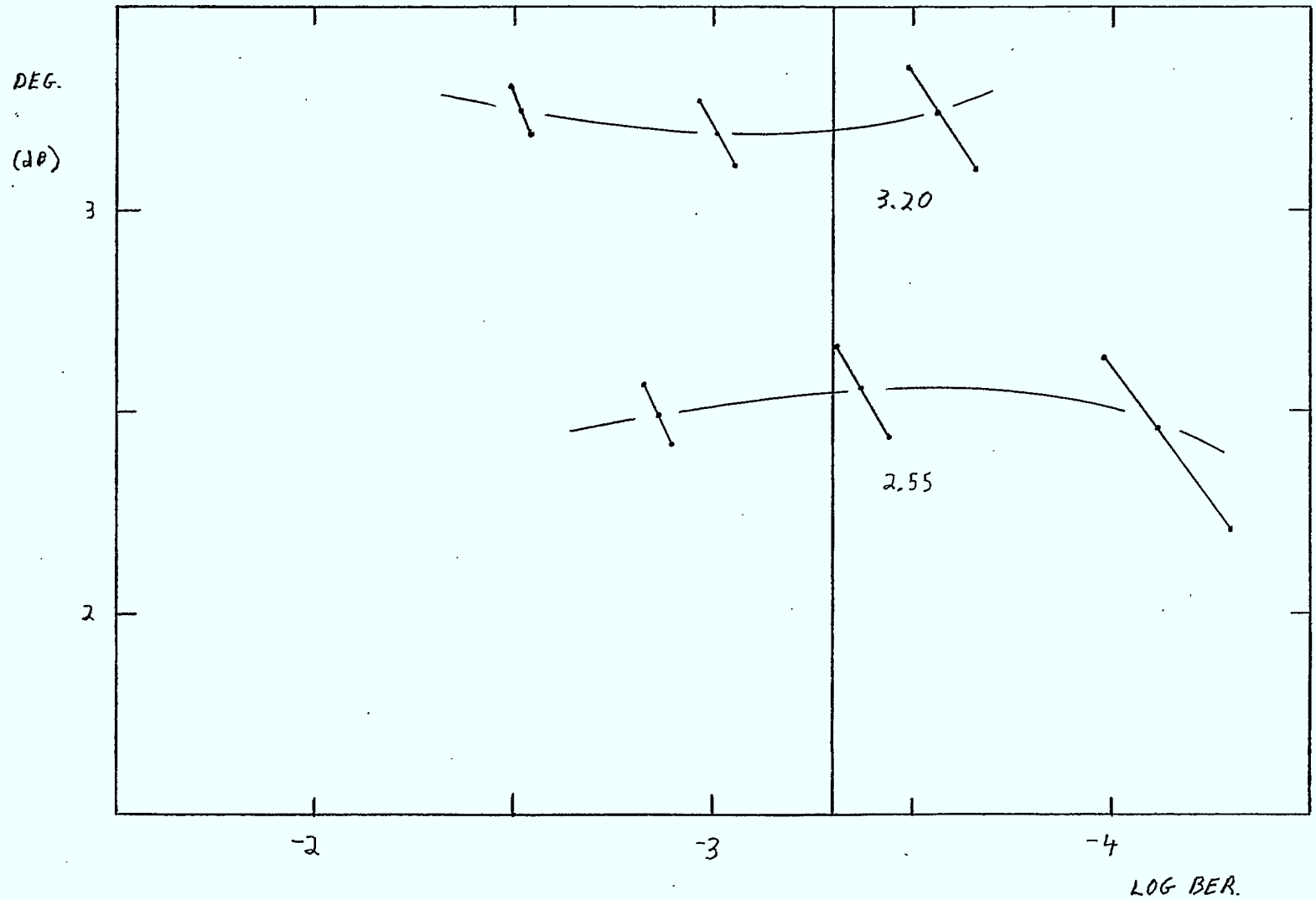

FIGURE B.6-3 Simulated DMSK With:
RX = 4th Order Butterworth Equalized (BT = 1.1)
Group Delay. (A = 0.6)

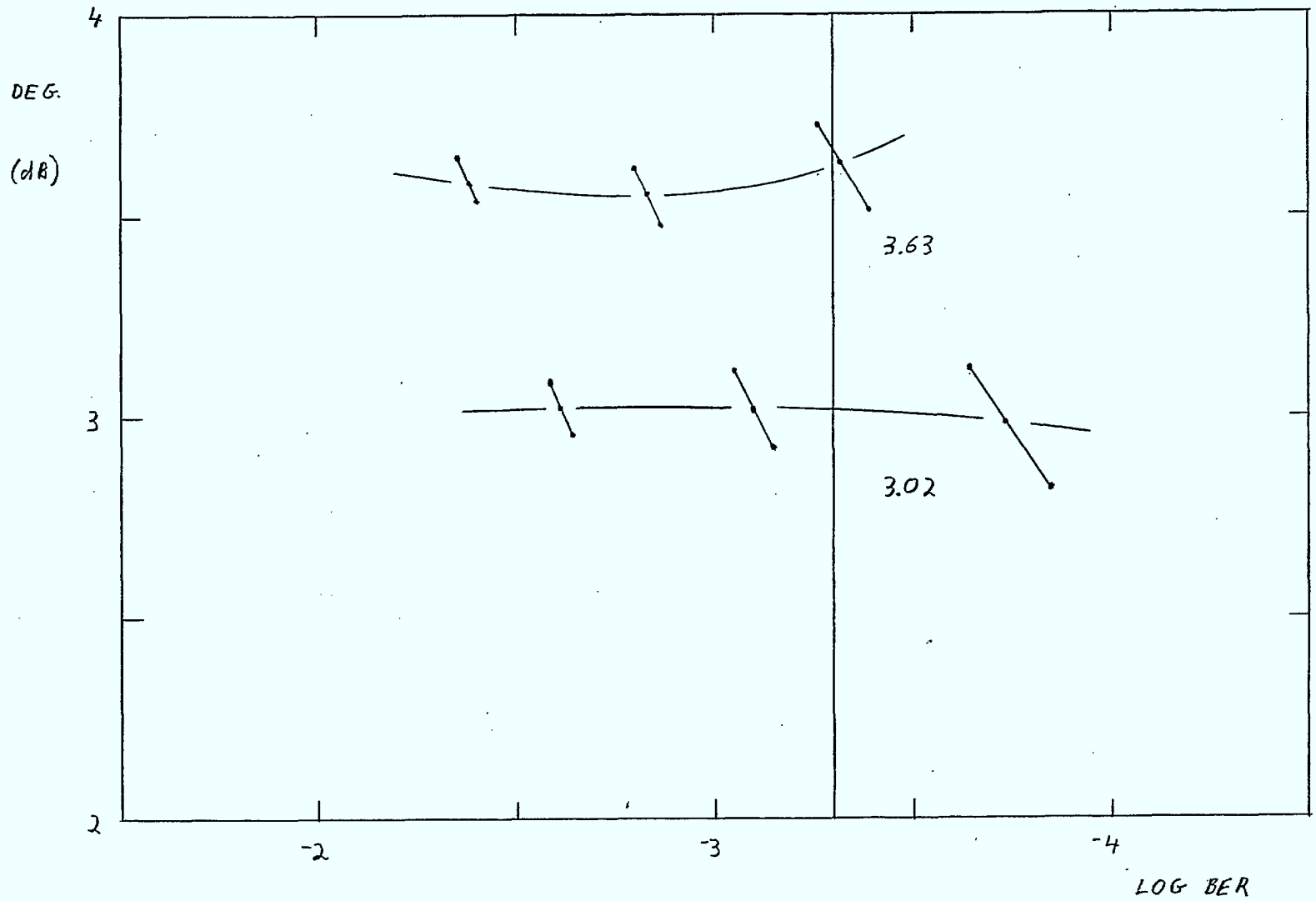

FIGURE B.6-4 Simulated DMSK With:
RX = 4th Order Butterworth Equalized (BT=1.1)
Group Delay (A = 0.8)

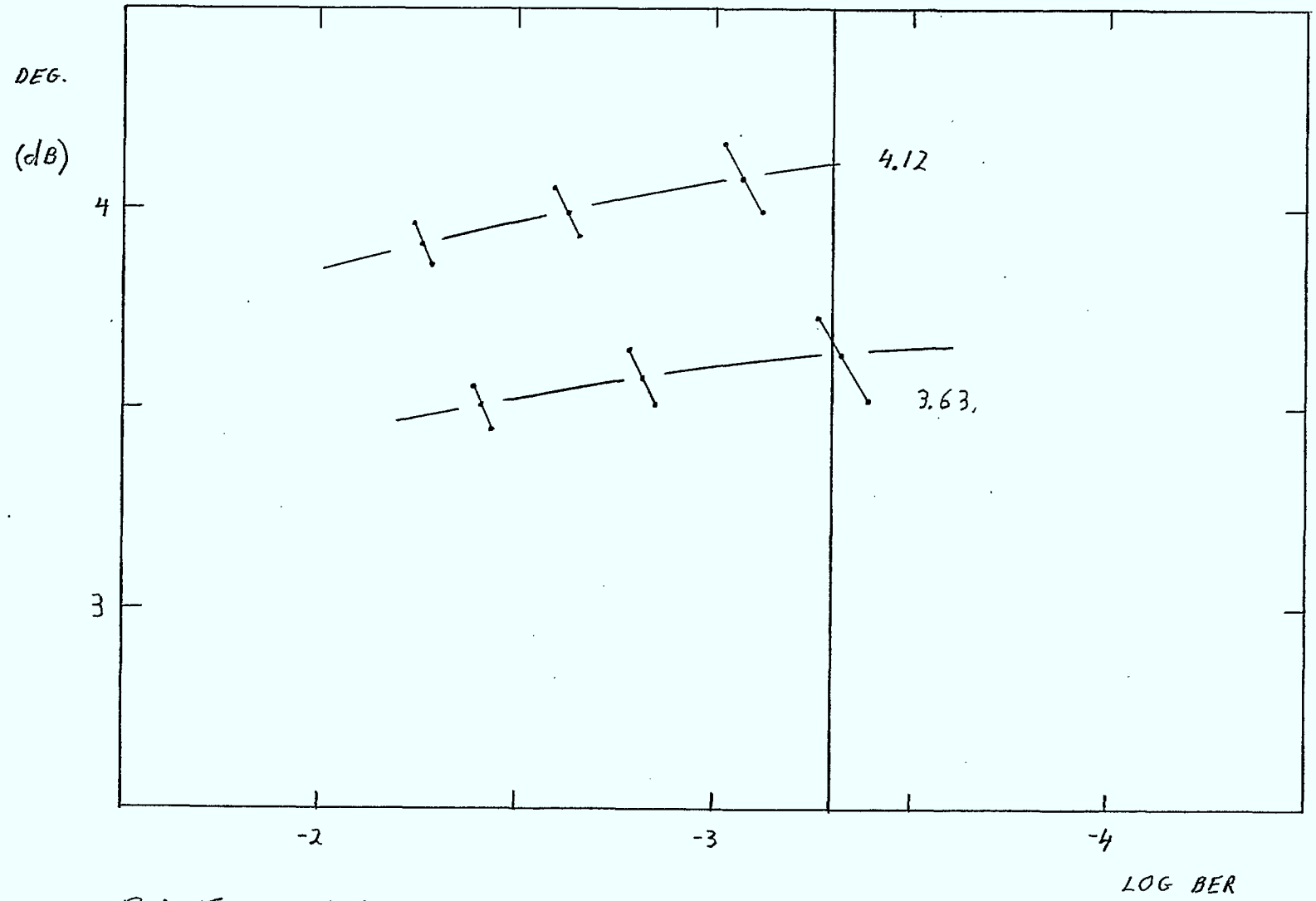

FIGURE B.6-5 Simulated DMSK With:
RX = 4th Order Butterworth Equalized (BT=1.1)
Group Delay (A= 1.0)

APPENDIX C

ANALYTICAL DETAILS OF ISI INVESTIGATION

C.1 <u>rivation of Equation (6.16)</u>

om (6.10),

$$k) = \mathrm{Re}\left[-jr(k)\ r^*(k-1)\right]$$

$$= \mathrm{Re}\left(-j\left[-j\alpha e(k+1)+e(k)+j\alpha e(k-1)\right]u^*(k)\cdot\right.$$

$$\left.\cdot\left[j\alpha e(k)+e(k-1) - j\alpha e(k-2)\right]u(k-1)\right)$$

$$t\ u^*(k)\cdot u(k-1) = u^*(k)(-j)u(k) = -j\left|u(k)\right|^2 = -j$$

us

$$d(k) = \mathrm{Re}\left(\left[j\alpha e(k+1)-e(k)-j\alpha e(k-1)\right]\cdot\right.$$

$$\left.\cdot\left[j\alpha e(k)+e(k-1)-j\alpha e(k-2)\right]\right)$$

$$= -\alpha^2 e(k+1)e(k)+\alpha^2 e(k+1)e(k-2)-e(k)e(k-1)$$

$$+ \alpha^2 e(k-1)e(k) - \alpha^2 e(k-1)e(k-2)$$

$$= a(k)-\alpha^2 a(k)-\alpha^2 a(k-1)a(k)a(k+1)+\alpha^2 a(k-1)$$
$$+\alpha^2 a(k+1)$$

$$= a(k)\left[1-\alpha^2-\alpha^2 a(k-1)a(k+1)\right]+\alpha^2 a(k-1)+\alpha^2 a(k+1) \tag{6.16}$$

C.2 Derivation of Equation (6.17)

From (6.11)

$$d(k) = \mathrm{Re}\left[r(k)\cdot r^*(k-2)\right]$$

$$= \mathrm{Re}\Big(\left[-j\alpha e(k+1)+e(k)+j\alpha e(k-1)\right]u^*(k)\cdot$$

$$\cdot\left[j\alpha e(k-1)+e(k-2)-j\alpha e(k-3)\right]u(k-2)\Big)$$

But $u^*(k)\ u(k-2) = (-j)^2\left|u(k)\right|^2 = -1$

Thus

$$d(k) = \mathrm{Re}\Big(\left[j\alpha e(k+1)-e(k)-j\alpha e(k-1)\right]\cdot$$

$$\left[j\alpha e(k-1)+e(k-2)-j\alpha e(k-3)\right]\Big)$$

$$= -\alpha^2 e(k+1)e(k-1)+\alpha^2 e(k+1)e(k-3)-e(k)e(k-2)$$

$$+\ \alpha^2 e(k-1)e(k-1)\ -\ \alpha^2 e(k-1)e(k-3)$$

$$= p(k)+\alpha^2+\alpha^2 p(k-1)+\alpha^2 p(k+1)+\alpha^2 p(k-1)p(k+1) \tag{6.17}$$

C.3 Derivation of Equations (6.25) and (6.26)

From (6.24), (6.23) and (6.15)

$$w(k) = \mathrm{Re}\big([-j\alpha e(k) - e(k-1)+j\alpha e(k-2)]u(k)n(k)$$

$$+[-\alpha e(k+1)-je(k)+\alpha e(k-1)]u^*(k)n^*(k-1)$$

$$-jn(k)n^*(k-1)] \qquad (*)$$

Aside

$$u(k)n(k) = \underbrace{[u_e(k)n_I(k)-u_o(k)n_Q(k)]}_{A} + j\underbrace{[u_e(k)n_Q(k)+u_o(k)n_I(k)]}_{B}$$

$$u^*(k)n^*(k-1) = \underbrace{[u_e(k)n_I(k-1)-u_o(k)n_Q(k-1)]}_{C}-j\underbrace{[u_e(k)n_Q(k-1) + u_o(k)n_I(k-1)]}_{D}$$

$$n(k)n^*(k-1) = \underbrace{[n_I(k)n_I(k-1)+n_Q(k)n_Q(k-1)]}_{F}+j\underbrace{[n_Q(k)n_I(k-1)-n_I(k)n_Q(k-1)}_{G}$$

From (*)

$$w(k)= \alpha e(k)B - e(k-1)A - \alpha e(k-2)B$$

$$-\alpha e(k+1)C - e(k)D + \alpha e(k-1)C + G$$

Assuming independent Noise samples

$$E[w(k)] = 0 \qquad (6.25)$$

And

$$E[w^2(k)] = \alpha^2 E[B^2]+E[A^2]+\alpha^2 E[B^2]$$

$$+\alpha^2 E[C^2]+E[D^2]+\alpha^2 E[C^2]+E[G^2]$$

$$-\alpha^2 e(k)(k-2)E[B^2]-\alpha^2 e(k+1)e(k-1)E[C^2]$$

as the expected value of all cross terms are zero.

Now $E[A^2] = E[B^2] = E[C^2] = E[D^2] = \sigma^2$

$$E[G^2] = 2\sigma^4$$

Thus $E[w^2(k)] = \alpha^2\sigma^2+\sigma^2+\alpha^2\sigma^2$

$$+\alpha^2\sigma^2+\sigma^2+\alpha^2\sigma^2+2\sigma^4$$

$$-\alpha^2 e(k)e(k-2)\sigma^2-\alpha^2 e(k+1)e(k-1)\sigma^2$$

$$= 2(\sigma^2+\sigma^4)+\alpha^2(4+p(k)+p(k+1))\sigma^2 \qquad (6.26)$$

C.4 Derivation of Equations (6.29) and (6.30)

From (6.28), (6.23) and (6.15)

$$w(k) = \mathrm{Re}\,\big([-j\alpha e(k-1)-e(k-2)+j\alpha e(k-3)]u(k)n(k)$$

$$+ [-j\alpha e(k+1)+e(k)+j\alpha e(k-1)]u^*(k)n^*(k-2)$$

$$+n(k)n^*(k-2)\big) \qquad (*)$$

Aside

$$u(k)n(k) = \underbrace{[u_e(k)n_I(k)-u_o(k)n_Q(k)]}_{A}+j\underbrace{[u_e(k)n_Q(k)+u_o(k)n_I(k)]}_{B}$$

$$u^*(k)n^*(k-2)=\underbrace{[u_e(k)n_I(k-2)-u_o(k)n_Q(k-2)]}_{C}-j\underbrace{[u_e(k)n_Q(k-2)+u_o(k)n_I(k-2)]}_{D}$$

$$n(k)n^*(k-2)=\underbrace{[n_I(k)n_I(k-2)+n_Q(k)n_Q(k-2)]}_{F}+j\underbrace{[n_Q(k)n_I(k-2)-n_I(k)n_Q(k-2)]}_{G}$$

From (*)

$$w(k) = \alpha e(k-1)B-\alpha(k-2)A-\alpha e(k-3)B$$

$$-\alpha e(k+1)D+e(k)C+\alpha e(k-1)D+F$$

Assuming independent noise samples

$$E[w(k)] = 0 \qquad (6.29)$$

And

$$E[w^2(k)] = \alpha^2 E[B^2]+E[A^2]+\alpha^2 E[B^2]$$

$$+\alpha^2 E[D^2]+E[C^2]+\alpha^2 E[D^2]+E[F^2]$$

$$-\alpha^2 e(k-1)e(k-3)E[B^2]-\alpha^2 e(k+1)e(k-1)E[D^2]$$

as the expected value of all cross terms are zero.

Now $E[A^2] = E[B^2] = E[C^2] = E[D^2] = \sigma^2$

$$E[F^2] = 2\sigma^4$$

Thus

$$E[w^2(k)] = \alpha^2\sigma^2+\sigma^2+\alpha^2\sigma^2$$

$$+\alpha^2\sigma^2+\sigma^2+\alpha^2\sigma^2+2\sigma^4$$

$$-\alpha^2 e(k-1)e(k-3)\sigma^2-\alpha^2 e(k+1)e(k-1)\sigma^2$$

$$= 2(\sigma^2+\sigma^4)+\alpha^2(4+p(k-1)+p(k+1))\sigma^2 \qquad (6.30)$$

APPENDIX D

ANALYTICAL DETAILS OF MAXIMUM LIKELIHOOD SYNCHRONIZATION APPROACH

D.1 Derivation of the MSK Preamble Matched Filter

As a first step towards the analysis of performance, we derive the MSK preamble matched filter (PMF) and its corresponding ambiguity function below. In fact, we shall actually do this twice - first from a conventional viewpoint and then from the serial MSK (SMSK) viewpoint. This will confirm that both viewpoints give the same result (as they must), and serves as a means for introducing the two equivalent PMF implementations.

