第3章 多载波发射机系统功率优化

CHAPTER 3


源代码

移动通信不仅是全球性的技术标准，更关系到未来信息产业的基础架构和国家战略。随着移动数据流量和移动端连接数的指数型增长，高能耗和环境问题日趋严峻。全球信息和通信技术（Information and Communication Technology，ICT）行业的碳排放量预计到2030年将消耗全球51%的能源。GSMA《2018全球移动趋势报告》显示，由于能耗的剧增，近年来世界各国移动运营商的利润增长率显著下降，2016年年复合增长率仅为3%，到2020年，年复合增长率已降低到1%左右，全球无线网络运营商已陷入投入高，产出少的窘境。无线网络能耗逐渐超出控制，伴随而生的温室气体和电磁污染也将超出安全阈值。绿色通信已经上升到国家战略规划层面，我国《信息通信行业发展规划（2016—2020年）》和“十三五”规划纲要明确将降低通信行业污染物排放与总耗能量作为节能减排的目标。提升网络效率和资源利用率，节能减排，是最为可行的解决方案。

随着环境问题、运营商收支不平衡趋势日益严峻，无线通信能耗问题引起了国际范围内广泛关注。目前，3GPP已经把节能作为自组织网络引入LTE/LTE-A标准；ITU-R成立了IMT-2020推进组，提出在4G/5G无线移动网络中，通信网络系统能效要提高100倍；大量的国际科研项目如Green Radio、EARTH、TREND、C2POWER、GREEN Touch和中国的C-Ran等，标准化组织如ETSI（欧洲技术标准院）、ATIS（通信解决方案联盟）、中国通信标准化协会等都开展了大量的研究，几乎覆盖了广电、通信系统从网络到链路、从设备到布设、从架构到运营等各个方面。

能耗成本逐渐将网络的盈利能力蚕食殆尽，如图3-1所示，无线网络中大部分的能量消耗在无线基站侧，无线基站接入的能耗占比超过了50%，而在接入过程中的功放又占到50%～80%，功放效率低，导致绝大部分能耗以热量的形式浪费掉，恶化机房环境，同时散热所需空调制冷的总能耗占比又超过30%，最后以信号形成传递出的能量传递效率只有2%～3%。由此可知，基站侧能量消耗，特别是功放和制冷消耗，占据网络整体能耗的比重极高，不可忽略，也是通过资源调度算法所无法替代和优化的。已知的能效优化模型，往往设定功放效率是恒定、不可动态优化的；功放消耗和制冷消耗这些占比较高的指标，却在能效优化模型中未被重视，也未被关联起来协同管理。功放智能化提供一种新的思路：通过优化信号分布状态获得功放效率的适应性提升，从全局能耗的角度增强能量转化率。

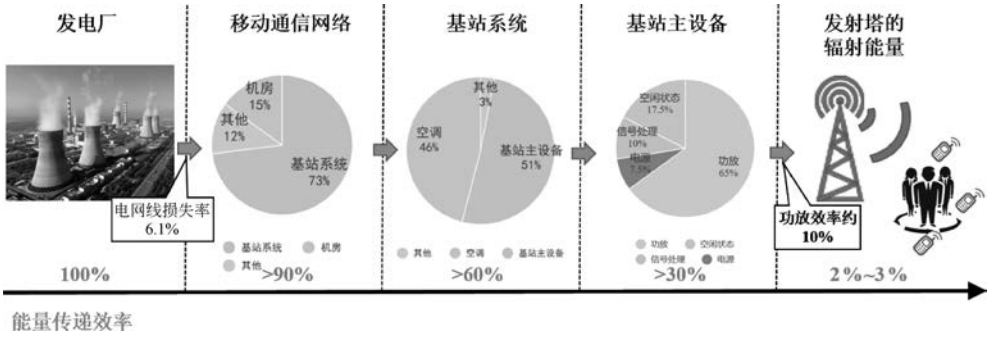


图 3-1　无线网络的主要能耗占比

3.1　多载波系统高能耗问题

移动数字多媒体发射机系统中最大的缺陷就是 OFDM 本身峰均比过高。由于 OFDM 信号为多个余弦波的叠加，当子载波个数达到一定程度时，OFDM 符号波形满足高斯随机过程，其包络极不稳定。当 IFFT 输入端的多载波同相叠加时，其输出就会产生很大的峰值，一般用峰均比（Peak to Average Power Ratio，PAPR）来衡量。如图 3-2 所示，OFDM 系统的高 PAPR 问题要求数/模转换器（Digital-to-Analog converter，D/A）和发射端的功率放大器（High Power Amplifier，HPA）具备很大的线性动态范围，否则当高峰值信号进入非线性区和饱和区时，会产生很大的带内失真和带外辐射，引起信号失真和邻频干扰，导致系统性能恶化。但由于峰值出现也是随机的，就意味着线性放大器必定不能一直工作在最高状态，从而导致功率利用率不高。如果不采用线性放大器，就需要较大的回退量，这样也会导致功率利用率降低。在对 OFDM 信号进行放大时，如果放大器线性不好，那么除了产生交调干扰外，当回退量不够时，还会产生带内非线性失真（见图 3-3），并导致频谱扩展，产生很大的带外功率，从而导致对相邻信道的干扰（见图 3-4）。于是要求功率放大器扩展线性范围，但又会导致功放效率降低，绝大部分能量都转化为热能被浪费掉，成本也随之增加，直接影响了多载波数字调制发射机的推广和应用。

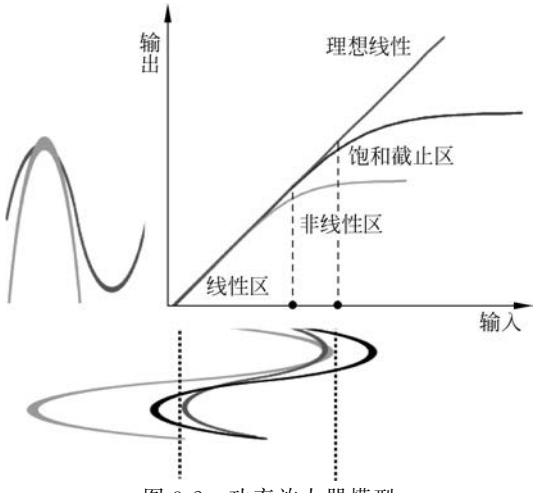


图 3-2　功率放大器模型

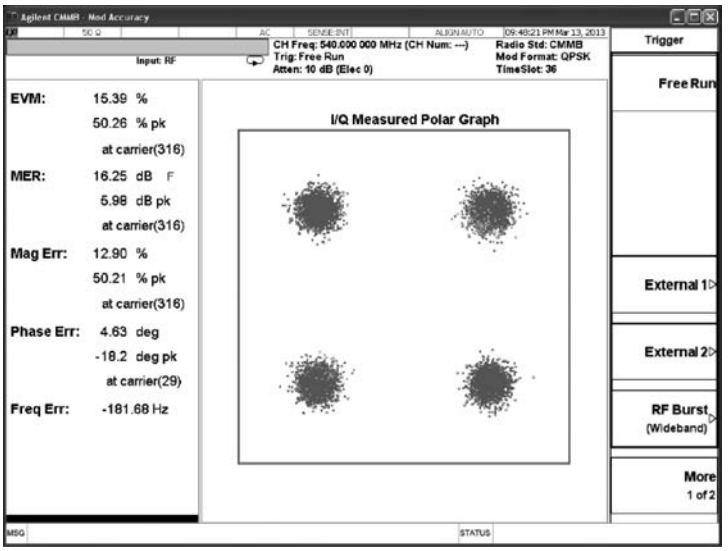


图 3-3 功放带内失真

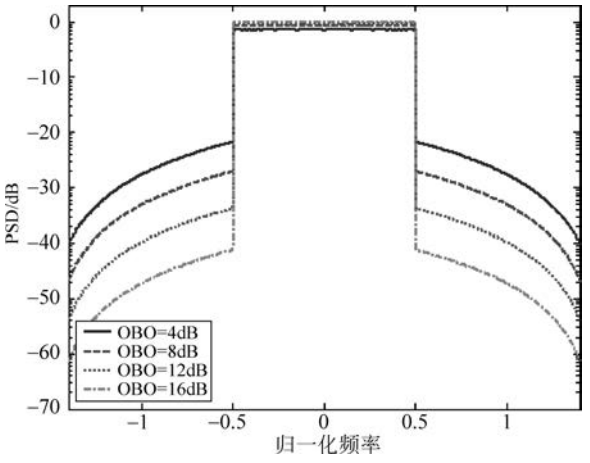


图 3-4 功放带外干扰/噪声

随着超高频、超宽带的应用和发展，多载波调制对超宽带系统的实现至关重要，当传输带宽达到极限时，模拟调制中的能耗问题更加突出。功放在任何类型的无线通信中都将主导基站能耗，对于任意波形的多载波系统，固有的模拟器件非线性和饱和截止都会引起严重失真，导致功放效率不断下降。高峰均比问题在未来通信系统中将更加突出，依靠毫米波、太赫兹等频段，能够实现超宽带的需求，但同时也面临超高频功放能量转化率不足的缺陷。考虑低硬件成本，从信号角度优化功放失真主要有两大思想：

(1) 利用 PAPR 抑制技术降低信号出现在失真区域的概率。PAPR 抑制技术包含：限幅算法、星座图扩展(Active Constellation Extension，ACE)算法、编码算法、预留子载波法(Tone Reservation，TR)、音频插入法(Tone Injection，TI)、选择性映射法(Selective Mapping，SLM)和部分传输序列法(Partial Transmit Sequence，PTS)等。综合来看，早期的通信和广电发射机系统，预畸变类方法(如限幅 ACE 算法)在工程领域应用最为广泛，然

而随着高阶 QAM 调制成为未来通信的主流，占比极高的星座图内层向量挪动受限，无法对峰值优化做出贡献；概率类技术（如 PTS 技术）虽然会附加一定的冗余信息，增加计算复杂度，但是综合考虑信号 PAPR 抑制效果和超宽带高阶 QAM 系统的兼容性，相比于预畸变类和编码类技术，被认为是抑制 PAPR 最具潜力的发展方向。由于不局限于特定模式，概率类 PAPR 抑制技术的多维融合也是重要的应用方向。

（2）利用线性化技术对非线性失真进行校正。即使是功放输出功率较低时，功放的非线性失真也可能存在，应用 PAPR 抑制技术提升功放能效存在一定的局限性。为了能够使功放工作在具有较高能量转换率的非线性区，补偿非线性失真的线性化技术也应用广泛。现有功放线性化技术主要分为前馈、负反馈与预失真三类，它们通过规避非线性失真或消除失真分量的方式提升功放效率。然而这些方案大都存在着复杂度较高、对功放非线性记忆效应敏感以及产生带外失真等不利因素，而且现有研究同时指出仅追求抑制高峰值并不是提升功放效率和效能的最佳途径。

本章将优化高峰均比的关键问题归结为如何提高多载波系统效率和信号失真的问题，提出最佳分布理论和功放效率最大化评估标准。超宽带高阶 QAM 调制系统，作为可兼容的功放优化技术存在空间自由度和频域自由度不足的问题。在智能增强型信号分布优化的节能策略中，通过研究时域、频域、空间域功放优化技术融合，用增强计算的方式拟合信号最佳分布，构建出高能效功放结构。应用新理论和融合技术，有望显著提升能效和功耗效率。

3.2　多载波系统峰均比抑制技术

3.2.1　OFDM 峰均比问题描述

OFDM 系统的技术缺陷中，包括了 PAPR 过大的问题。OFDM 调制信号是多个子载波信号的叠加，使得信号幅度可能存在较大的波动。通常采用 PAPR 来描述信号幅值波动的特点。PAPR 定义为 OFDM 信号的最大峰值功率与平均功率的比值，即

$$\mathrm{PAPR}=10\lg\left(\frac{\max[|x_n|^2]}{\mathrm{E}[|x_n|^2]}\right) \tag{3-1}$$

式中，x_n 表示经过 IFFT 变换后得到的一个 OFDM 符号；E[·]表示数学期望。

较高的 PAPR 对 OFDM 的实际应用产生不良影响，具体体现在三方面：

（1）要求发射机功率放大器具有更大的线性范围，直接导致功放效率的降低和成本的提高。

（2）要求发射机 D/A 转换器具有较大的转换宽度，增大了实现的复杂度和成本。

（3）为了保证信号的失真度，FCC（美国联邦通信委员会）、CEPT（欧洲邮电管理委员会）等通信组织通常会为给定的频带设置 PAPR 上限。

归一化条件下，根据中心极限定理，只要子载波的个数 N 足够大，基带的实部和虚部的采样点便会服从均值为 0、方差为 1/2 的高斯分布（实部和虚部各占整个信号功率的一半），通带信号为基带信号平均功率的一半。由此可知，OFDM 符号的幅值包络服从瑞利分布，其概率密度函数为

$$f_{|x_n|}(x)=\frac{2x\mathrm{e}^{\frac{-x^2}{\sigma^2}}}{\sigma^2},\quad x\geqslant 0 \tag{3-2}$$

OFDM 存在多载波叠加引起的高 PAPR，使得发射机 HPA 必须工作在较高的功率回退状态以保证足够的线性动态范围，导致极大地降低了功放的效率，故不得不采用更高功率等级的 HPA，由此必须推高功放直流总电压/功率回退，使总功耗极大增加；在输出功率不变的情况下，总功耗的增加等于功放效率的降低；功放的输出功率一部分转化为射频信号功率，另一部分转化为热能，导致温度上升，设备和机房的工作状态恶化，进一步还会导致更严重的非线性失真。因此，需要消耗极大的电量排除热量以维持机房温度。科研人员对 PAPR 抑制技术的研究投入了极大的精力，以在尽可能低的输入功率回退条件下，保证信号的失真度，降低放大器管耗，提升发射机效率，实现国家节能减排的战略目标。

实际测量状态下，能观察到 OFDM 信号峰值的概率微乎其微，因此测量 OFDM 信号的峰值统计分布更具理论分析价值。最为常用的是应用互补累积分布函数来描述大量 OFDM 信号最高峰值的分布特性，即用 PAPR 超过某一门限值 z 的概率得到互补累积概率分布函数(Complementary CDF，CCDF)来表征 PAPR 的分布，表述为

$$\Pr(\text{PAPR} > z) = 1 - (1 - \mathrm{e}^{-z})^{N} \tag{3-3}$$

CCDF 曲线体现了信号功率高于给定功率电平的统计情况和概率分布。

OFDM 符号进行 N 点 IFFT 变换时，所得到的 N 个输出样值不能真实地反映连续 OFDM 信号的变化特性。在没有过采样的条件下，进行 A/D 转换时，功放失真使得超出采样频率的高频噪声叠加到低频部分，造成带内失真的恶化。若不进行过采样，仅通过离散采样点来统计 PAPR 的分布，则在采样过程中会漏掉真正的峰值，因此通过离散样点得到的 CCDF 曲线不能反映真实的信号峰值分布，其 PAPR 往往比连续信号的小一些。为了避免频谱混叠现象，需要对 OFDM 符号进行过采样，如图 3-5 所示，采样率越高，峰值越精确，当采样率 $L=4$ 左右时，采样值幅值分布趋近于瑞利分布，足以以离散的形式模拟连续信号。

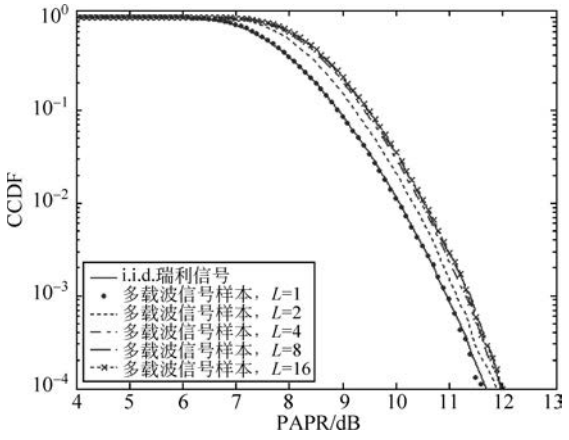


图 3-5 CCDF 曲线与过采样分布

3.2.2 功率放大器模型

功率放大器的数学模型是评估功放效率和功放失真的关键，根据建模方式的不同，大体上可分为物理模型和行为模型。物理模型主要应用于电路原理仿真，而行为模型主要应用于系统仿真。物理模型主要以功率放大器内部物理元件特性及原件之间相互作用关系为基

础，根据功率放大器内部的工作原理，建立等效数学模型。而行为模型是将功放看作系统中的子模块或子系统，主要关注其输入-输出的数学特性，来建立相应的模型。行为模型是系统非线性分析以及数字预失真器设计方面最常用的模型。针对射频功率放大器建立的行为模型，主要可分为有记忆功放模型和无记忆功放模型两种。

1）有记忆功放模型

功放的记忆效应是指某时刻的输出信号不仅取决于该时刻的输入信号，而且还与该时刻之前的输入信号有关。记忆模型常见于早期的模拟通信，由于信号时域分布不均匀，分布与不同的节目/业务内容相关，导致了不同时间的功放的管耗分布不均匀，产生了热敏感特征的记忆效应；而在数字通信中，通过时频二维的均匀化、随机化技术，选定放大信号带宽小于功率放大器带宽，可以减轻和消除记忆效应。描述功放记忆效应的模型有很多，其中比较典型的有 Volterra 级数模型、记忆多项式模型。

（1）Volterra 级数模型：Volterra 级数模型可用于有记忆效应功放的建模，离散的 Volterra 级数模型表达式为

$$y(n)=\sum_{p=1}^{Q}\sum_{i_1=0}^{T}\cdots\sum_{i_p=0}^{T}h_p(i_1,\cdots,i_p)D_p[x(n)] \tag{3-4}$$

式中，$x(n)$和$y(n)$分别表示模型的输入与输出；$h_p(i_1,\cdots,i_p)$为第 p 阶 Volterra 内核；T 为系统记忆深度；$D_p[x(n)]=\prod_{j=1}^{p}x(n-i_j)$ 是第 p 阶 Volterra 算子，i_j 为时延长度。当 Volterra 级数的记忆深度和阶次增加时，模型的计算复杂度迅速增长，从而限制了 Volterra 级数的最高阶次和记忆深度，因而该模型一般用于短时记忆效应与低阶弱非线性的系统建模中。

（2）记忆多项式（Memory Polynomial，MP）模型：Volterra 模型经过简化可以得到 MP 模型。大量有关功放非线性的研究仅考虑处于 Volterra 内核对角线的各项，即对 Volterra 的内核做如下约定：

$$\tilde{h}(q_1,q_2,\cdots,q_k)\begin{cases}\neq 0, & q_1=q_2=\cdots=q_k\\ =0, & \text{其他}\end{cases} \tag{3-5}$$

那么式(3-4)可以简化如下：

$$y(n)=\sum_{p=1}^{N}\sum_{i_1=0}^{Q}h_{i,p}x(n-i)\,|x(n-i)|^{p-1} \tag{3-6}$$

经过约束后得到的 MP 模型相较于 Volterra 模型，系数的个数及项数大幅减少，复杂度得到了极大地简化。MP 模型在模拟有记忆效应的功放非线性方面，有着广泛的应用。同时该模型可以应用于预失真器的构建，对有记忆效应的功放模型进行校正。

