参赛密码 _____

（由组委会填写）

第十届华为杯全国研究生数学建模竞赛

学　　校	南京师范大学
参赛队号	10319003
队员姓名	**1.**佟德宇
	2.顾　燕
	3.贾泽慧

参赛密码 _____

(由组委会填写)

第十届华为杯全国研究生数学建模竞赛

题 目 **功率放大器非线性特性及预失真建模**

摘 要

针对问题一中求解输入输出信号之间的非线性功放特性函数问题，采用了不同的多项式函数，运用最小二乘法或正则化后的最小二乘法进行拟合求解．并用参数 NMSE 来评价所建模型的准确度．结果发现在逼近函数选为函数基的情况下，采用正则化后的最小二乘法得出的模型准确度最好，其对应的参数 NMSE=-68.6294.

同时考虑计算量和模型准确度，在由多项式变形函数逼近功放的模型基础上，来进行预失真模型的建立．根据题中给出的原则和约束，可知预失真模型的表达式与功放模型的表达式是类似的，从而可建立相应的预失真模型.：

$$z(t)=\sum_{k=1}^{K} h_k \left|x(t)\right|^{k-1} x(t)$$

K=4 时，整体模型的放大倍数 g=1.8693，参数 NMSE=-32.5819，EVM=2.3491；K=5 时，g=1.8473，参数 NMSE=-37.1398, EVM=1.3900; K=7 时, g=1.8326，参数 NMSE=-46.0624, EVM=0.4976.

针对问题二，直接将功放的输入输出与题目中所提的“和记忆多项式”模型进行拟合，运用正则化后的最小二乘法进行求解，这很好的保证了模型的可解性．本题只考虑功放模型次数为 5 的情形．当记忆深度为 7 时，得 NMSE=-45.8394；当记忆深度为 3 时，得 NMSE=-44.5315．预失真模型的建立与问题一类似，文中以框图的方式建立了预失真处理的模型实现示意图，并对次数为 5、记忆深度为 3 的情形，求解出整体模型的放大倍数 g=9.4908，参数 NMSE=-37.8368, EVM=0.0128.

针对问题三，将所给的离散的、有限的输入输出数据作为随机过程的样本函数,通过其傅立叶变换得到功率谱密度函数．文中分别给出了输入信号、无预失真补偿的功率放大器输出信号、采用预失真补偿的功率放大器输出信号的功率谱密度图形．可解出它们的 ACPR 分别为-155.6610、-74.3340、-104.4904，最后对结果进行分析评价，得出采用

预失真补偿的功率放大器的输出信号效果比无预失真补偿的效果好.

关键字： 最小二乘法、Tikhonov 正则化、Fourier 变换

一、问题重述

信号的功率放大是电子通信系统的关键功能之一，其实现模块称为功率放大器(PA, Power Amplifier)，简称功放. 功放的输出信号相对于输入信号可能产生非线性变形，这将带来无益的干扰信号，影响信信息的正确传递和接收，此现象称为非线性失真.

功放非线性属于有源电子器件的固有特性，研究其机理并采取措施改善，具有重要意义. 目前已经提出了各种技术来克服功放的非线性失真，其中预失真技术是被研究的较多的一项技术，其最新的研究成果已经被运用于实际的产品中，但在新算法、实现复杂度、计算速度、效果精度等方面仍有相当的研究价值.

预失真的基本原理是：在功放前设置一个预失真处理模块，这两个模块的合成总效果使整体输入-输出特性线性化，输出功率得到充分利用.

文中给出了 NMSE、EVM 等参数评价所建模型其准确度，以及 ACPR 表示信道的带外失真的参数.

根据数据文件中给出的某功放无记忆效应、有记忆效应的复输入输出测试数据：

(1)我们建立此功放的非线性数学模型$G(\cdot)$，并用 NMSE 来评价所建模型的准确度.

(2)根据线性化原则以及“输出幅度限制”和“功率最大化”约束，计算线性化后最大可能的幅度放大倍数，建立预失真模型. 并运用评价指标参数 NMSE/EVM 评价预失真补偿的计算结果.

(3)应用问题二中所给的数据，计算功放预失真补偿前后的功率谱密度(输入信号、无预失真补偿的功率放大器输出信号、采用预失真补偿的功率放大器输出信号)，并用图形的方式表示了这三类信号的功率谱密度. 最后用相邻信道功率比 ACPR 对结果进行分析.

二、模型假设

1、假设题中所给的功放输入输出数据采样误差为 0.

2、假设题中所给的功放输入输出数据具有代表性、一般性.

3、假设存在这样的预失真处理器，能够做到将输入数据变为模型求解所得的预失真处理输出结果.

三、基本知识

§3.1 最小二乘方法

最小二乘方法[1][2]产生于数据拟合问题，它是一种基于观测数据与模型数据之间的差的平方和最小来估计数学模型中参数的方法. 输入数据t与输出数据y之间大致服从如下函数关系

$$y=\phi(x,t),$$

式中$x\in R^n$为待定参数. 为估计参数x的值，要先经过多次试验取得观测数据$(t_1,y_1),(t_2,y_2),\cdots,(t_m,y_m)$，然后基于模型输出值和实际观测值的误差平方和

$$\sum_{i=1}^{m}(y_i-\phi(x,t_i))^2$$

最小来求参数x的值，这就是最小二乘问题. 一般地，$m\gg n$.

引入函数

$$r_i(x)=y_i-\phi(x,t_i),\ i=1,2,\cdots,m,$$

并记

$$r(x)=(r_1(x),\ r_2(x),\ \cdots,\ r_m(x)),$$

则最小二乘问题即为

$$\min_{x\in R^n} r(x)^T r(x).$$

如果最小二乘问题中的模型函数估计准确，那么最小二乘问题的最优值是很靠近零的. 因此 $r(x)$ 常称作残量函数.

对于线性最小二乘问题，残量函数可以表示为

$$r(x)=b-Ax,$$

从而线性最小二乘问题可以表示为

$$\min_{x\in R^n}\|b-Ax\|^2. \tag{3.1.1}$$

若 A 是列满秩的，且考虑到二次凸函数的稳定点即为最小值点，可以直接得到 x 的求解公式，即

$$x=\left(A^TA\right)^{-1}A^Tb. \tag{3.1.2}$$

而对于复数域上的线性最小二乘问题

$$\min_{x\in C^n}\|b-Ax\|^2,$$

也可以直接得到 x 的求解公式，即为

$$x=\left(\overline{A}^TA\right)^{-1}\overline{A}^Tb, \tag{3.1.3}$$

其中，$\overline{A}^T$ 表示 A 的共轭转置.

§3.2 Tikhonov 正则化

在使用最小二乘方法进行参数估计的时候，由于 A 不一定是列满秩的，故 $\overline{A}^TA$ 不一定是可逆的，此时就不能够用上面所推得的公式进行直接的求解了. 为了克服这个困难，考虑 Tikhonov 正则化[3]方法，即给目标函数加上一个正则项(即一个邻近项)

$$\lambda_k\|x-x_k\|^2.$$

此时，最小二乘问题转化为

$$x_{k+1}=\min_{x\in C^n}\|b-Ax\|^2+\lambda_k\|x-x_k\|^2.$$

其中 x_k 是第 k 步迭代得到的解，λ_k 可以选为一个常数或一个单调下降趋于 0 的数列. 迭代的终止准则为

$$\|x_{k+1}-x_k\|\le\varepsilon,$$

其中 ε 是一个给定的误差上界.

考虑到二次凸函数的稳定点即为最小值点，这时问题

$$\min_{x\in C^n}\|b-Ax\|^2+\lambda_k\|x-x_k\|^2$$

是可以直接求解的，给出 x 的求解公式为

$$x=\left(\overline{A}^TA+\lambda_kI\right)^{-1}\left(\overline{A}^Tb+\lambda_kx_k\right).$$

显然，此时即使 A 非列满秩，问题也是可以求解的.

四、问题分析

问题一

题中已给出了某功放无记忆效应的复输入输出测试数据，现需要建立此功放的非线性特性数学模型，拟合出功放的特性函数$G(\cdot)$．根据函数逼近理论，功放的特性函数可以用多项式来表示，也可以用空间中的一组正交函数基来表示．然后采用最小二乘法或正则化后的最小二乘法，将这些情况都进行求解，得出功放的特性函数$G(\cdot)$．并在最后用参数 NMSE(归一化均方误差)来评价所建模型的准确度.

接着，在前面所建模型的基础上，选择一个计算量适当，且准确度较好的$G(\cdot)$的一个拟合模型．然后根据线性化原则以及“输出幅度限制”和“功率最大化”约束，建立预失真模型，使得整体模型线性化后放大倍数尽可能的大．通过对优化模型的分析可知，对预失真特性函数$F(\cdot)$的求解可以转化为对$\left(\frac{G}{g}\right)^{-1}$的求解，且预失真模型的表达式与功放模型的表达式是类似的．在求解$\left(\frac{G}{g}\right)^{-1}$时，可以对求解所用模型的次数进行不同的选取，分别得出整体模型的 g 和 NMSE、EVM 的值，用来评价预失真补偿的结果.

问题二

题中已给出了某功放有记忆效应的复输入输出测试数据，现需要建立此功放的非线性特性数学模型，拟合出功放的特性函数$G(\cdot)$．根据函数逼近理论，本文直接将功放的输入输出与题目中所提的“和记忆多项式”模型来进行拟合，在使用最小二乘方法求解时，我们对目标函数加了一个正则项，以保证求解的可实现性.

预失真处理器模型的建立与问题一类似，且给出了以框图的方式建立的预失真处理的模型实现示意图.

问题三

问题二中所给的输入输出数据是离散的、有限的，在这种情况下计算功率谱密度的函数可以用自相关函数法或对随机过程$\{x(t)\}$的样本函数作傅立叶变换得到，文中采取第二种方法来求解.

五、模型建立与求解

§5.1 问题一的模型与求解

§5.1.1 无记忆功放的特性函数$G(\cdot)$模型建立

文章中已给出某功放无记忆效应的复输入输出测试数据，这些数据是对功放输入$x(t)$/输出$z(t)$进行离散采样后得到的，它们的值为分别为$x(n)$/$z(n)$(采样过程符合 Nyquist 采样定理要求).

对于问题一，根据文章中所给的某功放无记忆效应的复输入输出测试数据，首先需要建立此功放的非线性特性数学模型，拟合出功放的特性函数$G(\cdot)$．根据函数逼近理论，可以采用

1、多项式的形式

2、多项式的变形的形式

3、空间中的一组正交函数基的线性组合来表示

4、正则化下，空间中的一组正交函数基的线性组合来表示

下面将这些情况都进行建模，来拟合功放的特性函数$G(\cdot)$，并在最后进行比较选择优者.

所求得的模型的数值计算结果业界常用 NMSE、EVM 等参数评价其准确度, NMSE 的具体定义如下. 采用归一化均方误差 (Normalized Mean Square Error, NMSE) 来表征计算精度，其表达式为

$$\mathrm{NMSE}=10\log_{10}\frac{\sum_{n=1}^{N}|z(n)-\hat{z}(n)|^2}{\sum_{n=1}^{N}|z(n)|^2}\ . \tag{5.1.1}$$

如果用z表示实际信号值，$\hat{z}$表示通过模型计算的信号值，NMSE 就反映了模型与实际模块的接近程度. 显然 NMSE 的值越小，模型的数值计算结果就越准确.