D.1.1 Conventional Viewpoint

The conventional way of looking at MSK is to visualize it as a form of staggered quadriphase modulation. Specifically, if the preamble signal p(t) is modulated with N bits $(a_o, a_1, \ldots, a_{N-1})$ and possesses a bit period of T seconds, then we may write [27]

$$p(t) = \Big[\sum_{k_{even}} a_k(-1)^{k/2} u(t-kT)\Big]\cos(2\pi f_c t)$$

$$+\Big[\sum_{k_{odd}} a_k(-1)^{(k-1)/2} u(t-kT)\Big]\sin(2\pi f_c t)$$

$$= p_I(t)\cos(2\pi f_c t) - p_Q(t)\sin(2\pi f_c t) \qquad \text{(D.1)}$$

where

$$a_k = \pm 1, \quad k=0,1,\ldots,N-1$$

$$u(t) = \sqrt{\frac{E_b}{T}} \cos\left(\frac{\pi t}{2T}\right), \quad |t| \leq T \quad \text{basic MSK symbol having energy } E_b \tag{D.2}$$

In complex baseband notation (denoted with ~), therefore

$$\tilde{p}(t) = p_I(t) + jp_Q(t)$$

$$= u(t) * \left[\sum_{k=0}^{N-1} a_k j^{3k} \delta(t-kT) \right] \tag{D.3}$$

where * denotes convolution and $j=(-1)^{\frac{1}{2}}$. Thus, the impulse response of the conventional PMF becomes

$$\tilde{h}_p^c = K\tilde{p}^*(T_o - t) \tag{D.4}$$

where K is an arbitrary constant and T_o is a large enough delay to ensure causality (i.e. realizability of the filter).

The normalized complex autocorrelation of $\tilde{p}(t)$ is

$$\tilde{R}_p(t) = \frac{T}{NE_b} \tilde{p}(t) * \tilde{p}^*(-t)$$

$$= \sum_{n=-(N-1)}^{N-1} r_n j^{3n} R_u(t-nT) \tag{D.5}$$

where $R_u(t)$ is the normalized autocorrelation of the basic MSK pulse shape

$$R_u(t) = \frac{T}{E_b} u(t)*u(-t)$$

$$= (1-\frac{|t|}{2T}) \cos(\frac{\pi t}{2T}) + \frac{1}{\pi} \sin\frac{\pi|t|}{2T} , \quad |t| \leq 2T \quad \text{(D.6)}$$

and r_n is the normalized autocorrelation of the preamble bits

$$r_n = \frac{1}{N} \sum_{k=0}^{N-1} a_k \, a_{k+n} \quad \text{(D.7)}$$

Thus, the output of a filter matched to $\tilde{p}(t)$ is simply (ignoring noise, K, T_o, and double frequency terms) [28]

$$R_p(t) = \frac{1}{2} \mathrm{Re}\{\tilde{R}_p(t) \exp(j2\pi f_c t)\}$$

$$= \frac{1}{2} \{R_{P_I}(t)\cos(2\pi f_c t)-R_{P_Q}(t)\sin(2\pi f_c t)\} \quad \text{(D.8)}$$

where

$$R_{P_I}(t) = \sum_{\substack{n=-(N-1) \\ n_{even}}}^{N-1} r_n j^n R_u(t-nT) \quad \text{(D.9a)}$$

$$R_{P_Q}(t) = \sum_{\substack{n = -(N-1) \\ n_{odd}}}^{N-1} r_n j^{n+1} R_u(t-nT) \quad \text{(D.9b)}$$

and the envelope of this signal becomes

$$\frac{1}{2}\left\{R^2_{P_I}(t) + R^2_{P_Q}(t)\right\}^{\frac{1}{2}} \tag{D.10}$$

So if the peak of $R_p(t)$ occurs at t_a (the true TOA), and we define the error in estimating t_a as τ, then the ambiguity function along the τ axis is the normalized square of (D.10)

$$\theta(\tau) = R^2_{P_I}(\tau) + R^2_{P_Q}(\tau) \tag{D.11a}$$

or equivalently

$$\theta(\tau) = \tilde{R}_p(\tau)\,\tilde{R}^*_p(\tau)$$

$$= \sum_{k=-(N-1)}^{N-1} \sum_{\ell=-(N-1)}^{N-1} r_k r_\ell j^{3(k-\ell)} Q_{k\ell}(\tau) \tag{D.11b}$$

where

$$Q_{k\ell}(\tau) = R_u(\tau-kT)\,R_u(\tau-\ell T) \tag{D.12}$$

Later we shall be interested in the second derivative of $\theta(\tau)$ at $\tau=0$. Taking this opportunity to evaluate it, we may write

$$\left.\frac{\partial^2}{\partial\tau^2}\theta(\tau)\right|_{\tau=0} = \sum_{k=-(N-1)}^{N-1} \sum_{\ell=-(N-1)}^{N-1} r_k r_\ell j^{3(k-\ell)} \left.\frac{\partial^2}{\partial\tau^2} Q_{k\ell}(\tau)\right|_{\tau=0} \tag{D.13}$$

We find

$$\frac{\partial^2}{\partial \tau^2} Q_{k\ell}(\tau)\Big|_{\tau=0} = \begin{cases} \dfrac{-\pi^2}{2T^2} & \text{for } (k,\ell) = (0,0) \\[2ex] \dfrac{1}{2T^2}(1+\pi^2/4) & \text{for } (k,\ell)=(-1,-1) \text{ and } (1,1) \\[2ex] \dfrac{1}{2T^2}(1-\pi^2/4) & \text{for } (k,\ell)=(-1,1) \text{ and } (1,-1) \\[2ex] 0 & \text{for all other } k \text{ and } \ell \end{cases} \tag{D.14}$$

and so

$$\frac{\partial^2}{\partial \tau^2} \theta(\tau)\Big|_{\tau=0} = \frac{-\pi^2}{2T^2}(1-r_1^2) \tag{D.15}$$

where r_1 is the normalized first sidelobe of the preamble bit pattern autocorrelation.

D.1.2 Serial MSK Viewpoint

The conventional viewpoint adopts the modulation centre frequency as the carrier frequency. However, this is rather arbitrary since no carrier tone is actually transmitted in MSK - any reasonable frequency will serve equally well for a reference. In particular, Amoroso and Kivett [29] have shown that adopting the mark or space frequency as a reference leads to a simplified MSK implementation (the so-called serial MSK, SMSK).

Let the MSK mark and space frequencies be, respectively

$$f_m = \frac{q}{2T} \tag{D.16a}$$

$$f_s = \frac{q+1}{2T} \tag{D.16b}$$

where q is an integer. Then Amoroso and Kivett indicate that the basic data pulse for general n is (shifting the centre of the pulse to t = 0),

$$b(t) = \frac{T}{\pi}\left(\frac{2q+2}{2q+1}\right) \cos\left(\frac{\pi t}{2T}\right)\cdot\cos(2\pi f_c t + \phi), \quad |t| \leq T \tag{D.17}$$

where ϕ is an arbitrary carrier phase and

$$f_c = \frac{(2q+1)}{4T} \tag{D.18}$$

is the nominal carrier frequency.

Now, for the case $\phi = 0$, Amoroso and Kivett have demonstrated that MSK can be generated and received with the scheme shown in Figure D.1*, where (shifting the centre to t=0)

$$\begin{aligned} h(t) &= \cos(\pi t/T)\cos(2\pi f_m t + q\pi/2) \\ &\quad - \sin(\pi t/T)\sin(2\pi f_m t + q\pi/2) \\ &= \mathrm{Re}\{2\tilde{h}(t)\exp[j(2\pi f_m t + q\pi/2)], \quad |t| \leq T/2 \end{aligned} \tag{D.19}$$

and

$$\tilde{h}(t) = \frac{1}{2}\{\cos(\pi t/T) + j\sin(\pi t/T)\}, \quad |t| \leq T/2 \tag{D.20}$$

*For $\phi \neq 0$ the scheme generates a signal identical to MSK except that its envelope is not strictly constant. However, for $q \gg 1$ the envelope fluctuations become negligibly small and true MSK results.

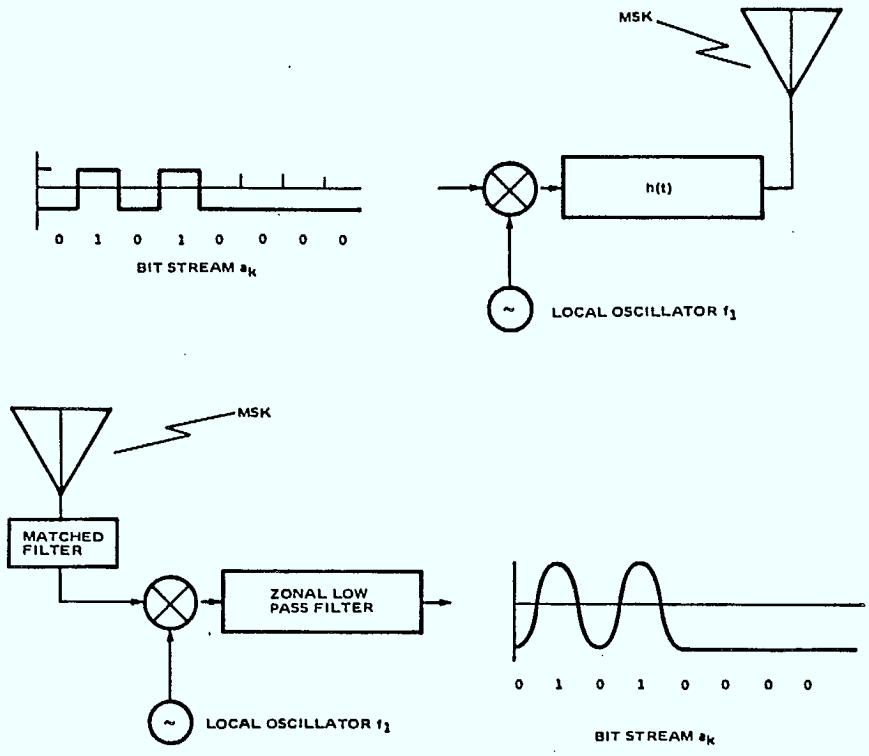

Figure D.1 Transmission and reception scheme [13]

From the SMSK viewpoint, therefore, the complex baseband preamble signal may be written as

$$\tilde{p}(t) = d(t)*g(t)*\tilde{h}(t) \tag{D.21}$$

where (ignoring multiplicative energy scaling)

$$d(t) = \sum_{k=0}^{N-1} a_k \delta(t-kT) \text{ is the preamble bit stream} \tag{D.22}$$

and

$$g(t) = \begin{cases} 1 \ , & |t| \leq T/2 \\ 0 \ , & \text{elsewhere} \end{cases} \text{ converts } d(t) \text{ to NRZ data} \tag{D.23}$$

Thus, the impulse response of the serial MSK PMF becomes

$$\tilde{h}_p^s = K\tilde{p}^*(T_o-t) \tag{D.24}$$

which is the same as before except that the "carrier" is now taken to be f_m instead of f_c.

Computing the normalized complex autocorrelation of $\tilde{p}(t)$ we obtain

$$\tilde{R}_p(t) = R_d(t)*[R_u(t)\{\cos(\pi t/2T) + j\sin(\pi t/2T)\}] \tag{D.25}$$

where $R_u(t)$ was given in (D.6) and

$$R_d(t) = \frac{1}{N}\, d(t)*d(-t)$$

$$= \sum_{n=-(N-1)}^{N-1} r_n \delta(t-nT) \qquad (D.26)$$

with r_n given in (D.7). So, the output of a filter matched to $\tilde{p}(t)$ is simply (ignoring noise, K, T_o, and double frequency terms) [28]

$$R_p(t) = \frac{1}{2}\, \mathrm{Re}\{\tilde{R}_p(t)\ \exp(j2\pi f_m t)\}$$

$$= \frac{1}{2}\{[R_{p_I}(t)\cos(\frac{\pi t}{2T}) - R_{p_Q}(t)\sin(\frac{\pi t}{2T})]\cos(2\pi f_m t)$$

$$- [R_{p_I}(t)\ \sin(\frac{\pi t}{2T}) + R_{p_Q}(t)\cos(\frac{\pi t}{2T})]\sin(2\pi f_m t)\}$$

$$= \frac{1}{2}\{R_{p_I}(t)\ \cos(2\pi f_c t) - R_{p_Q}(t)\sin(2\pi f_c t)\} \qquad (D.27)$$

which is exactly what was obtained in (D.8). Thus the equivalance of the two implementations is confirmed.

D.2 Time of Arrival Accuracy

Using results from the previous section we now estimate the TOA accuracy (i.e. timing jitter) assuming an ideal implementation and additive white Gaussian noise. For our estimate we shall follow usual radar analysis and employ the Cramer-Rao lower bound. This bound is fairly simple to compute, is known to be reasonably tight, and can serve as a useful benchmark for more exact calculations or simulation/experiment. For this preliminary study we shall assume there is no doppler or receiver AGC.

From Van Trees [20], the Cramer-Rao lower bound for TOA accuracy is

$$\sigma_\tau^2 \geq \left[\frac{2E}{N_o}\left(\frac{E}{E+N_o}\right)\right]^{-1}\left[\frac{\alpha^2}{\beta^2\alpha^2-\gamma^2}\right] \qquad (D.28)$$

where σ_τ^2 is the variance of the TOA error τ and

$$\beta^2 = \frac{1}{E}\int_{-\infty}^{\infty} (2\pi f)^2 \left|\tilde{P}(f)\right|^2 df \qquad \text{effective (Gabor) bandwidth of preamble} \qquad (D.29)$$

$$\alpha^2 = \frac{1}{E}\int_{-\infty}^{\infty} t^2 \left|p(t)\right|^2 dt \qquad \text{effective duration of preamble} \qquad (D.30)$$

$$\gamma^2 = \mathrm{Im}\left\{\frac{1}{E}\int_{-\infty}^{\infty} t\tilde{p}(t)\,\frac{\partial\tilde{p}^*(t)}{\partial t}\,dt\right\} \qquad (D.31)$$

Now, from (D.3)

$$\tilde{p}(t) = u(t)*\left[\sum_{k=0}^{N-1} a_k j^{3k}\ \delta(t-kT)\right]$$

$$=\sqrt{\frac{E_b}{T}}\sum_{k=0}^{N-1} a_k j^{3k}\ \cos\left[\frac{\pi}{2T}\ (t-kT)\right] \qquad (D.32)$$

So

$$\frac{\partial\tilde{p}^*(t)}{\partial t} = \frac{-2T}{\pi}\ \sqrt{\frac{E_b}{T}}\ \sum_{k=0}^{N-1} a_k j^{-3k}\sin\left[\frac{\pi}{2T}(t-kT)\right] \qquad (D.33)$$

Thus

$$\gamma^2 = \frac{-2E_b}{\pi}\ \mathrm{Im}\left\{\frac{1}{E}\int_{-\infty}^{\infty}\sum_{k=0}^{N-1}\sum_{\ell=0}^{N-1} a_k a_\ell j^{3(k-\ell)t}\right.$$

$$\left.\cos\left[\frac{\pi}{2T}\ (t-kT)\right]\sin\left[\frac{\pi}{2T}\ (t-\ell T)\right]\ dt\right\}$$

$$= \frac{-2}{N\pi}\ \mathrm{Im}\left\{\sum_{k=0}^{N-1}\sum_{\ell=0}^{N-1} a_k a_\ell j^{3(k-\ell)}\Gamma_{k\ell}\right\} \qquad (D.34)$$

where

$$\Gamma_{k\ell} = \int_{-\infty}^{\infty} t \cos[\tfrac{\pi}{2T}(t-kT)]\cdot\sin[\tfrac{\pi}{2T}(t-\ell T)]\, dt$$

$$= \frac{1}{2}\int_{-\infty}^{\infty} t \sin[\tfrac{\pi}{2}(k-\ell)]\, dt$$

$$= 0 \text{ for all } k \text{ and } \ell \qquad \text{(D.35)}$$

Hence, for the MSK preamble $\gamma^2=0$ and (D.28) simplifies to

$$\sigma_\tau^2 \geq \left[\frac{2E}{N_o}\left(\frac{E}{E+N_o}\right)\right]^{-1}\left[\frac{1}{\beta^2}\right] \qquad \text{(D.36)}$$

where $E = NE_b$.

Next, Van Trees indicates that

$$\beta^2 - \mu^2 = -\frac{1}{2}\frac{\partial^2}{\partial\tau^2}\theta(\tau,\omega)\Big|_{\substack{\tau=0\\ \omega=0}} \qquad \text{(D.37)}$$

where $\theta(\tau,\omega)$ is the ambiguity function on the TOA error/ doppler offset plane, and

$$\mu = \frac{1}{E}\int_{-\infty}^{\infty} (2\pi f)\left|\tilde{P}(f)\right|^2 df \qquad \text{(D.38)}$$

But, from (D.3)

$$\tilde{P}(f) = U(f)\left[\sum_{k=0}^{N-1} a_k j^{3k} e^{-j2\pi kfT}\right]$$

$$= U(f) \sum_{k=0}^{N-1} \exp\{-j[2\pi kfT - 3k\pi/2 - (a_k-1)\pi/2]\} \tag{D.39}$$

So

$$|\tilde{P}(f)|^2 = |U(f)|^2 [N+4\cdot \sum_{k=0}^{N-1} \sum_{\ell=k+1}^{N-1} \cos[2\pi(k-\ell)fT]$$

$$\cdot\cos[3(k-\ell)\pi/2 - (a_k-a_\ell)\pi/2]] \tag{D.40}$$

where [27]

$$|U(f)|^2 = \frac{16E_b}{\pi^2}\left[\frac{\cos 2\pi fT}{1-16f^2T^2}\right]^2 \tag{D.41}$$

Since (D.40) is symmetric about f=0 therefore $\mu=0$. So, from (D.37), we may rewrite (D.36) as

$$\sigma_\tau^2 \geqslant \left[\frac{E}{N_o}\left(\frac{E}{E+N_o}\right)\right]^{-1}\left[\frac{-\partial^2}{\partial\tau^2}\,\theta(\tau,\omega)\Big|_{\substack{\tau=0\\ \omega=0}}\right]^{-1} \tag{D.42}$$

and employing (D.15), this becomes

$$\left(\frac{\sigma_\tau^2}{T^2}\right) \geqslant \left[\frac{E}{N_o}\left(\frac{E}{E+N_o}\right)\right]^{-1}\left[\frac{\pi^2}{2}(1-r_1^2)\right]^{-1} \tag{D.43}$$

D.3 Probability of Not Correctly Acquiring

Up to this point we have assumed that the thresholding scheme has properly detected the main lobe of the ambiguity function , i.e. that the BTR circuit has correctly acquired timing (with some small timing jitter). Not correctly acquiring can take place if

(i) the threshold level λ is exceeded prematurely, i.e. the circuit suffers a false alarm, or

(ii) if there is no false alarm but the signal correlation peak is missed.

Specifically, let us define the probability of false alarm P_{FA} to be the probability that the time at which the threshold λ is crossed from below t_{λ_1}, occurs T seconds or more earlier than the true TOA t_a. That is

$$P_{FA} = \text{Prob}\left[t_{\lambda_1} \leq t_a - T\right] \tag{D.44}$$

Let us also define the probability of detection P_D to be the probability that the matched filter output squared envelope exceeds the threshold λ at some time within T seconds of the true TOA. That is

$$P_D = \text{Prob}\left[\left|t_{\lambda_1} - t_a\right| \leq T\right] \tag{D.45}$$

The probability of not correctly acquiring (P_{NCA}) thus becomes

$$\begin{aligned} P_{NCA} &= P_{FA} + (1-P_D)(1-P_{FA}) \\ &\approx P_{FA} + (1-P_D) \end{aligned} \tag{D.46}$$

where we shall sometimes refer to $P_M = 1-P_D$ as the probability of miss.

In this section we shall attempt to estimate P_{NCA} as a function of λ, E_b/N_o, and the preamble code. First we shall estimate the false alarm rate R_{FA}, and using an approximation compute P_{FA}. Then P_D will be calculated. Finally, P_{FA} and $(1-P_D)$ will be combined using (D.46) to give P_{NCA}.

D.3.1 False Alarm Rate

Since the BTR circuit employs a continuous threshold comparison, the relevant statistic of interest is the mean false alarm rate R_{FA}. DiFranco and Rubin [30] have computed R_{FA} in the case of a linear envelope detector. We extend their analysis below to the case of the square-law envelope detector used in the maximum likelihood BTR circuit.

A false alarm will occur whenever noise or noise plus ambiguity function sidelobes exceed a threshold value λ. For the purpose of this analysis,let us restrict ourselves to the mathematical model of Figure D.2. Here

$\tilde{p}(t)$ = MSK preamble having energy $E=NE_b$

$\tilde{n}(t)$ = AWGN with two-sided power spectral density $N_o/2$

$\tilde{R}_p(t)$ = preamble autocorrelation function

$\tilde{n}_1(t)$ = bandlimited, zero-mean Gaussian noise with variance σ_1^2.