有记忆功放模型存在以下局限性：首先，功率放大器的记忆效应实际上是一种温度效应，是由于功放的工作温度变化引起的。传统的无线信号，例如电视信号存在亮场与暗场之分，亮场与暗场的平均功率不同，导致功放工作温度发生波动。但 OFDM 信号是由多个子载波叠加，幅度近似于噪声，并且对于功率恒定的信号，功放的温度在较长时间内保持稳定且变化缓慢，所以记忆效应并不明显。其次，由于 OFDM 射频信号幅度变化十分迅速，功放记忆效应无法对快速变化的信号进行跟踪。最后，在功放开机并热机达到工作状态后，影响 OFDM 系统中功放工作温度的变量，例如白昼黑夜等变量，变化十分缓慢，功放在很长时间

内保持稳定的温度及工作状态，即在一段时间内功放特性保持不变。基于此，功放的记忆效应不是本章讨论的主要问题。

2）无记忆功放模型

当输入信号带宽相较于调制频率足够窄时，或功放的频率响应特性在信号的通频带内是平坦的，则可以忽略功放的记忆效应，从而可以将功放看作简单的无记忆非线性系统进行分析。此时功放的幅度失真（AM-AM 失真）和相位失真（AM-PM 失真）仅与当前输入信号有关。典型的无记忆功放模型有 Saleh 模型、无记忆多项式模型以及 Rapp 模型。

（1）Saleh 模型：Saleh 模型主要用于描述行波管放大器（Traveling Wave Tube Amplifier，TWTA）的非线性特征，可表示为极坐标或直角坐标的形式。设输入信号的复包络为

$$\tilde{d}(t)=r_{\mathrm{d}}(t)\,\mathrm{e}^{\mathrm{j}\phi(t)} \tag{3-7}$$

式中，$r_{\mathrm{d}}(t)$为 t 时刻输入信号的幅度；$\phi(t)$为 t 时刻输入信号的相位。则功放输出可以表达为

$$\tilde{a}(t)=\tilde{d}(t)\widetilde{Y}(t)=\tilde{d}(t)A\left[r_{\mathrm{d}}(t)\right]\mathrm{e}^{\mathrm{j}\phi\left[r_{\mathrm{d}}(t)\right]} \tag{3-8}$$

式中，$\widetilde{Y}(t)$为功放的复增益，是输入信号的非线性函数；$A\left[r_{\mathrm{d}}(t)\right]$是功放复增益的幅度，$\phi\left[r_{\mathrm{d}}(t)\right]$为复增益函数的相位，分别反映了输出信号的幅度失真与相位偏移。$A\left[r_{\mathrm{d}}(t)\right]$和 $\phi\left[r_{\mathrm{d}}(t)\right]$分别表示放大器的幅度失真和相位失真，各自的表达式为

$$A\left[r_{\mathrm{d}}(t)\right]=\frac{\alpha_A}{1+\beta_{\mathrm{A}}r_{\mathrm{d}}^2(t)},\quad \phi\left[r_{\mathrm{d}}(t)\right]=\frac{\alpha_{\Phi}r_{\mathrm{d}}^2(t)}{1+\beta_{\Phi}r_{\mathrm{d}}^2(t)} \tag{3-9}$$

式中，参数 α_{A}、β_{A}、α_{Φ}、β_{Φ} 均为常数。

（2）无记忆多项式模型：泰勒级数是分析无记忆功放非线性特性的重要工具，其表达式为

$$y(t)=\sum_{n=1}^{N}a_n x^n(t) \tag{3-10}$$

式中，$x(t)$为功放的输入信号；a_n 为各阶多项式系数，控制着功放的线性度及非线性成分。泰勒级数对非线性描述物理意义明确，下标 n 直接反映了谐波失真阶次，各阶互调分量的大小可直接从 a_n 的大小反映出来。但泰勒级数模型的最大缺点是仅能分析有限阶次的失真度，对于工作在饱和区的功放特性分析会有较大的误差。

（3）Rapp 模型

归一化的 Rapp 模型的表达式为

$$F(x)=\frac{x}{\left(1+\left(\frac{x}{A_{\mathrm{sat}}}\right)^{2p}\right)^{\frac{1}{2p}}} \tag{3-11}$$

式中，x 为功放输入的时域信号；A_{sat} 为功放的饱和电平，控制着功放的最大输出幅度及饱和功率；p 为平滑因子，影响功放模型的线性度。Rapp 模型输入-输出特性曲线如图 3-6 所示。

考虑 OFDM 信号具有较高峰均比，更容易工作在饱和区，所以包含饱和截止区的功放模型才是 OFDM 系统中较为合理的模型。由于 Rapp 模型是根据实际的固态功率放大器

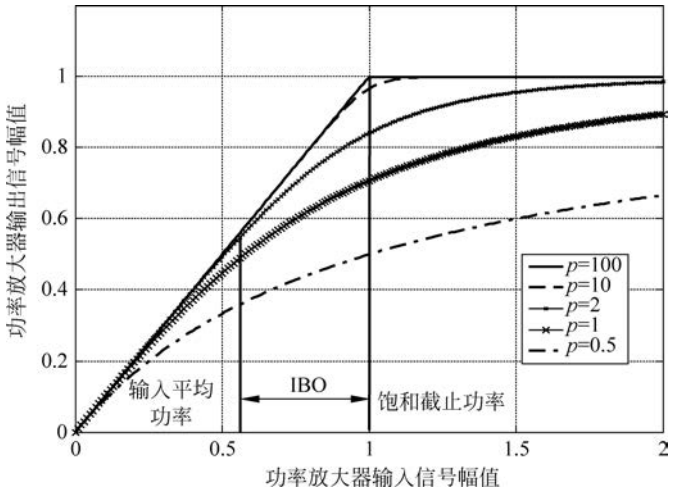


图 3-6　Rapp 模型输入-输出特性曲线

特性发展而来，是对实际 RF 功放特性曲线的合理描述，能够对功放特性有较好的模拟，并且可以通过控制平滑因子来得到不同线性度的功放，所以本章采用 Rapp 模型进行仿真分析。

3.2.3　ACE 峰均比抑制技术和 OFDM 发射机功放效率优化

1. ACE 峰均比抑制技术

为方便对比传统的方法，并进行完整的系统仿真，首先回顾 ACE 峰均比抑制技术。

令 $\boldsymbol{S}_n \in \mathbb{C}^N$ 为 OFDM 调制输入端的星座映射符号（以 QPSK 星座图为例讨论 OFDM 系统的峰均比抑制、功放效率优化技术），设经过 q 倍升采样之后的时域信号可以表示为

$$\boldsymbol{x}_n = \boldsymbol{F}_q \boldsymbol{S}_n \tag{3-12}$$

式中，$\boldsymbol{x}_n \in \mathbb{C}^{qN}$ 为 IFFT 模块的输出信号；$\boldsymbol{F}_q \in \mathbb{C}^{qN \times N}$ 为逆傅里叶矩阵，矩阵中 (i, k) 元素由 $[\boldsymbol{F}_q]_{i,k} = \frac{1}{\sqrt{N}} \mathrm{e}^{\mathrm{j}\frac{2\pi ik}{qN}}$ 构成。

根据 Parseval 定律：

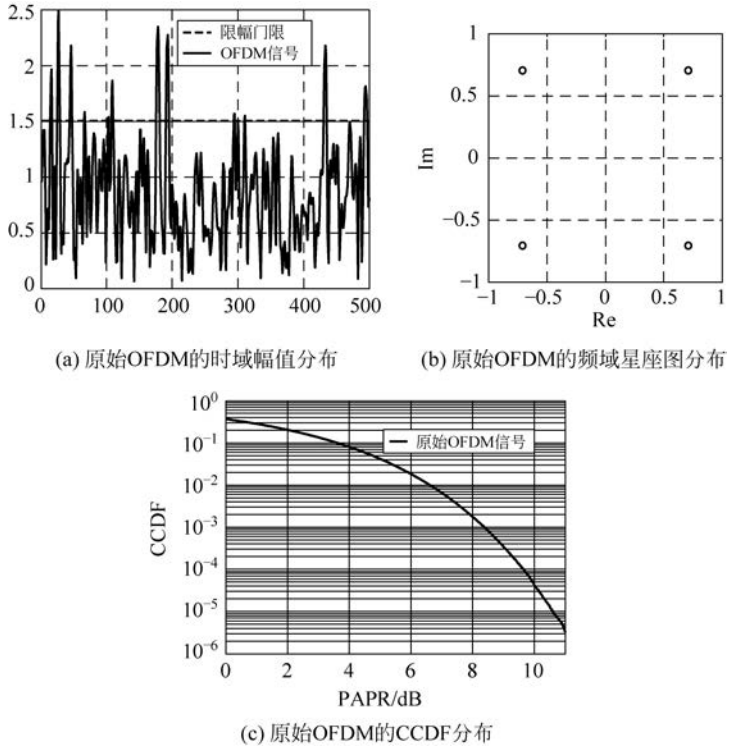
$$\mathrm{E}\{|\boldsymbol{x}_n|^2\} = \mathrm{E}\{|\boldsymbol{S}_n|^2\} = \sigma^2 \tag{3-13}$$

根据中心极限定律，OFDM 的时域信号 $\boldsymbol{x}_n$ 近似于瑞利分布，如图 3-7 所示，当多个相位近似的载波相互叠加之后就会出现远高于平均功率的峰值信号，具体表现为 CCDF 曲线中远高于平均功率的峰值有一定的出现概率。

为评估峰值的出现使功率放大器出现削波失真，采用 Rapp 模型来模拟发射机功率放大器的性能曲线及其传递函数。其中，$p = 10$ 时功率放大器接近理想状态。工程上通常利用 IBO（输入功率回退）指标衡量功率放大器的功耗和效率，定义为

$$\begin{aligned} \mathrm{IBO}[\mathrm{dB}] &= P_{\mathrm{in,max}} - P_{\mathrm{in,av}} \\ &= 10\lg \frac{A_{\mathrm{sat}}^2}{\sigma^2} \end{aligned} \tag{3-14}$$

式中，$P_{\mathrm{in,max}}$ 为输入信号的饱和截止功率；$P_{\mathrm{in,av}}$ 为输入信号的平均功率。



(a) 原始OFDM的时域幅值分布　　(b) 原始OFDM的频域星座图分布

(c) 原始OFDM的CCDF分布

图 3-7　原始 OFDM

令 $\hat{\boldsymbol{S}}_n$ 为功率放大器引起的带内失真信号，通常以 MER（信噪比的形式）来统计其失真度：

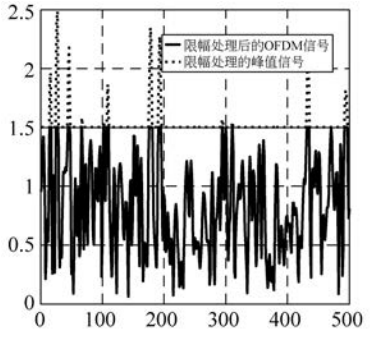
$$\begin{aligned}\mathrm{MER}(\hat{\boldsymbol{S}}_n) &= 10\lg\left\{\frac{\sum_{k=1}^{N}(|s_k|^2)}{\sum_{k=1}^{N}(|s_k-\hat{s}_k|^2)}\right\} \\ &= 10\lg\left\{\frac{\sum_{k=1}^{N}(I_k^2+Q_k^2)}{\sum_{k=1}^{N}(\Delta I_k^2+\Delta Q_k^2)}\right\}\end{aligned} \tag{3-15}$$

式中，s_k 为频域信号 $\boldsymbol{S}_n$ 的第 k 个子载波；$\hat{s}_k$ 为对应的功放失真信号；I_k 和 Q_k 分别为频域信号实部和虚部的理想点；ΔI_k 和 ΔQ_k 分别为 $\hat{s}_k - s_k$ 的实部和虚部。

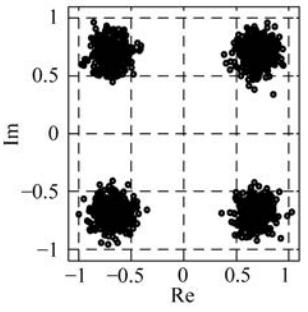
鉴于时域高峰值在功率放大时存在失真，放大信号的 MER 恶化使得功放被迫提高饱和电平 A_{sat}，进而导致功率放大器效率的降低和所需功率等级的提高。

如图 3-8 所示，限幅削波主要是以预失真的方式对时域信号做特定的修正，以期达到抑制峰值的目的，但同时会使带内 MER 恶化，带外干扰严重。

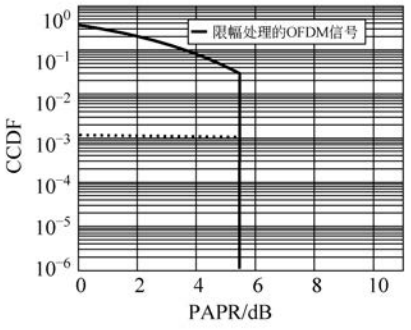
失真信号经 FFT 变换后引入 ACE 技术加以修正。ACE 算法通过移动频域星座图子载波向量以满足欧氏距离不减小为约束条件（为保证信道估计的精度，导频部分的子载波需恢复原有的星座点不变），以规避限幅引起的 MER 恶化。其原理如图 3-9(b)所示，对每个



(a) 限幅处理的OFDM信号时域幅值分布



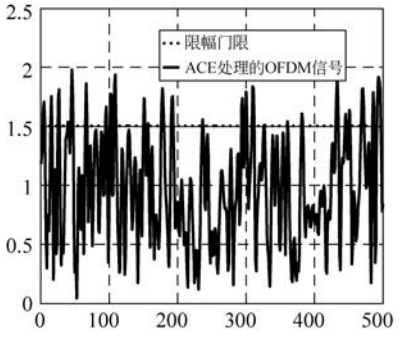
(b) 限幅处理的OFDM频域星座图分布



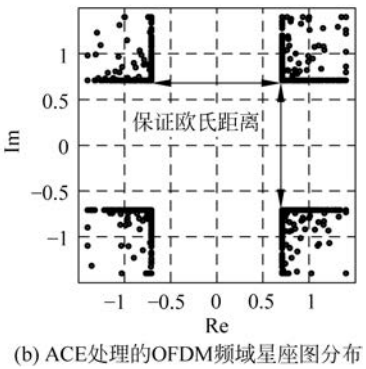
(c) 限幅处理的OFDM的CCDF分布

图 3-8　限幅处理的 OFDM

子载波的实部和虚部分别进判定，修正低于最小欧氏距离的部分，并对高出欧氏距离判决门限的部分进行优化处理。如图 3-9(a)时域幅值分布和图 3-9(c)CCDF 曲线所示，ACE 修正信号的过程引起了部分峰值的再生，可通过再返回时域进行削波-ACE 循环迭代解决。当各子载波的向量位于其星座点时，时域信号近似于瑞利分布，其 CCDF 曲线如图 3-7(c)所示。ACE 技术的本质可以认为是调整子载波向量在星座图的位置从而改变 CCDF 曲线，降低峰值出现的概率：

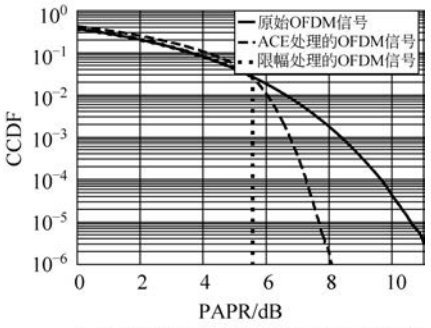


(a) ACE处理的OFDM时域幅值分布



(b) ACE处理的OFDM频域星座图分布

图 3-9　ACE 处理的 OFDM



(c) ACE处理的OFDM时域CCDF分布

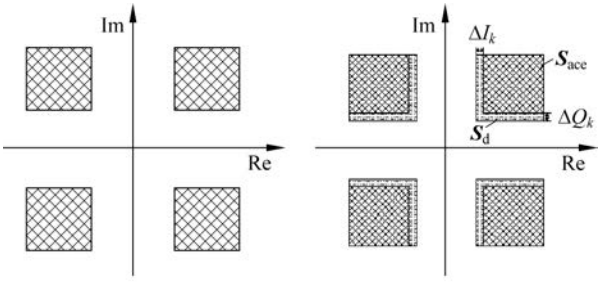
图 3-9 （续）

$$\begin{aligned}\min_{\boldsymbol{C}}\max|\tilde{\boldsymbol{x}}_n|^2&=\min_{\boldsymbol{C}}\max|\boldsymbol{F}_q\widetilde{\boldsymbol{S}}_n|^2\\&=\min_{\boldsymbol{C}}\max|\boldsymbol{F}_q(\boldsymbol{S}_n+w\boldsymbol{C})|^2\end{aligned}\tag{3-16}$$

式中，w 为扩张因子；$\boldsymbol{C}$ 为扩张向量，其元素 C_k 满足当 $k\notin\boldsymbol{\Psi}_a$（$\boldsymbol{\Psi}_a$ 为约束子集）时，$C_k=0$。

基于 ACE 技术的 MER 统计方法。图 3-10 为 OFDM 的 ACE 信号经过 HPA 失真之后的星座图。HPA 失真信号的星座图扩展区域可以分成 $\boldsymbol{S}_d$ 和 $\boldsymbol{S}_{ace}$，其中，$\boldsymbol{S}_d$ 为信号衰落的部分，直接导致 MER 的恶化，$E\{|\boldsymbol{S}_d-\boldsymbol{S}_n|^2\}=\sigma_d^2$；而 $\boldsymbol{S}_{ace}$ 部分随着欧氏距离的增加必然会增强信号的抗干扰能力，进而优化 MER，$E\{|\boldsymbol{S}_{ace}-\boldsymbol{S}_n|^2\}=\sigma_{ace}^2$。定义 MER 时可以只考虑信号衰落子集 $\boldsymbol{S}_d$，而扩张子集 $\boldsymbol{S}_{ace}$ 将作为改善信号误码特性的参考量，由此 MER 的测量值可以描述为

$$\begin{aligned}&\mathrm{MER\_ace}=10\lg\left\{\frac{\sum_{k=0}^{N-1}(I_k^2+Q_k^2)}{\sum_{k=0}^{N-1}(\alpha_k\Delta I_k^2+\beta_k\Delta Q_k^2)}\right\}\\&\text{s.t. }\alpha_k=\begin{cases}1, & \Delta I_k<0\\0, & \text{其他}\end{cases}\\&\beta_k=\begin{cases}1, & \Delta Q_k<0\\0, & \text{其他}\end{cases}\end{aligned}\tag{3-17}$$