误差矢量幅度 (Error Vector Magnitude, EVM)定义为误差矢量信号平均功率的均方根和参照信号平均功率的均方根的比值，以百分数形式表示. 如果用x表示理想的信号输出值，e表示理想输出与整体模型输出信号的误差，可用EVM衡量整体模型对信号的幅度失真程度:

$$\mathrm{EVM}=\sqrt{\frac{E[|e^2|]}{E[|X^2|]}}\times 100\%\ . \tag{5.1.2}$$

模型一 多项式的形式

首先根据函数逼近的 Weierstrass 定理，对解析函数采用简单的多项式来表示，可表示为

$$z(t)=\sum_{k=1}^{K}h_k x^k(t)\,. \tag{5.1.3}$$

因为此时是要将观测数据与形式已经固定的函数(5.1.3)进行拟合，而目的是求解该函数的各项系数，所以该问题其实就是最简单的线性最小二乘问题.

模型建立

$$\min_{h\in C^n}\sum_{n=1}^{N}\left\|z(n)-\sum_{k=1}^{K}h_k x^k(n)\right\|^2, \tag{5.1.4}$$

其中，$x(n)$和$z(n)$为文章中所给的输入和输出测试数据，这些数据是对功放输入$x(t)$、输出$z(t)$进行离散采样后得到的(采样过程符合 Nyquist 采样定理要求), N 为功放输入输出数据的总个数.

将问题(5.1.4)与(3.1.1)进行对应，由(3.1.3)可以直接得到系数的表达式为

$$h=\left(\bar{A}^T A\right)^{-1}\bar{A}^T z$$

其中

$$A=\begin{bmatrix} x(1) & x^2(1) & x^3(1) & \cdots & x^K(1)\\ x(2) & x^2(2) & x^3(2) & \cdots & x^K(2)\\ & & \vdots & & \\ x(N) & x^2(N) & x^3(N) & \cdots & x^K(N)\end{bmatrix},$$

$$h=\left(h_1,h_2,\ldots,h_K\right)^T,$$

$$z=\left(z(1),z(2),\ldots,z(N)\right)^T.$$

结果

当$K=3$时, (见附录 2.1.1)该表达式中的系数为

$$h_1 = 2.90853227839969 - 0.06065388325890i$$
$$h_2 = 0.21377599831493 - 0.43417026083854i\,.$$
$$h_3 = 0.19818563766673 + 0.27826757408010i$$

根据模型一以及(5.1.1)式, 可以求出 NMSE 的值如下：

$$\text{NMSE} = -13.4414169873254 \qquad (k=3)\,.$$

当$k=5$时, (见附录 2.1.2)表达式中的系数为

$$h_1 = 2.908037719327826 - 0.063527494375989\text{i}$$
$$h_2 = 0.343519806629302 - 0.388942747664566\text{i}$$
$$h_3 = 0.541211413428411 - 0.144422960285135\text{i}$$
$$h_4 = -0.399744749427209 - 0.558463329513045\text{i}$$
$$h_5 = -0.271952185146638 + 0.120559140060622\text{i}$$

根据模型一以及(5.1.1)式, 可以求出 NMSE 的值如下：

$$\text{NMSE} = -21.544782705381238 \qquad (k=5)\,.$$

模型二 多项式的变形

同时我们也考虑了多项式变形[4]的情形来对其进行表示, 其表示式为

$$z(t) = \sum_{k=1}^{K} h_k \left|x(t)\right|^{k-1} x(t)\,. \tag{5.1.5}$$

因为此时是要将观测数据与形式已经固定的函数(5.1.5)进行拟合, 而目的是求解该函数的各项系数, 所以该问题其实就是最简单的线性最小二乘问题.

模型建立

$$\min_{h\in C^n} \sum_{n=1}^{N} \left\| z(n) - \sum_{k=1}^{K} h_k \left|x(n)\right|^{k-1} x(n) \right\|^2 \tag{5.1.6}$$

其中 N 为所给功放输入输出数据的总个数, K 为表达式的次数. 将问题(5.1.6)与(3.1.1)进行对应, 由(3.1.3)可以直接得到系数的表达式为

$$h = \left(\overline{A}^T A\right)^{-1} \overline{A}^T z$$

其中

$$A = \begin{bmatrix} x(1) & |x(1)|x(1) & |x(1)|^2 x(1) & \cdots & |x(1)|^{K-1} x(1) \\ x(2) & |x(2)|x(2) & |x(2)|^2 x(2) & \cdots & |x(2)|^{K-1} x(2) \\ & & \vdots & & \\ x(N) & |x(N)|x(N) & |x(N)|^2 x(N) & \cdots & |x(N)|^{K-1} x(N) \end{bmatrix},$$

$$h = \left(h_1, h_2, h_3, \ldots, h_K\right)^T,$$

$$z = \left(z(1), z(2), z(3), \ldots, z(N)\right)^T.$$

分别考虑当$k=3$，$k=5$时，该表达式的具体形式(即确定表达式的系数).

结果

当$k=3$时, (见附录 2.1.3)表达式中的系数为

$$h_1 = 3.05118300539204 - 0.00000000000001i$$
$$h_2 = 0.00607190339398 + 0.00000000000005i \quad .$$
$$h_3 = -1.17015941262647 - 0.00000000000004i$$

根据上面所建立的模型以及(5.1.1)式，可以求出 NMSE 的值如下：

$$\mathrm{NMSE} = -29.7446547565428 \qquad (k = 3).$$

当 $k = 5$ 时, (见附录 2.1.4)表达式中的系数为

$$h_1 = 2.96798359725102 - 0.00000000000080i$$
$$h_2 = 0.30993164419760 + 0.00000000000873i$$
$$h_3 = -0.15366463690519 - 0.00000000002804i \ .$$
$$h_4 = -3.42450044595425 + 0.00000000003458i$$
$$h_5 = 2.20821239548647 - 0.00000000001446i$$

根据上面所建立的模型以及(5.1.1)式，可以求出 NMSE 的值如下：

$$\mathrm{NMSE} = -45.379717608769994 \qquad (k = 5)$$

模型三 空间中的一组正交函数基的线性组合

最后根据函数逼近理论，可采用空间中的一组正交函数基[4]的线性组合来表示该特性函数(参考文献 3 中的方法)，其表达式为

$$z(t) = \Psi h \ , \tag{5.1.7}$$

其中正交矩阵

$$\Psi = [\psi_1(x) \ \psi_2(x) \ \cdots \ \psi_k(x)],$$
$$\psi_k(x) = \sum_{l=1}^{k} (-1)^{l+k} \frac{(k+l)!}{(l-1)!(l+1)!(k-l)!} |x|^{l-1} x \ .$$

因为此时是要将观测数据与形式已经固定的函数(5.1.7)进行拟合，而目的是求解该函数的各项系数，所以该问题其实就是最简单的线性最小二乘问题.

模型建立

$$\min_{h \in C^n} \| z - \Psi h \|^2 \tag{5.1.8}$$

其中

$$h = (h_1, h_2, h_3, \ldots, h_K)^T \ ,$$
$$z = (z(1), z(2), z(3), \ldots, z(N))^T \ ,$$
$$\Psi = [\psi_1(x(n)) \ \psi_2(x(n)) \cdots \psi_k(x(n))],$$
$$\psi_k(x(n)) = \sum_{l=1}^{k} (-1)^{l+k} \frac{(k+l)!}{(l-1)!(l+1)!(k-l)!} |x(n)|^{l-1} x(n),$$

N 为功放的输入输出数据的总个数. 将问题(5.1.8)与(3.1.1)进行对应，由(3.1.3)可以直接得到系数的表达式为

$$h = (\overline{\Psi}^T \Psi)^{-1} \overline{\Psi}^T z \ .$$

由于计算量较大，我们选取 $k = 7$ 来进行拟合，得出表达式中的系数.

结果(见附录 2.1.5)

当 $k = 7$ 时，表达式中的系数为

$$h_1 = 3.287412936081622 - 7.322701472967097e-015i$$
$$h_2 = -0.091488124421954 - 2.16460963736731e-015i$$
$$h_3 = -0.066219774105875 + 5.035305939565804e-016i$$
$$h_4 = 0.038056322596937 + 2.726632938529483e-016i \ .$$
$$h_5 = 0.01014165858755 - 1.258894247527231e-016i$$
$$h_6 = -0.005283612035716 - 2.653720342429833e-016i$$
$$h_7 = -0.001265433154276 - 1.923256069376669e-016i$$

根据上面所建立的模型以及(5.1.1)式，可以求出 NMSE 的值如下：

$$\text{NMSE} = -60.5675309366592 \qquad (k = 7)$$

模型四 模型三正则化

模型建立

对于模型三，由于所给的数据较多，很难避免本文 3.2 节中所提到的 $\bar{\Psi}^T\Psi$ 奇异的情况，故对(5.1.8)再进行一个 Tikhonov 正则化. 即对(5.1.8)加一个正则项

$$\lambda_k \left\| h - h_k \right\|^2 .$$

问题转变为

$$h_{k+1} = \min_{h \in C^{(K\cdot M)\times 1}} \left\| z - \Psi h \right\|^2 + \lambda_k \left\| h - h_k \right\|^2 . \tag{5.1.9}$$

其中 h_k 是第 k 步迭代得到的解(计算机运行求解时是要给其赋一个初始值的)，而 λ_k 可以选为一个常数或一个单调下降趋于 0 的数列. 而迭代的终止准则为

$$\left\| h_{k+1} - h_k \right\| \le \varepsilon ,$$

其中 ε 是一个给定的误差上界.

考虑到二次凸函数的稳定点即为最小值点，问题(5.1.9)是可以直接求解的，得到 h 的求解公式为

$$h = \left(\bar{\Psi}^T\Psi + \lambda_k I \right)^{-1} \left(\bar{\Psi}^T z(n) + \lambda_k h_k \right). \tag{5.1.10}$$

此处，我们仍选取 $k = 7$ 来进行拟合，其中一些参数选取为

$$h_0 = 1 + 1i, \ \lambda_0 = 1, \ \lambda_{k+1} = 0.8\lambda_k, \ \varepsilon = 10^{-8} .$$

则可得出表达式(5.1.7)中的系数.

结果(见附录 2.1.6)

$$h_1 = 3.287399414051528 + 0.000008426827987i$$
$$h_2 = -0.091492245311883 + 0.000002568107767i$$
$$h_3 = -0.066218825186175 - 0.00000059135966i$$
$$h_4 = 0.038056824724197 - 0.000000312921951i$$
$$h_5 = 0.01014141261644 + 0.000000153287355i$$
$$h_6 = -0.005283977515731 + 0.000000227764411i$$
$$h_7 = -0.001265568675997 + 0.000000084456122i$$

根据上面所建立的模型以及(5.1.1)式，可以求出 NMSE 的值如下：

$$\text{NMSE} = -68.6293523598994 \qquad (k = 7)$$

模型一~模型四的总评价

对四种模型下参数 NMSE 的大小进行比较发现，当选用一组正交函数基，并运用正则化后的最小二乘方法来对功放特性函数进行拟合时(即模型四)，NMSE 的值是最小的. 也就是说

$$\frac{\sum_{n=1}^{N}|z(n)-\hat{z}(n)|^2}{\sum_{n=1}^{N}|z(n)|^2}$$

在模型四下是最靠近 0 的，故模型四是逼近效果最好的.