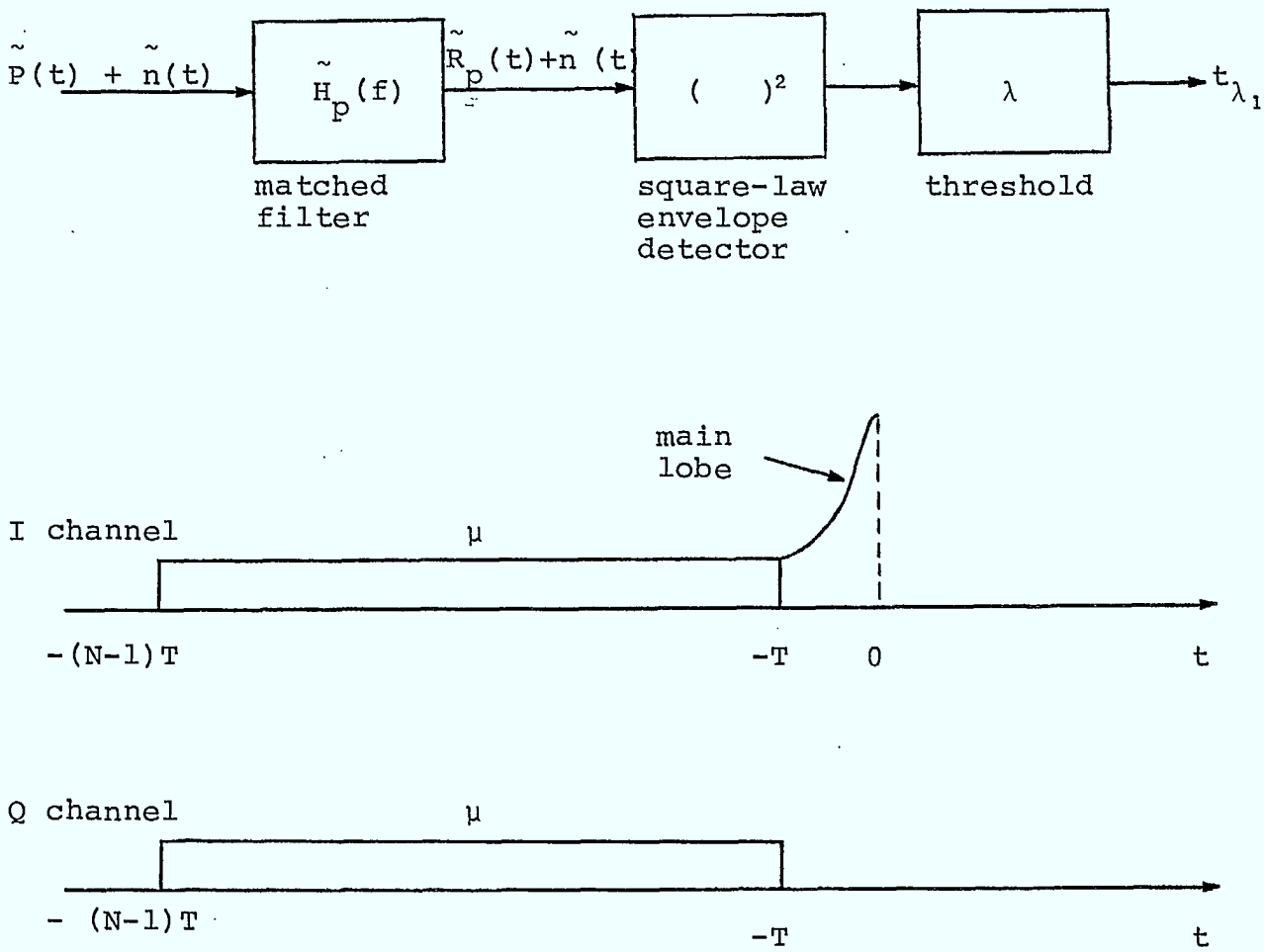

Figure D.2 Mathematical Model for Computation of False Alarm Rate

At this point we now make a couple of simplifying assumptions. First of all we recognize that non-zero sidelobes increase the likelihood of false alarms. Thus, a preamble code with low sidelobes is desirable. Barker codes are the optimum in this respect since the sidelobe magnitude never exceeds 1/N of the main lobe for a N-bit code. We will assume that Barker codes are used, and further approximate the in-phase and quadrature sidelobe level as a constant μ for $-(N-1)T \leq t \leq -T$. These simplifications are illustrated in Figure D.2.

Now consider the matched filter output during $(-[N-1]T,-T)$. Let us combine the noise $\tilde{n}_1(t)$ and constant sidelobes into a function $z(t)$:

$$z(t) = [x(t)+\mu]\cos \omega_c t - [y(t)+\mu]\sin \omega_c t \qquad (D.47)$$

$$= \nu_1(t)\cos \omega_c t - \nu_2(t)\sin \omega_c t$$

where $x(t)$ and $y(t)$ are quadrature noise components with

$$\text{Mean } [x(t)] = \text{Mean } [y(t)] = 0$$

$$\text{Mean } [\nu_1(t)] = \text{Mean } [\nu_2(t)] = \mu$$

$$\text{Var } [x(t)] = \text{Var } [y(t)] = \text{Var}[\nu_1(t)] = \text{Var } [\nu_2(t)]$$
$$= R_{xx}(0) = \sigma_1^2$$

The squared envelope $V(t)$ of $z(t)$ is the quantity of interest

$$V(t) = \nu_1^2(t)+\nu_2^2(t)$$

$$= x^2(t)+y^2(t) + {}^2\mu[x(t)+y(t)] + 2\mu^2 \qquad (D.48)$$

Whenever $V(t)$ exceeds λ from below (positive slope) a false alarm is declared. We hope to evaluate R_{FA} using the formula due to Rice [31]:

$$R_{FA} = \int_0^\infty \dot{V} p(\lambda, \dot{V}) d\dot{V} \tag{D.49}$$

where $\dot{V}(t) = dV(t)/dt$ and $p(\lambda, \dot{V})$ is the joint pdf of λ and $\dot{V}(t)$.

Now, the random waveforms $\dot{x}(t)$ and $\dot{y}(t)$, where $\dot{x}(t) = dx(t)/dt$ and $\dot{y}(t) = dy(t)/dt$, can be shown to be stationary Gaussian processes with zero mean and variance given by

$$\begin{aligned} \mathrm{Var}[\dot{x}(t)] &= \mathrm{Var}\,[\dot{y}(t)] = \mathrm{Var}[\dot{\nu}_1(t)] = \mathrm{Var}\,[\dot{\nu}_2(t)] \\ &= |\ddot{R}_{xx}(0)| \\ &= \sigma_2^2 \end{aligned} \tag{D.50}$$

Further, it can be shown that $x(t)$, $\dot{x}(t)$, $y(t)$, and $\dot{y}(t)$ (and hence also $\nu_1(t)$, $\dot{\nu}_1(t)$, $\nu_2(t)$, $\dot{\nu}_2(t)$) are uncorrelated at the same instant of time. We thus find

$$p(x, y, \dot{x}, \dot{y}) = \frac{1}{(2\pi)^2 \sigma_1^2 \sigma_2^2} \exp\left[-\frac{x^2+y^2}{2\sigma_1^2} - \frac{\dot{x}^2+\dot{y}^2}{2\sigma_2^2}\right] \tag{D.51}$$

$$p(\nu_1, \nu_2, \dot{\nu}_1, \dot{\nu}_2) = \frac{1}{(2\pi)^2 \sigma_1^2\ \sigma_2^2} \exp\left[-\frac{(\nu_1-\mu)^2+(\nu_2-\mu)^2}{2\sigma_1^2} - \frac{\dot{\nu}_1^2+\dot{\nu}_2^2}{2\sigma_2^2}\right] \tag{D.52}$$

Let us now transform to polar coordinates $A, \dot{A}, \theta, \dot{\theta}$ as follows:

$$\nu_1 = A \cos \theta \tag{D.53}$$

$$\nu_2 = A \sin \theta \tag{D.54}$$

$$\dot{\nu}_1 = \dot{A} \cos \theta + A \dot{\theta} \sin \theta \tag{D.55}$$

$$\dot{\nu}_2 = \dot{A} \sin \theta + A \dot{\theta} \cos \theta \tag{D.56}$$

and so

$$p(A,\dot{A},\theta,\dot{\theta}) = \frac{A^2}{(2\pi)^2 \sigma_1^2 \sigma_2^2} \exp\left[- \frac{A^2 - 2A\mu(\cos\theta + \sin\theta) + 2\mu^2}{2\sigma_1^2} - \frac{\dot{A}^2 + A^2\dot{\theta}^2}{2\sigma_2^2} \right] \tag{D.57}$$

The marginal pdf $p(A,\dot{A})$ is obtained by integrating the above with respect to θ over $(-\pi,\pi)$ and with respect to $\dot{\theta}$ over $(-\infty, \infty)$. We find

$$P(A,\dot{A}) = \frac{A\, I_o(\sqrt{2}\, A\mu/\sigma_1^2)}{\sqrt{2\pi\sigma_2^2}\, \sigma_1^2} \exp\left[- \frac{A^2 + 2\mu^2}{2\sigma_1^2} - \frac{\dot{A}^2}{2\sigma_2^2} \right] \tag{D.58}$$

where $I_o(\cdot)$ is the zeroth-order modified Bessel function.

But $V = A^2$ and $\dot{V} = 2A\dot{A}$. Hence the inverse mapping is $A = \sqrt{V}$, $\dot{A} = \dot{V}/(2\sqrt{V})$, and the Jacobian of the transformation becomes

$$J = \begin{vmatrix} \frac{\partial}{\partial V} \sqrt{V} & \frac{\partial}{\partial \dot{V}} \sqrt{V} \\ \frac{\partial}{\partial V} \left(\frac{\dot{V}}{2\sqrt{V}}\right) & \frac{\partial}{\partial \dot{V}} \left(\frac{\dot{V}}{2\sqrt{V}}\right) \end{vmatrix} = \frac{1}{4V} \tag{D.59}$$

So, transforming $p(A,\dot{A})$ to $p(V,\dot{V})$,

$$p(V,\dot{V}) = \frac{\left(\frac{1}{4V}\right)\sqrt{V}\; I_o\left(\frac{\sqrt{2V}\,\mu}{\sigma_1^2}\right)}{\sqrt{2\pi\sigma_2^2}\;\;\sigma_1^2} \exp\left[-\frac{V+2\mu^2}{2\sigma_1^2} - \frac{\dot{V}^2}{8V\sigma_2^2}\right] \tag{D.60}$$

and then substituting in (D.49),

$$R_{FA} = \frac{I_o(\sqrt{2\lambda}\;\mu/\sigma_1^2)}{4\sigma_1^2\;\sqrt{2\pi\lambda\sigma_2^2}} \exp\left[-\frac{\lambda+2\mu^2}{2\sigma_1^2}\right] \int_o^\infty \dot{V} \exp\left[\frac{-\dot{V}^2}{8\lambda\sigma_2^2}\right] d\dot{V}$$

$$= \frac{1}{\sigma_1^2}\left[\frac{\lambda\sigma_2^2}{2\pi}\right]^{\frac{1}{2}} \cdot I_o\left[\frac{\sqrt{2\lambda}\;\mu}{\sigma_1^2}\right] \cdot \exp\left[-\frac{\lambda+2\mu^2}{2\sigma_1^2}\right] \tag{D.61}$$

Our final task is to relate quantities in the above expression to known parameters. First,

$$\sigma_1^2 = \mathrm{Var}\left[n(t)\right] = \frac{N_o}{2}\int_{-\infty}^{\infty} |H_p(f)|^2\, df = \frac{N_o}{2} \tag{D.62}$$

were we define for convenience

$$\int_{-\infty}^{\infty} |H_p(f)|^2 df = 1 \tag{D.63}$$

Next,

$$\sigma_2^2 = \left|\ddot{R}_{xx}(o)\right|$$

$$= \left| \frac{\partial^2}{\partial t^2} \int_{-\infty}^{\infty} S_{xx}(f)\ e^{+j2\pi ft} df \right|_{t=0}$$

$$= \frac{N_o}{2} \int_{-\infty}^{\infty} (2\pi f)^2\ \left|H(f)\right|^2 df \qquad \text{(D.64)}$$

since $S_{xx}(f) = (N_o/2)\ \left|H_p(f)\right|^2$. But $H(f)$ is matched to $P(f)$ and the "energy" of $H_p(f)$ is unity from (D.63), thus from (D.29), (D.37), and (D.15)

$$\sigma_2^2 = \frac{N_o}{2}\ \beta^2$$

$$= \frac{N_o}{2} \left[-\tfrac{1}{2}\ \frac{\partial^2}{\partial \tau^2}\ \theta(\tau,\omega)\right]_{\substack{\tau=0\\ \omega=0}}$$

$$= \frac{N_o}{2} \left[\frac{-1}{2}\right]\left[\frac{-\pi^2}{2T^2}\ (1-r_1^2)\right]$$

$$= \frac{N_o \pi^2 (1-r_1^2)}{8T^2} \qquad \text{(D.65)}$$

Lastly, since H(f) is a matched filter

$$\left|\frac{\tilde{R}_p(0)}{\sigma_1^2}\right|^2 = \frac{2E}{N_o} \tag{D.66}$$

Thus, $\tilde{R}_p(0) = \sqrt{E}$ and for Barker codes $\mu = \sqrt{E/N}$ where N is the number of preamble bits. After the square-law device, however, the autocorrelation peak becomes E. Let us express λ as a positive fraction γ of this peak (i.e. $\lambda=\gamma E$). Then (D.61) becomes

$$R_{FA}=\frac{1}{2T}[\gamma N\pi(1-r_1^2)(E_b/N_o)]^{\frac{1}{2}}\cdot I_o[2\sqrt{2\gamma}\ (E_b/N_o)]\cdot\exp[-(\gamma N+2/N)(E_b/N_o)] \tag{D.67}$$

for $-(N-1)\ T \leqslant t \leqslant -T$

On the other hand, for $t \leqslant -(N-1)T$, when there are no sidelobes (i.e. $\mu=0$), then (D.61) is simply

$$R_{FA} = \frac{1}{\sigma_1^2}\left[\frac{\lambda\sigma_2^2}{2\pi}\right]^{\frac{1}{2}} \cdot \exp\left[\frac{-\lambda}{2\sigma_1^2}\right] \quad \text{for } t \leqslant -(N-1)T$$

$$= \frac{1}{2T}\left[\gamma N\pi(1-r_1^2)(E_b/N_o)\right]^{\frac{1}{2}} \cdot \exp\left[-\gamma N\ E_b/N_o\right] \tag{D.68}$$

Together, (D.67) and (D.68) approximately characterize the mean false alarm rate with noise+sidelobes and noise only, respectively.

D.3.2 Probability of False Alarm

Unfortunately, a false alarm rate by itself is not too

informative since the probability of false alarm then depends upon the amount of time the BTR circuit is active (the synchronization window - see Figure 7.5). To circumvent this dilemma, consider the ratio of the false alarm rates corresponding to noise+sidelobes and noise only:

$$\varepsilon = \frac{R_{FA}(\text{noise+sidelobes})}{R_{FA}(\text{noise})}$$

Let's choose some fairly conservative numbers: $\gamma=\frac{1}{2}$, $N=4$, $E_b/N_o = 10$. Then we find that $\varepsilon = I_o(20)\cdot e^{-5} \approx 3\text{x}10^5$.

Thus, the overall false alarm rate is very strongly dominated by false alarms occuring when the sidelobes are present. To a very good approximation we may therefore write

$$P_{FA} \approx (N-2)\ T \cdot R_{FA}\ (\text{noise+sidelobes})$$

$$= \frac{(N-2)}{2}\left[\gamma N\pi(1-r_1^2)(E_b/N_o)\right]^{\frac{1}{2}} \cdot I_o\left[2\sqrt{2\gamma}\ (E_b/N_o)\right]$$

$$\cdot \exp\left[-(\gamma N+2/N)(E_b/N_o)\right]$$

$$\approx \frac{(N-2)\sqrt{N(1-r_1^2)}}{4}\left(\frac{\gamma}{2}\right)^{\frac{1}{4}} \exp\left[2\sqrt{2\gamma} - \gamma N - 2/N)(E_b/N_o)\right] \tag{D.70}$$

where we have employed the approximation

$$I_o(x) \approx \frac{e^x}{\sqrt{2\pi x}} \qquad \text{for large } x \tag{D.71}$$

D.3.3 Probability of Detection

We have seen that the rate at which a threshold λ is crossed (from below) by the preamble matched filter squared envelope is very strongly dependent upon the autocorrelation level μ. We shall now make use of this observation a second time to compute the probability of detection P_D. Specifically, since the ambiguity function (i.e. squared magnitude of autocorrelation) falls off rapidly away from $\theta(0,0)$, we shall assume the probability that is λ is crossed in the vicinity of $\theta(0,0)$ is very closely approximated by the probability that the squared envelope exceeds λ at $\theta(0,0)$. That is, we shall replace the continuous time problem by the classical sampled time problem.

Recall that the equation (D.60) gives the joint pdf $p(V,\dot{V})$. Integrating over all $\dot{V}$, we may obtain the marginal pdf

$$p(V) = \frac{I_o(\sqrt{2V}\,\mu/\sigma^2)}{2\sigma_1^2} \exp\left[-\frac{V+2\mu^2}{2\sigma_1^2}\right] \qquad (D.72)$$

Hence, the probability of detection becomes

$$P_D = \int_{\lambda}^{\infty} p(V)dV$$

$$= Q\,(\sqrt{2}\mu/\sigma_1,\ \sqrt{\lambda/}\sigma_1) \qquad (D.73)$$

where

$$Q(\alpha,\beta) \overset{\Delta}{=} \int_{\beta}^{\infty} z \exp\left(-\frac{z^2+\alpha^2}{2}\right) I_o\,(\alpha z)dz \qquad (D.74)$$

is commonly called Marcum's Q function [32]. Relating (D.73) to known parameters we obtain (for the correlation peak)

$$P_D = Q(a, b) \tag{D.75}$$

where

$$a = 2\sqrt{NE_b/N_o} \tag{D.76}$$

$$b = \sqrt{2N\gamma E_b/N_o} \tag{D.77}$$

D.3.4 Computation of P_{NCA}

Recall that the probability of miss is $P_M = 1-P_D$. Using equations (A-3-1) and (A-3-6) of [32], we may express P_M in the computationally tractable from

$$P_M = Q(b,a) - \exp\left(-\frac{a^2+b^2}{2}\right)I_o(ab)$$

$$\approx \left\{\operatorname{erfc}\left(\frac{a-b}{\sqrt{2}}\right) + \frac{\exp[-(a-b)^2/2]}{\sqrt{2\pi ab}}\right\} - \exp\left[-(a^2+b^2)/2\right] I_o(ab) \tag{D.78}$$

The approximation is valid providing $a \gg 1$, $b \gg 1$ and $a \gg a-b > 0$, i.e.

$$N(E_b/N_o) \gg \tfrac{1}{4} \tag{D.79}$$

$$N\gamma(E_b/N_o) \gg \tfrac{1}{2} \tag{D.80}$$

and

$$2 \gg (2-\sqrt{2\gamma}) \geqslant 0 \tag{D.81}$$

The probability of not correctly acquiring follows from combining (D.70) with (D.78) according to (D.46).

APPENDIX E

PROGRAM DOCUMENTATION

This appendix contains an abbreviated compilation of software documentation for the DMSK modem. All routines have been written in the structured Fortran language FLECS, and are executable on the CRC Sigma 9 and 6 computers. Section E.1 briefly describes the program usage and Section E.2 contains an abbreviated set of FLECS listings.

E.1 Usage

Table E.1 contains a brief description of each of the main control routines. Only the listings of MSK1 and MSK32 are given in Section E.2, as all other control routines represent very minor modifications (or subsets) of these two main control routines. Table E.2 contains a brief description of all supporting subroutines, listing the subroutines by file. Table E.3 contains the list of channel designations used by the routines to interface with external devices or files. Table E.4 lists and describes the file name extensions use consistantly throughout.

An example printout from the MSK32 control routine is shown in Figure E.1.

MAIN CONTROL ROUTINES

Base Name	Description
MSK 1	- MSK modulated sequence - WGN added in time domain - Optimum CMSK demodulator Routines Required: PNSQNS, PULSHP, MSKMOD, CWGN1, OPTDEM
MSK 3	- MSK modulated sequence (differentially encoded) - WGN added in time domain - Rx filtering - Conventional DMSK and SEC Routines Required: PNSQNS, ENCODE, PULSHP, MSKMOD, CWGN1, FFT, REMOV, FILTER, MULT, CINTER, DEMOD, HARD, BERCNT, RXRES, SEC
MSK 32	MSK3 plus: - Delay error, E5, which imitates timing errors E3 and E4 (more efficient) - Hard limiters option - Adaptive thresholding option - Multipath option - Jamming option - Non-constant group delay option Additional Routines Required: HARDLIM, HARD(DEM2 file), MLTPTH, JAMMING, GRPDLY, DEGDB
MSK 33	MSK 32 plus: - RMS zero crossing jitter computed Additional Routines Required: RMSJITTE

TABLE E.1

MAIN CONTROL ROUTINES

Base Name	Description
MSK 4	MSK 3 plus: - Tx filtering - DEM filtering Additional Routines Required: TXRES, DEMRES
MSK 5	MSK 4 plus: - capability to handle multiple bit timing errors E3 and E4. (more efficient)
MSK 6	MSK 3 plus: - plots demodulated baseband signal Additional Routines Required: PLOTSIG
MSK 62	MSK 6 for an isolated pulse Additional Routines Required: MODTST instead of MOD,
MSK 7	MSK 4 plus: - Plots Eye Pattern Additional Routines Required: MSKEDG

Table E.1 (continued)

ROUTINE NAME KEY

File	Routine	Routine Description
CWGN1	CWGN1	- simulates a Complex Baseband Gaussian Noise Generator.
DEGDB	DEGDB	- computes the degradation of the BER from Ideal Antipodal Signalling
DEM	DEMOD	- Performs demodulation between the signal and a delayed version of itself.
	HARD	- Performs hard decisions on the signal
	HARDLIM	- Performs hard-limiter function on the signal.
	SEC	- Attempts to correct the conventional DMSK data stream using a SEC logic circuit.
	BERCNT	- Counts the discrepancies between the detected bit stream and the data source.
DEM2	HARD	- As above but with adaptive thresholding.
FILRES	TXRES	- Sets up complex frequency response of transmit filter.
	RXRES	- As TXRES but for receive filter.
	DEMRES	- As TXRES but for demod. filter.
	MLTPTH	- Adds multipath interference to signal.
	JAMMING	- Adds jamming interference to signal.
FRQ	FFT	- Calls FAST4 routine from Sigma 9 Math Library to compute the Fast Fourier Transform and its inverse.
	REMOV	- Adjusts the complex samples of the frequency response of a filter.