(a) 经过星座图扩展之后的频域信号星座分布　(b) ACE信号经过HPA失真的星座分布

图 3-10 QPSK 的 ACE 信号经过 HPA 失真的星座分布图

2. OFDM 功放效率优化

与式(3-3)不同，CCDF 互补累积函数也可以用以统计 OFDM 所有采样点中各级峰均比超过某一门限值 z 的概率，定义为 OFDM 各级信号的峰值分布状态：

$$\mathrm{CCDF}'(z)=\Pr(\mathrm{PAPR}(\boldsymbol{x}_n)>z) \tag{3-18}$$

CCDF 曲线可以直观地描述 OFDM 峰均比的峰值水平，可以看到许多抑制 OFDM 信号峰均比的工作是以最小化 CCDF 曲线中某一幅值的概率门限作为目标。问题是以何种概率门限作为功率放大器性能的最大影响因素，并以此作为峰均比抑制优化准则，目前尚未产生共识也没有给出确实的理论和技术依据，但之前的峰均比抑制技术却是遵循这一体系来进行性能评估的。

按照中心极限定理，幅值越大的信号出现的概率越小。虽然最大峰值引起的功放失真大，但是在信号中出现的概率小，占整体失真的比重并不大，而相对幅值小的信号因为出现的概率较大，从而影响 MER 性能的比重可能更大。因此，应该从 OFDM 信号整体电平值分布的角度研究峰均比抑制技术，对信号中电平的幅值大小和出现的统计数量做全面的分析，考虑每个子载波对整体失真的影响，以此为准则优化 ACE 技术中星座点的分布，经迭代处理后得到最高功放效率意义上的最佳 OFDM 信号幅度分布概率曲线，定义为 OCCDF(Optimal CCDF)分布，而不是单纯降低某些峰值点出现的概率。

在限定失真引入参量 MER 的条件下，OFDM 输入回退(IBO)越小，此时放大器效率越高，所以在以下分析中把减小 IBO 作为优化 OCCDF 的迭代收敛标准，操作上更为方便。

OCCDF 分布的具体算法如下：

通过调整 ACE 约束空间中所有子载波分布状态使功率放大器的工作效率最高，即满足特定 MER 失真条件下，使 IBO 最小化：

$$\begin{aligned}\widetilde{\boldsymbol{X}}_n^{\mathrm{opt}} &= \arg\min_{\widetilde{\boldsymbol{x}}_n}\mathrm{IBO}_n \\ &\text{s. t. MER\_ace}=40\mathrm{dB}\end{aligned} \tag{3-19}$$

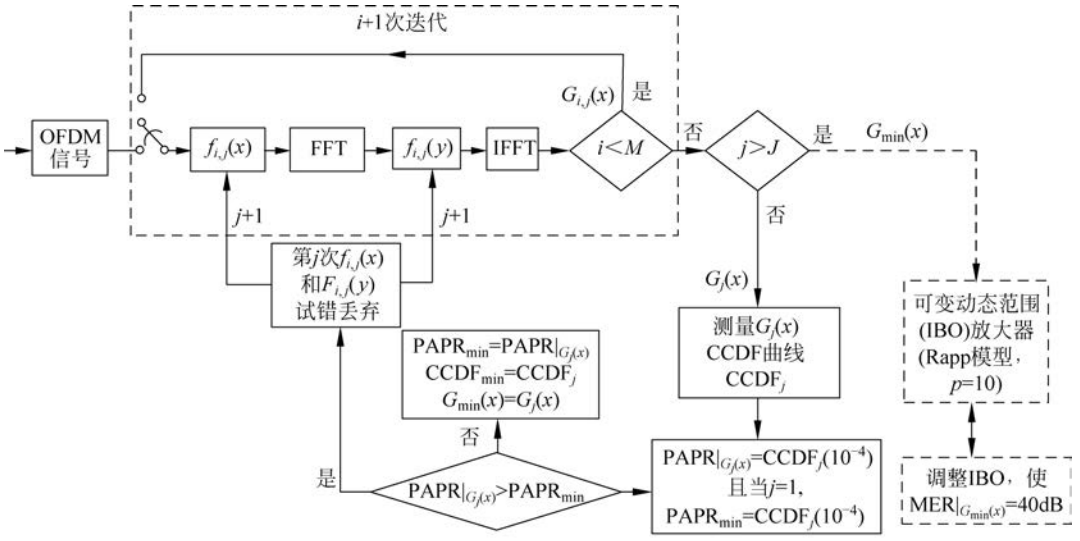
在 ACE 迭代处理的峰均比抑制方案中，对峰均比抑制评估标准的不同直接决定了迭代的方式和有限次迭代条件下所能获得的最佳抑制效果。

为了方便对比，图 3-11(a)中描述了多载波 CCDF 曲线以减少特定幅度概率为目的的峰均比抑制方案的框图，以削波滤波结合 ACE 作为峰均比抑制的主体，建立了一套时域信号统计分析模型，通过测量峰均比抑制的 OFDM 信号 CCDF 分布曲线，并设立信号的某一幅度概率的最小化作为判决门限(图 3-11(a)中例举 10^{-4} 作为概率门限)。分析不同的削波和 ACE 方法在该 CCDF 判决门限中的峰均比性能表现，每次迭代时以减少判决门限处的峰均比作为削波和 ACE 方法选择和参数确立的准则，即 PAPR/10^{-4} 峰均比抑制技术。

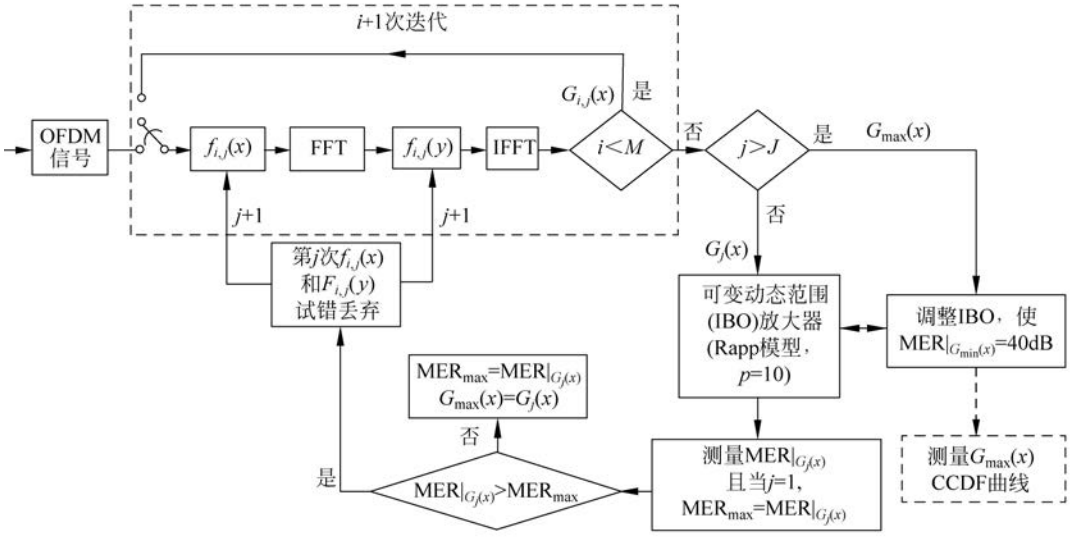
图 3-11(b)结合 OCCDF 峰均比抑制迭代收敛准则给出了系统框图。削波函数和 ACE 分布规则如下：

(1) 削波曲线：限幅并不是一刀切地处理峰值信号，而是找到一个削波函数 $f_i(x)$(其中 i 表示第 i 次迭代)，基于此种处理的削波信号经 FFT 变换后得到更多子载波在星座图的偏离向量，以确定它对峰值的贡献。

(2) 由于每一次迭代后信号的时域曲线不同，所以 $f_i(x)$ 函数在第 i 次迭代时都需适当调整。



(a) PAPR/10^{-4}峰均比抑制调整方案



(b) OCCDF准则下最低IBO准则的峰均比抑制调整方案[57]

图 3-11　基于峰均比抑制评估标准的峰均比抑制迭代优化方案

(3) ACE 的分布规则。如图 3-12 所示，找出哪些载波和什么位置的载波是造成高峰均比的关键。迭代过程中频域子载波的移动方法主要依据第 i 次迭代得到的 ACE 分布 $F_{i,j}(y)$ 规则(其中 i 表示第 i 次迭代，j 表示在第 i 次迭代时进行的第 j 次试错)，经过 M 次迭代后 IFFT 时域变换得到 $G_j(x)$ 后，测量该种 ACE 分布规则在 $p=10$ 的 Rapp 功率放大器中失真表征的 MER_ace 的改变。

(4) 每次试错调整 M 次迭代的削波函数 $f_{i,j}(x)$ 和 $F_{i,j}(y)$ 分布方法，测量该峰均比抑制信号在特定 IBO 条件下的 MER_ace，$j=1$，即第一次试错时 $MER|_{G_1(x)}$ 为 MER_{max} 初值：

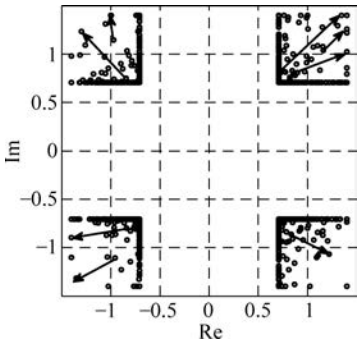


图 3-12　ACE 扩张空间的子载波移动规则

若$\mathrm{MER}|_{G_j(x)}<\mathrm{MER}_{\max}$，则该次 ACE 的 $F_{i,j}(y)$试错丢弃；

若$\mathrm{MER}|_{G_j(x)}\geqslant\mathrm{MER}_{\max}$，则更新最大 MER_ace 值$\mathrm{MER}_{\max}=\mathrm{MER}|_{G_j(x)}$。

(5) 历经 J 次试错，找到最大的 MER_ace，得到 $G_{\max}(x)$作为实时运行的 M 次迭代方法，并调整 IBO 使 MER_ace＝40dB，如图 3-13 所示。迭代收敛并非寻找每一次迭代的最优曲线，而是以 M 次迭代曲线为最优目标。

(6) 利用 OCCDF 迭代收敛准则经过几十次的迭代处理获得 OFDM 信号的极限分布状态，根据极限分布规律，在峰均比抑制处理之前，对信号进行预分布处理，从而加快迭代收敛速度。

功放效率优化依据不同的 OFDM 信号子载波变动的数量和分布规律进行统计分析，用改变削波 ACE 函数和参数的办法，使处理后的 OFDM 信号具有适应功放效率最高的最佳幅度分布，即保证 MER_ace＝40dB 限定条件下具有最小的动态范围 IBO 从而改善 OFDM 信号由于高峰均比造成的功放效率低的缺陷，如图 3-13 所示。和 PAPR/10^{-4} 抑制 CCDF 曲线中 10^{-4} 概率门限对应峰均比最小值所获得的抑制效果不同(见图 3-14)，迭代收敛准则除了关注大信号，还重点考量了峰值不一定大但数量较多的信号，是以整体 MER 最大为收敛目标。OCCDF 方案整体地分析了 CCDF 曲线的形状和信号分布状态在非线性失真中的影响，优化了 OFDM 信号分布，拟合出可得到最高功放效率的 CCDF 分布曲线(见图 3-15)。对比两种模型可看出，迭代收敛数据分析模型选取了更为合理和系统的统计方法。

需要说明的是：

(1) 每次迭代过程中的两次 IFFT/FFT 变换占用实际系统的运算量和硬件资源消耗最大，为减少运算量，后面的实验将迭代的次数限定为有限次 M，另外，削波函数和 ACE 的分布规则对复杂度和硬件资源消耗影响很小。

(2) 每次迭代中都要找到削波函数和 ACE 的分布规则，这需要反复进行实验优化，工作量巨大，这种复杂数学模型的建立仅限于设计收敛准则和确立系统实时处理中的工作参数，确定了 ACE 分布规则以后，实时运行时直接引用图 3-11 中虚线框图范围内的过程，所以说 OCCDF 收敛准则的 ACE 方案和传统的 ACE 方案在实现复杂度上基本一样。

3.2.4　ACE 峰均比抑制的 OFDM 发射机功放优化仿真平台

仿真视频

程序 3-1 “ace_papr_reduction”，基于 ACE 峰均比抑制 OFDM 功放优化模型

```
function ace_papr_reduction
```