但模型四的计算复杂度是很大，由所得的NMSE参数可发现模型二的计算精度也是不错的，但其计算的复杂度比模型四要小很多，故选择模型二来求解功放特性函数. 且在下面的无记忆功放模型的预失真处理建模中，功放特性函数是由模型二得出的.

§5.1.2 四种模型的输入输出幅度比较图与评价

下面将实际的与拟合的复输入输出幅度值进行作图，以便更直观的看出模型的逼近效果.

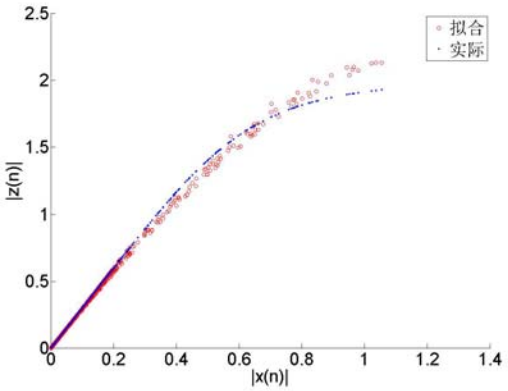


图 5.1 模型一 k=3 实际与拟合功放输入/输出幅度散点图

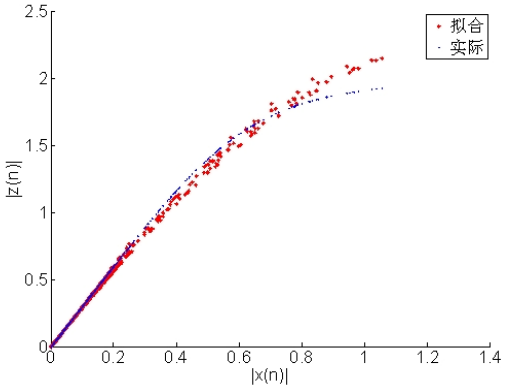


图 5.1 模型一 k=5 实际与拟合功放输入/输出幅度散点图

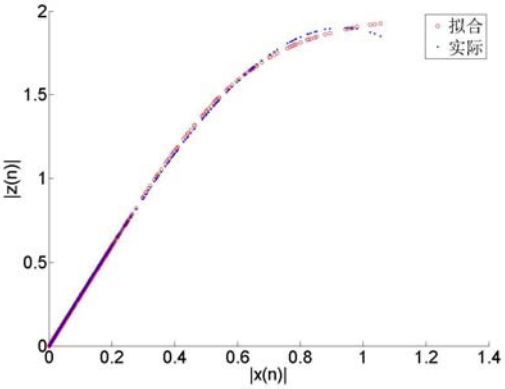


图 5.3 模型二 k=3 实际与拟合功放输入/输出幅度散点图

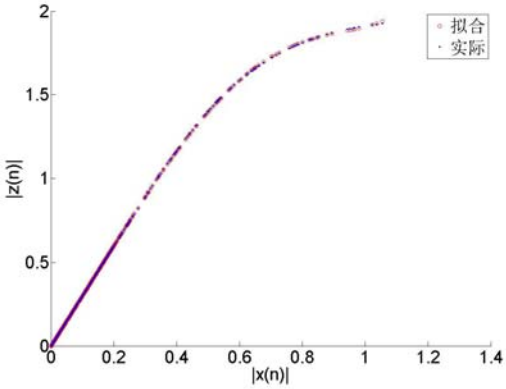


图 5.4 模型二 k=5 实际与拟合功放输入/输出幅度散点图

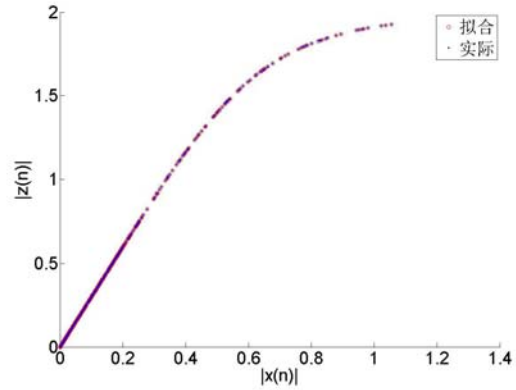


图 5.5 模型三实际与拟合的功放输入/输出幅度散点图

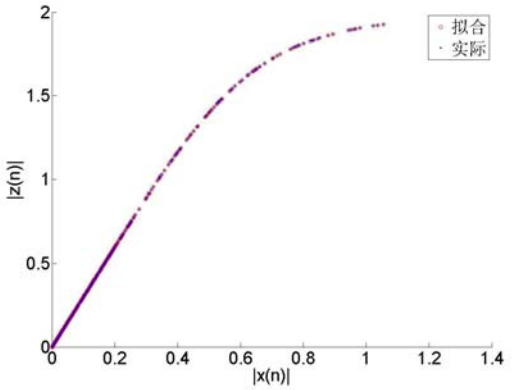


图 5.6 模型四实际与拟合的功放输入/输出幅度散点图

根据观察比较发现，当用正交的函数基或对其实行一个正则化(即模型三和模型四)，来对功放特性函数进行拟合的时候，拟合情形的输入输出幅度散点图与实际的输入输出幅度散点图的逼近效果是最佳的.

同时可观察到但模型二中的次数 $k=5$ 时，其散点图的逼近效果也是很好的.

§5.1.3 预失真处理模型建立

上面已经求得功放特性函数 $G(\cdot)$ 的多种模拟，现采用 Hammerstein 模型，即

$$z(n)=\sum_{k=1}^{K} b_k \left|x(n)\right|^{k-1} x(n)$$

来建立功放模型(即此处的功放特性函数是由模型二得出的). 下面建模的总体原则是使预失真和功放的联合模型呈线性后误差最小. 在此模型中，有两个约束需要考虑：

(1)输出幅度限制：即模型中的预失真处理的输出幅度不大于给出的功放输入幅度最大值.

(2)功率最大化：即模型的建立必需考虑尽可能使功放的信号平均输出功率最大，因此预失真处理后的输出幅度需尽可能提高.

用参数 NMSE、EVM 来评价模型的数值计算结果的准确度.

模型建立

由题知，目标误差函数为

$$\min \left\| g\cdot x - G\circ F(x) \right\|,$$

且满足上面提出的约束. 则得出优化问题为

$$\begin{aligned} &\min \left\| g\cdot x - G\circ F(x) \right\| \\ &s.t \quad \left\| F(x) \right\|_{\infty} \le \left\| x \right\|_{\infty} \\ &\qquad F(x)\text{尽可能大} \end{aligned} \tag{5.1.11}$$

其中

$$x=\left(x(1),x(2),x(3),\ldots,x(N)\right)^{T},$$

其中 N 为输入输出数据的总个数.

模型分析

在预失真模型(5.1.11)中，目标函数是使误差函数极小. 在分析这个优化问题时,可

以先假定目标函数能够取到最小，即

$$g \cdot x = G \circ F(x)$$

$$x = \frac{G}{g} \circ F(x)$$

其中 G 为多项式，g 为常数，所以可以求得

$$F(x) = \left(\frac{G}{g}\right)^{-1} \cdot x .$$

由于我们已经求得 G，且有

$$\frac{G(x)}{g} = \frac{z}{g},$$

其反函数的模型与功放模型类似，即

$$x(n) = \sum_{k=1}^{K} c_k \left|\frac{z(n)}{g}\right|^{k-1} \left|\frac{z(n)}{g}\right| \tag{5.1.12}$$

其中 $x(n)$, $z(n)$ 为已给的输入输出数据，则可以求得映射 $\left(\frac{G}{g}\right)^{-1}$，记作

$$H : \frac{z}{g} \to x ,$$

那么

$$y = H(x) .$$

另外，在模型(5.1.11)的约束条件中，要使 $F(x)$ 尽可能大，即目标函数中 g 尽可能大，也就是说使得 $\|F(x)\|_\infty$ 尽可能靠近 $\|x\|_\infty$，那么约束条件都可归为

$$0 \le \|x\|_\infty - \|F(x)\|_\infty \le e .$$

其中 e 为尽可能小的正常数. 下面我们将给出解决该优化问题的算法：

算法 5.1

初始化： $j = 2$， $g_2 = 1.1$，给定一个判断容限 e；

Step1：由(5.1.12)， $x(n) = \sum_{k=1}^{K} c_k \left|\frac{z(n)}{g_j}\right|^{k-1} \left|\frac{z(n)}{g_j}\right|$. 拟合函数 $H : \frac{z}{g} \to x$；

Step2：求 $y = H(x)$；

Step3：比较 $\|y\|_\infty$， $\|x\|_\infty$

若 $\|y\|_\infty \le \|x\|_\infty$，则判断

a1. 若 $\|x\|_\infty - \|y\|_\infty \ge e$，则 $g_{j+1} = g_j + \frac{1}{j}$, $j = j+1$; 返回 step1；

a2. 否则 stop，输出 g_j；

若 $\|y\|_\infty > \|x\|_\infty$，则 $g_{j+1} = g_j - \frac{1}{j}$, $j = j+1$; 返回 step1:；

结果与评价(见附录 2.1.7、2.1.8、2.1.9)

选定 $G(\cdot): z(n)=\sum_{k=1}^{K} b_k \left|x(n)\right|^{k-1} x(n)$ 的阶数 $K=5$，通过上面的算法可以得到当 F 取不同阶数的情况下， g, NMSE, EVM 的结果及图像

表 5.1　F 取不同阶数情况下 g, NMSE, EVM 的结果

F 的阶数 K	g	NMSE	EVM
4	1.86932497973065	-32.5819077399852	2.34911681195961%
5	1.84730161996524	-37.1398119663279	1.38998272147897%
7	1.83264461869445	-46.0624143395044	0.497598752653887%

由表 5.1 的结果可以看出当 F 的阶数越高时，得到的 g 的值越小(说明线性化后的幅度放大倍数越小)， NMSE、EVM 的值越小(说明模型的计算精度越高，整体模型对信号的幅度失真程度越小).