Table E.2

ROUTINE NAME KEY

File	Routine	Routine Description
	FILTER	- Multiplies the signal samples in the frequency domain by the frequency response of a filter.
	MULT	- Multiplies the signal samples by a constant.
	GRPDLY	- introduces a non-constant group delay.
	CINTER	- computes a sample by using linear interpolation between two samples.
MOD	PNSQNS	- Generates a PN sequence
	ENCODE	- Differentially encodes a binary input sequence.
	PULSHP	- Generates the baseband pulse shape used for modulation.
	MSKMOD	- Generates MSK signal samples.
MSKEDG	MSKEDG	- Generates an 'eyeplot' from a given set of signal samples.
OPTDEM	OPTDEM	- Optimally demodulates the received MSK signal.
PLTSG	PLOTSIG	- Plots the signal samples down the page.
RMS	RMSJITTE	- Finds the RMS error on the zero-crossing points.

Table E.2 (continued)

PROGRAM CHANNEL ASSIGNMENTS

Channel #	Description
3	= FLECS input source file
4	= FLECS FORTRAN output file
5	= FLECS errors output file
6	= FLECS listing output file
7	= user's console or results file
10	= input parameter file
11	= Tx filter response file
12	= Rx filter response file
13	= DEM filter response file

Table E.3

FILE NAME EXTENSION KEY

Name Extension	Description
-DF	= Data File
-DFL	= Data File Parameters Listing
-ER	= flecs output ERror file
-FLX	= flecs input source file
-FO	= FOrtran file
-L	= flecs Listing
-RO	= Relocatable Object module
-XFLX	= eXecute flecs command file
X	= eXecute LYNX command file

Note: Not all name extensions are valid for each of the base names. As an example the "X" extension only applies to main control routines, and not subroutine files.

Table E.4

```
INPUT PARAMETERS
  NUMBER OF BITS PER RUN =                    512
  PULSE SHAPE(1=SIN,2=RECT) =                   1
  NUMBER OF SAMPLES PER BIT=                    4
  BIT RATE (MBPS)=                       5.000000
  CARRIER FREQUENCY (MHZ) =              .000000
  SIGNAL DELAY ERROR 1 (BITS) =          .000000
  SIGNAL DELAY ERROR 2 (BITS) =          .000000
  SIGNAL DELAY ERROR 3 (BITS) =          .000000
  SIGNAL PHASE SHIFT ERROR(DEG.) =           .00
 CONVENTIONAL
  POS. THRESHOLD LEVEL =                 .000000
  NEG. THRESHOLD LEVEL =                 .000000
 SINGLE ERROR CORRECTION
  POS. THRESHOLD LEVEL =                 .000000
  NEG. THRESHOLD LEVEL =                 .000000
  NOISE SEED                                1357
  EB OVER NO IN DB =                        6.00
  NUMBER OF RUNS =                           200
  HARD LIMITERS IN USE:     1  2  3  4
    0-OFF, 1-ON.            0  0  0  0
  MULTIPATH IN USE:  NO
  JAMMING IN USE:    NO


RECEIVE FILTER PARAMETERS
  STARTING FREQUENCY (MHZ) =                -3.7500
  FREQUENCY SPACING OF RESPONSE SAMPLES=      .2500
  NUMBER OF POINTS =                             31


SIMULATION RESULTS
  TOTAL NUMBER OF BITS =                     102400
  CONVENTIONAL DMSK
    NUMBER OF BIT ERRORS =                     3535
    BIT ERROR RATE(BER) =                  .034521
    APPROX. BER RMS ERROR =                .000571
    LOG BER + RMS ERROR    =                -1.452
    DEGRADATION (DB)       =                 3.852
    LOG BER                =                -1.462
    DEGRADATION (DB)       =                 3.820
    LOG BER - RMS ERROR    =                -1.469
    DEGRADATION (DB)       =                 3.789
  WITH SINGLE ERROR CORRECTION(SEC)
    NUMBER OF BIT ERRORS =                     2167
    BIT ERROR RATE(BER) =                  .021162
    APPROX. BER RMS ERROR =                .000455
    LOG BER + RMS ERROR    =                -1.665
    DEGRADATION (DB)       =                 2.696
    LOG BER                =                -1.674
    DEGRADATION (DB)       =                 2.652
    LOG BER - RMS ERROR    =                -1.684
    DEGRADATION (DB)       =                 2.620
  PREDICTED SINGLE BIT ERRORS
    NUMBER OF BIT ERRORS =                     3176
    BIT ERROR RATE(BER) =                  .031016
    APPROX. BER RMS ERROR =                .000542
    LOG BER + RMS ERROR    =                -1.501
    LOG BER                =                -1.508
    LOG BER - RMS ERROR    =                -1.516
  SYNDROME ERRORS
    NUMBER OF BIT ERRORS =                     8279
    BIT ERROR RATE(BER) =                  .080850
    APPROX. BER RMS ERROR =                .000852
    LOG BER + RMS ERROR    =                -1.088
    LOG BER                =                -1.092
    LOG BER - RMS ERROR    =                -1.097

!
```

FIGURE E.1 Example Program Printout (MSK32)

E.2 Routine Listings

```
!
C MSK1-L
00001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00002 C
00003 C       PROGRAM: MSK1
00004 C       FUNCTION: SIMULATES AN MSK MODEM
00005 C       PROGRAMMER: STEWART CROZIER
00006 C       DATE: DEC 7/81
00007 C
00008 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00009 C
00010 C TRANSMIT DATA
00011        INTEGER NB,A(512)
00012 C
00013 C PULSE SHAPE
00014        INTEGER ISHAPE,NPT
00015        REAL PULSE(32)
00016 C
00017 C SIGNAL
00018        INTEGER NSPB,NS
00019        COMPLEX SIG(4096),SIG1(4096)
00020 C
00021 C NOISE
00022        INTEGER NSEED
00023        REAL SIGMA,EBNODB,EBNO,EB
00024        COMPLEX CWGN
00025 C
00026 C DETECTED DATA
00027        INTEGER AD(512)
00028 C
00029 C ERRORS
00030        INTEGER NER,TNB,NRUN
00031        REAL BER, RMSERR
00032 CCCCCCCCCCCCCCCCCC
00033 C
00034 C   MAIN PROGRAM
00035 C
00036 CCCCCCCCCCCCCCCCCC
00037        INITIALIZE-PARAMETERS
00038        GENERATE-PN-SEQUENCE
00039        GENERATE-MSK-SIGNAL
00040        COMPUTE-SIGMA-FOR-NOISE
00041        DO (IRUN=1,NRUN)
00042        .  ADD-NOISE
00043        .  PERFORM-OPTIMUM-DEMODULATION
00044        .  COUNT-ERRORS
00045        ...FIN
00046        COMPUTE-ERROR-RATES
00047        PRINT-ERROR-RATES
00048        STOP
```

```
00049        TO INITIALIZE-PARAMETERS
00050        .  GET-INPUT-PARAMETERS
```

```
00051        .  PRINT-INPUT-PARAMETERS
00052        .  NPT=4*NSPB
00053        .  NS=NB*NSPB
00054        .  NER=0
00055        ...FIN
```

```
00056        TO PRINT-INPUT-PARAMETERS
00057        .  WRITE(7,10)
00058        .  WRITE(7,20)NB
00059        .  WRITE(7,30)ISHAPE
00060        .  WRITE(7,40)NSPB
00061        .  WRITE(7,50)NSEED
00062        .  WRITE(7,60)EBNODB
00063        .  WRITE(7,70)NRUN
00064 10     .  FORMAT(///'INPUT PARAMETERS')
00065 20     .  FORMAT('  NUMBER OF BITS PER RUN =          ',I9)
00066 30     .  FORMAT('  PULSE SHAPE(1=SIN,2=RECT) =       ',I9)
00067 40     .  FORMAT('  NUMBER OF SAMPLES PER BIT=        ',I9)
00068 50     .  FORMAT('  NOISE SEED                        ',I9)
00069 60     .  FORMAT('  EB OVER NO IN DB =                ',F9.2)
00070 70     .  FORMAT('  NUMBER OF RUNS =                  ',I9)
00071        ...FIN
```

```
00072        TO GET-INPUT-PARAMETERS
00073        .  INPUT(10)NB
00074        .  INPUT(10)ISHAPE
00075        .  INPUT(10)NSPB
00076        .  INPUT(10)NSEED
00077        .  INPUT(10)EBNODB
00078        .  INPUT(10)NRUN
00079        ...FIN
```

```
00080        TO GENERATE-PN-SEQUENCE
00081        .  CALL PNSQNS(A,NB,9,5,9)
00082        ...FIN
```

```
00083        TO GENERATE-MSK-SIGNAL
00084        .  CALL PULSHP(NPT,ISHAPE,PULSE)
00085        .  CALL MSKMOD(A,NB,PULSE,NPT,NS,NSPB,SIG)
00086        ...FIN
```

```
00087        TO COMPUTE-SIGMA-FOR-NOISE
00088 C      .
00089 COMPUTE EB
00090        .  EB=0
```

```
00091        .  DO (I=1,NSPB*2)
00092        .  .  EB=EB+PULSE(I)**2
00093        .  ...FIN
00094 C      .
00095 COMPUTE EB OVER NO (SINGLE SIDED)
00096        .  EBNO=10**(EBNODB/10)
00097 C      .
00098 COMPUTE SIGMA
00099        .  SIGMA=SQRT(EB/EBNO/2)
00100        ...FIN
```

```
00101        TO ADD-NOISE
00102        .  DO (I=1,NS)
00103        .  .  CALL CWGN1(NSEED,SIGMA,CWGN)
00104        .  .  SIG1(I)=SIG(I)+CWGN
00105        .  ...FIN
00106        ...FIN
```

```
00107        TO PERFORM-OPTIMUM-DEMODULATION
00108        .  CALL OPTDEM(SIG1,NS,PULSE,NPT,NSPB,NB,AD)
00109        ...FIN
```

```
00110        TO COUNT-ERRORS
00111        .  DO (I=1,NB)
00112        .  .  IF (A(I).NE.AD(I)) NER=NER+1
00113        .  ...FIN
00114        ...FIN
```

```
00115        TO COMPUTE-ERROR-RATES
00116        .  TNB=NRUN*NB
00117        .  BER=FLOAT(NER)/TNB
00118        .  RMSERR=SQRT(BER*(1-BER)/TNB)
00119        ...FIN
```

```
00120        TO PRINT-ERROR-RATES
00121        .  WRITE(7,90)
00122        .  WRITE(7,100)TNB
00123        .  WRITE(7,110)NER
00124        .  WRITE(7,120)BER
00125        .  WRITE(7,130)RMSERR
00126 90     .  FORMAT(///'SIMULATION RESULTS')
00127 100    .  FORMAT('  TOTAL NUMBER OF BITS =            ',I9)
00128 110    .  FORMAT('  NUMBER OF BIT ERRORS =            ',I9)
00129 120    .  FORMAT('  BIT ERROR RATE (BER) =            ',F9.6)
00130 130    .  FORMAT('  APROX. BER RMS ERROR =            ',F9.6)
```

```
00131        ...FIN
00132        END
```

PROCEDURE CROSS-REFERENCE TABLE

```
00101 ADD-NOISE
      .00042

00115 COMPUTE-ERROR-RATES
      00046

00087 COMPUTE-SIGMA-FOR-NOISE
      00040

00110 COUNT-ERRORS
      00044

00083 GENERATE-MSK-SIGNAL
      00039

00080 GENERATE-PN-SEQUENCE
      00038

00072 GET-INPUT-PARAMETERS
      00050

00049 INITIALIZE-PARAMETERS
      00037

00107 PERFORM-OPTIMUM-DEMODULATION
      00043

00120 PRINT-ERROR-RATES
      00047

00056 PRINT-INPUT-PARAMETERS
      00051
```