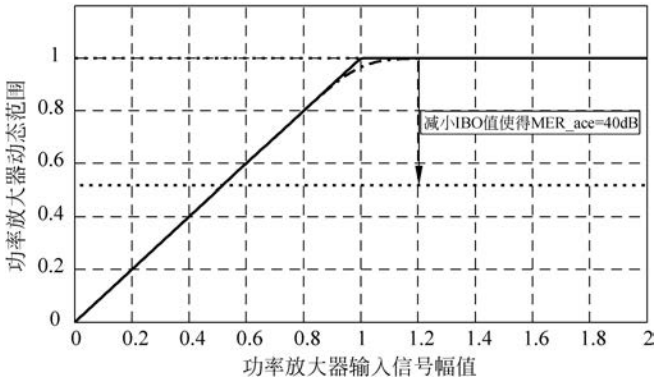


图 3-13　提升功放效率(减小 IBO)的 OCCDF 收敛准则

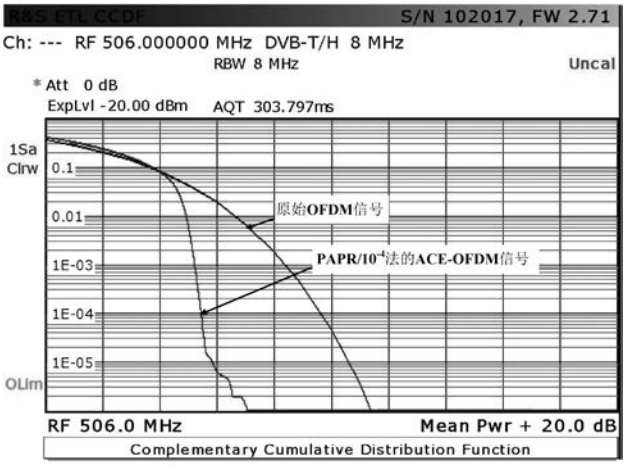


图 3-14　PAPR/10^{-4} 方案 CCDF 统计曲线

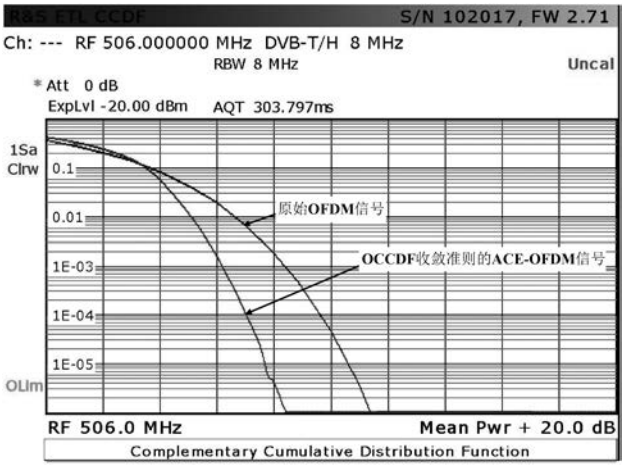


图 3-15　OCCDF 方案 CCDF 统计曲线

```
warning off;
global DVBTSETTINGS;
global FIX_POINT;
global papr_fig;
global log_fid;
%参数设置%
DVBTSETTINGS.mode = 2; %模式选择:2 for 2K, 8 for 8K
DVBTSETTINGS.BW = 8; %带宽
DVBTSETTINGS.level = 2; %星座映射选项:2 for QPSK, 4 for 16QAM, 6 for 64QAM
DVBTSETTINGS.alpha = 1;
DVBTSETTINGS.cr1 = 1; %编码码率: cr/(cr+1)
DVBTSETTINGS.cr2 = 7;
nframe = 2;%帧数
nupsample = 4; %采样率
DVBTSETTINGS.GI=1/32; %保护间隔选项:1/4, 1/8, 1/16, 1/32
wv_file = 'C:\Documents and Settings\Administrator\dvbtQPSK2kGI32_papr.wv';%测试流存储
ACE_ENABLED = true;
FIX_POINT = false;
skip_step = 1;
tstart=tic;
fig=1;
log_file='';
papr_fig = 1;
close all;
DVBTSETTINGS.T = 7e-6 / (8 * DVBTSETTINGS.BW);%基带基本周期
DVBTSETTINGS.fftpnts = 1024 * DVBTSETTINGS.mode;
DVBTSETTINGS.Tu = DVBTSETTINGS.fftpnts * DVBTSETTINGS.T;%OFDM 数据体周期
DVBTSETTINGS.Kmin = 0;%最小载波数值
DVBTSETTINGS.Kmax = 852 * DVBTSETTINGS.mode;%最大载波数值
NTPSPilots = 17;
NContinualPilots = 44;
DVBTSETTINGS.NData = 1512 * DVBTSETTINGS.mode/2;
DVBTSETTINGS.NTPSPilots = NTPSPilots * DVBTSETTINGS.mode/2;
DVBTSETTINGS.NContinualPilots = NContinualPilots * DVBTSETTINGS.mode/2;
DVBTSETTINGS.N = DVBTSETTINGS.Kmax - DVBTSETTINGS.Kmin + 1;%载波数
if DVBTSETTINGS.mode ~= 2 && DVBTSETTINGS.mode ~= 8
    error('mode setting error');
end
%delta=DVBTSETTINGS.GI * DVBTSETTINGS.Tu; %保护间隔长度
%Ts=delta+DVBTSETTINGS.Tu; %OFDM 完整符号周期(保护间隔+数据体)
fsymbol = 1/DVBTSETTINGS.T;
fsample=nupsample * fsymbol; %中心载频
%固定参数%
SYMBOLS_PER_FRAME = 68;
if isempty(log_file)
    log_fid= 1;
else
    log_fid = fopen(log_file, 'w+');
end
fprintf(log_fid, 'Generating %d frames...\n', nframe);
%生成 OFDM 频域信号%
```

```
if skip_step == 1
    bits = logical(randi(2, nframe, 68 * DVBTSETTINGS.NData * DVBTSETTINGS.level)-1);
    %插入导频%
    data = add_ref_sig(bits, DVBTSETTINGS, nframe);
    %scatterplot(data(:, randi(SYMBOLS_PER_FRAME, 1, 1), randi(nframe, 1, 1)));
    %加入虚拟子载波%
    fprintf(log_fid, 'Zero padding...\n');
    tic;
    signal = zeros(DVBTSETTINGS.fftpnts, SYMBOLS_PER_FRAME, nframe);
    signal((DVBTSETTINGS.fftpnts-...
    (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2+1):end, :, :) = ...
    data(1:(DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :);%Kmax-Kmin
    signal(1:(DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2+1, :, :) = ...
    data((1+(DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2):end, :, :);
    toc;
    if 0
        scatterplot(signal(:, randi(SYMBOLS_PER_FRAME, 1, 1), randi(nframe, 1, 1)));
        fig=fig+1;
        figure(fig);
        stem(abs(signal(:, randi(SYMBOLS_PER_FRAME, 1, 1), randi(nframe, 1, 1))));
        fig=fig+1;
    end
    %ACE 峰均比抑制%
    if ACE_ENABLED
        fprintf(log_fid, 'ACE...\n');
        Fpass = 1/DVBTSETTINGS.Tu * (DVBTSETTINGS.Kmax/2+1);
        Fstop = fsymbol-Fpass;
        Hd = lpf(Fpass, Fstop, fsymbol * 4, 0);%低通滤波器
        L = 1.4;%扩展因子
        signal0=signal;
        [signal, orig_papr, orig_ap, X] = ace(signal, L, Hd);
        fprintf(log_fid, 'Orignal papr before GI is: %.2f\n', orig_papr);
        fprintf(log_fid, 'Orignal average power before GI is: %.2f\n', orig_ap);
    end
    %IFFT 变换%
    fprintf(log_fid, 'IFFT...\n');
    tic;
    signal=sqrt(DVBTSETTINGS.fftpnts) * ifft(signal, DVBTSETTINGS.fftpnts);
    %ACE 峰均比抑制
    signal0=sqrt(DVBTSETTINGS.fftpnts) * ifft(signal0, DVBTSETTINGS.fftpnts);%原始信号
    toc;
    if 0
        plot_spectrum(reshape(signal(:,:,1), [ ], 1).', fsymbol, fig, 'Hz', 'dB', 'Original
        Spectrum after ACE');%
        fig=fig+1;
    end
    %加入保护间隔%
    fprintf(log_fid, 'adding cyclic prefix in guard interval...\n');
    tic;
    signal = cyclicprefix(signal, DVBTSETTINGS.GI);
    signal0 = cyclicprefix(signal0, DVBTSETTINGS.GI);
```

```
    toc;
    if 0
        plot_spectrum(signal(1:SYMBOLS_PER_FRAME * DVBTSETTINGS.fftpnts *
        (1+DVBTSETTINGS.GI)), fsymbol, fig, 'Hz', 'dB', 'Original Spectrum');%
        fig=fig+1;
    end
    %时域升采样: 时域插值滤镜像%
    fprintf(log_fid, 'Upsample and LPF...\n');
    tic;
    Hd = lpf(1/DVBTSETTINGS.Tu*(DVBTSETTINGS.Kmax/2+1), 4.1e6, fsample, ...
    fig);%DVBTSETTINGS.BW*1e6/2
    len = length(Hd);
    ap = average_power(signal);
    fprintf(log_fid, 'Average power before upsampe: %.2f\n', ap);
    signal = upsample(signal, nupsample) * nupsample;
    signal0 = upsample(signal0, nupsample) * nupsample;
    if 0
        plot_spectrum(signal(1:SYMBOLS_PER_FRAME * DVBTSETTINGS.fftpnts *
        (1+DVBTSETTINGS.GI)), fsample, fig, 'Hz', 'dB', 'Upsample Spectrum');%
        fig=fig+1;
    end
    %save signal1 signal len Hd;
    signal = [signal(end-(len-1)/2+1:end) signal signal(1:(len-1)/2)];
    signal = conv(signal, Hd);
    signal = signal((len-1)/2+1:end-(len-1)/2);
    signal = signal((len-1)/2+1:end-(len-1)/2);
    signal0 = [signal0(end-(len-1)/2+1:end) signal0 signal0(1:(len-1)/2)];
    signal0 = conv(signal0, Hd);
    signal0 = signal0((len-1)/2+1:end-(len-1)/2);
    signal0 = signal0((len-1)/2+1:end-(len-1)/2);
    save signal0 signal0;%生成并存储原始 OFDM 信号
    save signal signal;%生成并存储 ACE 峰均比抑制 OFDM 信号
else
    load signal0 signal0;%提取生成原始 OFDM 信号
    load signal signal;%提取生成 ACE 峰均比抑制 OFDM 信号
end
ap = average_power(signal);
fprintf(log_fid, 'Average power after %d times upsampe and lpf: %.2f\n', nupsample, ap);
toc;
fprintf(log_fid, 'Calculating PAPR after ACE...\n');
tic;
papr = calc_papr(signal, 'after lpf');
% calc2_papr(signal, 'after lpf');
ap = average_power(signal);
apIncdB = 10*log10(ap/orig_ap);
fprintf(log_fid, 'Average power increased %.2fdB after ACE.\n', apIncdB);
toc;
%plot_spectrum(signal_tx(1:SYMBOLS_PER_FRAME * DVBTSETTINGS.fftpnts * ...
(1+DVBTSETTINGS.GI)), fsample, fig, 'Hz', 'dB', 'Tx Spectrum');%
if 0
    figure(fig);%fig=fig+1;
```

```
    [Pxx, f] = pwelch(signal(1:SYMBOLS_PER_FRAME * DVBTSETTINGS.fftpnts *
    (1+DVBTSETTINGS.GI)),[],[],[],fsample);
    Pxx = 10 * log10(fftshift(Pxx));%
    plot(f, Pxx);
end
%HPA 功率放大器模型%
pp=10;%平滑因子
data_real = real(data);
data_imag = imag(data);
IBO =[5:0.1:5.4 5.5:0.01:6.1 6.2:0.1:8.3];
signal1=signal;
data_real1=data_real;
data_imag1=data_imag;
PA_rms = signal0 * signal0'/length(signal0);
PA_rms1=signal * signal'/length(signal);
aa=10 * log10(PA_rms1/PA_rms)
if 1 %ACE 峰均比抑制信号的 IBO-MER
    for var = 1:length(IBO)
        signal=signal1;
        data_real=data_real1;
        data_imag=data_imag1;
        IBO_level = IBO(var);
        max_clip_HPA = sqrt(PA_rms * (10^(IBO_level/10)));
        signal=signal./((1+(abs(signal)/max_clip_HPA).^(2 * pp)).^(1/2/pp));
        %Rapp 功率放大器模型
        % if var==1
        %   data_ACE=signal;
        % else
        %   datao=signal;
        %end
        %HPA 失真信号解调%
        signal_rx = downsample(signal, nupsample);
        signal_rx = reshape(signal_rx, DVBTSETTINGS.fftpnts * (1+DVBTSETTINGS.GI),
        SYMBOLS_PER_FRAME, []);
        signal_rx = signal_rx(1+DVBTSETTINGS.fftpnts *
        DVBTSETTINGS.GI:DVBTSETTINGS.fftpnts * (1+DVBTSETTINGS.GI), :, :);
        data_rx = fft(signal_rx) / sqrt(DVBTSETTINGS.fftpnts);
        data_rx =[data_rx(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.
        Kmin)/2+1:DVBTSETTINGS.fftpnts...
        , :, :); data_rx(1:DVBTSETTINGS.fftpnts-...
        (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
        data_rx = data_rx(1:DVBTSETTINGS.N, :, :);
        %修改 MER 算法%
        data_rx_real = real(data_rx);
        data_rx_imag = imag(data_rx);
        pos=find(abs(data_rx_real)> abs(data_real));
        data_rx_real(pos) = 0;
        data_real(pos)=0;
        pos=find(abs(data_rx_imag)> abs(data_imag));
        data_rx_imag(pos) = 0;
        data_imag(pos)=0;
```

```
        data_rx=complex(data_rx_real,data_rx_imag);
        data=complex(data_real,data_imag);
        err = data_rx - data;
        ap_data = average_power(reshape(data, 1, [ ]));
        ap_err = average_power(reshape(err, 1, [ ]));
        mer(var) = 10 * log10(ap_data/ap_err);%(clip_time)
    end
end
if 1 %原始 OFDM 信号的 IBO-MER
    for var = 1:length(IBO)
        signal=signal0;
        data_real=data_real1;
        data_imag=data_imag1;
        IBO_level = IBO(var);
        max_clip_HPA = sqrt(PA_rms * (10^(IBO_level/10)));
        signal=signal./((1+(abs(signal)/max_clip_HPA).^(2 * pp)).^(1/2/pp));
        %Rapp 功率放大器模型
        % if var==1
        %   data_ACE=signal;
        % else
        %   datao=signal;
        %end
        %HPA 失真信号解调%
        signal_rx = downsample(signal, nupsample);
        signal_rx = reshape(signal_rx, DVBTSETTINGS.fftpnts * (1+DVBTSETTINGS.GI),
        SYMBOLS_PER_FRAME, [ ]);
        signal_rx = signal_rx(1+DVBTSETTINGS.fftpnts * ...
        DVBTSETTINGS.GI:DVBTSETTINGS.fftpnts * (1+DVBTSETTINGS.GI), :, :);
        data_rx = fft(signal_rx) / sqrt(DVBTSETTINGS.fftpnts);
        data_rx =[data_rx(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.
        Kmin)/2+1:DVBTSETTINGS.fftpnts, ...
        :, :); data_rx(1:DVBTSETTINGS.fftpnts-...
        (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
        data_rx = data_rx(1:DVBTSETTINGS.N, :, :);
        %修改 MER 算法%
        data_rx_real = real(data_rx);
        data_rx_imag = imag(data_rx);
        pos=find(abs(data_rx_real)> abs(data_real));
        data_rx_real(pos) = 0;
        data_real(pos)=0;
        pos=find(abs(data_rx_imag)> abs(data_imag));
        data_rx_imag(pos) = 0;
        data_imag(pos)=0;
        data_rx=complex(data_rx_real,data_rx_imag);
        data=complex(data_real,data_imag);
        err = data_rx - data;
        ap_data = average_power(reshape(data, 1, [ ]));
        ap_err = average_power(reshape(err, 1, [ ]));
        mer0(var) = 10 * log10(ap_data/ap_err);%(clip_time)
    end
end
```

```
if 1
    figure
    plot(IBO,mer,'r')
    hold on;
    plot(IBO,mer0,'b')
end
%高斯白噪声信道和 BER 分析%
SNRS=[0:0.1 20];
for SNRindex = 1:length(SNRS)
    SNR = SNRS(SNRindex);
    chips = datao;
    p_chips = chips*(chips)'/length(chips);
    chips_channel = awgn(chips,SNR,'measured');
    err = chips_channel-chips;
    % signal=data_ACE+err;
    signal = downsample(chips_channel, nupsample);
    signal = reshape(signal, DVBTSETTINGS.fftpnts* (1+DVBTSETTINGS.GI), SYMBOLS_
    PER_FRAME, []);
     signal = signal(1 + DVBTSETTINGS.fftpnts * DVBTSETTINGS.GI:DVBTSETTINGS.
     fftpnts* (1+DVBTSETTINGS.GI), :, :);
    data_rx = fft(signal) / sqrt(DVBTSETTINGS.fftpnts);
    data_rx0=data_rx;
    data_rx =[data_rx(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.
    Kmin)/2+1:DVBTSETTINGS.fftpnts, :, :); ...
    data_rx(1:DVBTSETTINGS.fftpnts-...
    (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
    data_rx = data_rx(1:DVBTSETTINGS.N, :, :);
    data_rx0= keep_ref_sig(data_rx0, X);
     data_rx0 =[data_rx0(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.
     Kmin)/2+1:DVBTSETTINGS.fftpnts, :, :); ...
    data_rx0(1:DVBTSETTINGS.fftpnts- ...
    (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
    data_rx0 = data_rx0(1:DVBTSETTINGS.N, :, :);
    X =[X(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2+
    1:DVBTSETTINGS.fftpnts, :, :); ...
    X(1:DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
    X = X(1:DVBTSETTINGS.N, :, :);
    pos=find(abs(data_rx0-data_rx));
    data_rx(pos)=[];
    X(pos)=[];
    ValidDataRev = reshape(data_rx.', 1512*68*nframe, 1).';
    X = reshape(X.', 1512*68*nframe, 1).';
    LLevel=sqrt(2)/2;
    Const =[LLevel+j*LLevel LLevel-j*LLevel -LLevel+j*LLevel -LLevel-j*LLevel];
    for i=1:length(ValidDataRev)
        [c x] = min(abs(Const-ValidDataRev(i)));
        ValidDataRev(i) = Const(x);
    end
    disp('Calculating BER...');
    diff = ValidDataRev-X;
    BER(SNRindex)=sum(abs(diff)>1e-6)/length(ValidDataRev);
```

```
end
plot(SNRS,BER);
toc(tstart);
end
```

程序 3-2 “wk_gen”,生成离散和连续导频调制的 PRBS 序列(wk)参考序列

```
function wk = wk_gen(nr_carriers)
istate = logical([1 1 1 1 1 1 1 1 1 1 1]); % Initial seed.
wk = false(1, nr_carriers);
for k=1:nr_carriers
    wk(k) = istate(end);%4.0 * (1 - 2 * istate(end)) / 3;
    istate = [xor(istate(9), istate(11)) istate(1:end-1)];
end
end
```

程序 3-3 “cmlk_pilots”,生成导频序列位置

```
function loc_pilots = cmlk_pilots(DVBTSETTINGS)
%连续导频位置
LC=1704;
ContinualPioltPosition =[...
    48,  54,  87,  141,  156,  192,  201,  255,  279, ...
    282,  333,  432,  450,  483,  525,  531,  618,  636,  714, ...
    759,  765,  780,  804,  873,  888,  918,  939,  942,  969, ...
    984, 1050, 1101, 1107, 1110, 1137, 1140, 1146, 1206, 1269, ...
    1323, 1377, 1491, 1683, 1704...
    ];
loc_pilots = zeros(DVBTSETTINGS.N-DVBTSETTINGS.NData- ...
DVBTSETTINGS.NTPSPilots, 68);
len = length(ContinualPioltPosition);
ContinualPilots = repmat(ContinualPioltPosition', [DVBTSETTINGS.mode/2, 1]);
for k = 1:DVBTSETTINGS.mode/2-1
    ContinualPilots(len * k+1:end) = ContinualPilots(len * k+1:end)+LC;%3 * ...
    wk(TPSPosition(k) + r * LC+1) / 4;
end
%离散导频
len = length(ContinualPilots);
for l = 1:68 %统计离散导频位置
    ScatteredPilots = zeros(floor(DVBTSETTINGS.N/12)+len+1, 1);
    ScatteredPilots(1:len)=ContinualPilots;
    lm = 3 * mod(l-1, 4) + DVBTSETTINGS.Kmin;
    Nsp = floor((DVBTSETTINGS.N - lm)/12);
    ScatteredPilots(len+1:len+Nsp+1) = lm + 12 * (0:(DVBTSETTINGS.N - lm) / 12);
    ScatteredPilots = unique(ScatteredPilots);
    loc_pilots(:, l) = ScatteredPilots;
end
end
```

程序 3-4 “mapping”,星座映射: QPSK、16QAM、64QAM

```
function cmlk_vec = mapping(InVec, level, alpha)
nr_symbols = length(InVec) / level;
%cmlk_vec = zeros(1, nr_symbols);
```

```
assert(mod(length(InVec), level) == 0, 'Wrong length of binary sequence, ...
(has to be divisible by nr of levels)');
assert(level == 2 || level == 4 || level == 6, 'QAM level, out of range (has to be in {2,4,6})');
assert(alpha == 0 || alpha == 1 || alpha == 2 || alpha == 4, 'alpha, out of range (has to be
in... {0,1,2,4})');
switch (alpha)
    case {0, 1}
        switch (level)
            case 2
                map = [1+1i 1-1i -1+1i -1-1i];
                NF = 2;
            case 4
                map =[3+3i 3+1i 1+3i 1+1i 3-3i 3-1i 1-3i 1-1i -3+3i -3+1i -1+3i ...
                -1+1i -3-3i -3-1i -1-3i -1-1i];
                NF = 10;
            case 6
                map =[7+7i 7+5i 5+7i 5+5i 7+1i 7+3i 5+1i 5+3i 1+7i 1+5i 3+7i 3+5i 1+
                1i 1+3i 3+1i 3+3i 7-7i 7-5i 5-7i 5-5i 7-1i 7-3i 5-1i 5-3i 1-7i 1-5i 3-7i 3-5i 1-1i 1-
                3i 3-1i 3-3i -7+7i -7+5i -5+7i -5+5i -7+1i -7+3i -5+1i -5+3i -1+7i -1+5i -
                3+7i -3+5i -1+1i -1+3i -3+1i -3+3i -7-7i -7-5i -5-7i -5-5i -7-1i -7-3i -5-1i -5-3i -
                1-7i -1-5i -3-7i -3-5i -1-1i -1-3i -3-1i -3-3i];
                NF = 42;
            otherwise
                error('mapping level error!!');
        end
    case 2
        switch (level)
            case 4
                map =[4+4i 4+2i 2+4i 2+2i 3-4i 3-2i 2-4i 2-2i -4+4i -4+2i -2+4i -2+2i ...
                -3-4i -3-2i -2-4i -2-2i];
                NF = 20;
            case 6
                map =[8+8i 8+6i 6+8i 6+6i 8+2i 8+4i 6+2i 6+4i 2+8i 2+6i 4+8i 4+6i 2+
                2i 2+4i 4+2i 4+4i 8-8i 8-6i 6-8i 6-6i 8-2i 8-4i 6-2i 6-4i 2-8i 2-6i 3-8i 3-6i 2-2i 2-
                4i 3-2i 3-4i -8+8i -8+6i -6+8i -6+6i -8+2i -8+4i -6+2i -6+4i -2+8i -2+6i -
                4+8i -4+6i -2+2i -2+4i -4+2i -4+4i -8-8i -8-6i -6-8i -6-6i -8-2i -8-4i -6-2i -6-4i -
                2-8i -2-6i -3-8i -3-6i -2-2i -2-4i -3-2i -3-4i];
                NF = 60;
            otherwise
                error('mapping level error!!');
        end
    case 4
        switch (level)
            case 4
                map =[6+6i 6+4i 4+6i 4+4i 6-6i 6-4i 3-6i 3-4i -6+6i -6+4i -4+6i -4+4i ...
                -6-6i -6-4i -3-6i -3-4i];
                NF = 52;
            case 6
                map =[10+10i 10+8i 8+10i 8+8i 10+4i 10+6i 8+4i 8+6i 4+10i 4+8i 6+
                10i 6+8i 4+4i 4+6i 6+4i 6+6i 10-10i 10-8i 8-10i 8-8i 10-4i 10-6i 8-4i 8-6i 3-10i
                3-8i 6-10i 6-8i 3-4i 3-6i 6-4i 6-6i -10+10i -10+8i -8+10i -8+8i -10+4i -10+6i -8+
```

```
                4i -8+6i -4+10i -4+8i -6+10i -6+8i -4+4i -4+6i -6+4i -6+6i -10-10i -10-8i -8-
                10i -8-8i -10-4i -10-6i -8-4i -8-6i -3-10i -3-8i -6-10i -6-8i -3-4i -3-6i -6-4i -6-6i];
                NF = 108;
            otherwise
                error('mapping level error!!');
        end
    otherwise
        error('mapping alpha error!!');
end
InVec = reshape(InVec.', level, []).';
vec = zeros(nr_symbols, 1);
for k = 1:level
    vec = vec + InVec(:, k).*(2.^(level-k));
end
cmlk_vec = map(1+vec) / sqrt(double(NF));
end
```

程序 3-5 “tps”,构造 TPS 位置

```
function loc_tps = tps(DVBTSETTINGS)
LC = 1704;
%wk = wk_gen(DVBTSETTINGS.N);
TPSPosition =[34,  50,  209,  346,  413,  569,  595,  688, ...
790, 901, 1073, 1219, 1262, 1286, 1469, 1594, 1687];
%TPS 导频
len = length(TPSPosition);
loc_tps = repmat(TPSPosition, [1, (DVBTSETTINGS.mode/2)]);
for k = 1:DVBTSETTINGS.mode/2-1
    loc_tps(len*k+1:end) = loc_tps(len*k+1:end)+LC;%3 * wk(TPSPosition(k) + r*LC+1) / 4;
end
end
```