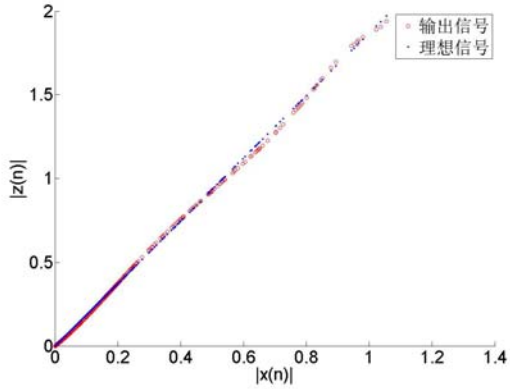


图 5.7 理想信号与所建模型得到的输出信号对比(K=4)

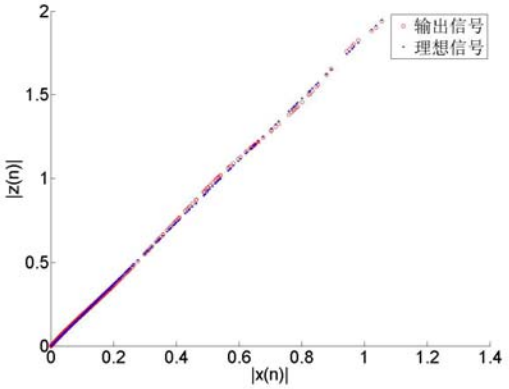


图 5.8 理想信号与所建模型得到的输出信号对比(K=5)

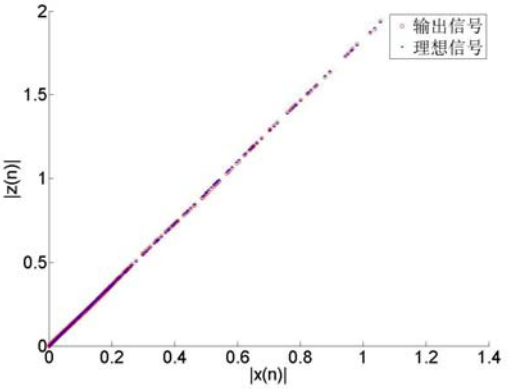


图 5.9 理想信号与所建模型得到的输出信号对比(K=7)

根据观察发现，当 K 的取值越大时，所建模型的输入输出幅度散点图与理想的输入输出幅度散点图的逼近效果越好.

§5.2 问题二的模型与求解

§5.2.1 有记忆功放的特性函数 $G(\cdot)$ 模型建立

对于问题二，根据文章中所给的某功放有记忆效应的复输入输出测试数据，首先需要建立此功放的非线性特性数学模型，拟合出功放的特性函数 $G(\cdot)$．此时功放不仅与此时刻输入有关，而且与此前某一时间段的输入有关，其可以写为

$$z(n)=\sum_{k=1}^{K}\sum_{m=0}^{M}h_{km}x^{k}(n-m)=h_{10}x(n)+h_{11}x(n-1)+\cdots+h_{1M}x(n-M)$$
$$+h_{20}x^{2}(n)+h_{21}x^{2}(n-1)+\cdots+h_{2M}x^{2}(n-M)\quad,\ n=0,1,2,\cdots,N.$$
$$+\cdots$$
$$+h_{K0}x^{K}(n)+h_{K1}x^{K}(n-1)+\cdots+h_{KM}x^{K}(n-M)$$

式中 M 表示记忆深度，h_{km} 为系数. 具有记忆效应的功放模型也可以用更一般的 Volterra 级数[5][6]表示，由于 Volterra 级数太复杂，简化模型有 Wiener、Hammersteint 等[4][7]. 由于常用复值输入-输出信号，上式也可表示为便于计算的“和记忆多项式”模型

$$z(n)=\sum_{k=1}^{K}\sum_{m=0}^{M}h_{km}x(n-m)\,|\,x(n-m)\,|^{k-1}\qquad n=0,1,2,\cdots,N \tag{5.2.1}$$

模型建立

本文采用“和记忆多项式”模型(5.2.1)式来进行拟合. 我们用最小二乘法来求解，由于本问中所给的输入输出的数据个数非常大，故现在选取其中的一部分来进行拟合，求得功放过程的模型. 我们选取输入输出数据的次数 n 为 $M+1$ 的倍数的数据来进行拟合，最小二乘公式即为

$$\min_{h\in C^{(K\times M)\times 1}}\ \sum_{\substack{(M+1)\mid n\\ n\le N}}\left\|z(n)-\sum_{k=1}^{K}\sum_{m=0}^{M}h_{km}x(n-m)\,|\,x(n-m)\,|^{k-1}\right\|^{2} \tag{5.2.2}$$

其中 N 是指所有的功放的输入数据总个数，K 表示所选模型的最高次数，M 表示记忆深度(本文在求解模型时是事先给定的)，$x(n)$ 是第 n 个复输入值，$z(n)$ 是第 n 个复输出值，h_{km} 为系数，$h=\left(h_{10},h_{20},\ldots,h_{K0},h_{12},h_{22},\ldots,h_{K2},\ldots\ \ldots,h_{1M},h_{2M},\ldots,h_{KM}\right)^{T}$.

由于所给的数据较多，即便是选取了部分数据进行拟合,但仍很难避免 3.2 节中所提到的 $\bar{A}^{T}A$ 奇异的情况，故对(5.2.2)再进行一个 Tikhonov 正则化. 即对(5.2.2)加一个正则项

$$\lambda_k\left\|h-h_k\right\|^{2},$$

则问题转变为

$$h_{k+1}=\min_{h\in C^{(K\times M)\times 1}}\ \sum_{\substack{(M+1)\mid n\\ n\le N}}\left\|z(n)-\sum_{k=1}^{K}\sum_{m=0}^{M}h_{km}x(n-m)\,|\,x(n-m)\,|^{k-1}\right\|^{2}+\lambda_k\left\|h-h_k\right\|^{2} \tag{5.2.3}$$

其中 h_k 是第 k 步迭代得到的解，而 λ_k 可以选为一个常数或一个单调下降趋于 0 的数列. 而迭代的终止准则为

$$\left\|h_{k+1}-h_k\right\|\le\varepsilon,$$

其中 ε 是一个给定的误差上界.

当给定一个记忆深度 M 后，我们可以将问题(5.2.3)化成如下形式的问题，即

$$\min_{h\in C^{n}}\left\|z(n)-Ah\right\|^{2}+\lambda_k\left\|h-h_k\right\|^{2} \tag{5.2.4}$$

其中 A 是一个 $\left(N/(M+1)\right)\times\left(K\cdot(M+1)\right)$ 的复矩阵，即

$$A=\begin{bmatrix} x(M+1) & x(M+1)|x(M+1)| & \cdots & x(M+1)|x(M+1)|^{K-1} & \cdots & x(1) & \cdots & x(1)|x(1)|^{K-1} \\ x(2M+2) & x(2M+2)|x(2M+2)| & \cdots & x(2M+2)|x(2M+2)|^{K-1} & \cdots & x(M+2) & \cdots & x(1)|x(1)|^{K-1} \\ & & & \vdots & & & & \\ & & & \vdots & & & & \end{bmatrix}$$

而

$$h=\left(h_{10},h_{20},\ldots,h_{K0},h_{11},h_{21},\ldots,h_{K1},\ldots\ \ldots,h_{1M},h_{2M},\ldots,h_{KM}\right)^T.$$

考虑到二次凸函数的稳定点即为最小值点，问题(5.2.4)是可以直接求解的，h 的求解公式为

$$h=\left(\overline{A}^T A+\lambda_k I\right)^{-1}\left(\overline{A}^T z(n)+\lambda_k h_k\right). \tag{5.2.5}$$

本题中已给出有记忆功放输入输出数据的总个数为 $N=73920$，并分别取 $M=7,\ K=5,\ \varepsilon=10^{-8}$ 和 $M=3,\ K=5,\ \varepsilon=10^{-8}$ 这两种情况．这样就可以根据(5.2.5)求得 h．

结果(见附录 2.2.1、2.2.2)

当 $M=7, K=5$ 时，由于系数共有 40 个，即 h 是一个 40×1 的大向量，故将该结果放到附录中．再根据上面所建立的模型及(5.1.1)式，求出该模型的 NMSE 值如下：

NMSE = -45.839408840847　　$M=7, K=5$.

当 $M=3, K=5$ 时，由于系数共有 20 个，即 h 是一个 20×1 的大向量，故将该结果放到附录中．再根据上面所建立的模型及(5.1.1)式，求出该模型的 NMSE 值如下：

NMSE = –44.5315001961471　　$M=3, K=5$.

§5.2.2 有记忆功放模型的输入输出幅度图

下面将实际与拟合的复输入输出幅度进行作图，以便更直观的看出模型的逼近效果.

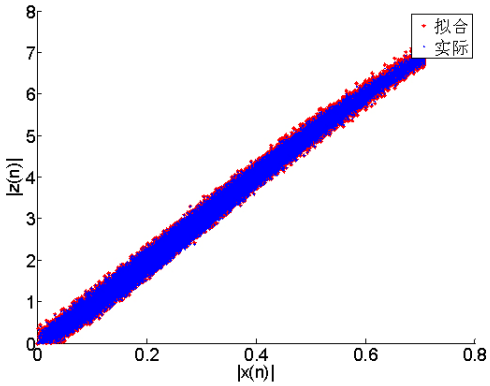


图 5.10　M=7 实际与拟合功放输入/输出幅度散点图

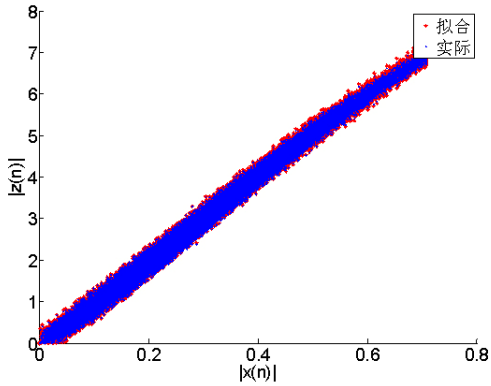


图 5.11　M=3 实际与拟合功放输入/输出幅度散点图

总评价

根据观察比较发现，尽管在用“和记忆多项式”模型进行拟合时，我们只选取了一部分输入输出测量数据进行模型的建构．但通过对上面两图的观察，当对所有的输入测量数据进行作图时，可发现拟合得到的输入输出幅度散点图与实际的输入输出幅度散点图

的逼近效果还是很好的.

§5.2.3 预失真处理模型建立

上面已求得功放特性函数$G(\cdot)$的模型，采用“和记忆多项式”模型

$$z(n)=\sum_{k=1}^{K}\sum_{m=0}^{M}h_{km}x(n-m)\,|\,x(n-m)\,|^{k-1}$$

建立的功放模型. 下面建模的总体原则是使预失真和功放的联合模型呈线性后误差最小. 在此模型中，有两个约束需要考虑：

(1)输出幅度限制：即模型中的预失真处理的输出幅度不大于给出的功放输入幅度最大值.

(2)功率最大化：即模型的建立必需考虑尽可能使功放的信号平均输出功率最大，因此预失真处理后的输出幅度需尽可能提高.

模型的数值计算结果用 NMSE、EVM 参数来评价其准确度.