TRANSPORT FLECS MAY78

1

!

```
!C MSK32-L
00001 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00002 C
00003 C       PROGRAM: MSK32
00004 C       FUNCTION: SIMULATES A DMSK MODEM WITH SEC
00005 C       PROGRAMMER: STEWART CROZIER
00006 C       DATE: DEC 17/81
00007 C
00008 CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
00009 C
00010 C TRANSMIT DATA
00011       INTEGER NB,A(512),B(512)
00012 C
00013 C PULSE SHAPE
00014       INTEGER ISHAPE,NPT
00015       REAL PULSE(32)
00016 C
00017 C SIGNAL
00018       INTEGER NSPB,NS,CHECK(5)
00019       REAL RATE,DFS,AMP,T,PHASE,E1,E2,E5,PIBY2,PHER,FC,GRPDLYA
00020       REAL THC1,THC2,THP1,THP2
00021       REAL THALPHA,THMLT
00022       REAL DEL1,DEL2,DEL3,DEL4,DEL5
00023       COMPLEX SIG(2048),SIG1(2048),SIG2(2048),SIG3(2048)
00024 C
00025 C
00026       INTEGER MLTPTHA(2)/' NO ',' YES'/
00027       INTEGER JAMCHCKA(2)/' NO ',' YES'/
00028 C
00029 C NOISE
00030       INTEGER NSEED
00031       REAL SIGMA,EBNODB,EBNO,EB
00032       COMPLEX CWGN
00033 C
00034 C FFTS
00035       INTEGER NSR2,ISIGN
00036       REAL W(1024)
00037 C
00038 C FILTERS
00039       INTEGER PRFLAG1,NPFIL,NPRX
00040       REAL SFRX,DFRX
00041       COMPLEX FILRX(101)
00042       DATA NPFIL/101/
00043 C
00044 C DETECTED DATA
00045       INTEGER AD(512),CD(512)
00046 C
00047 C ERRORS
00048       INTEGER TNB,NRUN,PRFLAG2
00049       INTEGER NERA,TNERA
00050       INTEGER NERC,TNERC
00051       INTEGER NERP,TNERP
00052       INTEGER NERS,TNERS
00053       INTEGER NSTEP
```

```
C
      REAL BERA,RMSERA,LUPA,LAVA,LOWA
      REAL BERC,RMSERC,LUPC,LAVC,LOWC
      REAL BERP,RMSERP,LUPP,LAVP,LOWP
      REAL BERS,RMSERS,LUPS,LAVS,LOWS
      REAL START,STEP
      REAL DBDEGUPA,DBDEGAVA,DBDEGOWA
      REAL DBDEGUPC,DBDEGAVC,DBDEGOWC
C
CCCCCCCCCCCCCCCCCC
C
C   MAIN PROGRAM
C
CCCCCCCCCCCCCCCCCC
      PERFORM-INITIALIZATION
      GENERATE-PN-SEQUENCE
      DIFF-ENCODE-PN-SEQUENCE
      GENERATE-MSK-SIGNAL
      COMPUTE-SIGMA-FOR-NOISE
      DO (IRUN=1,NRUN)
      .  GET-SIG
      .  USE-MULTIPATH
      .  ADD-NOISE
      .  JAM-SIGNAL
      .  PUT-SIGNAL-THROUGH-RX-FILTER
      .  GENERATE-CONVENTIONAL-DMSK-DECISIONS
      .  GENERATE-SEC-DECISIONS
      .  COUNT-ERRORS
      ...FIN
      COMPUTE-ERROR-RATES
      PRINT-ERROR-RATES
      STOP
```

```
      TO PERFORM-INITIALIZATION
      .  GET-INPUT-PARAMETERS
      .  PRINT-INPUT-PARAMETERS
C     .
C PULSE SHAPE
      .  NPT=4*NSPB
C     .
C SIGNAL
      .  NS=NB*NSPB
      .  DFS=RATE/NB
      .  PIBY2=3.141592654/2
      .  PHER=PHER/57.29578
      .  T=1/RATE
C     .
C FFTS
      .  NSR2=ALOG(NS)/ALOG(2)+0.5
      .  ISIGN=0
      .  CALL FFT(NSR2,NS,SIG1,W,ISIGN)
C     .
C FILTERS
```

```
00106        .  CALL RXRES(SFRX,DFRX,NPRX,FILRX,NPFIL,PRFLAG1)
00107        .  CALL REMOV(FILRX,SFRX,DFRX,NPRX,0.0)
00108 C      .
00109 C ERRORS
00110        .  TNERA=TNERC=TNERP=TNERS=0
00111        ...FIN
```

```
00112        TO PRINT-INPUT-PARAMETERS
00113        .  WRITE(7,5)
00114        .  WRITE(7,10)NB
00115        .  WRITE(7,15)ISHAPE
00116        .  WRITE(7,20)NSPB
00117        .  WRITE(7,25)RATE
00118        .  WRITE(7,30)FC
00119        .  WRITE(7,35)E1
00120        .  WRITE(7,40)E2
00121        .  WRITE(7,42)E5
00122        .  WRITE(7,43)GRPDLYA
00123        .  WRITE(7,45)PHER
00124        .  WRITE(7,46)
00125        .  WRITE(7,47)THC1,THC2
00126        .  WRITE(7,48)
00127        .  WRITE(7,47)THP1,THP2
00128        .  WRITE(7,50)NSEED
00129        .  WRITE(7,60)EBNODB
00130        .  WRITE(7,70)NRUN
00131        .  WRITE(7,71)
00132        .  WRITE(7,72)(CHECK(I),I=1,4)
00133        .  WRITE(7,400)MLTPTHA(MLTCHCK+1)
00134        .  IF(MLTCHCK.NE.0)
00135        .  .  WRITE(7,410)MLTDLY,CVIDB
00136        .  ...FIN
00137        .  CONTINUE
00138        .  WRITE(7,401)JAMCHCKA(JAMCHCK+1)
00139        .  IF(JAMCHCK.NE.0)
00140        .  .  WRITE(7,411)CVJDB
00141        .  ...FIN
00142 5      .  FORMAT(//'INPUT PARAMETERS')
00143 10     .  FORMAT('  NUMBER OF BITS PER RUN =          ',I9)
00144 15     .  FORMAT('  PULSE SHAPE(1=SIN,2=RECT) =       ',I9)
00145 20     .  FORMAT('  NUMBER OF SAMPLES PER BIT=        ',I9)
00146 25     .  FORMAT('  BIT RATE (MBPS)=                  ',F9.6)
00147 30     .  FORMAT('  CARRIER FREQUENCY (MHZ) =         ',F9.6)
00148 35     .  FORMAT('  SIGNAL DELAY ERROR 1 (BITS) =     ',F9.6)
00149 40     .  FORMAT('  SIGNAL DELAY ERROR 2 (BITS) =     ',F9.6)
00150 42     .  FORMAT('  SIGNAL DELAY ERROR 5 (BITS) =     ',F9.6)
00151 43     .  FORMAT('  GROUP DELAY PARAMETER A=          ',F9.6)
00152 45     .  FORMAT('  SIGNAL PHASE SHIFT ERROR(DEG.) = ',F9.2)
00153 46     .  FORMAT('  CONVENTIONAL ')
00154 47     .  FORMAT('    POS. THRESHOLD LEVEL =          ',F9.6,/T3,
00155       1.  '  NEG. THRESHOLD LEVEL =          ',F9.6)
00156 48     .  FORMAT('  SINGLE ERROR CORRECTION ')
00157 50     .  FORMAT('  NOISE SEED                        ',I9)
```

```
00158 60    . FORMAT('  EB OVER NO IN DB =                ',F9.2)
00159 70    . FORMAT('  NUMBER OF RUNS =                  ',I9)
00160 71    . FORMAT('  HARD LIMITERS IN USE:    1  2  3  4')
00161 72    . FORMAT('    0-OFF, 1-ON',10X,4I3)
00162 400   . FORMAT('  MULTIPATH IN USE: ',20X,R4)
00163 401   . FORMAT('  JAMMING IN USE:   ',20X,R4)
00164 410   . FORMAT('  MULTIPATH SIGNAL DELAY =',I18,/T4,'ATTENUATION IN DB =',
00165       1. F22.2)
00166 411   . FORMAT('  JAMMING ATTENUATION IN DB = ',F14.2)
00167       ...FIN
```

```
00168       TO GET-INPUT-PARAMETERS
00169       . NB=512
00170       . ISHAPE=1
00171       . NSPB=4
00172       . INPUT(10)RATE
00173       . INPUT(10)FC
00174       . INPUT(10)E1,E2,E5,GRPDLYA
00175       . INPUT(10)(CHECK(I),I=1,4)
00176       . INPUT(10)MLTCHCK,MLTDLY,CVIDB
00177       . INPUT(10)JAMCHCK,CVJDB
00178       . INPUT(10)PHER
00179       . INPUT(10)THALPHA,THMLT
00180       . INPUT(10)NSEED
00181       . INPUT(10)EBNODB
00182       . INPUT(10)NRUN
00183       . INPUT(10)PRFLAG1,PRFLAG2
00184       . THC1=THMLT*THALPHA**2
00185       . THC2=-THC1
00186       . THP1=2.0*THALPHA**2
00187       . THP2=0.0
00188       ...FIN
```

```
00189       TO GENERATE-PN-SEQUENCE
00190       . CALL PNSQNS(A,NB,9,5,9)
00191       ...FIN
```

```
00192       TO DIFF-ENCODE-PN-SEQUENCE
00193       . CALL ENCODE(A,NB,B)
00194       ...FIN
```

```
00195       TO GENERATE-MSK-SIGNAL
00196       . CALL PULSHP(NPT,ISHAPE,PULSE)
00197       . CALL MSKMOD(B,NB,PULSE,NPT,NS,NSPB,SIG)
00198       ...FIN
```

```
00199        TO COMPUTE-SIGMA-FOR-NOISE
00200 C      .
00201 COMPUTE EB
00202        .  EB=0
00203        .  DO (I=1,NS)
00204        .  .  EB=EB+CABS(SIG(I))**2
00205        .  ...FIN
00206        .  EB=EB/NB
00207 C      .
00208 COMPUTE EB OVER NO (SINGLE SIDED)
00209        .  EBNO=10**(EBNODB/10)
00210 C      .
00211 COMPUTE SIGMA
00212        .  SIGMA=SQRT(EB/EBNO/2)
00213        ...FIN
```

```
00214        TO GET-SIG
00215        .  DO (I=1,NS)SIG1(I)=SIG(I)
00216        ...FIN
```

```
00217        TO USE-MULTIPATH
00218        .  IF(MLTCHCK.NE.0)
00219        .  .  CALL MLTPTH(SIG1,NS,MLTDLY,CVIDB,IRUN,NRUN)
00220        .  ...FIN
00221        ...FIN
```

```
00222        TO ADD-NOISE
00223        .  DO (I=1,NS)
00224        .  .  CALL CWGN1(NSEED,SIGMA,CWGN)
00225        .  .  SIG1(I)=SIG1(I)+CWGN
00226        .  ...FIN
00227        ...FIN
```

```
00228        TO JAM-SIGNAL
00229        .  IF(JAMCHCK.NE.0)
00230        .  .  CALL JAMMING(SIG1,NS,CVJDB,IRUN,NRUN)
00231        .  ...FIN
00232        ...FIN
```

```
00233        TO PUT-SIGNAL-THROUGH-RX-FILTER
00234        .  CALL HARDLIM(SIG1,CHECK(1),NS)
00235        .  ISIGN=-1
00236        .  CALL FFT(NSR2,NS,SIG1,W,ISIGN)
00237        .  CALL FILTER (SIG1,DFS,0.0,NS,FILRX,SFRX,DFRX,NPRX)
```

```
C     .
      ...FIN
```

```
      TO GENERATE-CONVENTIONAL-DMSK-DECISIONS
      . DO (I=1,NS) SIG3(I)=SIG2(I)=SIG1(I)
C     .
      . AMP=1.0
      . DEL1=T*(1+E1)
      . PHASE=0.0
      . CALL GRPDLY(SIG1,DFS,NS,AMP,DEL1,PHASE,T,GRPDLYA)
      . DEL2=T*E5
      . CALL MULT(SIG2,DFS,NS,AMP,-DEL2,PHASE)
C     .
      . ISIGN=1
      . CALL FFT(NSR2,NS,SIG1,W,ISIGN)
      . CALL HARDLIM(SIG1,CHECK(2),NS)
C     .
      . CALL FFT(NSR2,NS,SIG2,W,ISIGN)
      . CALL HARDLIM(SIG2,CHECK(3),NS)
C     .
      . PHASE=PIBY2+PHER
      . DEL4=DEL1-DEL2
      . CALL DEMOD(SIG2,SIG1,NS,FC,DEL4,PHASE)
C     .
      . CALL HARD(SIG1,NS,NB,AD,THC1,THC2)
      ...FIN
```

```
      TO GENERATE-SEC-DECISIONS
      . AMP=1.0
      . DEL3=T*(2+E2)
      . PHASE=0.0
      . CALL GRPDLY(SIG3,DFS,NS,AMP,DEL3,PHASE,T,2*GRPDLYA)
C     .
      . ISIGN=1
      . CALL FFT(NSR2,NS,SIG3,W,ISIGN)
      . CALL HARDLIM(SIG3,CHECK(4),NS)
C     .
      . DEL5=DEL3-DEL2
      . CALL DEMOD(SIG2,SIG3,NS,FC,DEL5,PHASE)
C     .
      . CALL HARD(SIG3,NS,NB,CD,THP1,THP2)
C     .
      . CALL SEC (AD,CD,NB,NERP,NERS)
      ...FIN
```

```
      TO COUNT-ERRORS
      . CALL BERCNT(A,AD,CD,NB,IRUN,NERA,NERC,PRFLAG2)
      . TNERA=TNERA+NERA
      . TNERC=TNERC+NERC
```

```
00284        .  TNERP=TNERP+NERP
00285        .  TNERS=TNERS+NERS
00286        ...FIN

      ----------------------------------------

00287        TO COMPUTE-ERROR-RATES
00288        .  TNB=NRUN*NB
00289        .  ALMOST0=1.0E-30
00290        .  START=10.0
00291        .  STEP=8.0
00292        .  NSTEP=11
00293 C      .
00294        .  BERA=FLOAT(TNERA)/TNB
00295        .  RMSERA=SQRT(BERA*(1-BERA)/TNB)
00296        .  LUPA=ALOG10(BERA+RMSERA+ALMOST0)
00297        .  LAVA=ALOG10(BERA+ALMOST0)
00298        .  LOWA=ALOG10(BERA-RMSERA+ALMOST0)
00299        .  CALL DEGDB(EBNODB,LUPA,START,STEP,NSTEP,DBDEGUPA)
00300        .  CALL DEGDB(EBNODB,LAVA,START,STEP,NSTEP,DBDEGAVA)
00301        .  CALL DEGDB(EBNODB,LOWA,START,STEP,NSTEP,DBDEGOWA)
00302 C      .
00303        .  BERC=FLOAT(TNERC)/TNB
00304        .  RMSERC=SQRT(BERC*(1-BERC)/TNB)
00305        .  LUPC=ALOG10(BERC+RMSERC+ALMOST0)
00306        .  LAVC=ALOG10(BERC+ALMOST0)
00307        .  LOWC=ALOG10(BERC-RMSERC+ALMOST0)
00308        .  CALL DEGDB(EBNODB,LUPC,START,STEP,NSTEP,DBDEGUPC)
00309        .  CALL DEGDB(EBNODB,LAVC,START,STEP,NSTEP,DBDEGAVC)
00310        .  CALL DEGDB(EBNODB,LOWC,START,STEP,NSTEP,DBDEGOWC)
00311 C      .
00312        .  BERP=FLOAT(TNERP)/TNB
00313        .  RMSERP=SQRT(BERP*(1-BERP)/TNB)
00314        .  LUPP=ALOG10(BERP+RMSERP+ALMOST0)
00315        .  LAVP=ALOG10(BERP+ALMOST0)
00316        .  LOWP=ALOG10(BERP-RMSERP+ALMOST0)
00317 C      .
00318        .  BERS=FLOAT(TNERS)/TNB
00319        .  RMSERS=SQRT(BERS*(1-BERS)/TNB)
00320        .  LUPS=ALOG10(BERS+RMSERS+ALMOST0)
00321        .  LAVS=ALOG10(BERS+ALMOST0)
00322        .  LOWS=ALOG10(BERS-RMSERS+ALMOST0)
00323        ...FIN

      ----------------------------------------

00324        TO PRINT-ERROR-RATES
00325        .  WRITE(7,200)
00326        .  WRITE(7,210)TNB
00327        .  WRITE(7,220)
00328        .  WRITE(7,230)TNERA
00329        .  WRITE(7,240)BERA
00330        .  WRITE(7,250)RMSERA
00331        .  WRITE(7,260)LUPA
00332        .  WRITE(7,320)DBDEGUPA
```

```
00333        .  WRITE(7,270)LAVA
00334        .  WRITE(7,320)DBDEGAVA
00335        .  WRITE(7,280)LOWA
00336        .  WRITE(7,320)DBDEGOWA
00337        .  WRITE(7,290)
00338        .  WRITE(7,230)TNERC
00339        .  WRITE(7,240)BERC
00340        .  WRITE(7,250)RMSERC
00341        .  WRITE(7,260)LUPC
00342        .  WRITE(7,320)DBDEGUPC
00343        .  WRITE(7,270)LAVC
00344        .  WRITE(7,320)DBDEGAVC
00345        .  WRITE(7,280)LOWC
00346        .  WRITE(7,320)DBDEGOWC
00347        .  WRITE(7,300)
00348        .  WRITE(7,230)TNERP
00349        .  WRITE(7,240)BERP
00350        .  WRITE(7,250)RMSERP
00351        .  WRITE(7,260)LUPP
00352        .  WRITE(7,270)LAVP
00353        .  WRITE(7,280)LOWP
00354        .  WRITE(7,310)
00355        .  WRITE(7,230)TNERS
00356        .  WRITE(7,240)BERS
00357        .  WRITE(7,250)RMSERS
00358        .  WRITE(7,260)LUPS
00359        .  WRITE(7,270)LAVS
00360        .  WRITE(7,280)LOWS
00361 200    .  FORMAT(///'SIMULATION RESULTS')
00362 210    .  FORMAT('  TOTAL NUMBER OF BITS =              ',I9)
00363 220    .  FORMAT('  CONVENTIONAL DMSK')
00364 230    .  FORMAT('    NUMBER OF BIT ERRORS =            ',I9)
00365 240    .  FORMAT('    BIT ERROR RATE(BER) =             ',F9.6)
00366 250    .  FORMAT('    APPROX. BER RMS ERROR =           'F9.6)
00367 260    .  FORMAT('    LOG BER + RMS ERROR   =           'F9.3)
00368 270    .  FORMAT('    LOG BER               =           'F9.3)
00369 280    .  FORMAT('    LOG BER - RMS ERROR   =           'F9.3)
00370 290    .  FORMAT('  WITH SINGLE ERROR CORRECTION(SEC)')
00371 300    .  FORMAT('  PREDICTED SINGLE BIT ERRORS')
00372 310    .  FORMAT('  SYNDROME ERRORS')
00373 320    .  FORMAT('    DEGRADATION (DB)      =           'F9.3)
00374        ...FIN
00375        END
```

PROCEDURE CROSS-REFERENCE TABLE

```
00222 ADD-NOISE
      00076

00287 COMPUTE-ERROR-RATES
      00083

00199 COMPUTE-SIGMA-FOR-NOISE
```

```
        00072

00280 COUNT-ERRORS
        00081

00192 DIFF-ENCODE-PN-SEQUENCE
        00070

00240 GENERATE-CONVENTIONAL-DMSK-DECISIONS
        00079

00195 GENERATE-MSK-SIGNAL
        00071

00189 GENERATE-PN-SEQUENCE
        00069

00263 GENERATE-SEC-DECISIONS
        00080

00168 GET-INPUT-PARAMETERS
        00087

00214 GET-SIG
        00074

00228 JAM-SIGNAL
        00077

00086 PERFORM-INITIALIZATION
        00068

00324 PRINT-ERROR-RATES
        00084

00112 PRINT-INPUT-PARAMETERS
        00088

00233 PUT-SIGNAL-THROUGH-RX-FILTER
        00078

00217 USE-MULTIPATH
        00075


TRANSPORT FLECS MAY78
1

!
```

```
!C CWGN1-L
 00001        SUBROUTINE CWGN1(IX,SIGMA,CWGN)
 00002 C      THIS SUBROUTINE SIMULATES A COMPLEX BASEBAND GAUSSIAN
 00003 C      NOISE GENERATOR
 00004 C
 00005 C REFERENCE
 00006 C =========
 00007 C      BOX AND MULLER, ANN. MATH. STAT., VOL 28,
 00008 C      PP. 610-611, 1958.
 00009 C
 00010 C INPUTS
 00011 C ======
 00012 C      IX      = SEED FOR GENERATING RANDOM NUMBERS, MUST BE
 00013 C                ODD AND POSITIVE
 00014 C      SIGMA   = STANDARD DEVIATION OF GENERATED GAUSSIAN
 00015 C                DEVIATES
 00016 C
 00017 C OUTPUT
 00018 C ======
 00019 C      CWGN    = COMPLEX GAUSIAN NOISE SAMPLE
 00020 C
 00021 C THIS SUBROUTINE CALLS THE SUBROUTINE UNFORMRN
 00022 C
 00023 C
 00024        COMPLEX CWGN
 00025        TWOPI=6.28318530717959
 00026        CALL UNFORMRN(IX,JX,PP)
 00027        IX=JX
 00028        A=SQRT(-2.0*ALOG(PP))*SIGMA
 00029        CALL UNFORMRN(IX,JX,PP)
 00030        IX=JX
 00031        PHI=TWOPI*PP
 00032        CWGN=A*CMPLX(COS(PHI),SIN(PHI))
 00033        RETURN
 00034        END

       TRANSPORT FLECS MAY78
1

!
```

```
!

C DEGDB-L
00001        SUBROUTINE DEGDB(EBNODB,LBER,START,STEP,NSTEPS,DBDEG)
00002 C;+
00003 C
00004 C      DEGDB
00005 C
00006 C      FUNCTION:     THIS ROUTINE COMPUTES THE DEGRADATION IN DB
00007 C                    FROM IDEAL ANTIPODAL SIGNALLING, USING A
00008 C                    BINARY SEARCH METHOD.
00009 C
00010 C      INPUTS:       EBNODB  = EB/NO (DB) USED TO OBTAIN THE BER.
00011 C                    LBER    = BIT ERROR RATE
00012 C                    START   = START EB/NO (DB) FOR BINARY SEARCH
00013 C                    STEP    = INITAL STEP SIZE (DB)
00014 C                    NSTEPS  = THE NUMBER OF STEPS FOR REQUIRED ACCURACY
00015 C
00016 C      OUTPUTS:      DBDEG   = DEGRADATION FROM IDEAL (DB)
00017 C
00018 C      EFFECTS:      NONE
00019 C
00020 C;-
00021 C
00022        DIMENSION QD(0:400)
00023        COMMON/QVAR/ QD,E,RTPI,QST
00024        REAL LBER
00025        E=2.718281828
00026        RTPI=2.506628275
00027        QST=0.01
00028        INDEX=400
00029        BER=10**LBER
00030        CALL QSAMP(QD,INDEX,QST)
00031        STP=STEP
00032        ENDB=START
00033        DO (I=1,NSTEPS)
00034        .  EBNO=10**(ENDB/10)
00035        .  PE=QC(SQRT(2*EBNO))
00036        .  WHEN(PE.LT.BER)ENDB=ENDB-STP
00037        .  ELSE ENDB=ENDB+STP
00038        .  STP=STP/2
00039        ...FIN
00040        DBDEG=EBNODB-ENDB
00041        RETURN
00042        END

      TRANSPORT FLECS MAY78
1
00043        SUBROUTINE QSAMP(Q,N,ST)
00044 C
00045 C      THIS SUBROUTINE COMPUTES A VECTOR OF SAMPLED
00046 C      VALUES OF THE Q-FUNCTION FROM 0 TO 4 STANDARD
00047 C      DEVIATIONS
00048 C
00049 C      INPUTS:       N   = HIGHEST INDEX FOR THE Q VECTOR
```

```
C                    ST  = INTEGRATION STEP SIZE USED FOR
C                          SIMPSON'S RULE
C
C     OUTPUTS:       Q(0:N) = SAMPLED VALUES OF THE Q-FUNCTION
C                             IN ST INTERVALS
C
C     LOCAL VARIABLES:
C
C                    E     = THE CONSTANT E
C                    RTPI  = ROOT TWO PI CONSTANT
C                    G(X)  = THE GUASSIAN PDF FUNCTION
C                    S     = INTEGRATION SUM
      DIMENSION Q(0:N)
      DOUBLE PRECISION E,G,RTPI,S
      G(X)=E**(-X*X/2)/RTPI
      E=2.718281828
      RTPI=2.506628275
      S=0
      Q(0)=0.5
      DO (I=1,N)
      .  S=S+(G(I*ST)+4*G(I*ST-ST/2)+G(I*ST-ST))*ST/6
      .  Q(I)=0.5-S
      ...FIN
      RETURN
      END
```

TRANSPORT FLECS MAY78

1

```
      FUNCTION QC(X)
C     THIS FUNCTION COMPUTES AN APPROXIMATION TO THE
C     CONTINUOUS Q-FUNCTION OF X WHICH IS ACCURATE TO WITHIN
C     0.9 PERCENT.
C     FOR THE MAGNITUDE OF X LESS THAN 4, INTERPOLATION IS
C     USED BETWEEN THE DISCRETE Q-FUNCTION SAMPLES
C     OF VECTOR QD FOR ABS(X)>  4 AN ACCURATE LOWER
C     BOUND IS USED.
C
C     INPUTS:        E    = THE CONSTANT E
C                    RTPI = ROOT TWO PI CONSTANT
C                    STEP = INTEGRATION STEP SIZE USED FOR QD
C                    QD   = VECTOR OF DISCRETE Q-FUNCTION SAMPLES
C
C     LOCAL VARIABLES:
C
C                    AX        = ABSOLUTE VALUE OF X
C                    I         = INTEGER INDEX FOR QD
C                    CI        = CONTINUOUS INDEX FOR QD TO,INTERPOLATE
C                    TAIL(X)   = THE Q-FUNCTION APPROXIMATION FOR X> 4
      COMMON/QVAR/ QD(0:400),E,RTPI,STEP
      TAIL(X)=(X*X-1)*E**(-X*X/2)/X**3/RTPI
      AX=ABS(X)
      I=CI=AX/STEP
      WHEN (AX.LT.4) QC= QD(I)+(CI-I)*(QD(I+1)-QD(I))
      ELSE QC=TAIL(AX)
```

```
00101        IF (X.LT.0) QC=1-QC
00102        RETURN
00103        END
```

TRANSPORT FLECS MAY78

1

!

```
!C DEM-L
 00001        SUBROUTINE DEMOD(SIG,SIGD,NS,FC,DEL,PHA)
 00002 C;+
 00003 C      DEMOD
 00004 C
 00005 C      FUNCTION: TO PERFORM THE DEMODULATION PROCESS BETWEEN THE
 00006 C                SIGNAL SIG AND THE DELAYED VERSION SIGD OF
 00007 C                THIS SIGNAL.
 00008 C
 00009 C      INPUTS: SIG = RECEIVED MSK SIGNAL
 00010 C              SIGD = DELAYED MSK SIGNAL
 00011 C              NS = NUMBER OF SAMPLES
 00012 C              FC = CARRIER FREQUENCY(MHZ)
 00013 C              DEL = BRANCH DELAY (USEC)
 00014 C              PHA = BRANCH PHASE SHIFT
 00015 C
 00016 C      OUTPUTS: SIGD = DEMODULATED SIGNAL
 00017 C
 00018 C      EFFECTS: SIGD INPUT IS LOST
 00019 C
 00020 C;-
 00021 C
 00022        INTEGER NS
 00023        REAL FC,DEL,PHA,PHI
 00024        COMPLEX SIG(NS),SIGD(NS),EJPHI
 00025 C
 00026        TWOPI=6.2831853
 00027        PHI=AMOD(TWOPI*FC*DEL,TWOPI)-PHA
 00028        EJPHI= CMPLX(COS(PHI),SIN(PHI))
 00029 C
 00030 C  CALCULATE DEMODULATED SIGNAL SAMPLES
 00031        DO (I=1,NS)
 00032        .  SIGD(I)=CMPLX(REAL(SIG(I)*CONJG(SIGD(I))*EJPHI),0.0)
 00033        ...FIN
 00034        RETURN
 00035        END

       TRANSPORT FLECS MAY78
1
 00036        SUBROUTINE HARD(SIG,NS,NB,A,THHLD)
 00037 C;+
 00038 C      HARD
 00039 C
 00040 C      FUNCTION: TO PERFORM HARD DECISIONS ON THE BIT
 00041 C                INTERVAL SPACED SAMPLES OF THE INPUT SIGNAL.
 00042 C
 00043 C      INPUTS: SIG = INPUT DEMODULATED SIGNAL
 00044 C              NS = NUMBER OF SAMPLES
 00045 C              NB = NUMBER OF BITS
 00046 C              THHLD = THE THRESHOLD LEVEL
 00047 C
 00048 C      OUTPUTS: A = OUTPUT DECISIONS
 00049 C
```

```
C     EFFECTS: NONE
C
C;-
C
      INTEGER NS,NB,NSPB,A(NB)
      REAL THHLD
      COMPLEX SIG(NS)
C
      NSPB=NS/NB
C
C  COMPUTE HARD DECISIONS
      J=1
      DO (I=1,NB)
      .  WHEN (REAL(SIG(J)).GT.THHLD) A(I)=1
      .  ELSE A(I)=-1
      .  J=J+NSPB
      ...FIN
      RETURN
      END
```