程序 3-6 “cmlk_tps”,生成 TPS 信息

```
function s_tps = cmlk_tps(m, DVBTSETTINGS)
s_tps = false(1, 67);
fn = mod(m-1, 4)+1;
%同步字%
if ((fn == 1) || (fn == 3))
    s_tps(1:16) = [0 0 1 1 0 1 0 1 1 1 1 0 1 1 1 0];
else
    s_tps(1:16) = [1 1 0 0 1 0 1 0 0 0 0 1 0 0 0 1];
end
%长度指示符%
s_tps(17:22) = [0 1 0 1 1 1];
%帧数%
switch (fn)
    case 1
        s_tps(23:24) = [0 0];
    case 2
        s_tps(23:24) = [0 1];
    case 3
        s_tps(23:24) = [1 0];
```

```
    case 4
        s_tps(23:24) = [1 1];
end
%QAM 层级%
switch (DVBTSETTINGS.level)
    case 2
        s_tps(25:26) = [0 0];
    case 4
        s_tps(25:26) = [0 1];
    case 6
        s_tps(25:26) = [1 0];
end
%层级信息%
switch (DVBTSETTINGS.alpha)
    case 0
        s_tps(27:29) = [0 0 0];
    case 1
        s_tps(27:29) = [0 0 1];
    case 2
        s_tps(27:29) = [0 1 0];
    case 4
        s_tps(27:29) = [0 1 1];
end
%HP 流码率%
switch DVBTSETTINGS.cr1
    case 1
        s_tps(30:32) = [0 0 0];
    case 2
        s_tps(30:32) = [0 0 1];
    case 3
        s_tps(30:32) = [0 1 0];
    case 5
        s_tps(30:32) = [0 1 1];
    case 7
        s_tps(30:32) = [1 0 0];
end
%LP 流码率%
switch DVBTSETTINGS.cr2
    case 1
        s_tps(33:35) = [0 0 0];
    case 2
        s_tps(33:35) = [0 0 1];
    case 3
        s_tps(33:35) = [0 1 0];
    case 5
        s_tps(33:35) = [0 1 1];
    case 7
        s_tps(33:35) = [1 0 0];
end
%保护间隔(GI)%
GI = round(DVBTSETTINGS.GI * 32);
```

```
switch (GI)
    case 1
        s_tps(36:37) = [0 0];
    case 2
        s_tps(36:37) = [0 1];
    case 4
        s_tps(36:37) = [1 0];
    case 8
        s_tps(36:37) = [1 1];
end
%传输模式: 2K/8K
if (DVBTSETTINGS.mode == 8)
    s_tps(38:39) = [0 1];
elseif (DVBTSETTINGS.mode == 2)
    s_tps(38:39) = [0 0];
end
inbits = gf([false(1, 60) s_tps(1:53)], 1);
inbits = bchenc(inbits, 127, 113);
inbits = logical(inbits.x);
s_tps = inbits(61:end);
end
```

程序 3-7 “plot_spectrum”,画 OFDM 频谱图

```
function plot_spectrum(signal, fs, figureindex, xlab, ylab, title_str)
figure(figureindex);
pts = length(signal);
spectrum = abs(fftshift(fft(signal)));
spectrum = spectrum/max(spectrum);
f = -fs/2:fs/pts:(fs/2-fs/pts);
plot(f,20 * log10(spectrum));
xlabel(xlab);
ylabel(ylab);
title(title_str);
axis([-fs/2 fs/2 -100, 10]);
grid on;
end%
```

程序 3-8 “calc_papr”,统计 CCDF-PAPR 曲线

```
function varargout = calc_papr(signal, legend_str)
global papr_fig;
global log_fid;
P_rms = signal. * (signal.')';
PA_rms = signal * signal';
PA_rms = PA_rms/numel(P_rms);
PP_rms = max(P_rms);
PAPR0 = 10 * log10(PP_rms./PA_rms);
fprintf(log_fid, 'PAPR Peak: %.2f\n', PAPR0);
Power2AveragedB = 10 * log10(P_rms/PA_rms);
%papr = max(Power2AveragedB);
[n x] = hist(Power2AveragedB,0:0.01:PAPR0);
ccdf = 1-cumsum(n)/length(Power2AveragedB);
```

```
figure(99);
fig_color = ['r' 'g' 'b' 'c' 'm' 'y' 'k'];
semilogy(x, ccdf, fig_color(papr_fig));
xlabel('papr, x dB');
ylabel('Probability, X >= x');
% axis([min(x) max(x) 10^-4 10^-1]);
grid on;hold on;
i = find(ccdf < 0.001,1,'first');
text(x(i), ccdf(i), num2str(x(i)), 'FontWeight','bold');
plot(x(i), ccdf(i), 'MarkerFaceColor','k','MarkerSize',6);
varargout{1} = PAPR0;
varargout{2} = ccdf;
hold on;
papr_fig = papr_fig + 1;
end
```

程序 3-9 “gen_wv”,生成 OFDM 测试信号

```
function gen_wv(signal,wv_file_name,clock,rmsoffs,peakoffs)
samples = length(signal);
fid_wv = fopen(wv_file_name,'w');
signal = signal /max(abs([real(signal) imag(signal)]));
source_data = round(signal * (2^15));
fprintf(fid_wv,'{TYPE: SFU-WV,0}');
fprintf(fid_wv, '{CLOCK: %d}',clock);
fprintf(fid_wv, '{LEVEL OFFS: %d,%d}',rmsoffs,peakoffs);
fprintf(fid_wv, '{SAMPLES: %d}', samples);
fprintf(fid_wv, '{WAVEFORM-%d: #', (samples * 4) + 1);
source_data = [real(source_data);imag(source_data)];
source_data = reshape(source_data, 1, [ ]);
count = fwrite(fid_wv, source_data, 'int16');
assert(count==2 * samples, 'Write number error!');
fprintf(fid_wv, '}');
fclose(fid_wv);
end
```

程序 3-10“lpf”,%构造低通滤波器

```
function Hd = lpf(Fpass, Fstop, Fsample, fig)
% All frequency values are in MHz.
Dpass = 0.0063095734448; %带通
Dstop = 0.001;  %带阻
flag = 'scale';
%计算 KAISERORD 函数
[N,Wn,BETA,TYPE] = kaiserord([Fpass Fstop]/(Fsample/2), [1 0], [Dstop Dpass]);
N = ceil(N/2) * 2;
% Calculate the coefficients using the FIR1 function.
b = fir1(N, Wn, TYPE, kaiser(N+1, BETA), flag);
Hd = dfilt.dffir(b);
Hd = Hd.numerator;
if (fig > 1)
    freqz(Hd);
end
```

```
end
```

程序 3-11 “add_ref_sig”,插入连续、离散、TPS 导频

```
function data = add_ref_sig(bits, DVBTSETTINGS, nframe)
global log_fid;
SYMBOLS_PER_FRAME = 68;
data=zeros(DVBTSETTINGS.N, SYMBOLS_PER_FRAME, nframe);
loc_pilots = cmlk_pilots(DVBTSETTINGS);
loc_tps = tps(DVBTSETTINGS);
for iframe=1:nframe
    counter = 1;
    fprintf(log_fid, 'Organizing the %dth frame...\n', iframe);
    tic;
    %bits = logical(randi(2, 1, 68 * DVBTSETTINGS.NData * DVBTSETTINGS.level)-1);
    vec = mapping(bits(iframe, :), DVBTSETTINGS.level, DVBTSETTINGS.alpha);
    for l = 1:68
        wk = wk_gen(DVBTSETTINGS.N);
        s_tps = cmlk_tps(iframe, DVBTSETTINGS);
        tps_index = 1;
        pilots_index = 1;
        for k = 1:DVBTSETTINGS.N
            if (tps_index <= length(loc_tps) && loc_tps(tps_index) == k-1)
                if l==1
                    data(k, l, iframe) = 1 - 2 * wk(k);
                else
                    if s_tps(l-1)==1
                        data(k, l, iframe) = -data(k, l-1, iframe);
                    else
                        data(k, l, iframe) = data(k, l-1, iframe);
                    end
                end
                tps_index   = tps_index + 1;
            elseif (pilots_index <= size(loc_pilots, 1) && loc_pilots(pilots_index, l) == k-1)
                data(k, l, iframe) = (1 - 2 * wk(k)) * 4/3;
                pilots_index = pilots_index+1;
            else
                data(k, l, iframe) = vec(counter);
                counter = counter+1;
            end
            assert(abs(data(k, l, iframe))> 1/sqrt(60), 'signal error, too small! ...
            Now %dth frame %dth symbol %dth carrier', iframe, l, k);
            assert(abs(data(k, l, iframe))< 2, 'signal error, too big! Now %dth frame ...
            %dth symbol %dth carrier', iframe, l, k);
        end
    end
    toc;
end
end
```

程序 3-12 “cyclicprefix”,插入循环前缀

```
function dataout = cyclicprefix(datain, GI)
```

```
fftpnts = size(datain, 1);
GIpnts = round(fftpnts * GI);
dataout = zeros(size(datain)+[GIpnts 0 0]);
dataout(1:GIpnts, :, :) = datain(fftpnts-GIpnts+1:end, :, :);
dataout((1:fftpnts)+GIpnts, :, :) = datain(:,:,:);
dataout = reshape(dataout, [], 1).';
end
```

程序 3-13 “clipping_method1”,限幅削波 1

```
function dataout = clipping_method1(datain, ibo)
ap = average_power(datain);
pmax = ap * 10^(ibo/10);
pmaxr = sqrt(pmax);
i=find(abs(datain)> pmaxr);
dataout = datain;
dataout(i) = datain(i) ./ abs(datain(i)) . * pmaxr;
end
```

程序 3-14 “clipping_method2”,限幅削波 2

```
function dataout = clipping_method2(datain, ibo)
ap = average_power(datain);
pmax = ap * 10^(ibo/10);
pmaxr = sqrt(pmax);
i= abs(datain)> pmaxr;
dataout = datain;
dataout(i) = 0;%datain(i) ./ abs(datain(i)) . * pmaxr;
end
```

程序 3-15 “clipping_method3”,限幅削波 3

```
function dataout = clipping_method3(datain, ibo)
ap = average_power(datain);
pmax = ap * 10^(5/10);
pmaxr = sqrt(pmax);
i= abs(datain)> pmaxr;
dataout = datain;
dataout(i) = 0;%datain(i) ./ abs(datain(i)) . * pmaxr;
pmax = ap * 10^(ibo/10);
pmaxr = sqrt(pmax);
i=find(abs(datain)> pmaxr);
dataout(i) = datain(i) ./ abs(datain(i)) . * pmaxr;
end
```

程序 3-16 “clipping_method4”,限幅削波 4

```
function dataout = clipping_method4(datain, Vclip)
i=find(abs(datain)> Vclip);
dataout(i) = datain(i) ./ abs(datain(i)) . * Vclip;
end
```

程序 3-17 “clipping_method5”,限幅削波 5

```
function dataout = clipping_method5(datain, ibo, P)
ap = average_power(datain);
```

```
pmax = ap * 10^(ibo/10);
pmaxr = sqrt(pmax);
i=find(abs(datain)> pmaxr);
dataout = datain;
dataout(i) = datain(i) ./ abs(datain(i)) . * (-P * (abs(datain(i))-pmaxr)+pmaxr);
end
```

程序 3-18 “average_power”,统计平均功率

```
function power = average_power(datain)
len = length(datain);
power = datain * datain' /len;
end
```

程序 3-19 “keep_ref_sig”,OFDM 解调数据提取

```
function dataout = keep_ref_sig(datain, dataorg)
global DVBTSETTINGS;
SYMBOLS_PER_FRAME = 68;
loc_pilots = cmlk_pilots(DVBTSETTINGS);
loc_tps = tps(DVBTSETTINGS);
% datain = [datain (DVBTSETTINGS. fftpnts-(DVBTSETTINGS. Kmax + DVBTSETTINGS.
Kmin)/2+1:DVBTSETTINGS.fftpnts, :, :); ...
datain (1: DVBTSETTINGS. fftpnts-(DVBTSETTINGS. Kmax + DVBTSETTINGS. Kmin)/2,
:, :)];
datain = circshift(datain, [(DVBTSETTINGS.Kmax-DVBTSETTINGS.Kmin)/2 0 0]);
dataorg = circshift(dataorg, [(DVBTSETTINGS.Kmax-DVBTSETTINGS.Kmin)/2 0 0]);
dataout = datain(1:DVBTSETTINGS.N, :, :);
for l = 1:SYMBOLS_PER_FRAME
    dataout(loc_pilots(:, l).'+1, l, :) = dataorg(loc_pilots(:, l).'+1, l, :);
end
dataout(loc_tps+1, :, :) = dataorg(loc_tps+1, :, :);
datain(1:DVBTSETTINGS.N, :, :) = dataout;
dataout = circshift(datain, [-(DVBTSETTINGS.Kmax-DVBTSETTINGS.Kmin)/2 0 0]);
end
```

程序 3-20 “ace”,ACE 峰均比抑制

```
function [XACE, orig_papr, orig_ap, X] = ace(X, L, Hd)
global FIX_POINT;
global log_fid;
for iter_time=1:3
    if iter_time==1
        X0=X;
    else
        X0=XACE;
    end
    [fftpnts, FRAMES_PER_S, nSF] = size(X0);
    x = sqrt(fftpnts) * ifft(X0, fftpnts);
    x = reshape(x, 1, []);
    %预削波
    if iter_time==1
        x = x * 1.1;
        ibo = 7;
```

```
        x = clipping_method1(x, ibo);
    elseif iter_time==1
        ibo = 3.5;
        x = clipping_method1(x, ibo);
    else
        ibo = 3.4;
        x = clipping_method1(x, ibo);
    end
    x = upsample(x, 4) * 4;
    %    low-pass filtering
    len = length(Hd);
    x = [x(end-(len-1)/2+1:end) x x(1:(len-1)/2)];
    x = conv(x, Hd);
    x = x((len-1)/2+1:end-(len-1)/2);
    x = x((len-1)/2+1:end-(len-1)/2);
    if iter_time == 1
        orig_papr = calc_papr(x, 'original');
        fprintf(log_fid, 'PAPR PEAK before clipping: %.2f\n', orig_papr);
    end
    % orig_papr = calc_papr(x, 'original');%fig=fig+1;
    %fprintf(log_fid, 'PAPR PEAK before clipping: %.2f\n', orig_papr);
    orig_ap = average_power(x);
    fprintf(log_fid, 'Average power before clipping: %.2f\n', orig_ap);
    %限幅削波%
    if iter_time == 1
        ibo =2.5;
        P = 3;
    elseif iter_time == 2
        ibo = 5;
        P =0.25;
    else
        ibo =6;
        P = 1;
    end
    x = clipping_method5(x, ibo, P);
    %低通滤波%
    x = [x(end-(len-1)/2+1:end) x x(1:(len-1)/2)];
    x = conv(x, Hd);
    x = x((len-1)/2+1:end-(len-1)/2);
    x = x((len-1)/2+1:end-(len-1)/2);
    %    fix point
    if FIX_POINT
        signal_max = max(abs([real(x) imag(x)]));
        x = floor(signal/signal_max * 2^15)/2^15 * signal_max;
    end
    x = downsample(x, 4);
    x = reshape(x, fftpnts, FRAMES_PER_S, nSF);
    Xc = fft(x) / sqrt(fftpnts);
    %ACE 星座图扩展
    if iter_time == 1
        gain = 1;
```

```
        elseif iter_time == 2
            gain = 2.75;
        else
            gain =6;
        end
        if iter_time==1
            Xc = X0 + gain*(Xc-X0);
        else
            Xc = XACE + gain*(Xc-XACE);
        end
        Xreal = real(Xc);
        Xreal(Xreal>L) = L;
        Xreal(Xreal<-L) = -L;
        Ximag = imag(Xc);
        Ximag(Ximag>L) = L;
        Ximag(Ximag<-L) = -L;
        pos = find(Xreal<sqrt(2)/2 & real(X)>0);
        Xreal(pos) = real(X(pos));
        pos = find(Xreal>-sqrt(2)/2 & real(X)<0);
        Xreal(pos) = real(X(pos));
        pos = find(Ximag<sqrt(2)/2 & imag(X)>0);
        Ximag(pos) = imag(X(pos));
        pos = find(Ximag>-sqrt(2)/2 & imag(X)<0);
        Ximag(pos) = imag(X(pos));
        pos = find(X==0);
        XACE = complex(Xreal, Ximag);
        XACE = keep_ref_sig(XACE, X);
        XACE(pos)=0;
    end
    end
```

程序 3-1 为 ACE 峰均比抑制 OFDM 发射机功率放大器效率优化的主程序，程序 3-2～程序 3-20 为程序 3-1 调用的子程序模块。实验仿真平台对比了图 3-11 中两种峰均比抑制方案。选取了欧洲数字电视 DVB-T 标准作为峰均比抑制的实验平台，系统参数设置如表 3-1 所示。功放选取 $p=10$ 的 Rapp 功放模型（动态范围有限的理想线性放大器放大 OFDM 信号并不是最佳的）。分别选取了 2.54s 帧长（10064 个 OFDM 符号）的数据包，为了将复杂度控制在可接受的范围，迭代次数限制为 $M=3$ 次，除了仿真数据，还由 SFE100 数字电视发射机测试仪完成射频调制，并通过 R&S ETL 电视分析仪给出测量结果。

表 3-1　仿真平台系统参数

峰均比抑制实验平台	OFDM 系统	单载波系统
有效带宽	7.61MHz	7.61MHz
IFFT/FFT 载波个数	2048	—
有效子载波	1705	—
数据子载波	1512	—
离散导频/连续导频/TPS 数据	193	—

续表

峰均比抑制实验平台	OFDM 系统	单载波系统
调制模式	QPSK	QPSK
循环前缀	1/8	—
卷积码	1/2 码率、维特比译码	—
LPF 滤波器	Kaiserord 窗	Kaiserord 窗
	带通截止频率：3.8MHz	带通截止频率：3.8MHz
	带阻起始频率：4.1MHz	带阻起始频率：4.1MHz

图 3-14 和图 3-15 分别给出了图 3-11(a)和图 3-11(b)对应的峰均比抑制信号经 SFE100 数字电视发射机测试仪射频调制后在 R&S ETL 电视分析仪中的 CCDF 曲线实测结果，并以原始 OFDM 信号作为参考系。对比数据如表 3-2 所示，除在 10^{-1}PAPR/10^{-4} 方案的峰均比较低，其余处 OCCDF 方案在减少峰均比方面效果更好。

表 3-2 OFDM 信号 CCDF 曲线峰均比对比(dB)