模型建立

$$\begin{aligned}&\min\left\|g\cdot x-G\circ F(x)\right\|\\&s.t\quad \left\|F(x)\right\|_{\infty}\le\left\|x\right\|_{\infty}\\&\qquad F(x)\text{尽可能大}\end{aligned}\tag{5.2.6}$$

模型分析

在预失真模型(5.1.11)中，目标函数是使误差函数极小. 在分析这个优化问题时，可以先假定目标函数能够取到最小，即

$$g\cdot x=G\circ F(x)$$

$$x=\frac{G}{g}\circ F(x)$$

其中 G 为多项式，g 为常数，所以可以求得

$$F(x)=\left(\frac{G}{g}\right)^{-1}\cdot x$$

由于我们已经求得 G，且有

$$\frac{G(x)}{g}=\frac{z}{g},$$

其反函数的模型与功放模型类似，即为

$$x(n)=\sum_{k=1}^{K}\sum_{m=0}^{M}l_{km}\left|\frac{z(n-m)}{g}\right|^{k-1}\left|\frac{z(n-m)}{g}\right|\tag{5.2.7}$$

其中 $x(n)$, $z(n)$ 为已给的输入输出数据，则可以求得映射$\left(\frac{G}{g}\right)^{-1}$,记作

$$H:\frac{z}{g}\to x,$$

那么

$$y=H(x).$$

另外，在模型(5.2.6)的约束条件中，要使$F(x)$尽可能大，即目标函数中 g 尽可能大，也就是说使得$\left\|F(x)\right\|_{\infty}$尽可能靠近$\left\|x\right\|_{\infty}$，那么约束条件都可归为

$$0 \le \|x\|_\infty - \|F(x)\|_\infty \le e.$$

其中 e 为一个尽可能小的正常数.

模型实现示意图

我们以框图的方式给出预失真处理的模型实现示意图：

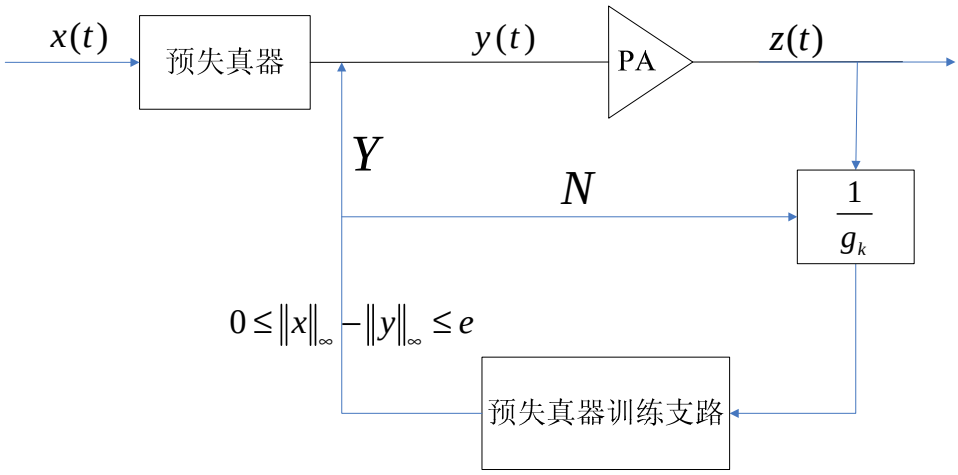


图5.12 预失真处理模型实现示意图

下面我们将给出解决该优化问题的算法：

算法 5.2

初始化： $j=2$， $g_2=1.1$，给定判断容限 e；

Step1：由(5.2.7)， $x(n)=\sum_{k=1}^{K}\sum_{m=0}^{M} l_{km}\left|\frac{z(n-m)}{g_j}\right|^{k-1}\frac{z(n-m)}{g_j}$. 拟合函数 $H:\frac{z}{g}\to x$；

Step2：求 $y=H(x)$；

Step3：比较 $\|y\|_\infty$， $\|x\|_\infty$

若 $\|y\|_\infty \le \|x\|_\infty$，则判断

a1. 若 $\|x\|_\infty - \|y\|_\infty \ge e$，则 $g_{j+1}=g_j+\frac{1}{j}$, $j=j+1$; 返回 step1；

a2. 否则 stop，输出 g_j；

若 $\|y\|_\infty > \|x\|_\infty$，则 $g_{j+1}=g_j-\frac{1}{j}$, $j=j+1$; 返回 step1；

结果与评价(见附录 2.2.3)

选定 $G(\cdot)$: $z(n)=\sum_{k=1}^{K}\sum_{m=0}^{M} h_{km}x(n-m)\,|x(n-m)|^{k-1}$ 的阶数为 $K=5$. 因数据量很大且算法较复杂，本文对 F 进行多次计算，发现当阶数为 $K=5$ 的时候与更高阶相比，效果就已经很好了，故下面只给出阶数为 $K=5$ 时 g, NMSE, EVM 的结果.

本文取定记忆深度为 $M=3$，现根据算法 5.2 可求得

$$g=9.490829228013789,$$

由于系数一共有 20 个，即 h 是一个 20×1 的向量，故将此结果放到附录中.

根据上面所建模型以及(5.1.1)、(5.1.2)式，可求出该模型的 NMSE、EVM 值如下：.

NMSE = -37.836849855461956

EVM = 0.012827957346961

$M=3, K=5$

由所得数据，可以发现在该算法下，得到的 g 的值比较大(说明线性化后的幅度放大倍数大), NMSE、EVM 的值较小(说明模型的计算精度越高，整体模型对信号的幅度失真程度越小).

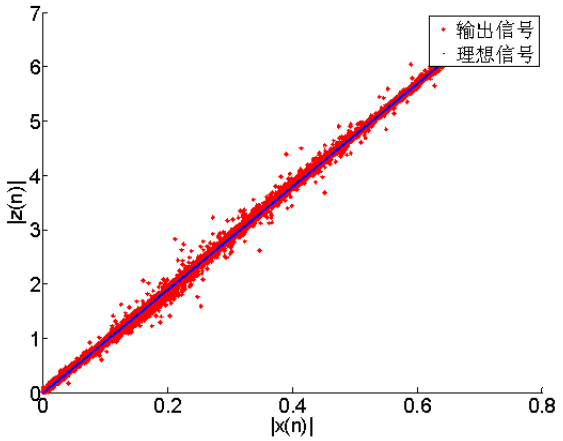


图 5.13　M=3, K=5 实际与拟合功放输入/输出幅度散点图

观察图 5.13 发现，该情况下所建模型的输入输出幅度散点图与理想的输入输出幅度散点图逼近效果还是较好的. 故该模型是可行的.

§5.3 问题三的模型与求解

§5.3.1 背景知识

功率谱的概念是针对功率有限信号的，所表现的是单位频带内信号功率随频率的变化情况. 保留了频谱的幅度信息，但是丢掉了相位信息，所以频谱不同的信号其功率谱是可能相同的. 功率谱是随机过程的统计平均概念，平稳随机过程的功率谱是一个确定函数；而频谱是随机过程样本的 Fourier 变换，对于一个随机过程而言，频谱也是一个“随机过程”(随机的频域序列).

功率谱密度(PSD)，它定义了信号或者时间序列的功率如何随频率分布. 这里功率可能是实际物理上的功率，或者更经常便于表示抽象的信号，被定义为信号数值的平方，也就是当信号的负载为 1 欧姆(ohm)时的实际功率.

由于平均值不为零的信号不是平方可积的，所以在这种情况下就没有傅立叶变换. 维纳-辛钦定理(Wiener-Khinchin theorem)提供了一个简单的替换方法. 如果信号可以看作是平稳随机过程，那么功率谱密度就是信号自相关函数的傅立叶变换. 信号的功率谱密度当且仅当信号是广义的平稳过程的时候才存在；如果信号不是平稳过程，那么自相关函数一定是两个变量的函数，这样就不存在功率谱密度，但是可以使用类似的技术估计时变谱密度. 随机信号是时域无限信号，不具备可积分条件，因此不能直接进行傅氏变换. 一般用具有统计特性的功率谱来作为谱分析的依据. 功率谱与自相关函数是一个傅氏变换对.

一般的功率谱密度都是针对平稳随机过程的，由于平稳随机过程的样本函数一般不是绝对可积的，因此不能直接对它进行傅立叶分析. 可以有三种办法来重新定义谱密度，

来克服上述困难.

1. 用相关函数的傅立叶变换来定义谱密度；
2. 用随机过程的有限时间傅立叶变换来定义谱密度；
3. 用平稳随机过程的谱分解来定义谱密度.

§5.3.2 模型建立

计算功率谱密度函数通常有两种方法[8]. 一种叫做标准的自相关函数法, 其表达式为：

$$G_x^{(1)}(f)=4\int_0^{\infty}R_x(\tau)\cos 2\pi f\tau \mathrm{d}\tau \tag{5.3.1}$$

其中 $R_x(\tau)$ 表示某个各态历经的随机过程 $\{x(t)\}$ 的自相关函数；另一种叫做直接法, 即是直接对随机过程 $\{x(t)\}$ 的样本函数作傅立叶变换得到功率谱密度函数, 其表达式为：

$$G_x^{(2)}(f)=\lim_{T\to\infty}\frac{2}{T}\left|\int_0^T x(t)e^{-j2\pi ft}\mathrm{d}t\right|^2 \tag{5.3.2}$$

在计算机上计算功率谱密度函数时, 要求输入的数据必须是离散数值, 所以要对连续观测的数据记录必须做离散化处理. 这叫做数据采样. 离散化的数据值叫做采样数据. 实际计算时, 要求参加运算的采样数据的个数是有限的(即是说, 在有限的时间区段 0-T 上进行计算). 在记录是离散的、有限的情况下, 计算功率谱密度函数的公式可以分别近似地表示为：

$$G_x^{(1)}(f)=2\Delta t\left[R_0+2\sum_{r=1}^{M-1}R_r\cos 2\pi fr\Delta t+R_M\cos 2\pi fM\Delta t\right] \tag{5.3.3}$$

和

$$G_x^{(2)}(f)=\frac{2}{N\Delta t}\left|\Delta t\sum_{i=0}^{N-1}x_i e^{-j2\pi fi\Delta t}\right|^2 \tag{5.3.4}$$

这里, 将(5.3.4)式整理为

$$P(f)=\frac{1}{N}\left|X(f)\right|^2 \tag{5.3.5}$$

其中 $X(f)$ 是 $x(n)$ 的傅里叶变换, 在计算过程中可以直接调用 FFT 函数.

另外由题意可设出, F_{per} 表示每个点上的频率, 其表达式为

$$F_{per}=\frac{F_s}{N}.$$

M 表示每个信道所含的点的个数, 其表达式为

$$M=\frac{F_0}{F_{per}}.$$

其中 F_0 表示每个传输信道上的频率. 故传输信道就只包含 M 个点, 相邻信道也只包含 M 个点.

由于非线性效应产生的新频率分量会对邻道信号有一定的影响, 现用相邻信道功率比(Adjacent Channel Power Ratio, ACPR)表示信道的带外失真的参数, 衡量由于非线性效应所产生的新频率分量对邻道信号的影响程度. 其定义为

$$ACPR = 10\log_{10}\frac{\int_{f_2}^{f_3} s(f)df}{\int_{f_1}^{f_2} s(f)df} \tag{5.3.6}$$

其中 $s(f)$ 为信号的功率谱密度函数，$[f_1, f_2]$ 为传输信道，$[f_2, f_3]$ 为相邻信道. 功率谱密度的计算可通过对信号的自相关函数进行 Fourier 变换计算，也可以通过直接法等计算(假定本题涉及的信号为时间平稳信号).