TRANSPORT FLECS MAY78

1

```
      SUBROUTINE HARDLIM(SIG,CHECK,NS)
C;+
C     HARD LIMITTER
C
C     FUNCTION:   TO PERFORM THE HARD LIMITTER FUNCTION TO THE SIGNAL
C                 IN THE TIME DOMAIN IF IT IS CALLED FOR BY CHECK.
C
C     INPUTS: SIG   = INPUT SIGNAL IN THE TIME DOMAIN
C             CHECK = DECISION CHECK:  0 DONT PERFORM FUNCTION
C                                      1 PERFORM FUNCTION
C
C     OUTPUTS: SIG = OUTPUT SIGNAL IN TIME DOMAIN
C
C     EFFECTS: CHANGES VALUES OF THE SIGNAL
C
C;-
C
      INTEGER CHECK,I,NS
      COMPLEX SIG(NS)
C
      IF(CHECK.NE.0)
      .  DO(I=1,NS)
      .  .  IF(CABS(SIG(I)).EQ.0)SIG(I)=CMPLX(0.707,0.707)
      .  .  SIG(I)=SIG(I)/CABS(SIG(I))
      .  ...FIN
      ...FIN
      RETURN
      END
```

TRANSPORT FLECS MAY78

1

```
      SUBROUTINE SEC(A,B,NB,NERP,NERS)
```

```
00098 C
00099 C;+
00100 C     SEC
00101 C
00102 C     FUNCTION: THIS ROUTINE ATTEMPTS TO CORRECT THE
00103 C               CONVENTIONAL DMSK DATA STREAM USING
00104 C               THE GENERATED PARITY BIT STREAM AND THE
00105 C               SINGLE ERROR CORRECTING(SEC) LOGIC CIRCUIT.
00106 C
00107 C     INPUTS: A = DATA SOURCE
00108 C             B = PARITY SOURCE
00109 C             NB = NUMBER OF BITS
00110 C
00111 C     OUTPUTS: B = CORRECTED DATA STREAM
00112 C              NERP = NUMBER OF SINGLE ERRORS PREDICTED COUNTER
00113 C              NERS = NUMBER OF SYNDROME ERRORS COUNTER
00114 C
00115 C     EFFECTS: THE CORRECTED DATA STREAM IS DELAYED BY
00116 C              ONE BIT INTERVAL, AND THE PARITY BITS ARE
00117 C              LOST.  THE FIRST BIT IS NOT CORRECTED.
00118 C
00119 C;-
00120 C
00121       INTEGER A(NB),B(NB),D,E,EM1,F,FM1,NERP,NERS,NB,NBM2
00122       NERP=0
00123       NERS=0
00124       NBM2=NB-2
00125       F=-1
00126       E=A(NB)*A(NB-1)*B(NB)
00127       DO (K=1,NB)
00128       . KM1=MOD(K+NBM2,NB)+1
00129       . EM1=E
00130       . FM1=F
00131       . D=A(K)*A(KM1)*B(K)
00132       . IF (D.EQ.1) NERS=NERS+1
00133       . E=-FM1*D
00134       . WHEN (D+EM1.EQ.2)
00135       . . F=1
00136       . . NERP=NERP+1
00137       . ...FIN
00138       . ELSE F=-1
00139       . B(K)=-F*A(KM1)
00140       ...FIN
00141       RETURN
00142       END

      TRANSPORT FLECS MAY78
1
00143       SUBROUTINE BERCNT(A,AD,CD,NB,IRUN,NERA,NERC,PRFLAG)
00144 C;+
00145 C     BERCNT
00146 C
00147 C     FUNCTION: THIS ROUTINE COUNTS THE DISCREPANCIES BETWEEN
00148 C               THE DETECTED BIT STREAM AND THE DATA SOURCE
```

```
C                 FOR BOTH THE CONVENTIONAL AND SEC SEQUENCES.
C
C     INPUTS: A = DATA SOURCE
C             AD = DETECTED DATA
C             CD = CORRECTED DETECTED DATA
C             NB = NUMBER OF BITS
C             IRUN = RUN NUMBER
C             PRFLAG = PRINT FLAG
C                    = 0 FOR NONE
C                    = 1 FOR SOME
C                    = 2 FOR ALL
C
C     OUTPUTS: NERA = NUMBER OF ERRORS IN A
C              NERC = NUMBER OF ERRORS IN C
C
C     EFFECTS: NONE
C
C;-
C
      INTEGER A(NB),AD(NB),CD(NB),NB,IRUN,NERA,NERC,NBM1,PRFLAG
      INTEGER DELA,DELC,IERRA(512),IERRC(512)
C
      NERA=NERC=0
      DELA=0
      DELC=1
      NBM1=NB-1
C
COUNT THE BIT ERRORS
      DO (I=1,NB)
      . IA=MOD(I-DELA+NBM1,NB)+1
      . IC=MOD(I-DELC+NBM1,NB)+1
      . IF (A(IA).NE.AD(I))
      . . NERA=NERA+1
      . . IERRA(NERA)=I
      . ...FIN
      . IF (A(IC).NE.CD(I))
      . . NERC=NERC+1
      . . IERRC(NERC)=I
      . ...FIN
      ...FIN
C
C  PRINT POSITION OF THE BIT ERRORS
      IF (PRFLAG.GE.1)
      . WRITE(7,100) IRUN
      . WRITE(7,110)
      . WRITE(7,120) NERA
      ...FIN
      IF (NERA.NE.0.AND.PRFLAG.EQ.2) WRITE(7,130),(IERRA(I),I=1,NERA)
      IF (PRFLAG.GE.1)
      . WRITE(7,140)
      . WRITE(7,120) NERC
      ...FIN
      IF (NERC.NE.0.AND.PRFLAG.EQ.2) WRITE(7,130),(IERRC(I),I=1,NERC)
      RETURN
```

```
00203 100   FORMAT(/,'SIMULATION RUN NUMBER',T29,I4)
00204 110   FORMAT('  CONVENTIONAL DMSK          ')
00205 120   FORMAT('    NUMBER OF BIT ERRORS     ',I4)
00206 130   FORMAT('    BIT ERROR NUMBERS        ',52(T29,10I4,/))
00207 140   FORMAT('  DMSK WITH SEC')
00208       END

      TRANSPORT FLECS MAY78
1

!
```

```
!
C DEM2-L
00001        SUBROUTINE DEMOD(SIG,SIGD,NS,FC,DEL,PHA)
00002 C;+
00003 C      DEMOD
00004 C
00005 C      FUNCTION: TO PERFORM THE DEMODULATION PROCESS BETWEEN THE
00006 C                SIGNAL SIG AND THE DELAYED VERSION SIGD OF
00007 C                THIS SIGNAL.
00008 C
00009 C      INPUTS: SIG = RECEIVED MSK SIGNAL
00010 C              SIGD = DELAYED MSK SIGNAL
00011 C              NS = NUMBER OF SAMPLES
00012 C              FC = CARRIER FREQUENCY(MHZ)
00013 C              DEL = BRANCH DELAY (USEC)
00014 C              PHA = BRANCH PHASE SHIFT
00015 C
00016 C      OUTPUTS: SIGD = DEMODULATED SIGNAL
00017 C
00018 C      EFFECTS: SIGD INPUT IS LOST
00019 C
00020 C;-
00021 C
00022        INTEGER NS
00023        REAL FC,DEL,PHA,PHI
00024        COMPLEX SIG(NS),SIGD(NS),EJPHI
00025 C
00026        TWOPI=6.2831853
00027        PHI=AMOD(TWOPI*FC*DEL,TWOPI)-PHA
00028        EJPHI=CMPLX(COS(PHI),SIN(PHI))
00029 C
00030 C  CALCULATE DEMODULATED SIGNAL SAMPLES
00031        DO (I=1,NS)
00032        .  SIGD(I)=CMPLX(REAL(SIG(I)*CONJG(SIGD(I))*EJPHI),0.0)
00033        ...FIN
00034        RETURN
00035        END

      TRANSPORT FLECS MAY78
1
00036        SUBROUTINE HARD(SIG,NS,NB,A,THHLDPL,THHLDMI)
00037 C;+
00038 C      HARD
00039 C
00040 C      FUNCTION: TO PERFORM HARD DECISIONS ON THE BIT
00041 C                INTERVAL SPACED SAMPLES OF THE INPUT SIGNAL.
00042 C
00043 C      INPUTS: SIG = INPUT DEMODULATED SIGNAL
00044 C              NS = NUMBER OF SAMPLES
00045 C              NB = NUMBER OF BITS
00046 C              THHLDPL = PLUS THRESHOLD LEVEL
00047 C              THHLDMI = MINUS THRESHOLD LEVEL
00048 C
00049 C      OUTPUTS: A = OUTPUT DECISIONS
00050 C
```

```
C     EFFECTS: NONE
C
C;-
C
      INTEGER NS,NB,NSPB,A(NB)
      REAL THHLDPL,THHLDMI
      COMPLEX SIG(NS)
C
      NSPB=NS/NB
C
C  COMPUTE HARD DECISIONS
      J=1
      WHEN(REAL(SIG(J)).GT.(THHLDPL+THHLDMI)/2.0)A(1)=1
      ELSE A(1)=-1
      DO (I=2,NB)
      . J=J+NSPB
      . WHEN(A(I-1).GT.0.0)
      . . WHEN (REAL(SIG(J)).GT.THHLDPL) A(I)=1
      . . ELSE A(I)=-1
      . ...FIN
      . ELSE
      . . WHEN (REAL(SIG(J)).GT.THHLDMI) A(I)=1
      . . ELSE A(I)=-1
      .. ...FIN
      ...FIN
      RETURN
      END
```

TRANSPORT FLECS MAY78

1

```
      SUBROUTINE HARDLIM(SIG,CHECK,NS)
C;+
C     HARD LIMITTER
C
C     FUNCTION:   TO PERFORM THE HARD LIMITTER FUNCTION TO THE SIGNAL
C                 IN THE TIME DOMAIN IF IT IS CALLED FOR BY CHECK.
C
C     INPUTS: SIG   = INPUT SIGNAL IN THE TIME DOMAIN
C             CHECK = DECISION CHECK:  0 DONT PERFORM FUNCTION
C                                      1 PERFORM FUNCTION
C
C     OUTPUTS: SIG = OUTPUT SIGNAL IN TIME DOMAIN
C
C     EFFECTS: CHANGES VALUES OF THE SIGNAL
C
C;-
C
      INTEGER CHECK,I,NS
      COMPLEX SIG(NS)
C
      IF(CHECK.NE.0)
      . DO(I=1,NS)
      . . IF(CABS(SIG(I)).EQ.0)SIG(I)=CMPLX(0.707,0.707)
      . . SIG(I)=SIG(I)/CABS(SIG(I))
      . ...FIN
```

```
00103        ...FIN
00104        RETURN
00105        END

     TRANSPORT FLECS MAY78
1
00106        SUBROUTINE SEC(A,B,NB,NERP,NERS)
00107 C
00108 C;+
00109 C      SEC
00110 C
00111 C      FUNCTION: THIS ROUTINE ATTEMPTS TO CORRECT THE
00112 C                CONVENTIONAL DMSK DATA STREAM USING
00113 C                THE GENERATED PARITY BIT STREAM AND THE
00114 C                SINGLE ERROR CORRECTING(SEC) LOGIC CIRCUIT.
00115 C
00116 C      INPUTS: A = DATA SOURCE
00117 C              B = PARITY SOURCE
00118 C             NB = NUMBER OF BITS
00119 C
00120 C      OUTPUTS: B = CORRECTED DATA STREAM
00121 C               NERP = NUMBER OF SINGLE ERRORS PREDICTED COUNTER
00122 C               NERS = NUMBER OF SYNDROME ERRORS COUNTER
00123 C
00124 C      EFFECTS: THE CORRECTED DATA STREAM IS DELAYED BY
00125 C               ONE BIT INTERVAL, AND THE PARITY BITS ARE
00126 C               LOST.  THE FIRST BIT IS NOT CORRECTED.
00127 C
00128 C;-
00129 C
00130        INTEGER A(NB),B(NB),D,E,EM1,F,FM1,NERP,NERS,NB,NBM2
00131        NERP=0
00132        NERS=0
00133        NBM2=NB-2
00134        F=-1
00135        E=A(NB)*A(NB-1)*B(NB)
00136        DO (K=1,NB)
00137        .  KM1=MOD(K+NBM2,NB)+1
00138        .  EM1=E
00139        .  FM1=F
00140        .  D=A(K)*A(KM1)*B(K)
00141        .  IF (D.EQ.1) NERS=NERS+1
00142        .  E=-FM1*D
00143        .  WHEN (D+EM1.EQ.2)
00144        .  .  F=1
00145        .  .  NERP=NERP+1
00146        .  ...FIN
00147        .  ELSE F=-1
00148        .  B(K)=-F*A(KM1)
00149        ...FIN
00150        RETURN
00151        END

     TRANSPORT FLECS MAY78
1
```

```
      SUBROUTINE BERCNT(A,AD,CD,NB,IRUN,NERA,NERC,PRFLAG)
C;+
C     BERCNT
C
C     FUNCTION: THIS ROUTINE COUNTS THE DISCREPANCIES BETWEEN
C               THE DETECTED BIT STREAM AND THE DATA SOURCE
C               FOR BOTH THE CONVENTIONAL AND SEC SEQUENCES.
C
C     INPUTS: A = DATA SOURCE
C             AD = DETECTED DATA
C             CD = CORRECTED DETECTED DATA
C             NB = NUMBER OF BITS
C             IRUN = RUN NUMBER
C             PRFLAG = PRINT FLAG
C                    = 0 FOR NONE
C                    = 1 FOR SOME
C                    = 2 FOR ALL
C
C     OUTPUTS: NERA = NUMBER OF ERRORS IN A
C              NERC = NUMBER OF ERRORS IN C
C
C     EFFECTS: NONE
C
C;-
C
      INTEGER A(NB),AD(NB),CD(NB),NB,IRUN,NERA,NERC,NBM1,PRFLAG
      INTEGER DELA,DELC,IERRA(512),IERRC(512)
C
      NERA=NERC=0
      DELA=0
      DELC=1
      NBM1=NB-1
C
COUNT THE BIT ERRORS
      DO (I=1,NB)
      .  IA=MOD(I-DELA+NBM1,NB)+1
      .  IC=MOD(I-DELC+NBM1,NB)+1
      .  IF (A(IA).NE.AD(I))
      .  .  NERA=NERA+1
      .  .  IERRA(NERA)=I
      .  ...FIN
      .  IF (A(IC).NE.CD(I))
      .  .  NERC=NERC+1
      .  .  IERRC(NERC)=I
      .  ...FIN
      ...FIN
C
C  PRINT POSITION OF THE BIT ERRORS
      IF (PRFLAG.GE.1)
      .  WRITE(7,100) IRUN
      .  WRITE(7,110)
      .  WRITE(7,120) NERA
      ...FIN
      IF (NERA.NE.0.AND.PRFLAG.EQ.2) WRITE(7,130),(IERRA(I),I=1,NERA)
      IF (PRFLAG.GE.1)
```

```
00207        .  WRITE(7,140)
00208        .  WRITE(7,120) NERC
00209        ...FIN
00210        IF (NERC.NE.0.AND.PRFLAG.EQ.2) WRITE(7,130),(IERRC(I),I=1,NERC)
00211        RETURN
00212 100    FORMAT(/,'SIMULATION RUN NUMBER',T29,I4)
00213 110    FORMAT('  CONVENTIONAL DMSK          ')
00214 120    FORMAT('    NUMBER OF BIT ERRORS     ',I4)
00215 130    FORMAT('    BIT ERROR NUMBERS        ',52(T29,10I4,/))
00216 140    FORMAT('  DMSK WITH SEC')
00217        END

       TRANSPORT FLECS MAY78
1

!
```

```
!C FILRES-L
00001        SUBROUTINE TXRES(SFTX,DFTX,NPTX,FILTX,NPFIL,PRFLAG)
00002 C;+
00003 C      TXRES
00004 C
00005 C      FUNCTION: TO SET UP THE COMPLEX FREQUENCY RESPONSE OF THE
00006 C                TRANSMIT FILTER.
00007 C
00008 C      METHOD: READ IN AMPLITUDE (IN DB) AND PHASE(DEGREES)
00009 C              FROM CHANNEL 11.
00010 C
00011 C      INPUTS: NPFIL = MAXIMUM NUMBER OF FILTER SAMPLE POINTS
00012 C              PRFLAG= PRINT FLAG
00013 C                    = 0 FOR NONE
00014 C                    = 1 FOR PART
00015 C                    = 2 FOR ALL
00016 C
00017 C      OUTPUTS: SFTX = START FREQUENCY (MHZ)
00018 C               DFTX = DELTA FREQUENCY, IE SPACING (MHZ)
00019 C               NPTX = NUMBER OF POINTS
00020 C               FILTX = COMPLEX FREQUENCY SAMPLES OF THE FILTER
00021 C                       RESPONSE.
00022 C
00023 C      EFFECTS: NONE
00024 C
00025 C;-
00026 C
00027        INTEGER NPTX,NPFIL,PRFLAG
00028        REAL SFTX,DFTX
00029        COMPLEX FILTX(NPFIL)
00030 C
00031        RADFACT=3.141592654/180
00032 C
00033        INPUT(11) SFTX,DFTX,NPTX
00034        IF (NPTX.GT.NPFIL)
00035        .  OUTPUT(7) 'INVALID NPTX',NPTX
00036        .  OUTPUT(7) 'ERROR OCCURRED IN ROUTINE TXRES'
00037        .  STOP
00038        ...FIN
00039        IF (PRFLAG.GE.1)
00040        .  WRITE(7,10)
00041 10     .  FORMAT(//,'TRANSMIT FILTER PARAMETERS')
00042        .  WRITE(7,20)SFTX
00043 20     .  FORMAT('  STARTING FREQUENCY (MHZ) =              ',F9.4)
00044        .  WRITE(7,30)DFTX
00045 30     .  FORMAT('  FREQUENCY SPACING OF RESPONSE SAMPLES= ',F9.4)
00046        .  WRITE(7,40)NPTX
00047 40     .  FORMAT('  NUMBER OF POINTS =                     ',I9)
00048        ...FIN
00049        IF (PRFLAG.EQ.2) WRITE(7,50)
00050 50     FORMAT('  SAMPLE   AMP(DB)  PHASE(DEGREES)')
00051        DO (I=1,NPTX)
00052        .  INPUT(11),AMPDB,PHASE
00053        .  IF (PRFLAG.EQ.2) WRITE(7,60) I,AMPDB,PHASE
```

```
00054 60     .  FORMAT(I6,F11.5,F11.3)
00055 C      .
00056        .  RADIANS=PHASE*RADFACT
00057        .  AMP=10**(AMPDB/20)
00058        .  FILTX(I)=AMP*CMPLX(COS(RADIANS),SIN(RADIANS))
00059        ...FIN
00060        RETURN
00061        END

      TRANSPORT FLECS MAY78
1
00062        SUBROUTINE RXRES(SFRX,DFRX,NPRX,FILRX,NPFIL,PRFLAG)
00063 C;+
00064 C      RXRES
00065 C
00066 C      FUNCTION: TO SET UP THE COMPLEX FREQUENCY RESPONSE OF THE
00067 C                RECEIVE FILTER.
00068 C
00069 C      METHOD: READ IN AMPLITUDE (IN DB) AND PHASE(DEGREES)
00070 C              FROM CHANNEL 12.
00071 C
00072 C      INPUTS: NPFIL = MAXIMUM NUMBER OF FILTER SAMPLE POINTS
00073 C              PRFLAG= PRINT FLAG
00074 C                    = 0 FOR NONE
00075 C                    = 1 FOR PART
00076 C                    = 2 FOR ALL
00077 C
00078 C      OUTPUTS: SFRX = START FREQUENCY (MHZ)
00079 C               DFRX = DELTA FREQUENCY, IE SPACING (MHZ)
00080 C               NPRX = NUMBER OF POINTS
00081 C               FILRX = COMPLEX FREQUENCY SAMPLES OF THE FILTER
00082 C                       RESPONSE.
00083 C
00084 C      EFFECTS: NONE
00085 C
00086 C;-
00087 C
00088        INTEGER NPRX,NPFIL,PRFLAG
00089        REAL SFRX,DFRX
00090        COMPLEX FILRX(NPFIL)
00091 C
00092        RADFACT=3.141592654/180
00093 C
00094        INPUT(12) SFRX,DFRX,NPRX
00095        IF (NPRX.GT.NPFIL)
00096        .  OUTPUT(7) 'INVALID NPRX',NPRX
00097        .  OUTPUT(7) 'ERROR OCCURRED IN ROUTINE RXRES'
00098        .  STOP
00099        ...FIN
00100        IF (PRFLAG.GE.1)
00101        .  WRITE(7,10)
00102 10     .  FORMAT(//,'RECEIVE FILTER PARAMETERS')
00103        .  WRITE(7,20)SFRX
00104 20     .  FORMAT('  STARTING FREQUENCY (MHZ) =              ',F9.4)
00105        .  WRITE(7,30)DFRX
```

```
00106 30    .  FORMAT('  FREQUENCY SPACING OF RESPONSE SAMPLES= ',F9.4)
00107       .  WRITE(7,40)NPRX
00108 40    .  FORMAT('  NUMBER OF POINTS =                    ',I9)
00109       ...FIN
00110       IF (PRFLAG.EQ.2) WRITE(7,50)
00111 50    FORMAT('  SAMPLE   AMP(DB)  PHASE(DEGREES)')
00112       DO (I=1,NPRX)
00113       .  INPUT(12),AMPDB,PHASE
00114       .  IF (PRFLAG.EQ.2) WRITE(7,60) I,AMPDB,PHASE
00115 60    .  FORMAT(I6,F11.5,F12.3)
00116 C     .
00117       .  RADIANS=PHASE*RADFACT
00118       .  AMP=10**(AMPDB/20)
00119       .  FILRX(I)=AMP*CMPLX(COS(RADIANS),SIN(RADIANS))
00120       ...FIN
00121       RETURN
00122       END
```