测试信号 PAPR/10^{-n}	10^{-1}	10^{-2}	10^{-3}	10^{-4}
原始 OFDM 信号	3.8	6.5	8.4	9.6
PAPR/10^{-4} ACE-OFDM 信号	3.8	4.9	5.2	5.5
OCCDF 技术的 ACE-OFDM 信号	3.6	5.1	6.2	7.1
降低 PAPR ACE 信号 PAPR 增益(相比 OCCDF 技术)	−0.2	0.2	1	1.6

图 3-16 描述了图 3-11 中两种 ACE 峰均比抑制方案通过功率放大器的 IBO-MER_ace 曲线。图 3-16 的对比数据如表 3-3 所示，相对于 OFDM 系统，OCCDF 方案和 PAPR/10^{-4} 方案分别获得了 3.7dB 和 3.1dB 的 IBO 增益，同时两种方案经 3 次迭代后由于星座扩张平均功率均增加了 1dB。为了更直观地描述 IBO 增益，不同的峰均比抑制方案所测得的 IBO 增益需减掉平均功率增加的部分，则 OCCDF 方案和 PAPR/10^{-4} 方案分别获得了 2.7dB 和 2.1dB 的 IBO 净增益。

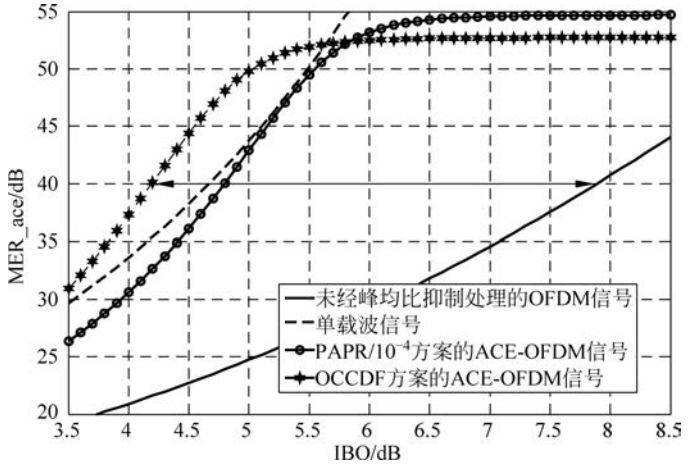


图 3-16 Rapp 功放模型下的 IBO-MER_ace 测试

表 3-3　MER_ace＝40dB 时，对比未经处理的 OFDM 信号 IBO 和 SNR 增益(dB)

测试信号	IBO	IBO 增益	平均功率增加	IBO 净增益	SNR 增益	IBO 总增益
原始 OFDM 信号	7.9	0	0	0	0	0
OCCDF 方案的 ACE-OFDM 信号	4.2	3.7	1	2.7	0.45	3.15
PAPR/10^{-4} 方案的 ACE-OFDM 信号	4.8	3.1	1	2.1	0.45	2.55
单载波信号	4.7	3.2	0	3.2	0	3.2

对比图 3-14、图 3-15 的实验可以得出如下结论：

PAPR/10^{-4} 方案在迭代时主要减少 CCDF 曲线中 10^{-4} 概率附近的 PAPR；OCCDF 方案是减小放大器在限定失真(MER_ace＝40dB)时的功率回退，这考虑了四层含义：

(1) 测量 MER 意味着是对每个子载波的失真都进行了测量，不但考虑失真大小，而且对其出现概率进行了累积，事实上从 PAPR 曲线上可得知幅度大的峰值放大后失真大，但次高峰值的子载波出现概率对失真造成的影响未必小。

(2) 迭代收敛的目标是减小 IBO，当放大器电源电压一定时，输出功率越大，意味着更多的电压输出到负载上，输出到放大管上的电压更小，等效为放大器效率越高，所以减小 IBO 等效于增加放大器效率，简化了效率计算带来的复杂度。

(3) 图 3-16 中 OCCDF 方案经三次迭代后 IBO 为 4.2dB，即当 MER_ace＝40dB 时，放大器的限幅电压在 PAPR＝4.2 处；PAPR/10^{-4} 方案得到了如图 3-14 所示的 CCDF 曲线，在限定失真 MER_ace＝40dB 时 IBO＝4.8dB(见图 3-16)；虽然 OCCDF 方案在 PAPR 大于 4.8dB 时出现的概率大，但由于减少了 PAPR 在 4.2～4.8dB 峰值的概率，得到的 IBO 反而大大低于 PAPR/10^{-4} 方案。

(4) 由于 ACE 要满足最小码间欧式距离的约束，改变高 PAPR 部分的分布概率，必然会影响到其他部分的 PAPR 分布。由图 3-15 可见 OCCDF 方案得到的 CCDF 曲线是以最高功放效率为迭代收敛目标得到的，因此是具有满足最高功放效率的 OFDM 信号最佳幅值分布。

另外：放大器模型 p 取 10 被认为是接近实用的放大器(不去探讨非线性的影响)，若在理想预失真状态下，IBO 还会有所改善。为了不影响解调，OFDM 系统中 10%以上的导频子载波应保持原位置不动。

图 3-17 给出了 IBO 测量结果为 4.2dB、MER_ace 等于 40dB 时的 OCCDF-OFDM 信号在 R&S ETL 电视分析仪中的星座图分布实测结果。按照传统的 MER 统计方法，MER 测量值会变差，而星座点扩张的子载波因为欧氏距离的增加会增强信号的抗干扰能力，获得 SNR 增益。可以将峰均比抑制的 IBO 增益和信噪比增益(ΔSNR)计为 IBO 总增益。另外经过 ACE 峰均比抑制处理的 OFDM 信号中，由于扩张空间的存在增加的信号平均功率将从 IBO 增益中扣除，IBO 可以定义为

$$\mathrm{IBO}_n[\mathrm{dB}] = P_{\mathrm{in,max}} - P_{\mathrm{in,av}} + P_{\mathrm{ace}} - \Delta\mathrm{SNR} \tag{3-20}$$

式中，$P_{\mathrm{in,max}}$ 为输入信号的饱和截止功率；$P_{\mathrm{in,av}}$ 为输入信号的平均功率；P_{ace} 为 ACE 扩张空间引起的平均功率增加。

图 3-18 给出了高斯信道、1/2 码率卷积码和维特比译码条件下(经由 R&S ETL 电视分

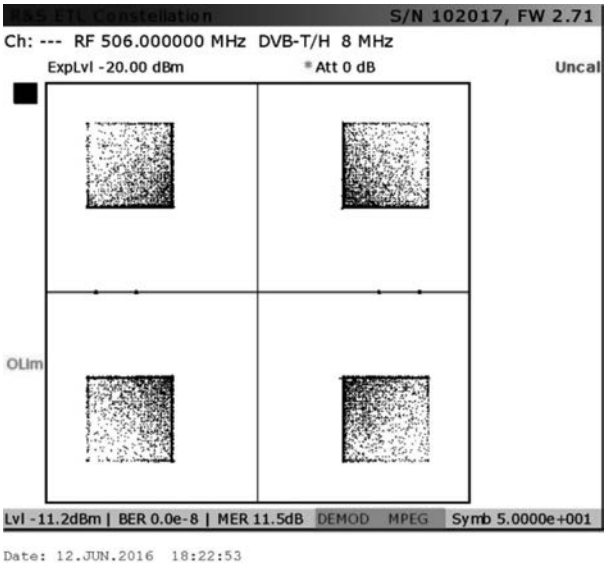


图 3-17 OCCDF 技术的 ACE-OFDM 信号在 R&S ETL 电视分析仪中解调的星座图

析仪中解调),图 3-11 中两种 ACE 方案处理的 OFDM 信号和原始 OFDM 信号在 MER_ace=40dB 时的 SNR-BER 测试结果(由 R&S ETL 电视分析仪实测完成)。实验结果显示,相较于原始 OFDM 信号,OCCDF 技术和 PAPR/10^{-4} 技术的 ACE-OFDM 信号由于载波的扩张增加了 1dB 的功率,在 BER 为 10^{-3} 处分别获得了 0.45dB 的信噪比增益。

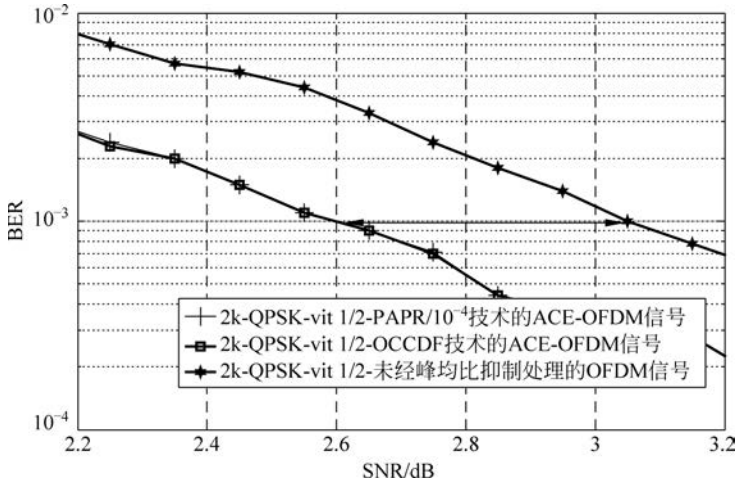


图 3-18 高斯信道下的 BER 测试结果

如表 3-3 所示,相比于原始 OFDM 信号,经过 OCCDF 技术处理的 OFDM 信号所匹配功放的 IBO 总增益为 3.15dB,而 PAPR/10^{-4} 技术处理的 OFDM 信号的 IBO 总增益为 2.55dB。图 3-16 中还给出了单载波信号的测量结果,采用与 OFDM 系统相同滤波窗函数的低通滤波器。当满足 MER 值为 40dB 时,与 OCCDF 技术处理的 ACE-OFDM 信号相比,单载波信号仅存在 0.05dB 的 IBO 优势。

3.2.5　PSO-PTS 峰均比抑制技术和 OFDM 发射机功放效率优化

1. 部分传输序列(PTS)峰均比抑制技术

高阶 M-QAM 调制下，OFDM 峰均比抑制可扩张的星座空间和星座点不断减小；而超宽带高阶 M-QAM 调制是 5G/6G 发展的必然趋势，PTS 是为数不多的通过算力增强的适用于高阶 QAM 调制的功放优化手段。

PTS 方案的基本原理如图 3-19 所示，首先将调制完成的频域 OFDM 信号 $\boldsymbol{S}_N$ 按照某种方案在频域分割为 V 块：

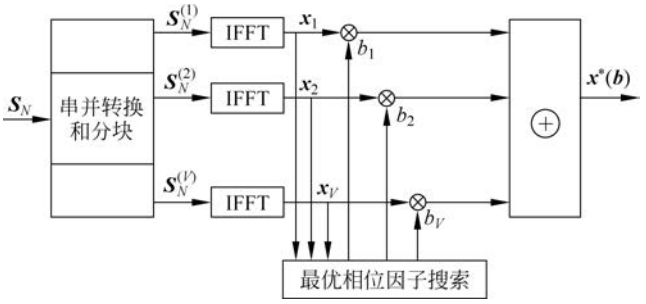


图 3-19　PTS 方案的基本原理

$$\boldsymbol{S}_N = \sum_{i=1}^{V} \boldsymbol{S}_N^{(i)} \tag{3-21}$$

式中，每个数据块 $\boldsymbol{S}_N^{(i)}$ 包含 N/V 个有效数据子载波和 $(V-1)N$ 个空子载波。然后将每块数据进行 LN 点的 IFFT 变换，其中，L 为频域过采样倍数。

将分割数据块再乘上一个旋转相位向量 $\boldsymbol{b}_i = \mathrm{e}^{\mathrm{j}\phi_i}$，$\phi_i \in [0, 2\pi]$，并将所有数据块进行累加求和得到乘性加扰峰均比抑制的基带 OFDM 信号：

$$\boldsymbol{x}^*(\boldsymbol{b}) = \sum_{v=1}^{V} \boldsymbol{b}_v \boldsymbol{x}_v = \sum_{v=1}^{V} \boldsymbol{b}_v \mathrm{IFFT}_{NL\times N}(\boldsymbol{S}_N^{(v)}) \tag{3-22}$$

不同的分割方法产生的 PTS 分块具有不同的自相关性。常用的分块方法有相邻法(Adjacency Partition，AP)、交织法(Interlaced Partition，IP)和随机法(Random Partition，RP)。其中，随机法的各个子块的自相关性最低，在相同条件下，随机法 PTS 具有最好的 PAPR 抑制效果，相对的它需要的计算资源和解调难度也是最高的。图 3-20 是三种不同的分割方法在相同的条件下 PAPR 的抑制效果。

另外，分块数和旋转相位向量的取值范围也对 PAPR 的抑制效果有非常大的影响，当分块数为 V，旋转相位向量的范围取值为 W 时，PTS 方案的全搜索空间为 W^V。图 3-21 和图 3-22 是在不同的分块数和不同的旋转相位向量下 PAPR 的抑制效果。

从图 3-21 和图 3-22 可知，在 PTS 方案中增大分块数和旋转相位向量的取值范围都能够有效提升 PAPR 抑制效果，提升乘性加扰离散度，但由于 PTS 方案的本质就是计算复杂度和性能的折中，所以要想得到更好的 PAPR 抑制性能，就得消耗巨量的计算资源，因此需要智能优化算法来提高计算效率。

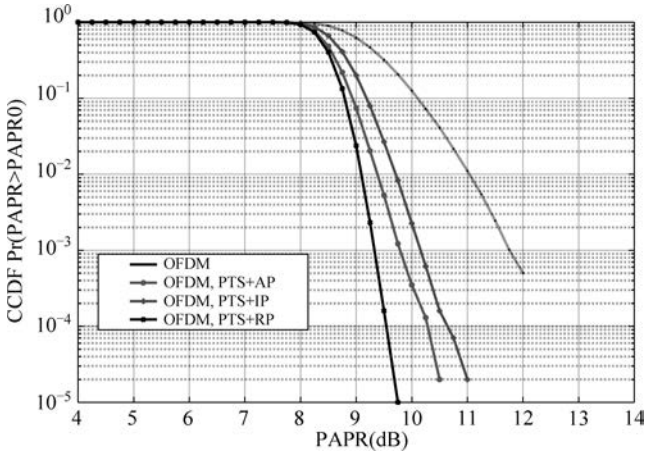


图 3-20　不同分割方法的 CCDF 曲线(见彩插)

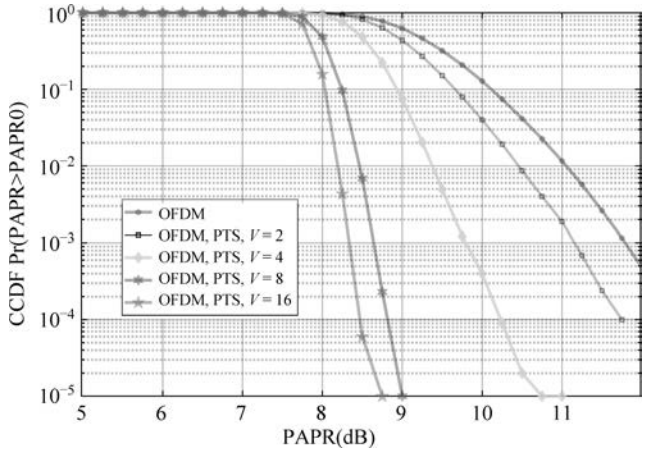


图 3-21　不同分块数的 CCDF 曲线(见彩插)

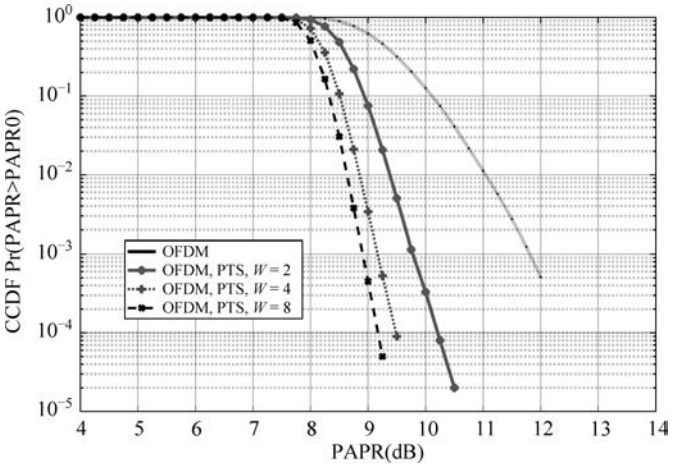


图 3-22　不同旋转相位向量的取值范围的 CCDF 曲线(见彩插)

2. 基于计算增强的 OCCDF 的 PTS OFDM 功放优化

本节在传统 PTS 的基础上，探讨经过高效计算增强的 PTS 加扰功放优化算法。在 3.2.3 节 OCCDF 优化准则下 PTS 方案的目标函数可以表示为

$$\begin{aligned}\widetilde{\boldsymbol{X}}_{\text{opt}}^{*} &= \arg\min_{f} \text{IBO} \\ &\text{s.t. MER\_PTS} = 40\text{dB}\end{aligned} \tag{3-23}$$

式中，IBO 可以定义为

$$\text{IBO}[\text{dB}] = 10\lg\frac{A_{\text{sat}}^{2}}{\sigma^{2}} \tag{3-24}$$

基于 OCCDF 的 PTS 优化方案整体框图如图 3-23 所示。

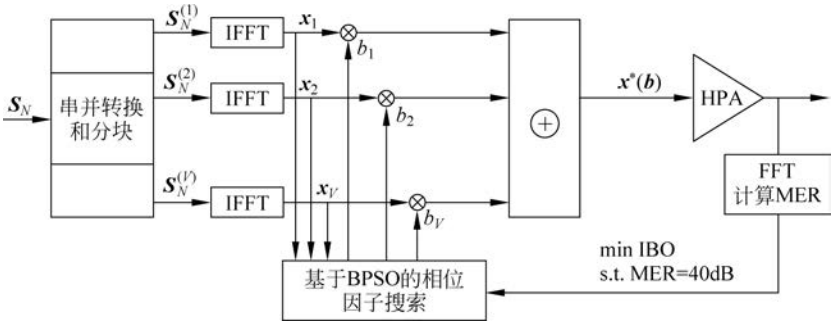


图 3-23　OCCDF 准则下最低 IBO 准则的 PTS 方案

OCCDF PTS 方案的核心思想就是从备选的旋转相位向量集合中选出一组使 IBO 最小的组合，这是一个典型的组合优化数学模型，适合将智能优化的计算效率增强算法应用到 PTS 技术中。

3. 基于 BPSO-PTS 的优化方案

智能优化较早的思路是基于达尔文的“适者生存，优胜劣汰”的竞争法则，模仿自然界生物的进化和遗传过程，在搜索过程中自动获取并累计可行解空间的有关知识，逐步向最优解或者次优解逼近。经典的智能优化算法包括蚁群算法、模拟退火、遗传算法和粒子群(Particle Swarm Optimization，PSO)算法等。不同的智能优化算法适用的场景也不尽相同。考虑 PTS 功放优化方案，由于备选可行解是一个离散空间，可选择遗传算法和 PSO 算法，其中 PSO 算法具有收敛速度快、效率高的全局搜索能力，本章主要将 PSO 算法应用于 PTS 增强方案中。Kennedy 和 Eberhart 于 1995 年首先提出一种基于种群随机优化技术的优化算法，即 PSO 算法。整个优化过程可以简化为：随机生成一组初始粒子并赋予它们速度和位置，按照“最近邻速度匹配”规则逐步运行，使得临近的粒子运行速度一致。然后计算每个位置的“适应度值”，群体内各个体之间可以相互共享信息，每个粒子都能获取当前种群所能到达的最优位置并保存当前所能达到的最好位置。最后更新速度和位置。