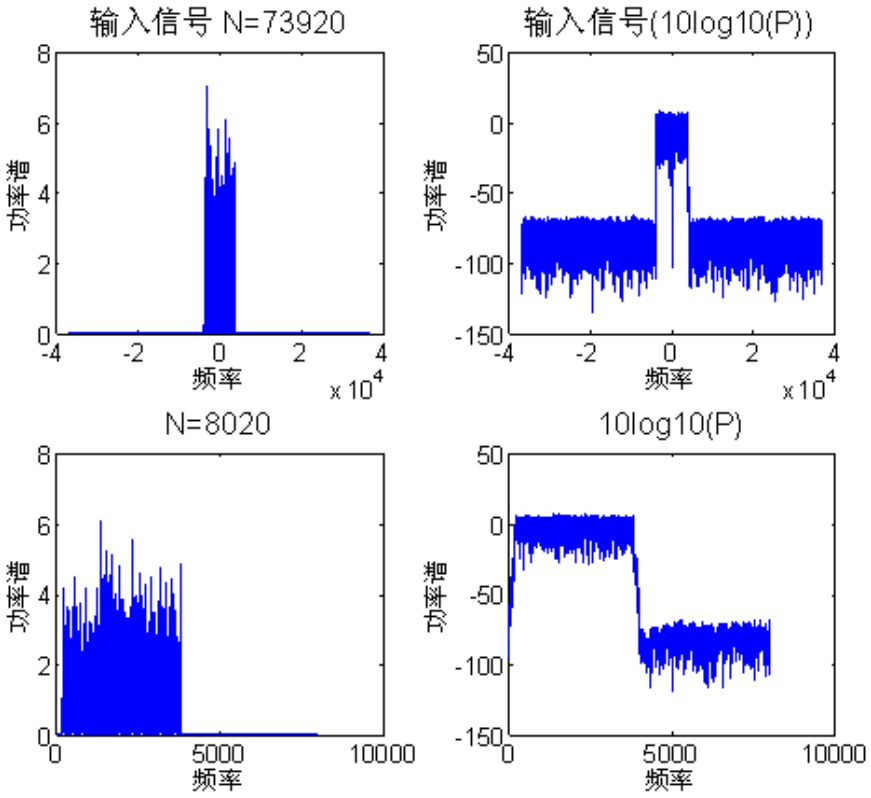
§5.3.3 模型求解

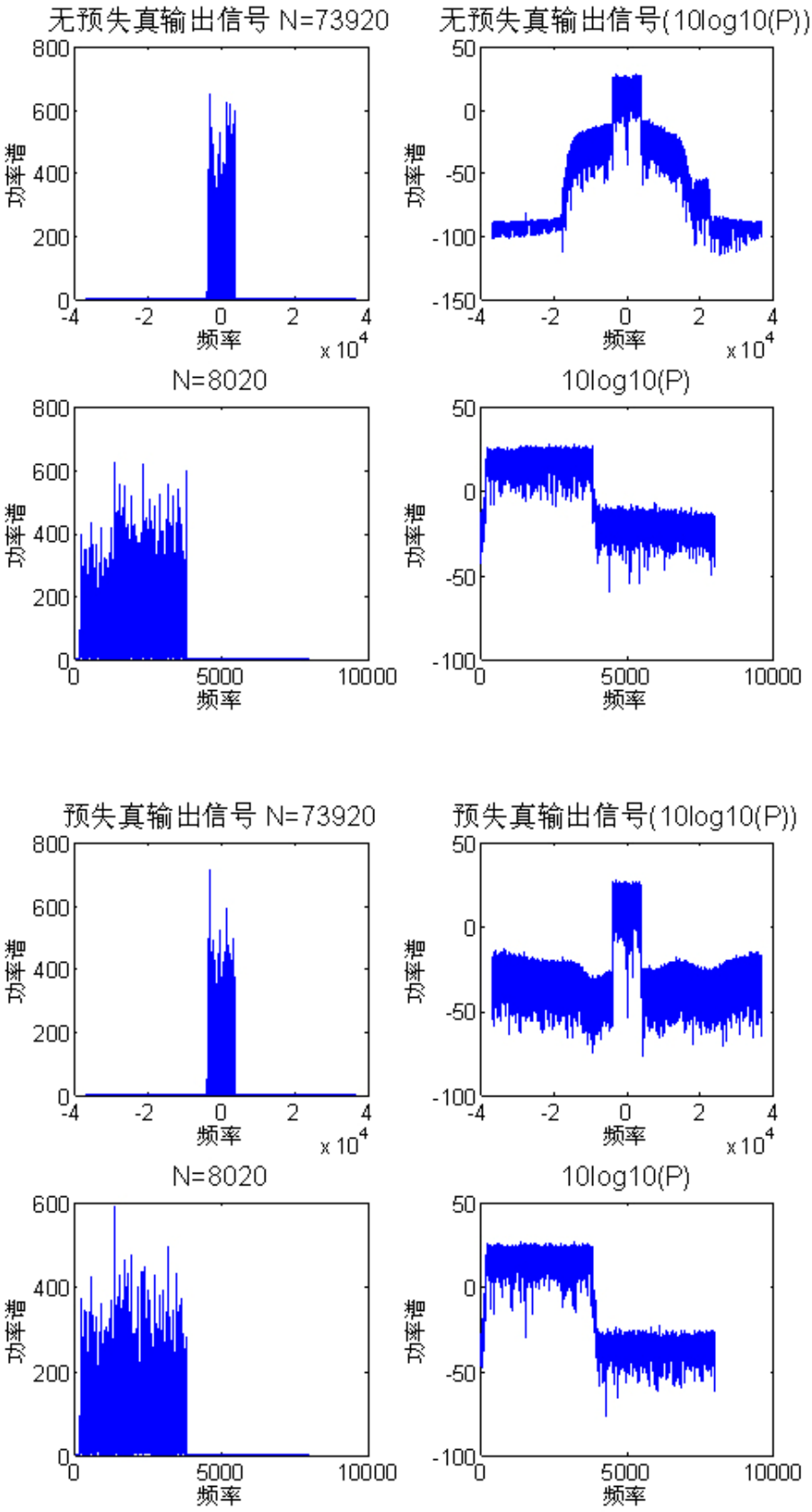
由题目可知数据采样频率 $F_s = 30.72\times 12\text{MHz}$，已给数据个数 $N = 73920$，传输信道 $F_0 = 20\text{MHz}$，则

$$F_{per} = \frac{F_s}{N} = \frac{30.72\times 12\times 10^6}{73920} \approx 4987\text{Hz},$$

$$M = \frac{F_0}{F_{per}} = \frac{20\times 10^6}{4987} \approx 4010.$$

得出输入信号、无预失真补偿的功率放大器输出信号、采用预失真补偿的功率放大器输出信号的功率谱密度图为(见附录 2.3.1)
:





同时也求解出 ACPR 值，如下表:

表 5.2　三种信号的 ACPR 值

信号	ACPR
输入信号	-155.6610424101818
无预失真补偿的功率放大器输出信号	-74.33401486315535
采用预失真补偿的功率放大器输出信号	-104.4904083834696

从表 5.2 中可看出，采用预失真补偿的功率放大器的输出信号效果比无预失真补偿的效果要好.

六、参考文献

[1] 王宜举，修乃华，非线性最优化理论与方法，北京：科学出版社, 2012.1：112-117

[2] 袁亚湘，非线性优化计算方法. 北京：科学出版社, 2008.2:

[3] Francisco Facchinei and Jong-Shi Pang, Finite-Dimensional Variational Inequalities and Complementarity Problems, New York: Springer-Verlag, 2003:1125-1135

[4] Raviv Raich, et al. Orthogonal Polynomials for Power Amplifier Modeling and Predistorter Design.　IEEE Trans. Vehicular technology, 2004, 53(5):1468-1479

[5] John Tsimbinos, Identification and Compensation of Nonlinear Distortion, PhD Dissertation, School of Electronic Engineering, University Of South Australia, Adelaide, February 1995.

[6] Tianhai Wang, et al. Volterra-Mapping-Based Behavioral Modeling of Nonlinear Circuits and Systems for High Frequencies. IEEE Trans. Microwave Theory and Techniques, 2003, 51(5):1433-1440

[7] Dennis R.Morgan et al. A Generalized Memory Polynomial Model for Digital Predistortion of RF Power Amplifiers. IEEE Trans. Signal Processing , 2006, 54(10):3852-3860

[8] 范顺庭，关于功率谱密度函数计算中的几个问题，海洋科学, 2, 44-48, 1980

七、附录

1、文中结果

1.1　文中 5.2.1 有记忆功放特性函数 G 的模型

当 $M=7$, $K=5$ 时，系数为

$$h=\left(h_{10},h_{20},\ldots,h_{K0},h_{11},h_{21},\ldots,h_{K1},\ldots\ \ldots,h_{1M},h_{2M},\ldots,h_{KM}\right)^{T}$$

其中

$$\begin{pmatrix}h_{10}\\h_{20}\\\vdots\\h_{50}\end{pmatrix}=\begin{pmatrix}7.543619044690026\ -0.048615467637612i\\7.346631293043551+6.898899384271611i\\-0.675263689679702-15.799913239635858i\\-33.85640854642059+10.283470901226687i\\33.94284744283753+1.134660159968797i\end{pmatrix}$$

$$\begin{pmatrix} h_{11} \\ h_{21} \\ \vdots \\ h_{51} \end{pmatrix} = \begin{pmatrix} 0.856505298006137 - 1.674198769833794i \\ 3.792606390936379 + 1.595599179635527i \\ -35.12508228425808 - 16.080485382172558i \\ 105.74727685466114 + 42.11489340994745i \\ -97.44217186061887 - 36.43985671826266i \end{pmatrix}$$

$$\begin{pmatrix} h_{12} \\ h_{22} \\ \vdots \\ h_{52} \end{pmatrix} = \begin{pmatrix} -0.188426478481042 - 3.516280113401766i \\ -0.67348101261251 + 1.055345303702035i \\ 38.25500942778418 + 10.879314896601993i \\ -146.49136835026673 - 43.83134379952031i \\ 153.20750367283546 + 46.33228007729713i \end{pmatrix}$$

$$\begin{pmatrix} h_{13} \\ h_{23} \\ \vdots \\ h_{53} \end{pmatrix} = \begin{pmatrix} 0.129137809709903 + 2.216831722151192i \\ 2.437281422023494 + 0.139947836359978i \\ -39.092296632485514 - 10.651910606407894i \\ 163.40350907816693 + 46.186002203218344i \\ -176.1592905457562 - 50.09376954399058i \end{pmatrix}$$

$$\begin{pmatrix} h_{14} \\ h_{24} \\ \vdots \\ h_{54} \end{pmatrix} = \begin{pmatrix} 0.579199959210455 + 4.857324472159618i \\ -3.780952137722379 - 0.986780215642203i \\ 32.55542417653291 + 7.809694024811261i \\ -147.29205412727606 - 40.688839758279606i \\ 152.60989030442204 + 42.81016215924786i \end{pmatrix}$$

$$\begin{pmatrix} h_{15} \\ h_{25} \\ \vdots \\ h_{55} \end{pmatrix} = \begin{pmatrix} -1.921424766255614 + 0.297871281986365i \\ 1.419450380042762 + 0.759472248855102i \\ -17.908896709016993 - 3.319219343465521i \\ 99.67208894993098 + 26.434098603525335i \\ -96.78810593030363 - 27.05810799258672i \end{pmatrix}$$

$$\begin{pmatrix} h_{16} \\ h_{26} \\ \vdots \\ h_{56} \end{pmatrix} = \begin{pmatrix} 1.911123541886597 - 6.785358197936043i \\ -0.375184673558697 + 0.544557123983112i \\ 11.339406251617223 - 0.187523447938984i \\ -54.725877179646446 - 11.70250038063823i \\ 46.705835550616655 + 12.076406957853028i \end{pmatrix}$$

$$\begin{pmatrix} h_{17} \\ h_{27} \\ \vdots \\ h_{57} \end{pmatrix} = \begin{pmatrix} -1.024663376871514 + 3.490258159156221i \\ 0.029814188603163 - 0.483604710444412i \\ -5.1081199146394 + 0.098177739273448i \\ 18.51088653848523 + 3.666032297737733i \\ -13.987947358226732 - 3.459359784054443i \end{pmatrix}$$