TRANSPORT FLECS MAY78

1

```
00123       SUBROUTINE DEMRES(SFDEM,DFDEM,NPDEM,FILDEM,NPFIL,PRFLAG)
00124 C;+
00125 C     DEMRES
00126 C
00127 C     FUNCTION: TO SET UP THE COMPLEX FREQUENCY RESPONSE OF THE
00128 C               DEMOD.  FILTER.
00129 C
00130 C     METHOD: READ IN AMPLITUDE (IN DB) AND PHASE(DEGREES)
00131 C             FROM CHANNEL 13.
00132 C
00133 C     INPUTS: NPFIL = MAXIMUM NUMBER OF FILTER SAMPLE POINTS
00134 C             PRFLAG= PRINT FLAG
00135 C                   = 0 FOR NONE
00136 C                   = 1 FOR PART
00137 C                   = 2 FOR ALL
00138 C
00139 C     OUTPUTS: SFDEM = START FREQUENCY (MHZ)
00140 C              DFDEM = DELTA FREQUENCY, IE SPACING (MHZ)
00141 C              NPDEM = NUMBER OF POINTS
00142 C              FILDEM = COMPLEX FREQUENCY SAMPLES OF THE FILTER
00143 C                       RESPONSE.
00144 C
00145 C     EFFECTS: NONE
00146 C
00147 C;-
00148 C
00149       INTEGER NPDEM,NPFIL,PRFLAG
00150       REAL SFDEM,DFDEM
00151       COMPLEX FILDEM(NPFIL)
00152 C
00153       RADFACT=3.141592654/180
00154 C
00155       INPUT(13) SFDEM,DFDEM,NPDEM
00156       IF (NPDEM.GT.NPFIL)
00157       .  OUTPUT(7) 'INVALID NPDEM',NPDEM
```

```
00158        .  OUTPUT(7) 'ERROR OCCURRED IN ROUTINE DEMRES'
00159        .  STOP
00160        ...FIN
00161        IF (PRFLAG.GE.1)
00162        .  WRITE(7,10)
00163 10     .  FORMAT(//,'DEMOD.  FILTER PARAMETERS')
00164        .  WRITE(7,20)SFDEM
00165 20     .  FORMAT('  STARTING FREQUENCY (MHZ) =              ',F9.4)
00166        .  WRITE(7,30)DFDEM
00167 30     .  FORMAT('  FREQUENCY SPACING OF RESPONSE SAMPLES= ',F9.4)
00168        .  WRITE(7,40)NPDEM
00169 40     .  FORMAT('  NUMBER OF POINTS =                     ',I9)
00170        ...FIN
00171        IF (PRFLAG.EQ.2) WRITE(7,50)
00172 50     FORMAT('  SAMPLE   AMP(DB)  PHASE(DEGREES)')
00173        DO (I=1,NPDEM)
00174        .  INPUT(13),AMPDB,PHASE
00175        .  IF (PRFLAG.EQ.2) WRITE(7,60) I,AMPDB,PHASE
00176 60     .  FORMAT(I6,F11.5,F11.3)
00177 C      .
00178        .  RADIANS=PHASE*RADFACT
00179        .  AMP=10**(AMPDB/20)
00180        .  FILDEM(I)=AMP*CMPLX(COS(RADIANS),SIN(RADIANS))
00181        ...FIN
00182        RETURN
00183        END
```

TRANSPORT FLECS MAY78

1

```
00184        SUBROUTINE MLTPTH(SIG,NS,MLTDLY,CVIDB,IRUN,NRUN)
00185 C;+
00186 C      MLTPTH
00187 C
00188 C      FUNCTION:  TO ADD TO THE SIGNAL THE INTERFERENCE OF A REFLECTED
00189 C                 (DELAYED VERSION) OF THE SIGNAL
00190 C
00191 C      INPUTS: SIG    = THE SIGNAL (IN THE TIME DOMAIN)
00192 C              NS     = NUMBER OF SAMPLES
00193 C              NB     = NUMBER OF BITS
00194 C              MLTDLY = THE DELAY OF THE SIGNAL COMPARED
00195 C                       TO THE DIRECT SIGNAL
00196 C              CVIDB  = CARRIER TO INTERFERENCE RATIO IN DB
00197 C              IRUN   = CURRENT RUN NUMBER
00198 C              NRUN   = NUMBER OF RUNS
00199 C
00200 C      OUTPUTS: SIG   = THE SIGNAL WITH THE INTERFERENCE ADDED
00201 C
00202 C      EFFECTS: CHANGES THE VALUE OF THE SIGNAL IN THE MAIN PROGRAM
00203 C
00204 C;-
00205 C
00206        COMPLEX EJTHETA,SIG(2048),SIGTR(2048)
00207        INTEGER I,J,NS,IRUN,NRUN
00208        REAL ATEN,PHASE
00209 C
```

```
00210        TWOPI=3.14159265*2.0
00211        PHASE=TWOPI*IRUN/FLOAT(NRUN)
00212        EJTHETA=CEXP(CMPLX(0,PHASE))
00213        ATEN=10.0**(-CVIDB/20.0)
00214 C
00215        DO(I=1,NS)
00216        .  J=I-MLTDLY
00217        .  IF(J.LE.0)J=J+NS
00218        .  SIGTR(I)=SIG(I)+ATEN*EJTHETA*SIG(J)
00219        ...FIN
00220 C
00221        DO(I=1,NS)
00222        .  SIG(I)=SIGTR(I)
00223        ...FIN
00224        RETURN
00225        END

      TRANSPORT FLECS MAY78
1
00226        SUBROUTINE JAMMING(SIG,NS,CVJDB,IRUN,NRUN)
00227 C;+
00228 C      JAMMING
00229 C
00230 C      FUNCTION:  TO SIMULATE THE EFFECTS OF A JAMMING
00231 C                 SIGNAL ON THE TRANSMITTED SIGNAL
00232 C
00233 C      INPUTS: SIG   = TRANSMITTED SIGNAL (IN TIME DOMAIN)
00234 C              NS    = NUMBER OF SAMPLES
00235 C              CVJDB = ATENUATION OF JAMMING SIGNAL
00236 C              IRUN  = THE CURRENT RUN NUMBER
00237 C              NRUN  = THE TOTAL NUMBER OF RUNS
00238 C
00239 C      OUPUTS: SIG  = THE JAMMED SIGNAL
00240 C
00241 C      EFFECTS:  CHANGES THE VALUE OF THE SIGNAL
00242 C
00243 C;-
00244 C
00245        COMPLEX SIG(2048),EJTHETA
00246        INTEGER I,IRUN,NRUN,NS
00247        REAL ATEN,PHASE,CVJDB
00248 C
00249        ATEN=10.0**(-CVJDB/20.0)
00250        TWOPI=6.283185308
00251        PHASE=TWOPI*IRUN/NRUN
00252        EJTHETA=ATEN*CEXP(CMPLX(0,PHASE))
00253 C
00254        DO(I=1,NS)
00255        .  SIG(I)=SIG(I)+EJTHETA
00256        ...FIN
00257        RETURN
00258        END

      TRANSPORT FLECS MAY78
1
```

```
C FRQ-L
!00001       SUBROUTINE FFT(M,NS,X,W,ISIGN)
 00002 C;+
 00003 C     FFT
 00004 C
 00005 C     FUNCTION: THIS ROUTINE CALLS THE FAST4
 00006 C               ROUTINE IN THE SIGMA 9 MATH LIBRARY
 00007 C               TO COMPUTE THE FAST FOURIER
 00008 C               TRANSFORM AND ITS INVERSE.
 00009 C
 00010 C     METHOD: FAST4 IN SIGMA 9 LIBRARY
 00011 C
 00012 C     INPUTS: M=SAMPLE NUMBER OF POINTS RADIX 2,IE NS=2**M
 00013 C             NS=NUMBER OF SAMPLES
 00014 C             X =COMPLEX SIGNAL SAMPLES
 00015 C             ISIGN= 0 WHEN THE COSINE TABLE ONLY IS TO
 00016 C                     BE SET UP FOR M.
 00017 C                  = -1 FOR FFT
 00018 C                  = +1 FOR IFFT
 00019 C
 00020 C     OUTPUTS: W = WORK VECTOR OF DIMENSION NS/4
 00021 C              X = TRANSFORMED SIGNAL SAMPLES
 00022 C
 00023 C     EFFECTS: THE COMPLEX FREQUENCY SAMPLES ARE REARRANGED
 00024 C              SO THAT THE MIDDLE SAMPLE CORRESPONDS TO THE
 00025 C              CARRIER FREQUENCY.
 00026 C
 00027 C;-
 00028 C
 00029       INTEGER M,NS,ISIGN
 00030       REAL W(NS/4)
 00031       COMPLEX X(NS),T
 00032 C
 00033       N2=NS/2
 00034 C
 00035       SELECT (ISIGN)
 00036 C     .
 00037 C SET UP
 00038       .  (0)
 00039       .  .  CALL FAST4(M,X,W,ISIGN)
 00040       .  ...FIN
 00041 C     .
 00042 C PERFORM FFT
 00043       .  (-1)
 00044       .  .  CALL FAST4(M,X,W,ISIGN)
 00045 C     .  .
 00046 C     .  .  REARRANGE FREQUENCY SAMPLES AND NORMALIZE
 00047       .  .  DO (I=1,N2)
 00048       .  .  .  J=N2+I
 00049       .  .  .  T=X(I)
 00050       .  .  .  X(I)=X(J)/NS
 00051       .  .  .  X(J)=T/NS
 00052       .  .  ...FIN
```

```
00053          .  ...FIN
00054 C        .
00055 C PERFORM IFFT
00056          .  (1)
00057 C        .  .  REARRANGE FREQUENCY SAMPLES
00058          .  .  DO (I=1,N2)
00059          .  .  .  J=N2+I
00060          .  .  .  T=X(I)
00061          .  .  .  X(I)=X(J)
00062          .  .  .  X(J)=T
00063          .  .  ...FIN
00064 C        .  .
00065          .  .  CALL FAST4(M,X,W,ISIGN)
00066          .  ...FIN
00067 C        .
00068          .  (OTHERWISE)
00069          .  .  OUTPUT(7)'INVALID ISIGN',ISIGN
00070          .  .  OUTPUT(7)'ERROR OCCURRED IN ROUTINE FFT'
00071          .  .  STOP
00072          .  ...FIN
00073          ...FIN
00074          RETURN
00075          END
```

TRANSPORT FLECS MAY78

1

```
00076          SUBROUTINE REMOV(FILT,SF,DF,NP,CF)
00077 C;+
00078 C        REMOV
00079 C
00080 C        FUNCTION: THIS ROUTINE ADJUSTS THE COMPLEX SAMPLES OF THE
00081 C                  FREQUENCY RESPONSE OF A FILTER SUCH THAT THE
00082 C                  AMPLITUDE RESPONSE AT THE CARRIER FREQUENCY IS
00083 C                  UNITY, AND THE LINEAR AND CONSTANT COMPONENT OF
00084 C                  THE PHASE RESPONSE ARE ZERO.
00085 C
00086 C        INPUTS: FILT = COMPLEX SAMPLES OF THE FREQ. RESPONSE
00087 C                SF = STARTING FREQUENCY OF THE FREQ. RESPONSE
00088 C                DF = FREQ. SPACING OF THE FREQ. RESPONSE
00089 C                NP = NUMBER OF SAMPLES OF THE FREQ. RESPONSE
00090 C                CF = CARRIER FREQ.
00091 C
00092 C        OUTPUTS: FILT = MODIFIED SAMPLES OF THE FREQ. RESPONSE
00093 C
00094 C        EFFECTS: NONE
00095 C
00096 C;-
00097 C
00098          INTEGER NP,I1,I2
00099          REAL CF,DF,SF,PH,AMP,AMP1,PHA1,PHA2,PHASE,SLOPE
00100          COMPLEX FILT(NP)
00101 C
00102 C  CALCULATE THE AMPLITUDE AT THE CARRRIER FREQUENCY
00103          X=(CF-SF)/DF+1.0
```

```
00104        I1=X
00105        I2=I1+1
00106        AMP1=1.0
00107        AMP=CABS(FILT(I1)-(FILT(I1)-FILT(I2))*(X-I1))
00108        IF(AMP.GT.1.0E-30) AMP1=1.0/AMP
00109 C
00110 C  CALCULATE THE SLOPE AND THE PHASE OF THE PHASE RESPONSE
00111 C  OF THE FILTER AT THE CARRIER FREQUENCY.
00112        PHA1=0.0
00113        PHA2=0.0
00114        IF (AMP.GT.1.0E-30)
00115        .  PHA1=ATAN2(AIMAG(FILT(I1)),REAL(FILT(I1)))
00116        .  PHA2=ATAN2(AIMAG(FILT(I2)),REAL(FILT(I2)))
00117        ...FIN
00118        SLOPE=PHA2-PHA1
00119        PHASE=PHA1+SLOPE*(X-I1)
00120 C
00121 C  REMOVE THE LINEAR AND CONSTANT PHASE COMPONENTS
00122 C  FROM THE PHASE RESPONSE
00123        DO (I=1,NP)
00124        .  PH=(I-X)*SLOPE+PHASE
00125        .  FILT(I)=FILT(I)*AMP1*CMPLX(COS(PH),-SIN(PH))
00126        ...FIN
00127        RETURN
00128        END

      TRANSPORT FLECS MAY78
1
00129        SUBROUTINE FILTER(RFS,DFS,CF,NS,FILT,SF,DF,NP)
00130 C;+
00131 C      FILTER
00132 C
00133 C      FUNCTION: THIS ROUTINE MULTIPLIES THE SIGNAL SAMPLES IN THE
00134 C                FREQUENCY DOMAIN BY THE FREQUENCY RESPONSE OF A
00135 C                FILTER.
00136 C
00137 C      INPUTS:  RFS = SIGNAL SAMPLES INPUT TO THE FILTER
00138 C               DFS = FREQUENCY SPACING BETWEEN SIGNAL SAMPLES
00139 C               NS = NUMBER OF SIGNAL SAMPLES
00140 C               CF = CARRIER FREQUENCY
00141 C               FILT = SAMPLES OF THE FREQUENCY RESPONSE OF THE FILTER
00142 C               SF = STARTING FREQUENCY OF THE FREQ. RESPONSE
00143 C               DF = FREQUENCY SPACING BETWEEN SAMPLES OF THE
00144 C                    FREQUENCY RESPONSE
00145 C               NP = NUMBER OF SAMPLES OF THE FREQ. RESPONSE
00146 C
00147 C      OUTPUTS: RFS = SIGNAL SAMPLES AT THE FILTER OUTPUT
00148 C
00149 C      EFFECTS: NONE
00150 C
00151 C;-
00152 C
00153        INTEGER NS,NP,NM
00154        REAL DFS,SF,DF,UB,FI,DELF
```

```
00155        COMPLEX RFS(NS),FILT(NP),CMPLX0,CINTER
00156 C
00157        NM=NS/2+1
00158        DELF=CF-SF
00159        CMPLX0=CMPLX(0.0,0.0)
00160 C
00161 C  CALCULATE THE  UPPER BOUND FREQUENCY OF THE RESPONSE.
00162 C  THE FREQUENCY IS RELATIVE TO 'SF'
00163        UB=(NP-1)*DF
00164 C
00165 C  CALCULATE THE SIGNAL SAMPLES AT THE FILTER OUTPUT
00166        DO (I=1,NS)
00167        .  FI=(I-NM)*DFS+DELF
00168        .  WHEN (FI.GT.UB.OR.FI.LT.0.0) RFS(I)=CMPLX0
00169        .  ELSE RFS(I)=RFS(I)*CINTER(FILT,DF,NP,FI)
00170        ...FIN
00171        RETURN
00172        END
```

TRANSPORT FLECS MAY78

1

```
00173        SUBROUTINE MULT(RFS,DFS,NS,AMP,TAU,PHI)
00174 C;+
00175 C      MULT
00176 C
00177 C      FUNCTION: THIS ROUTINE MULTIPLIES THE SIGNAL SAMPLES
00178 C                IN THE FREQUENCY DOMAIN BY THE CONSTANT
00179 C                AMP*EXP[J*(W*TAU+PHI)].
00180 C
00181 C      INPUTS: RFS = SIGNAL SAMPLES IN THE FREQUENCY DOMAIN
00182 C              DFS = FREQ. SPACING BETWEEN THE SIGNAL SAMPLES
00183 C              NS  = NUMBER OF SIGNAL SAMPLES
00184 C              AMP = RELATIVE AMPLITUDE
00185 C              TAU = TIME SHIFT
00186 C              PHI = PHASE SHIFT
00187 C
00188 C      OUTPUTS: RFS = SIGNAL SAMPLES ADJUSTED IN AMPLITUDE,
00189 C                     SHIFTED IN TIME AND PHASE.
00190 C
00191 C      EFFECTS: NONE
00192 C
00193 C;-
00194 C
00195        INTEGER NS,NM
00196        REAL AMP,DFS,TAU,PHI,WI,TWOPI,PHASE
00197        COMPLEX RFS(NS)
00198 C
00199        TWOPI=6.2831853
00200        NM=NS/2+1
00201 C
00202        DO (I=1,NS)
00203 C      .
00204 C  CALCULATE THE FREQUENCY OF THE I-TH SIGNAL SAMPLE
00205        .  WI=TWOPI*(I-NM)*DFS
```

```
C     .
C  MULTIPLY THE SIGNAL SAMPLE BY THE COMPLEX CONSTANT
      .  PHASE=WI*TAU+PHI
      .  RFS(I)=RFS(I)*AMP*CMPLX(COS(PHASE),SIN(PHASE))
      ...FIN
      RETURN
      END
```

TRANSPORT FLECS MAY78

1

```
      SUBROUTINE GRPDLY(RFS,DFS,NS,AMP,TAU,PHI,T,A)
C;+
C     GRPDLY
C
C     FUNCTION: THIS ROUTINE MULTIPLIES THE SIGNAL SAMPLES
C               IN THE FREQUENCY DOMAIN BY AMP*EXP[J*PHASE]
C               WHERE PHASE = -2*PI*[FI*TAU+(FT)**2*A/2]+PHI
C               I.E. GROUP DELAY = TAU+(AT)*(FT)
C
C     INPUTS: RFS = SIGNAL SAMPLES IN THE FREQUENCY DOMAIN
C             DFS = FREQ. SPACING BETWEEN THE SIGNAL SAMPLES
C             NS  = NUMBER OF SIGNAL SAMPLES
C             AMP = RELATIVE AMPLITUDE
C             TAU = CONSTANT DELAY
C             PHI = PHASE SHIFT
C             T   = BIT PERIOD
C             A   = LINEAR GROUP DELAY PARAMETER
C
C     OUTPUTS: RFS = SIGNAL SAMPLES ADJUSTED IN AMPLITUDE,
C                    SHIFTED IN PHASE.
C
C     EFFECTS: NONE
C
C;-
C
      INTEGER NS,NM
      REAL AMP,DFS,TAU,PHI,FI,TWOPI,PHASE,T,A
      COMPLEX RFS(NS)
C
      TWOPI=6.2831853
      NM=NS/2+1
C
      DO (I=1,NS)
C     .
C  CALCULATE THE FREQUENCY OF THE I-TH SIGNAL SAMPLE
      .  FI=(I-NM)*DFS
C     .
C  MULTIPLY THE SIGNAL SAMPLE BY THE COMPLEX CONSTANT
      .  PHASE=-TWOPI*(FI*TAU+(FI*T)**2*A/2)+PHI
      .  RFS(I)=RFS(I)*AMP*CMPLX(COS(PHASE),SIN(PHASE))
      ...FIN
      RETURN
      END
```