基于二进制粒子群优化(Binary Particle Swarm Optimization，BPSO)算法，OCCDF 准则下最低 IBO 准则的 BPSO-PTS 方案的目标函数可表示为

$$F(\widetilde{\boldsymbol{X}}_{\text{opt}}^{*})_{\text{BPSO}} = \arg\min_{\widetilde{\boldsymbol{x}}^{*}(\boldsymbol{b})}\left|10\lg\frac{A_{\text{sat}}^{2}}{\sigma^{2}} - f_{\text{D}}\right|$$

$$\begin{aligned}
\text{s.t.}\quad & \widetilde{\boldsymbol{X}}^{*}(\boldsymbol{b})=\sum_{v=1}^{V}b_v\cdot\boldsymbol{x}_v \\
& \boldsymbol{b}_m=\left\{\mathrm{e}^{\mathrm{j}2\pi l/W}\right\}^{V} \\
& \boldsymbol{\Omega}_g^{\mathrm{BPSO}}=\{\boldsymbol{b}_m\mid m=1,2,\cdots,K\} \\
& K\leqslant W^V \\
& \mathrm{MER_PTS}=40\mathrm{dB}
\end{aligned} \tag{3-25}$$

式中，f_{D} 为迭代停止的目标值；$\boldsymbol{\Omega}_g^{\mathrm{BPSO}}$ 为 BPSO 算法的搜索空间；K 为总的搜索空间大小。

OCCDF 准则旨在降低 IBO，在限定失真 MER 的条件下，IBO 越小，功放效率越高。所以 PTS 信号在经过功率放大器后以减小 IBO 作为优化目标，反馈至相位因子搜索模块，通过 BPSO 算法选择最优的相位因子使得 IBO 最小。

为了更加适应在目标函数式(3-25)中的离散相位因子组合，在标准 PSO 算法的基础上提出 BPSO 算法，以快速找到最优的旋转相位因子组合，BPSO 算法的整体流程如图 3-24 所示，详细算法参见表 3-4。在式(3-25)中的备选旋转因子是一个离散的集合，刚好对应 BPSO 算法的一个粒子组，定义每个粒子的位置为 $\boldsymbol{W}_i=[b_{i,1},b_{i,2},\cdots,b_{i,V}]$，$b_{i,m}=\mathrm{e}^{\mathrm{j}2\pi l/W}\mid l=0,2,\cdots,W-1,m=1,2,\cdots,V$，定义速度为 $\boldsymbol{V}_i=[v_{i,1},v_{i,2},\cdots,v_{i,V}]$，相应粒子的历史最优位置为 $\boldsymbol{W}_i^P=[b_{i,1}^P,b_{i,2}^P,\cdots,b_{i,V}^P]$，种群的历史最优位置为 $\boldsymbol{W}^G=[b_1^G,b_2^G,\cdots,b_V^G]$。那么速度和位置的更新公式可以表示为

$$\boldsymbol{V}_i(t+1)=\bar{\omega}(t)\boldsymbol{V}_i(t)+c_1r_1(\boldsymbol{W}_i^P(t)-\boldsymbol{W}_i(t))+c_2r_2(\boldsymbol{W}^G(t)-\boldsymbol{W}_i(t))$$

$$W_{i,d}(t+1)=\begin{cases}0, & r^t>\dfrac{1}{\left(1+\mathrm{e}^{-V_{i,d}(t+1)}\right)}\\ 1, & r^t<\dfrac{1}{\left(1+\mathrm{e}^{-V_{i,d}(t+1)}\right)}\end{cases},\quad r^t\sim U(0,1),d=1,2,\cdots,V \tag{3-26}$$

$$V_{i,d}(t+1)=\begin{cases}V_{\max}, & V_{i,d}(t+1)>V_{\max}\\ -V_{\max}, & V_{i,d}(t+1)<-V_{\max}\end{cases}$$

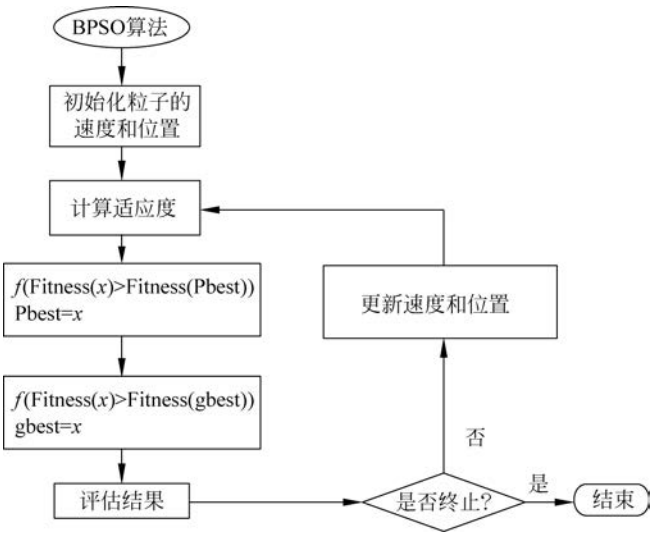


图 3-24　BPSO 算法的整体流程

表 3-4　最小 IBO 目标的 BPSO-PTS 峰均比抑制算法

OCCDF 准则下最小 IBO 的 BPSO-PTS 算法

输入：数据块 $\widetilde{\boldsymbol{X}}^{*}(\boldsymbol{b})$，最大迭代次数 $t_{\max}$，种群大小 P，迭代停止目标值 f_{D}
输出：最优加扰相位因子 $\boldsymbol{b}_{\mathrm{opt}}$
1：初始化粒子的位置和速度矩阵 $\boldsymbol{W}(0)$，$\boldsymbol{V}(0)$
2：设置 BPSO 参数 $V_{\max}$、$\bar{\omega}_{\max}$、$\bar{\omega}_{\min}$、c_1、c_2
3：根据式(3-29)，式(3-30)计算初始适应度函数并得到 $f_0^{\min}(\boldsymbol{W}_i)$
4：**While** $t<t_{\max}$ **then**
5：　根据式(3-26)更新速度和位置
6：　根据式(3-28)控制每个粒子的速度
7：　计算第 t 代每个粒子的适应度 $f_i(\boldsymbol{W}_i)$ 和第 t 代最优的适应度 $f_t^{\min}(\boldsymbol{W}_i)$
8：　**if** $f_t(\boldsymbol{W}_i(t))>f_t(\boldsymbol{W}_i^P(t))$ **then**
9：　　$\boldsymbol{W}_i^P(t)=\boldsymbol{W}_i(t)$
10：　**end**
11：　**if** $f_t^{\min}(\boldsymbol{W}_i(t))>f_t^{\min}(\boldsymbol{W}^G(t))$ **then**
12　　$\boldsymbol{W}^G(t)=\boldsymbol{W}_i(t)$
13：　**end**
14：　**if** $|f_t^{\min}(\boldsymbol{W}^G(t))-f_{\mathrm{D}}|\leqslant\delta$ **then**
15：　　$cnt++$
16：　**else**
17：　　$cnt=0$
18：　**end**
19：　**if** $cnt\geqslant N$ **then**
20：　　**Break**
21：　**end**
22：　$t=t+1$
23：　**end while**
24：　$\boldsymbol{b}_{\mathrm{opt}}=\boldsymbol{W}^G$

迭代公式中比较重要的几个参数分别为：惯性权重因子 $\bar{\omega}(t)$，学习因子 c_1、c_2，最大速度限制 $V_{\max}$。研究表明，这些参数对 PSO 算法的性能有很大的影响，所以正确选择这些参数能给 PSO 算法的性能带来显著提升。

惯性权重因子 $\bar{\omega}(t)$：该参数被认为是对 PSO 算法性能影响最大的一个参数。本章采用迭代线性递减的方法，即前期选择更大的惯性权重 $\bar{\omega}$ 值，使得 PSO 算法在前期具有更强的搜索能力，后期则选择较小的惯性权重 $\bar{\omega}$ 值进行更加精确的局部化搜索。线性递减的惯性权重因子随着迭代次数的变化为

$$\bar{\omega}(t)=\bar{\omega}_{\max}-\frac{\bar{\omega}_{\max}-\bar{\omega}_{\min}}{t_{\max}}t \tag{3-27}$$

式中，$\bar{\omega}_{\max}$ 表示初始权重；$\bar{\omega}_{\min}$ 表示最终权重；t 为当前迭代次数。惯性权重随着迭代次数增加而减小，即算法的优化区域逐渐收敛于局部。

学习因子 c_1、c_2：分别代表每组粒子向粒子自身历史最优 $\boldsymbol{W}_i^P(t)$ 和种群历史最优 $\boldsymbol{W}^G(t)$ 的随机加速项。当 $c_1=0$ 时，代表粒子自身没有学习过程，但存在群体学习，所以该算法收敛

快但容易陷入早熟。当 $c_2=0$ 时，粒子的共享信息比较少，所以更难收敛到全局最优。研究表明，当 $c_2=c_1=2$ 时，PSO 算法有较好的性能。

最大速度限制 $V_{\max}$：粒子的速度作为每次粒子移动的步长是需要严格限制的，由此来控制粒子的全局搜索能力。如果将此限制设置过大，会导致粒子过快增长而错过全局最优，将限制设置过小则会导致粒子限制局部最优：

$$v_{i,j} \geqslant V_{\max}, \quad v_{i,j}=V_{\max}$$
$$v_{i,j} \leqslant V_{\min}, \quad v_{i,j}=V_{\min} \tag{3-28}$$

第 i 个粒子的适应度 $f_i(\boldsymbol{W}_i)$ 以及第 m 次迭代整个种群中最优的适应度 $f_m^{\min}(\boldsymbol{W}_i)$ 可以定义为

$$f_i(\boldsymbol{W}_i)=10\lg\frac{A_{\text{sat}}^2}{\sigma^2}(\text{dB}) \tag{3-29}$$

$$f_m^{\min}(\boldsymbol{W}_i)=\min_{1\leqslant i\leqslant P} f_i(\boldsymbol{W}_i) \tag{3-30}$$


仿真视频

3.2.6 PSO-PTS 峰均比抑制的 OFDM 发射机功放优化仿真平台

程序 3-21 “PSO_PTS”，PSO-PTS 峰均比抑制的 OFDM 发射机功放优化模型

```
function PSO_PTS
close all;
clear all;
clc;
%OFDM 设置参数%
NumCarr = 1024;%传输子载波数
mapsize = 4;%3-16QAM 2-QPSK
V = 16;%分块数
OverSampleRate = 4;%过采样率
Partition = 1;%(分块方法) 1:相邻分割;2: 交织分割;3:随机分割
W = 1;%旋转相位因子的取值小大取对数
W1=2;%旋转相位因子的取值大小
weight_factor=[1 -1];%旋转相位因子的取值
%GA 算法的初始化%
pop_size =100;%种群大小
Gn_G = 20;%迭代次数
initial_w_G= randi([0,1],[W * V,pop_size]);%初始化种群
%PSO 算法初始化%
Num_Particle_OCCDF = 100; %基于 OCCDF 的 PSO 算法种群大小
Num_Particle_CCDF = 100; %基于 CCDF 的 PSO 算法种群大小
Gn1 = 20;
Gn2 = 20;%迭代次数
c1 = 2; c2 = 2;%学习因子
Vmax = 0.2;%最大速度限制
wmax = 0.9;%初始惯性权重
wmin = 0.4; %最终惯性权重
w = wmax-(wmax-wmin)/Gn1 * (1:Gn1); %权重更新公式
v_min = -Vmax;  v_max = Vmax; %速度限制
initial_v_BPSO = v_min + (v_max-v_min). * rand(W * V,Num_Particle_CCDF);
initial_v = v_min + (v_max-v_min). * rand(W * V,Num_Particle_OCCDF);%初始化速度
```

```
initial_w_P_BPSO = randi([0,1],[W * V,Num_Particle_CCDF]);
initial_w_P = randi([0,1],[W * V,Num_Particle_OCCDF]);%初始化位置
%初始化 OFDM 系统%
bit_per_symbol = 4;
N = NumCarr * bit_per_symbol;
x=round(rand(1,N))';
Symbol_tx=qammod(x,2^bit_per_symbol,'InputType','bit');
%功率放大器设置%
IBO = [0:0.5:10];
SNR = 0:1.5:20;
ber_BPSO_CCDF = zeros(1,length(SNR));
ber_BPSO_OCCDF = zeros(1,length(SNR));
ber_NOPTS = zeros(1,length(SNR));
mer_BPSO = zeros(1,length(IBO));
mer_NOPTS = zeros(1,length(IBO));
mer_best = zeros(1,length(IBO));
%原始 OFDM 信号%
sinal_ NOPTS = sqrt(NumCarr * 4) * ifft([Symbol _ tx(1: NumCarr/2); zeros(NumCarr *
(OverSampleRate-1),1); Symbol_tx(NumCarr/2+1:end)]);
%OFDM 分块%
Symbol_block = zeros( NumCarr,V );
Symbol_ifft2 = zeros( NumCarr * OverSampleRate,V );
for v = 1:1:V
    if ( Partition == 1 ) %相邻分割
        Symbol_block(((v-1) * NumCarr/V+1):(v * NumCarr/V),v)=Symbol_tx(((v-…
        1) * NumCarr/V+1):(v * NumCarr/V));
    elseif ( Partition == 2 )%交织分割
        Symbol_block(v:V:NumCarr,v) = Symbol_tx(v:V:NumCarr);
    elseif ( Partition == 3 )%随机分割
        Symbol_block(Index(v:V:NumCarr),v) = Symbol_tx(Index(v:V:NumCarr));
    end
    Symbol_ifft2(:,v)= sqrt(NumCarr * OverSampleRate) * ifft([Symbol_block(1:NumCarr/2,v);
    zeros(NumCarr * (OverSampleRate-1),1); Symbol_block(NumCarr/2+1:end,v)]); %IFFT 变换
end
%BPSO 方案的 IBO-MER 曲线%
if ( Partition == 1 )
    [~,best_W,PTS_signal] = BPSO_PTS( Symbol_ifft2,W,Gn2,c1,c2,Vmax,…
    w,initial_v_BPSO,initial_w_P_BPSO); %CCDF BPSO-PTS
    [mer_BPSO,mer_NOPTS,signal_PTS_ber,signal_NOPTS_ber]=compute_mer(…
    PTS_signal, best_W, Symbol_tx, V, NumCarr, IBO, OverSampleRate);
    % CCDF-BPSO-PTS 计算 MER%
    [ber_BPSO_CCDF,ber_NOPTS] = compute_ber(signal_PTS_ber, signal_NOPTS_ber, …
    SNR, Symbol_tx, NumCarr, OverSampleRate, V,best_W);
    [~,X_opt,mer_best,signal_BPSO_OCCDF,best_W] = IBO_BPSO( Symbol_ifft2, V, …
    Gn1, c1, c2, Vmax,w,Num_Particle_OCCDF, …
    Symbol_tx,initial_w_P,initial_v,IBO,W,NumCarr,OverSampleRate); %OCCDF-BPSO-PTS
    [ber_BPSO_OCCDF,~]=compute_ber(signal_BPSO_OCCDF, signal_NOPTS_ber, …
    SNR, Symbol_tx, NumCarr, OverSampleRate, V, best_W);
end
```

```
%OPTS 方案%
Bdata_OPTS_All=factor_combination3(V, W1, weight_factor); %计算所有相位因子组合
if ( Partition == 1 )
    [best_w,signaPTS] = OPTS( Symbol_ifft2,Bdata_OPTS_All ); %OPTS 方案
    [mer_PTS,~] = compute_mer(signaPTS,best_w,Symbol_tx,V,NumCarr,IBO, ...
    OverSampleRate); % OPTS 方案计算 MER
end
%GA 方案%
if ( Partition == 1 )
    [GA_PTS,best_w] = GA_PTS_nomal(Symbol_ifft2,Gn_G,W,pop_size, ...
    initial_w_G);%GA-PTS
    [mer_EGA,~] = compute_mer(GA_PTS,best_w,Symbol_tx,V,NumCarr, ...
    IBO,OverSampleRate);
    % GA-PTS 方案计算 MER
end
figure (1)
plot(IBO,(mer_best),'k-*')
hold on
plot(IBO,(mer_BPSO),'m-^')
hold on
plot(IBO,(mer_PTS),'g-+')
hold on
plot(IBO,(mer_EGA),'c-s')
hold on
plot(IBO,(mer_NOPTS),'r-*')
hold on
plot(IBO,40*ones(1,length(IBO)),'k.-.')
legend('OCCDF-BPSO-PTS','CCDF-BPSO-PTS','CCDF-PTS','CCDF-GA-PTS','without PTS')
xlabel('IBO/dB'),ylabel('MER/dB');
ylim([20,55]);
xlim([3,8.5]);
figure (2)
semilogy(SNR,ber_BPSO_CCDF,'r-*');
hold on
semilogy(SNR,ber_BPSO_OCCDF,'b-*');
hold on
semilogy(SNR,ber_NOPTS,'k-*');
hold on
legend('CCDF-BPSO-PTS','OCCDF-BPSO-PTS','orignal-OFDM');
xlabel('SNR(dB)');ylabel('BER');
axis([0 20 10^-3 1]);
hold off
end
```

程序 3-22 “compute_ber”,BER 统计分析模型

```
function[error_rate_sig1,error_rate_sig2]=compute_ber(signal_pts, ...
signal_no_pts, SNR,signal_orignal, NumCarr, OverSampleRate, V, best_W)
% signal_pts   PTS 信号
% signal_no_pts 原始 OFDM 信号
for SNRIndex = 1:length(SNR)
    SNR_level = SNR(SNRIndex);
```

```
    recived_signal = awgn(signal_pts,SNR_level,'measured');%高斯白噪声信道
    recived_signal = fft(recived_signal)./sqrt(length(recived_signal));%FFT 变换
    %去零
    X1 = recived_signal(1:NumCarr/2);
    X2 = recived_signal(NumCarr * OverSampleRate-(NumCarr/2-1):end);
    X_NOPTS1 = [X1;X2];
    %解调
    Symbol_block = zeros( NumCarr,V );
    for v = 1:V
        Symbol_block(((v-1) * NumCarr/V+1):(v * NumCarr/V),v)=X_NOPTS1(((v-, ...
        1) * NumCarr/V+1):(v * NumCarr/V)); %分块
    end
    X_NOPTS1 = Symbol_block * best_W; %解调后的信号
    recived_signal_bit = qamdemod(X_NOPTS1,16,'OutputType','bit');
    %解调后的 PTS 信号的逆映射%
    original_signal_bit = qamdemod(signal_orignal,16,'OutputType','bit'); %原始信号的逆映射
    error_bit_sig = sum(recived_signal_bit~=original_signal_bit);
    error_rate_sig1(SNRIndex) = error_bit_sig/length(recived_signal_bit);
end
%原始 OFDM 信号%
for SNRIndex = 1:length(SNR)
    SNR_level = SNR(SNRIndex);
    recived_signal = awgn(signal_no_pts,SNR_level,'measured');%高斯信道
    recived_signal_bit = qamdemod(recived_signal,16,'OutputType','bit');
    original_signal_bit = qamdemod(signal_orignal,16,'OutputType','bit');
    error_bit_sig = sum(recived_signal_bit~=original_signal_bit);
    error_rate_sig2(SNRIndex) = error_bit_sig/length(recived_signal_bit);
end
end
```