当 $M=3, K=5$ 时，系数为

$$h=\left(h_{10},h_{20},\ldots,h_{K0},h_{11},h_{21},\ldots,h_{K1},\ldots\ \ldots,h_{1M},h_{2M},\ldots,h_{KM}\right)^{T}$$

其中

$$\begin{pmatrix}h_{10}\\h_{20}\\\vdots\\h_{50}\end{pmatrix}=\begin{pmatrix}14.780236738188762-3.407272982859118i\\7.5589799128771+6.798584802871691i\\-4.968156860890714-18.239256176244496i\\-17.573520752070838+17.550986315296576i\\17.01609092382657-5.375911876183807i\end{pmatrix}$$

$$\begin{pmatrix}h_{11}\\h_{21}\\\vdots\\h_{51}\end{pmatrix}=\begin{pmatrix}-23.8730694321638+6.44146042981748i\\2.280114375211962+1.06559650482386i\\-17.435878246983794-5.361314727169957i\\42.84976617812742+14.26011350677256i\\-30.669623552294826-11.835155488489631i\end{pmatrix}$$

$$\begin{pmatrix}h_{12}\\h_{22}\\\vdots\\h_{52}\end{pmatrix}=\begin{pmatrix}29.350554924136034-8.563195493139947i\\2.560105155027932+2.165153671382142i\\5.157372551733523-6.260596396933706i\\-25.8385917484162+3.301284121077502i\\22.045633038847896+2.424240202428063i\end{pmatrix}$$

$$\begin{pmatrix}h_{13}\\h_{23}\\\vdots\\h_{53}\end{pmatrix}=\begin{pmatrix}-12.395243880633434+4.384201919362976i\\-1.934779747384792-0.88979555527754i\\0.650336538692571+3.822101463942476i\\7.084504811234023-4.012958492220953i\\-7.486107002375+0.469785546979324i\end{pmatrix}$$

1.2 文中 5.2.3 有记忆功预失真函数 F 的模型

当 $K=5,\ M=3$ 时，系数为：

$$h=\left(h_{10},h_{20},\ldots,h_{K0},h_{11},h_{21},\ldots,h_{K1},\ldots\ \ldots,h_{1M},h_{2M},\ldots,h_{KM}\right)^{T}$$

其中

$$\begin{pmatrix}h_{10}\\h_{20}\\\vdots\\h_{50}\end{pmatrix}=\begin{pmatrix}0.922911223019879 - 0.60744700393722\mathrm{i}\\0.245417330250641 + 0.09332920348482\mathrm{i}\\-1.16963554048467 - 0.024528042793277\mathrm{i}\\1.187212606655683 - 0.165029976931694\mathrm{i}\\-0.03713399904953 + 0.168432724793623\mathrm{i}\end{pmatrix}$$

$$\begin{pmatrix} h_{11} \\ h_{21} \\ \vdots \\ h_{51} \end{pmatrix} = \begin{pmatrix} -2.091909358726769 + 1.371266133905037i \\ 0.047014283264433 - 0.13081718239165i \\ 0.710017055603604 - 0.54646093373777i \\ -0.033948879919108 + 1.485726896624385i \\ -1.1639739395996 - 0.994221118675401i \end{pmatrix}$$

$$\begin{pmatrix} h_{12} \\ h_{22} \\ \vdots \\ h_{52} \end{pmatrix} = \begin{pmatrix} 1.580468468866203 - 1.182175322195181i \\ -0.32496706199653 - 0.083415969201986i \\ 0.218125428321858 + 1.294458625242992i \\ -1.308344452821484 - 2.629434977381017i \\ 1.660657243626061 + 1.558671507932425i \end{pmatrix}$$

$$\begin{pmatrix} h_{13} \\ h_{23} \\ \vdots \\ h_{53} \end{pmatrix} = \begin{pmatrix} 0.7382043669404 + 0.549776316258908i \\ -0.953647638466643 - 0.885639003305225i \\ 2.045890564324863 + 2.102220315329139i \\ -1.278922836381994 - 2.114552195869492i \\ 0.122952737857115 + 0.886566632382652i \end{pmatrix}$$

2、主程序

2.1 问题一

2.1.1 模型一 三次多项式（见 Workspace\Q1_1\Poly3\Poly3Main.m）

Poly3Main.m:

```
x1=pa_in_memoryless;
x2=pa_in_memoryless.^2;
x3=pa_in_memoryless.^3;
I=[x1.';x2.';x3.'];
I=I.';
Poly3=inv(I'*I)*I'*pa_out_memoryless;
pa_out_simulate=I*Poly3;
MNSE=CalMNSE(pa_out_simulate,pa_out_memoryless);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_memoryless);
```

2.1.2 模型一 五次多项式（见 Workspace\Q1_1\Poly5\Poly5Main.m）

Poly5Main.m:

```
x1=pa_in_memoryless;
x2=pa_in_memoryless.^2;
x3=pa_in_memoryless.^3;
x4=pa_in_memoryless.^4;
x5=pa_in_memoryless.^5;
I=[x1.';x2.';x3.';x4.';x5.'];
I=I.';
Poly5=inv(I'*I)*I'*pa_out_memoryless;
```

```
pa_out_simulate=I*Poly5;
MNSE=CalMNSE(pa_out_simulate,pa_out_memoryless);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_memoryless);
```

2.1.3 模型二：多项式变形 K=3（见 Workspace\Q1_1\B3\B3Main.m）

B3Main.m:

```
pa_in_abs=abs(pa_in_memoryless);
x1=pa_in_memoryless;
x2=x1.*pa_in_abs;
x3=x2.*pa_in_abs;
I=[x1.';x2.';x3.'];
I=I.';
B3=inv(I'*I)*I'*pa_out_memoryless;
pa_out_simulate=I*B3;
MNSE=CalMNSE(pa_out_simulate,pa_out_memoryless);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_memoryless);
```

多项式变形 K=5（见 Workspace\Q1_1\B5\B5Main.m）

B5Main.m:

```
pa_in_abs=abs(pa_in_memoryless);
x1=pa_in_memoryless;
x2=x1.*pa_in_abs;
x3=x2.*pa_in_abs;
x4=x3.*pa_in_abs;
x5=x4.*pa_in_abs;
I=[x1.';x2.';x3.';x4.';x5.'];
I=I.';
B5=inv(I'*I)*I'*pa_out_memoryless;
pa_out_simulate =I*B5;
MNSE=CalMNSE(pa_out_simulate,pa_out_memoryless);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_memoryless);
```

2.1.4 模型三：函数基 K=7（见 Workspace\Q1_1\B7\B7Main.m）

B7Main.m:

```
pa_in_abs=abs(pa_in_memoryless);
x1=pa_in_memoryless;
x2=pa_in_abs.*x1.*4-x1.*3;
x3=pa_in_abs.^2.*x1.*15-pa_in_abs.*x1+x1.*6;
x4=pa_in_abs.^3.*x1.*56-pa_in_abs.^2.*x1.*105+pa_in_abs.*x1.*60-x1.*10;
x5=pa_in_abs.^4.*x1.*210-pa_in_abs.^3.*x1.*504+pa_in_abs.^2.*x1.*420-pa_in_abs.*x1.*1
40+x1.*15;
x6=pa_in_abs.^5.*x1.*792-pa_in_abs.^4.*x1.*2310+pa_in_abs.^3.*x1.*2520-pa_in_abs.^2.*
x1.*1260+pa_in_abs.*x1.*280-x1.*21;
x7=pa_in_abs.^6.*x1.*3003-pa_in_abs.^5.*x1.*10296+pa_in_abs.^4.*x1.*13860-pa_in_abs.
```

```
^3.*x1.*9240+pa_in_abs.^2.*x1.*3150-pa_in_abs.*x1.*504+x1.*28;
I=[x1.';x2.';x3.';x4.';x5.';x6.';x7.'];
I=I.';
B7=inv(I'*I)*I'*pa_out_memoryless;
pa_out_simulate =I*B7;
MNSE=CalMNSE(pa_out_simulate,pa_out_memoryless);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_memoryless);
```

2.1.5 模型四：函数基(正则化)K=7（见 Workspace\Q1_1\Reg\RegMain.m）

RegMain.m:主程序

```
RegCoef=Regularization(pa_in_memoryless,pa_out_memoryless);
Matrix7=GetMatrix(pa_in_memoryless);
pa_out_simulate=Matrix7*RegCoef;
MNSE=CalMNSE(pa_out_simulate,pa_out_memoryless);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_memoryless);
```

2.1.6 预失真模型： K=4（见 Workspace\Q1_2\K4\K4Main.m）

K4Main.m:主程序

```
x=pa_in_memoryless;
y=pa_out_memoryless;
g=1.1;
k=2;
A=GetMatrix(y,g);
Coef=inv(A'*A)*A'*x;
ANormal=GetNormalMatrix(x);
y_simulate=ANormal*Coef;
ymax=abs(max(y_simulate));
xmax=abs(max(x));
while (ymax>xmax)||((xmax-ymax)>=1E-6)
    if xmax-ymax>=1E-6
        g=g+1/k;
        k=k+1;
    else
        g=g-1/k;
        k=k+1;
    end
    A=GetMatrix(y,g);
    Coef=inv(A'*A)*A'*x;
    ANormal=GetNormalMatrix(x);
    y_simulate=ANormal*Coef;
    ymax=abs(max(y_simulate));
    xmax=abs(max(x));
end
GCoef=CalGCoef(pa_in_memoryless,pa_out_memoryless);
pa_G_Matrix=GetNormalMatrix_5(y_simulate);
pa_out_simulate=pa_G_Matrix*GCoef;
pa_out_standard=pa_in_memoryless.*g;
MNSE=CalMNSE(pa_out_simulate,pa_out_standard);
EVM=CalEVM(pa_out_simulate,pa_out_standard);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_standard);
```