TRANSPORT FLECS MAY78

1

```
00256        COMPLEX FUNCTION CINTER(SAMPLE,DF,NP,FI)
00257 C;+
00258 C      CINTER
00259 C
00260 C      FUNCTION: THIS ROUTINE COMPUTES A SAMPLE BY USING
00261 C                LINEAR INTERPOLATION BETWEEN TWO SAMPLES
00262 C                WHICH ENCLOSE THE WANTED ONE.
00263 C
00264 C      INPUTS: SAMPLE = SAMPLES TO BE INTERPOLATED
00265 C              DF = FREQUENCY SPACING BETWEEN SAMPLES
00266 C              NP = NUMBER OF SAMPLES
00267 C              FI = FREQUENCY OF THE WANTED SAMPLE OFFSET
00268 CC                  FROM THE FREQUENCY OF THE FIRST SAMPLE
00269 C                   OF THE ARRAY 'SAMPLE'.
00270 C
00271 C      OUTPUTS: CINTER = VALUE OF THE WANTED SAMPLE
00272 C
00273 C      EFFECTS; NONE
00274 C
00275 C;-
00276 C
00277        INTEGER NP,INDEX1,INDEX2
00278        REAL DF,FI,FRACT
00279        COMPLEX SAMPLE(NP)
00280 C
00281 C  FIND THE ARRAY INDICES OF THE SAMPLES THAT ENCLOSE
00282 C  THE WANTED ONE
00283        X=FI/DF+1.0
00284        INDEX1=IFIX(X)
00285        WHEN (INDEX1.LE.0.OR.INDEX1.GE.NP) CINTER=CMPLX(0,0)
00286        ELSE
00287        .  INDEX2=INDEX1+1
00288 C      .
00289 C  FIND THE WANTED SAMPLE
00290        .  FRACT=X-INDEX1
00291        .  CINTER=SAMPLE(INDEX1)-(SAMPLE(INDEX1)-SAMPLE(INDEX2))*FRACT
00292        ...FIN
00293        RETURN
00294        END
```

TRANSPORT FLECS MAY78

1

!

```
!C MOD-1
00001        SUBROUTINE PNSQNS(A,NB,NSTAGE,NX,NY)
00002 C;+
00003 C      PNSQNS
00004 C
00005 C      FUNCTION: THIS ROUTINE GENERATES A NB-BIT LONG PN
00006 C                SEQUENCE
00007 C
00008 C      METHOD: NSTAGE-STAGE SHIFT REGISTER WITH FEEDBACK
00009 C              TAPS FROM STAGE NX AND STAGE NY IS USED.
00010 C              THE SHIFT REGISTER IS INITIALIZED TO 10000...
00011 C
00012 C      INPUTS: NB = LENGTH OF THE PN SEQUENCE
00013 C              NSTAGE = NUMBER OF STAGES
00014 C              NX,NY = POSITION OF FEEDBACK TAPS
00015 C
00016 C      OUTPUTS: A = NB-BIT LONG PN SEQUENCE
00017 C
00018 C      EFFECTS: THE ALL ZERO'S STATE IS INSERTED AT THE
00019 C               BEGINNING.
00020 C
00021 C;-
00022 C
00023        INTEGER NB,NSTAGE,NX,NY,A(NB)
00024 C
00025 C  INITIALIZE THE SHIFT REGISTER
00026        DO (I=1,NSTAGE) A(I)=-1
00027        A(NSTAGE+1)=1
00028 C
00029 C  GENERATE THE REMAINING BITS OF THE SEQUENCE
00030        DO (I=NSTAGE+2,NB)
00031        .  A(I)=-1
00032        .  IF (A(I-NX).NE.A(I-NY))A(I)=1
00033        ...FIN
00034        RETURN
00035        END

      TRANSPORT FLECS MAY78
1
00036        SUBROUTINE ENCODE(A,NB,B)
00037 C;+
00038 C      ENCODE
00039 C
00040 C      FUNCTION: THIS ROUTINE GENERATES A DIFFERENTIALLY ENCODED
00041 C                DATA SEQUENCE FROM THE DATA SEQUENCE A
00042 C
00043 C      METHOD: B(N)=+1 IF A(N) .NE. B(N-1)
00044 C                   =-1 OTHERWISE
00045 C
00046 C      INPUTS: A = INPUT DATA SEQUENCE
00047 C              NB = NUMBER OF BITS
00048 C
00049 C      OUTPUTS: B = DIFFERENTIALLY ENCODED SEQUENCE
```

```
C
C       EFFECTS: NONE
C
C;-
C
        INTEGER NB,A(NB),B(NB)
C
C  THE FIRST ENCODED BIT IS +1
        B(1)=1
C
C  GENERATE THE REMAINING ENCODED BITS
        DO (I=2,NB) B(I)=-A(I)*B(I-1)
        RETURN
        END
```

TRANSPORT FLECS MAY78

1

```
        SUBROUTINE PULSHP(NPT,ISHAPE,PULSE)
C;+
C       PULSHP
C
C       FUNCTION: THIS ROUTINE GENERATES NPT SAMPLES OVER
C                 ONE PERIOD OF THE PULSE SHAPE USED IN THE
C                 MODULATOR.
C
C       INPUTS: NPT = NUMBER OF SAMPLES TO BE GENERATED
C               ISHAPE = TYPE OF PULSE SHAPE
C                            1 - SINUSOIDAL
C                            2 - RECTANGULAR
C
C       OUTPUTS: PULSE = GENERATED SAMPLE VALUES
C
C       EFFECTS: NONE
C;-
C
        INTEGER NPT,ISHAPE,NPT2
        REAL DELPHI,PHI,PULSE(NPT)
C
        SELECT (ISHAPE)
C       .
C       .  GENERATE SINUSOIDAL PULSE
        .  (1)
        .  .  DELPHI=6.2831853/NPT
        .  .  PHI=0.0
        .  .  DO (I=1,NPT)
        .  .  .  PULSE(I)=SIN(PHI)
        .  .  .  PHI=PHI+DELPHI
        .  .  ...FIN
        .  ...FIN
C       .
C       .  GENERATE RECTANGULAR PULSE
        .  (2)
        .  .  NPT2=NPT/2
        .  .  DO (I=1,NPT2)
```

```
00101          . . .  PULSE(I)=1.0
00102          .  .  .  PULSE(I+NPT2)=-1.0
00103          .  .  ...FIN
00104          .  ...FIN
00105          ...FIN
00106          RETURN
00107          END

       TRANSPORT FLECS MAY78
1
00108          SUBROUTINE MSKMOD(B,NB,PULSE,NPT,NS,NSPBIT,SIG)
00109 C$+
00110 C        MSKMOD
00111 C
00112 C        FUNCTION: THIS ROUTINE GENERATES MSK SIGNAL SAMPLES
00113 C                  AT THE RATE NSPBIT SAMPLES PER BIT.
00114 C
00115 C        METHOD: COMPLEX ENVELOPE NOTATION IS USED TO REPRESENT
00116 C                THE SIGNAL SAMPLES
00117 C
00118 C                S(T) = I(T) - J Q(T)
00119 C
00120 C        INPUTS: B = DIFFERENTIALLY ENCODED DATA SEQUENCE
00121 C                NB = NUMBER OF BITS
00122 C                PULSE = SAMPLE VALUES OF THE MOD. PULSE SHAPE
00123 C                NPT = NUMBER OF SAMPLES IN PULSE
00124 C                NS = NUMBER OF SIGNAL SAMPLES TO BE GENERATED
00125 C                NSPBIT = NUMBER OF SAMPLES PER BIT
00126 C
00127 C        OUTPUTS: SIG = MSK SIGNAL SAMPLES
00128 C
00129 C        EFFECTS: NONE
00130 C
00131 C$-
00132 C
00133          INTEGER NB,NPT,IB,II,IQ,IS,IPI,IPQ,B(NB)
00134          REAL SI,SQ,PULSE(NPT)
00135          COMPLEX SIG(NS)
00136 C
00137          I=1
00138          IPI=NSPBIT+1
00139          IPQ=1
00140 C
00141 C  GENERATE THE SIGNAL SAMPLES
00142          DO (IB=1,NB)
00143 C        .
00144 C  CALCULATE THE I AND Q DATA BIT INDICES
00145          .  K=IB-1
00146          .  IQ=2*(K/2)+2
00147          .  II=2*((K+1)/2)+1
00148          .  II=MOD(II-1,NB)+1
00149 C        .
00150 C  GENERATE SIGNAL SAMPLES FROM I AND Q DATA BITS AND
00151 C  THE MODULATING PULSE SHAPE. FIRST SAMPLE AT TIME 0.
```

```
00152        .  DO (J=1,NSPBIT)
00153        .  .  SI=B(II)*PULSE(IPI)
00154        .  .  SQ=B(IQ)*PULSE(IPQ)
00155        .  .  SIG(I)=CMPLX(SI,-SQ)
00156        .  .  I=I+1
00157        .  .  IPI=MOD(IPI,NPT)+1
00158        .  .  IPQ=MOD(IPQ,NPT)+1
00159        .  ...FIN
00160        ...FIN
00161       RETURN
00162       END
```

TRANSPORT FLECS MAY78

1

!

```
!C MSKEDG-L
00001        SUBROUTINE MSKEDG(SIG,NS,NB,NFIRST,NBIT)
00002 C;+    EDPLOT
00003 C
00004 C      FUNCTION : THIS SUBPROGRAM PLOTS THE EYE-PATTERN WHICH CONSISTS OF
00005 C                 NBIT SYMBOLS STARTING AT THE NFIRST-TH SYMBOL. THE PATT
00006 C                 IS PLOTTED IN A FRAME OF ONE SYMBOL PERIOD.
00007 C
00008 C      INPUTS :   SIG    ... SIGNAL SAMPLES IN THE TIME DOMAIN
00009 C                 NS     ... NUMBER OF SIGNAL SAMPLES
00010 C                 NB     ... NUMBER OF SIMULATION SYMBOLS
00011 C                 NFIRST ... BIT NUMBER AT WHICH THE PLOT STARTS
00012 C                 NBIT   ... NUMBER OF BITS IN THE PLOT
00013 C
00014 C      OUTPUTS :            THE EYE-PATTERN PLOTS ARE PRINTED ON THE
00015 C                           LINE PRINTER
00016 C
00017 C      EFFECTS :  NONE
00018 C
00019 C;-
00020 C
00021 C
00022        INTEGER NS,NFIRST,NBIT,IS
00023        INTEGER IX,IY,NCOLUM,NROW,NSPBIT,NSF,NROW2
00024 C
00025        INTEGER     PATTERN(33,76)
00026 C
00027        REAL AMP,AMPMAX,AMP1,AMP2,DELP,SAMP,SCALE
00028 C
00029        COMPLEX SIG(2048)
00030 C
00031        INTEGER     BLANK/' '/,ASTRX/'*'/,CHARAC/'+'/
00032
00033 C
00034        NCOLUM=76
00035        NROW=33
00036        NSPBIT=NS/NB
00037        NSF=NCOLUM/NSPBIT
00038        NROW2=17
00039        IF((NFIRST.LT.1).OR.(NFIRST.GE.NB))NFIRST=1
00040        IF((NFIRST+NBIT).GT.NB)NBIT=NB-NFIRST
00041 C
00042        IS=NSPBIT*(NFIRST-0.5)+1
00043        DO (I=1,NROW)
00044        .  DO (J=1,NCOLUM)
00045        .  .  PATTERN(I,J)=BLANK
00046        .  ...FIN
00047        ...FIN
00048 C
00049 C      FIND THE MAXIMUM ABSOLUTE VALUE OF THE SIGNAL
00050 C
00051        AMPMAX=0.0
00052        DO (I=0,NSPBIT*NBIT)
00053 C      .
```

```
C     .  TAKE REAL PART OF SIG
C     .
      .  AMP=ABS(REAL(SIG(I+IS)))
      .  IF(AMP.GT.AMPMAX)AMPMAX=AMP
      ...FIN
C
C     FIND THE SCALING FACTOR
C
      SCALE=(NROW2 )/AMPMAX
C
C     STORE THE EYE PATTERN IN THE ARRAY PATTERN
C
      DO(I=1,NBIT)
      .  IX=0
      .  DO(J=1,NSPBIT)
      .  .  AMP1=REAL(SIG(IS))
      .  .  AMP2=REAL(SIG(IS+1))
      .  .  SAMP=AMP1
      .  .  DELP=(AMP2-AMP1)/NSF
      .  .  IS=IS+1
      .  .  DO(K=1,NSF)
      .  .  .  IX=IX+1
      .  .  .  IY=IFIX(-SAMP*SCALE)+NROW2
      .  .  .  PATTERN(IY,IX)=CHARAC
      .  .  .  SAMP=SAMP+DELP
      .  .  ...FIN
      .  ...FIN
      ...FIN
      IX=(NSPBIT/2)*NSF+1
      PATTERN(1,IX)=ASTRX
      PATTERN(NROW,IX)=ASTRX
C
C     PLOT THE EYE-PATTERN
C
      WRITE(7,6000)
      WRITE(7,6020)
      SAMP=AMPMAX
      AMP1=0.5*AMPMAX
      IY=2
      WRITE(7,6040)SAMP,(PATTERN(1,K),K=1,NCOLUM),SAMP
      DO(I=1,4)
      .  WRITE(7,6030)((PATTERN(J,K),K=1,NCOLUM),J=IY,IY+6)
      .  IY=IY+8
      .  SAMP=SAMP-AMP1
      .  WRITE(7,6040)SAMP,(PATTERN(IY-1,K),K=1,NCOLUM),SAMP
      ...FIN
      WRITE(7,6050)
      WRITE(7,6060)
      WRITE(7,6070)NFIRST,NBIT
6070  FORMAT(/3X,' FIRST BIT SAMPLED = ',I8,/T4,
     1       ' NUMBER OF BITS PLOTTED =',I4)
      RETURN
6000  FORMAT(1H1,T29,'***  E Y E    P A T T E R N  ***')
6020  FORMAT(/T7,'.00',T26,'.25',T45,'.50',T64,'.75',
     1        T83,'1.00',/T2,'-----',4('I',18('-')),'I','----')
```

```
00109 6030  FORMAT(T6,'X',76A1,'X')
00110 6040  FORMAT(T1,F5.1,'I',76A1,'I',F5.1)
00111 6050  FORMAT(T2,'-----',4('I',18('-')),'I','----',/T7,'.00',
00112      1       T26,'.25',T45,'.50',T64,'.75',T83,'1.0')
00113 6060  FORMAT(/T10,'* SAMPLING INSTANT.')
00114       END

      TRANSPORT FLECS MAY78
1

!
```

```
!C OPTDEM-L
 00001        SUBROUTINE OPTDEM(SIG,NS,PULSE,NPT,NSPB,NB,B)
 00002 C
 00003 C;+    OPTDEM
 00004 C
 00005 C      FUNCTION: THIS ROUTINE OPTIMALLY DEMODULATES THE
 00006 C                RECEIVED MSK SIGNAL, ASSUMING NO ISI
 00007 C                CAUSED BY PREFILTERING.
 00008 C
 00009 C      METHOD:   MATCHED FILTERING THE I AND Q CHANNEL
 00010 C                SIGNALS WITH THE PULSE SHAPE USED IN THE
 00011 C                MODULATOR.
 00012 C
 00013 C      INPUTS:   SIG=RECEIVED MSK SIGNAL SAMPLES
 00014 C                NS=NUMBER OF SAMPLES
 00015 C                PULSE=SAMPLED VALUES OF THE MODULATOR PULSE
 00016 C                NPT=NUMBER OF POINTS IN THE PULSE
 00017 C                NSPB=NUMBER OF SAMPLES PER BIT
 00018 C                NB=NUMBER OF BITS
 00019 C
 00020 C      OUTPUTS:  B=DECODED DATA SEQUENCE WITHOUT DIFFERENTIAL
 00021 C                  DECODING
 00022 C
 00023 C      EFFECTS:  NONE
 00024 C;-
 00025 C
 00026        INTEGER B(NB)
 00027        REAL PULSE(NPT)
 00028        COMPLEX SIG(NS)
 00029 C
 00030        NSPB2=NSPB*2
 00031 C
 00032 C I IS NSPB AHEAD OF Q
 00033        II=NS+1-NSPB
 00034        IQ=IP=1
 00035        DO (IB=1,NB,2)
 00036        . SUMI=SUMQ=0
 00037        . DO (J=1,NSPB2)
 00038        . . SUMI=SUMI+REAL(SIG(II))*PULSE(IP)
 00039        . . SUMQ=SUMQ+AIMAG(SIG(IQ))*PULSE(IP)
 00040        . . II=MOD(II,NS)+1
 00041        . . IQ=IQ+1
 00042        . . IP=MOD(IP,NPT)+1
 00043        . ...FIN
 00044        . WHEN (SUMI.GE.0) B(IB)=1
 00045        . ELSE B(IB)=-1
 00046        . WHEN (SUMQ.GE.0) B(IB+1)=-1
 00047        . ELSE B(IB+1)=1
 00048        ...FIN
 00049        RETURN
 00050        END
```

```
!C PLTSG-L
 00001        SUBROUTINE PLOTSIG(SIGP,SCALE,ISAMPLE,NSAMPLE)
 00002        COMPLEX SIGP(2048)
 00003        REAL SCALE
 00004        INTEGER SAMPLE
 00005        INTEGER BLANK,STAR
 00006        BLANK=64
 00007        STAR=92
 00008 C
 00009        OFFSET=25
 00010 C
 00011        WRITE(7,515)(STAR,J=1,OFFSET+16)
 00012 515    FORMAT(///56R1)
 00013        DO(SAMPLE=ISAMPLE,ISAMPLE+NSAMPLE)
 00014        .  AMPL=REAL(SIGP(SAMPLE))+0.00003
 00015        .  WRITE(7,510)SAMPLE,AMPL,(BLANK,J=1,SCALE*AMPL+OFFSET),STAR
 00016 510    .  FORMAT(I6,F10.4,50R1)
 00017        ...FIN
 00018        RETURN
 00019        END

       TRANSPORT FLECS MAY78
1

!
```

```
C RMS-L
00001        SUBROUTINE RMSJITTE(SIG,NS,NB,NRMS,RMS)
00002 C;+
00003 C      RMSJITTE
00004 C
00005 C      FUNCTION: TO FIND THE RMS ERROR ON THE
00006 C                ZERO-CROSSING POINTS.
00007 C
00008 C      INPUTS: SIG  = INPUT DEMODULATED SIGNAL
00009 C              NS   = NUMBER OF SAMPLES
00010 C              NB   = NUMBER OF BITS
00011 C              NRMS = NUMBER OF ZERO CROSSINGS TO CHECK
00012 C
00013 C      OUTPUTS: RMS = RMS ERROR ON THE ZERO CROSSINGS
00014 C
00015 C      EFFECTS: NONE
00016 C
00017 C;-
00018 C
00019        INTEGER TT,NS,NB,NSPB,NRMS,T,NSRMS
00020        REAL RMS,A
00021        COMPLEX SIG(NS)
00022 C
00023        RMS = 0
00024        NSPB = NS/NB
00025        NSRMS = NRMS*NSPB
00026        I = 0
00027        TT = 0
00028 C
00029        REPEAT UNTIL (TT.GE.NRMS)
00030        .  I = I+1
00031        .  SIGI = REAL(SIG(I))
00032        .  SIGI1 = REAL(SIG(I+1))
00033        .  IF (SIGI*SIGI1.LT.0)
00034        .  .  A = I+1-SIGI1/(SIGI1-SIGI)
00035        .  .  T = 4*INT((A-1.0)/4)+3
00036        .  .  RMS = RMS+((T-A)/NSPB)**2
00037        .  .  TT = TT+1
00038        .  ...FIN
00039        .  IF (SIGI.EQ.0)
00040        .  .  T = 4*INT(I-1.0/4)+3
00041        .  .  RMS = RMS+((T-I)/NSPB)**2
00042        .  .  TT = TT+1
00043        .  ...FIN
00044        ...FIN
00045        RMS = SQRT(RMS/NRMS)
00046        RETURN
00047        END
```

TRANSPORT FLECS MAY78

1

!