2.1.7 预失真模型： K=5（见 Workspace\Q1_2\K5\K5Main.m）

K5Main.m:主程序

```
x=pa_in_memoryless;
y=pa_out_memoryless;
g=1.1;
k=2;
A=GetMatrix(y,g);
Coef=inv(A'*A)*A'*x;
ANormal=GetNormalMatrix(x);
y_simulate=ANormal*Coef;
ymax=abs(max(y_simulate));
xmax=abs(max(x));
while (ymax>xmax)||((xmax-ymax)>=1E-6)
    if xmax-ymax>=1E-6
        g=g+1/k;
        k=k+1;
    else
        g=g-1/k;
        k=k+1;
    end
    A=GetMatrix(y,g);
    Coef=inv(A'*A)*A'*x;
    ANormal=GetNormalMatrix(x);
    y_simulate=ANormal*Coef;
    ymax=abs(max(y_simulate));
    xmax=abs(max(x));
end
GCoef=CalGCoef(pa_in_memoryless,pa_out_memoryless);
pa_G_Matrix=GetNormalMatrix_5(y_simulate);
pa_out_simulate=pa_G_Matrix*GCoef;
pa_out_standard=pa_in_memoryless.*g;
MNSE=CalMNSE(pa_out_simulate,pa_out_standard);
EVM=CalEVM(pa_out_simulate,pa_out_standard);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_standard);
```

2.1.8 预失真模型： K=7（见 Workspace\Q1_2\K7\K7Main.m）

K7Main.m:主程序

```
x=pa_in_memoryless;
y=pa_out_memoryless;
g=1.1;
k=2;
A=GetMatrix(y,g);
Coef=inv(A'*A)*A'*x;
ANormal=GetNormalMatrix(x);
y_simulate=ANormal*Coef;
ymax=abs(max(y_simulate));
xmax=abs(max(x));
while (ymax>xmax)||((xmax-ymax)>=1E-6)
    if xmax-ymax>=1E-6
```

```
        g=g+1/k;
        k=k+1;
    else
        g=g-1/k;
        k=k+1;
    end
    A=GetMatrix(y,g);
    Coef=inv(A'*A)*A'*x;
    ANormal=GetNormalMatrix(x);
    y_simulate=ANormal*Coef;
    ymax=abs(max(y_simulate));
    xmax=abs(max(x));
end
GCoef=CalGCoef(pa_in_memoryless,pa_out_memoryless);
pa_G_Matrix=GetNormalMatrix_5(y_simulate);
pa_out_simulate=pa_G_Matrix*GCoef;
pa_out_standard=pa_in_memoryless.*g;
MNSE=CalMNSE(pa_out_simulate,pa_out_standard);
EVM=CalEVM(pa_out_simulate,pa_out_standard);
DrawComparePic(pa_in_memoryless,pa_out_simulate,pa_out_standard);
```

2.2 问题二

2.2.1 K=5、M=3：（见 Workspace\Q2_1\M3\M3Main.m）

M3Main.m:主程序

```
x=pa_in_memory;
y=pa_out_memory;
Q=3;
T=4;
K=5;
[X Y xnew]=Bianxing(x,y,Q,K,T);
b0 =Regularization(X,Y,K,Q);
pa_out_simulate=GetMatrixX(pa_in_memory)*b0;
pa_in_simulate=pa_in_memory(4:73920);
pa_out_standard=pa_out_memory(4:73920);
MNSE=CalMNSE(pa_out_simulate,pa_out_standard);
DrawComparePic(pa_in_simulate,pa_out_simulate,pa_out_standard);
```

Regularization.m:

```
function b0 =Regularization(X,Y,K,Q)
namta=1;
b0=ones(K*(Q+1),1)+ones(K*(Q+1),1).*1i;
b=(X'*X+2*namta*eye(K*(Q+1)))\(X'*Y+2*namta*b0);
while(norm(b-b0)^2>1E-8)
    b0=b;
    b=(X'*X+2*namta*eye(K*(Q+1)))\(X'*Y+2*namta*b0);
    namta=.8*namta;
end
end
```

GetMatrixX.m:

```
function MatrixX=GetMatrixX(pa_in)
MatrixX=zeros(73917,20);
for i=1:73917
    for j=1:4
        x1=pa_in(i+j-1);
        a=abs(x1);
        MatrixX(i,(j-1)*5+1: (j-1)*5+5)=[x1 a*x1 a^2*x1 a^3*x1 a^4*x1];
    end
end
end
```

2.2.2 K=5、M=7：（见 Workspace\Q2_1\M7\M7Main.m）

M7Main.m 主程序：

```
x=pa_in_memory;
y=pa_out_memory;
Q=7;
T=8;
K=5;
[X Y xnew]=Bianxing(x,y,Q,K,T);
b0 =Regularization(X,Y,K,Q);
pa_out_simulate=GetMatrixX(pa_in_memory)*b0;
pa_in_simulate=pa_in_memory(8:73920);
pa_out_standard=pa_out_memory(8:73920);
MNSE=CalMNSE(pa_out_simulate,pa_out_standard);
DrawComparePic(pa_in_simulate,pa_out_simulate,pa_out_standard);
```

2.2.3 K=5、M=3：（见 Workspace\Q2_2\Q2_2Main.m）

Q2_2Main.m:主程序

```
x=pa_in_memory;
y=pa_out_memory;
g=5;
k=2;
Q=3;
T=4;
K=5;
MatrixFixed=GetNormalMatrixX(x);
[X Y xnew]=Bianxing(y./g,x,Q,K,T);
b0=Regularization(X,Y,K,Q);
y_simulate=MatrixFixed*b0;
ymax=abs(max(y_simulate));
xmax=abs(max(x));
count=0;
while (ymax>xmax)||((xmax-ymax)>=1E-4)
    if count>1000
        break;
    end
    if xmax-ymax>=1E-4
        g=g+1/k;
```

```
        k=k+1;
    else
        g=g-1/k;
        k=k+1;
    end
    count=count+1;
    [X Y xnew]=Bianxing(y./g,x,Q,K,T);
    b0=Regularization(X,Y,K,Q);
    y_simulate=MatrixFixed*b0;
    ymax=abs(max(y_simulate));
    xmax=abs(max(x));
end
[XOri YOri xnewOri]=Bianxing(x,y,Q,K,T);
GCoef =Regularization(XOri,YOri,K,Q);
MatrixG=GetMatrixX(y_simulate);
pa_out_simulate=MatrixG*GCoef;
pa_out_standard=pa_in_memory*g;
pa_out_standard=pa_out_standard(7:73920);
pa_in_standard=pa_in_memory(7:73920);
MNSE=CalMNSE(pa_out_simulate,pa_out_standard);
EVM=CalEVM(pa_out_simulate,pa_out_standard);
DrawComparePic(pa_in_standard,pa_out_simulate,pa_out_standard);
```

2.3 问题三

2.3.1

simulate_in_out.mat 为 matlab 数据文件，存放有 pa_in_memory、pa_out_memory 和第二题第二问解算出的预失真输出信号 pa_out_simulate。

（见 Workspace\Q3\simulate_in_out.mat 及 Workspace\Q3\Q3Main.m）

Q3Main.m:主程序

```
N=length(pa_in_memory);
per=30.72*12*10^6/N;
n=fix(20*10^6/per);
x=fft(pa_in_memory);
y=fftshift(x);
P_in_Ori=abs(x).^2/N;
P_in=abs(y).^2/N;
figure

subplot(2,2,1);
plot([1:1:N]-N/2*ones(1,N),P_in);
title('输入信号  N=73920');
xlabel('频率');
ylabel('功率谱');

subplot(2,2,2);
plot([1:1:N]-N/2*ones(1,N),10*log10(P_in));
title('输入信号(10log10(P))');
xlabel('频率');
```

```
ylabel('功率谱');

subplot(2,2,3);
plot(P_in_Ori(1:2*n));
title('N=8020');
xlabel('频率');
ylabel('功率谱');

subplot(2,2,4);
plot(10*log10(P_in_Ori(1:2*n)));
title('10log10(P)');
xlabel('频率');
ylabel('功率谱');

ACPR=10*log10((sum(abs(P_in_Ori(n+1:2*n)).^2)-0.5*((abs(P_in_Ori(n+1)))^2+(abs(P_in_
Ori(2*n)))^2))...
      /(sum(abs(P_in_Ori(1:n)).^2)-0.5*((abs(P_in_Ori(1)))^2+(abs(P_in_Ori(n)))^2)))
%原始输入数据图像

N=length(pa_out_memory);
per=30.72*12*10^6/N;
n=fix(20*10^6/per);
x=fft(pa_out_memory);
y=fftshift(x);
P_out_Ori=abs(x).^2/N;
P_out=abs(y).^2/N;
figure

subplot(2,2,1);
plot([1:1:N]-N/2*ones(1,N),P_out);
title('无预失真输出信号  N=73920');
xlabel('频率');
ylabel('功率谱');

subplot(2,2,2);
plot([1:1:N]-N/2*ones(1,N),10*log10(P_out));
title('无预失真输出信号(10log10(P))');
xlabel('频率');
ylabel('功率谱');;

subplot(2,2,3);
plot(P_out_Ori(1:2*n));
title('N=8020');
xlabel('频率');
ylabel('功率谱');


subplot(2,2,4);
```

```
plot(10*log10(P_out_Ori(1:2*n)));
title('10log10(P)');
xlabel('频率');
ylabel('功率谱');;

P_abs=abs(P_out_Ori);
up=sum(P_abs(n+1:2*n).^2);
up=up-0.5*(P_abs(n+1))^2;
up=up-0.5*(P_abs(2*n))^2;
down=sum(P_abs(1:n).^2);
down=down-0.5*(P_abs(1))^2;
down=down-0.5*(P_abs(n))^2;
ACPR_out=10*log10(up/down)
%原始输出数据图像

N=length(pa_out_simulate);
per=30.72*12*10^6/N;
n=fix(20*10^6/per);
x=fft(pa_out_simulate);
y=fftshift(x);
P_simulate_Ori=abs(x).^2/N;
P_simulate=abs(y).^2/N;
figure

subplot(2,2,1);
plot([1:1:N]-N/2*ones(1,N),P_simulate);
title('预失真输出信号  N=73920');
xlabel('频率');
ylabel('功率谱');

subplot(2,2,2);
plot([1:1:N]-N/2*ones(1,N),10*log10(P_simulate));
title('预失真输出信号(10log10(P))');
xlabel('频率');
ylabel('功率谱');;

subplot(2,2,3);
plot(P_simulate_Ori(1:2*n));
title('N=8020');
xlabel('频率');
ylabel('功率谱');

subplot(2,2,4);
plot(10*log10(P_simulate_Ori(1:2*n)));
title('10log10(P)');
xlabel('频率');
ylabel('功率谱');;

P_abs=abs(P_simulate_Ori);
```

```
up=sum(P_abs(n+1:2*n).^2);
up=up-0.5*(P_abs(n+1))^2;
up=up-0.5*(P_abs(2*n))^2;
down=sum(P_abs(1:n).^2);
down=down-0.5*(P_abs(1))^2;
down=down-0.5*(P_abs(n))^2;
ACPR_Simulate=10*log10(up/down)
%模型输出数据图像
```