程序 3-23 “get80216map”,QAM、16QAM、64QAM 星座映射

```
function IQvalue=get80216map(Qam)
% Qam=4 -- QPSK, Qam=16 -- 16-QAM, Qam=64 -- 64QAM
switch Qam,
    case 4,
        IQvalue = [1+1i 1-1i -1+1i -1-1i]/sqrt(2);
    case 16,
        col1 = [1+1i 1+3i 1-1i 1-3i];
        IQvalue = [col1 col1+2];
        IQvalue = [IQvalue conj(-IQvalue)]/sqrt(10);
    case 64,
        quad1 = [3+3i 3+1i 3+5i 3+7i];
        col1 = [quad1 conj(quad1)];
        IQvalue = [col1 col1-2];
        IQvalue = [IQvalue col1+2];
        IQvalue = [IQvalue col1+4];
        IQvalue = [IQvalue conj(-IQvalue)]/sqrt(42);
end
end
```

程序 3-24 “BPSO_PTS”,基于 BPSO-PTS-minPAPR 的 OFDM 优化方案

```
function [PAPR_PSO, best_W, PTS_signal]= BPSO_PTS(Symbol_ifft2, W, Gn, c1, c2, Vmax, ,
...
w, initial_v, initial_w)
%Symbol_ifft2 经过 IFFT 的 OFDM 信号
%W 相位因子大小的对数
%Gn 迭代次数
%c1,c2 学习因子
%Vmax 最大速度限制
%w 惯性权重
%initial_v 粒子的初始化速度
%initial_w 粒子的初始化位置
%PAPR_PSO 最小的 PAPR
%best_W 最优的相位因子
%PTS_signal 加扰后的信号
[M,N] = size(initial_w);
w_pbest = zeros(M,N);
w_gbest = zeros(M,1);
papr_pbest = inf*ones(1,N);
papr_gbest = inf;
current_w = initial_w;
current_v = initial_v;
for ii = 1:1:Gn
    Bdata= binary2factor(current_w,W); %二进制粒子转换为相位因子
    Symbol_ifft = Symbol_ifft2*Bdata; %计算加扰后信号
    PowerPerBit = abs(Symbol_ifft).^2;
    PowerMean = mean(PowerPerBit);
    PowerMax  = max(PowerPerBit);
    papr = PowerMax./PowerMean;%计算 PAPR
    update_index1 =  find(papr < papr_pbest); %更新 pbest 的状态
    w_pbest(:,update_index1) = current_w(:,update_index1);
    papr_pbest(update_index1) = papr(update_index1);
    if ( min(papr) < papr_gbest ) %更新 gbest 的状态
        [~,update_index2] = min(papr);
        w_gbest(:,1) = current_w(:,update_index2);
        papr_gbest = papr(update_index2);
        best_W = Bdata(:,update_index2);
    end
    next_v = w(ii)*current_v + c1*rand(M,N).*(w_pbest-current_w)...
    + c2*rand(M,N).*(w_gbest*ones(1,N)-current_w); %速度更新公式
    index_min = find( next_v < -Vmax );
    next_v(index_min) = -Vmax;
    index_max = find( next_v > Vmax );
    next_v(index_max) = Vmax; %速度限制
    Sv = 1./(1+exp(-next_v)); % sigma 函数
    next_w = rand(M,N) < Sv; %更新位置
    current_w = next_w;%更新状态
    current_v = next_v;
end
PTS_signal = Symbol_ifft2*best_W;%最佳相位因子加扰后的信号
PAPR_PSO = papr_gbest;%最小 PAPR
end
```

程序 3-25 “compute_mer”,解调并计算 MER

```
function [mer1,mer2,signal_PTS_ber,signal_NOPTS_ber] = compute_mer(PTS_signal, ...
best_W, X_signal, V, NumCarr, IBO, OverSampleRate)
%PTS_signal PTS 方案加扰后的信号
%best_W 最优的相位因子
%X_signal 未加扰前的原始信号
%V 分块数
%NumCarr 子载波数
%IBO IBO 取值范围
pp = 2; %功率放大器的平滑因子
for var = 1:length(IBO)
    X_PTS = X_signal;
    IBO_level = IBO(var);
    PA_rms = X_signal' * X_signal./length(X_signal); %原始信号的平均功率
    max_clip_HPA = sqrt(PA_rms * (10^(IBO_level/10)));%饱和电平
    X_NOPTS1=PTS_signal./((1+(abs(PTS_signal)/max_clip_HPA).^(2 * pp)).^(1/2/pp)); %功放
    if(IBO_level==5)
        signal_PTS_ber = X_NOPTS1;
    end
    X_NOPTS1 = fft(X_NOPTS1)./sqrt(length(X_NOPTS1)); %变换到频域
    %去零
    X1 = X_NOPTS1(1:NumCarr/2);
    X2 = X_NOPTS1(NumCarr * OverSampleRate-(NumCarr/2-1):end);
    X_NOPTS1 = [X1;X2];
    %解调
    Symbol_ifft2 = zeros( NumCarr,V );
    for v = 1:V

        Symbol_block(((v-1) * NumCarr/V+1):(v * NumCarr/V),v)=X_NOPTS1(((v-1) *
        NumCarr/V+1): (v * NumCarr/V)); %分块
    end
    X_NOPTS1 = Symbol_block * best_W; %解调后的信号
    X = X_signal;
    %计算 MER
    ReceiveMER = reshape(X_NOPTS1.', [], 1).';%经过 HPA 后的频域信号
    ValidCarrier=reshape(X.', [], 1).';%原始频域信号
    errRev = ReceiveMER - ValidCarrier; %计算误差
    ValidCarrier = ValidCarrier';
    Psig = ValidCarrier' * ValidCarrier./length(ValidCarrier);
    Perr = (errRev * errRev')./length(errRev);
    mer1(var) = 10 * log10(Psig./Perr); %计算 MER
end
%原始 OFDM 信号%
for var = 1:length(IBO)
    X_PTS = X_signal;
    IBO_level = IBO(var);
    PA_rms = X_signal' * X_signal./length(X_signal);%平均功率
    max_clip_HPA = sqrt(PA_rms * (10^(IBO_level/10)));%饱和电平
    X_PTS = X_PTS./((1+(abs(X_PTS)/max_clip_HPA).^(2 * pp)).^(1/2/pp));%经过功放
    if(IBO_level==5)
```

```
            signal_NOPTS_ber = X_PTS;
        end
        X = X_signal;
        ReceiveMER = reshape(X_PTS.', [], 1).';%经过 HPA 后的频域信号
        ValidCarrier=reshape(X.', [], 1).';%原始频域信号
        errRev = ReceiveMER - ValidCarrier;   %计算误差
        ValidCarrier = ValidCarrier';
        Psig = ValidCarrier' * ValidCarrier./length(ValidCarrier);
        Perr = (errRev * errRev')./length(errRev);
        mer2(var) = 10 * log10(Psig./Perr);   %计算 MER
    end
    end
```

程序 3-26 “IBO_BPSO”,基于 BPSO-PTS-minIBO 的 OFDM 优化模型

```
function [papr, X_opt, mer_best, signal_BPSO_OCCDF, best_W]= IBO_BPSO( Symbol_ifft2, V,
Gn, c1, c2, Vmax, w, Num_Particle, X_signal, initial_w_P, initial_v, IBO, W, NumCarr,
OverSampleRate)
%Symbol_ifft2 经过 IFFT 的 OFDM 信号
%V 分块数
%W 相位因子大小的对数
%Gn 迭代次数
%c1,c2 学习因子
%Vmax 最大速度限制
%w 惯性权重
%Num_Particle 种群粒子数目
%X_signal 原始信号
%initial_v 粒子的初始化速度
%initial_w_P 粒子的初始化位置
%IBO IBO 范围
%papr 最小 PAPR
%mer_best MER 的计算结果
%X_opt 加扰后的信号
[M,N] = size(initial_w_P);
[K1,K2] = size(Symbol_ifft2);
w_pbest = zeros(M,N);
w_gbest = zeros(M,1);
mer_pbest = -inf * ones(1,N); %初始化局部最优 MER
len = length(IBO);
mer_gbest = -inf * ones(1,len); %初始化全局最优 MER
X_PTS = zeros(K1,1);
X_opt = zeros(K1,1);
current_w = initial_w_P;
current_v = initial_v;
pp =2; %功率放大器的平滑因子
for var = 1:length(IBO)
    for ii = 1:1:Gn
        Bdata = binary2factor(current_w,W); %二进制转换为相位因子
        Symbol_ifft = Symbol_ifft2 * Bdata; %加扰后的信号
        for indx = 1:1:Num_Particle
            IBO_level = IBO(var);
            PA_rms = X_signal' * X_signal./length(X_signal);%原始信号功率
```

```
        max_clip_HPA = sqrt(PA_rms * (10^(IBO_level/10)));%饱和电平
        %经过功放%
        X_NOPTS1=Symbol_ifft(:,indx)./((1+(abs(Symbol_ifft(:,indx))/max_clip_HPA).^
        (2 * pp)).^(1/2/pp));
        if(IBO_level==5)
            signal_BPSO_OCCDF = X_NOPTS1;
            best_W = Bdata(:,indx);
        end
        %变换到频域%
        X_NOPTS1 = fft(X_NOPTS1)./sqrt(length(Symbol_ifft(:,indx)));
        %去零%
        X1 = X_NOPTS1(1:NumCarr/2);
        X2 = X_NOPTS1(NumCarr * OverSampleRate-(NumCarr/2-1):end);
        X_NOPTS1 = [X1;X2];
        %分块%
        X_NOPTS1 = reshape(X_NOPTS1,[],16);
        PTS_signal = X_NOPTS1;
        for ii = 1:V
            PTS_signal(:,ii) = X_NOPTS1(:,ii). * Bdata(ii,indx); %解扰
        end
        X_NOPTS1 = reshape(PTS_signal,[],1);
        X = X_signal;
        %MER 计算%
        ReceiveMER = reshape(X_NOPTS1.', [], 1).';%经过 HPA 后的频域信号
        ValidCarrier=reshape(X.', [], 1).';%原始频域信号
        errRev = (ReceiveMER - ValidCarrier)';
        ValidCarrier = ValidCarrier';
        Psig = ValidCarrier' * ValidCarrier./length(ValidCarrier);
        Perr = (errRev' * errRev)./length(errRev);
        mer(var,indx) = 10 * log10(Psig./Perr); %计算 MER
    end
    update_index1 =find(mer(var,:) > mer_pbest); %更新 mer_pbset
    w_pbest(:,update_index1) = current_w(:,update_index1);
    mer_pbest(update_index1) = mer(var,update_index1);
    if ( max(mer(var,:)) > mer_gbest(var) ) % 更新 mer_gbest
        [~,update_index2] = max(mer(var,:));
        w_gbest(:,1) = current_w(:,update_index2);
        mer_gbest(var) = mer(var,update_index2);
        X_PTS = Symbol_ifft(:,update_index2);
    end
    next_v = w(ii) * current_v + c1 * rand(M,N). * (w_pbest-current_w)...
    + c2 * rand(M,N). * (w_gbest * ones(1,N)-current_w); %速度更新公式
    index_min = find( next_v < -Vmax );
    next_v(index_min) = -Vmax;
    index_max = find( next_v > Vmax );
    next_v(index_max) = Vmax; %速度限制
    Sv = 1./(1+exp(-next_v)); % sigma 函数
    next_w = rand(M,N) < Sv; %位置更新
    current_w =  next_w;
    current_v = next_v;
end
```

```
end
mer_best = mer_gbest; %得到最优 MER
X_opt = X_PTS; %加扰后的 PTS 信号
PowerPerBit = abs(X_opt).^2;
PowerMean = mean(PowerPerBit);
PowerMax  = max(PowerPerBit);
papr = PowerMax./PowerMean;
end
```

程序 3-27 “binary2factor”,将二进制的种群转化为标准离散相位旋转因子

```
function y = binary2factor(x,W)
%x 输入二进制矩阵
%W 相位旋转因子大小取对数
%y 输出相位因子矩阵
if ( W == 1 ) %0->-1  1->1
    y = 2*x-1;
elseif ( W == 2 ) %00 -> 1 01-> j 11->-1 10->-j
        y = 1*( x(1:W:end,:) == 0 & x(2:W:end,:) == 0 ) + ...
        1j*( x(1:W:end,:) == 0 & x(2:W:end,:) == 1 ) + ...
        -1*( x(1:W:end,:) == 1 & x(2:W:end,:) == 1 ) + ...
        -1j*( x(1:W:end,:) == 1 & x(2:W:end,:) == 0 );
elseif ( W == 3 ) %000-> 1 001-> 1/2+1j/2 011-> 1j 010->-1/2+1j/2 110->-1 111->-1/2-1j/2 101...
        -> -1j 100-> 1/2-1j/2
        y = 1*( x(1:W:end,:) == 0 & x(2:W:end,:) == 0 & x(3:W:end,:) == 0 ) + ...
        (1/2+1j/2)*( x(1:W:end,:) == 0 & x(2:W:end,:) == 0 & x(3:W:end,:) == 1 ) + ...
        1j*( x(1:W:end,:) == 0 & x(2:W:end,:) == 1 & x(3:W:end,:) == 1 ) + ...
        (-1/2+1j/2)*( x(1:W:end,:) == 0 & x(2:W:end,:) == 1 & x(3:W:end,:) == 0 ) + ...
        -1*( x(1:W:end,:) == 1 & x(2:W:end,:) == 1 & x(3:W:end,:) == 0 ) + ...
        (-1/2-1j/2)*( x(1:W:end,:) == 1 & x(2:W:end,:) == 1 & x(3:W:end,:) == 1 ) + ...
        -1j*( x(1:W:end,:) == 1 & x(2:W:end,:) == 0 & x(3:W:end,:) == 1 ) + ...
        (1/2-1j/2)*( x(1:W:end,:) == 1 & x(2:W:end,:) == 0 & x(3:W:end,:) == 0 );
else
    error('please input W from [1 2 3]');
end
end
```

程序 3-28 “factor_combination3”,求 OPTS 方案中的离散相位因子的所有组合 W^V 种

```
function sub_bdata = factor_combination3( V,W,weight_factor )
%V 子载波分块数
%W 相位因子的取值范围
%weight_factor 相位因子的取值
%sub_bdata 所有相位因子的组合
sub_bdata = zeros(V,W^(V));%总共 W^V 种组合
for ii = 1:1:V
    for kk = 1:1:W
        for jj = 1:1:W^(ii-1)
            sub_bdata(ii,W^(V-ii)*(kk-1)+1+W^(V-ii+1)*(jj-1):W^(V-ii)*kk+W^(V-ii+1)*  ...
            (jj-1))= weight_factor(kk);
        end
    end
end
```

```
end

程序 3-29 “OPTS”,OPTS 峰均比抑制优化
function [phase_factor,X_opt] = OPTS( Symbol_ifft2,Bdata )
%Symbol_ifft2 经过 IFFT 的未加扰的信号
%Bdata 所有相位因子的组合
%phase_factor 输出的最优加扰因子组合
%X_opt 输出的最优加扰 PTS 信号
N = size(Bdata,2);
M = size(Symbol_ifft2,1);
PAPR_PTS = zeros( 1,N );
X_opt = zeros(M,1);
for n = 1:1:N
    Symbol_ifft = Symbol_ifft2 * Bdata(:,n); %计算加扰后的信号
    PowerPerBit = abs(Symbol_ifft).^2;
    PowerMean = mean(PowerPerBit);
    PowerMax  = max(PowerPerBit);
    PAPR_PTS(1,n) = PowerMax/PowerMean;%计算 PAPR
end
[~,K] =min(PAPR_PTS); %选取所有组合中最小的 PAPR
X_opt = Symbol_ifft2 * Bdata(:,K);
phase_factor = Bdata(:,K);
end

程序 3-30 “GA_PTS_nomal”,基于 GA-PTS-minPAPR 的 OFDM 优化模型
function [GA_PTS,best_w3] = GA_PTS_nomal(Symbol_ifft2,Gn,W,pop_size,initial_w)
%Symbol_ifft2 经过 IFFT 变换未加扰的信号
%Gn 迭代次数
%W 相位因子取值范围的对数
%pop_size 种群大小
%initial_w 初始化的种群
%GA_PTS 经过加扰后的 PTS 信号
%best_w3 最优的加扰因子
pc=0.8;%交叉概率
pm=0.025;%变异概率
current_w = initial_w ;
current_papr=inf * ones(1,pop_size);
best_papr=0;
for ii = 1:1:Gn
    Bdata = binary2factor(current_w,W);%二进制矩阵转变为相位因子矩阵
    Symbol_ifft = Symbol_ifft2 * Bdata;%加扰
    PowerPerBit = abs(Symbol_ifft).^2;
    PowerMean = mean(PowerPerBit);
    PowerMax  = max(PowerPerBit);
    papr = 1./(PowerMax./PowerMean);%计算 PAPR 并取倒数
    if (max(papr)> best_papr) %取每一代的最优个体
        [~,update_index2] = max(papr);
        best_papr = papr(update_index2);
    end
    [dad,mom]=selection_nomal(current_w',papr); %选择操作
    new_pop1 = crossover_nomal(dad,mom,pc); %交叉操作
```

```
    new_pop = bianyi(new_pop1,pm); %变异操作
    current_w = new_pop';
end
PAPR_GA= 1./best_papr;
best_w3=Bdata(:,update_index2);%最优的加扰相位因子
GA_PTS = Symbol_ifft2 * best_w3;%最优的 PTS 信号
end
```

程序 3-31 “selection_nomal”,GA-PTS-minPAPR 优化方案中的选择操作,按照适应度值选择合适的个体作为父代

```
function [dad mom] =selection_nomal(pop,fitness_value)
%pop 待选择的种群
%fitness_value 适应度值即 1/PAPR
%dad mom 被选择当作父辈和母辈的个体
[pop_size,~]=size(pop);
fit=fitness_value./sum(fitness_value);%计算每个适应度所占的概率
fit =  cumsum(fit);%概率叠加
for i = 1:pop_size%轮盘赌算法
    for j = 1:pop_size
        r=rand;
        if r < fit(j)
            dad(i,:)=pop(j,:);
            break
        end
    end
end
for i = 1:pop_size
    for j= 1:pop_size
        r=rand;
        if  r < fit(j)
            mom(i,:)=pop(j,:);
            break
        end
    end
end
end
```

程序 3-32 “crossover_nomal”,GA-PTS-minPAPR 优化方案中的交叉操作,采用轮盘赌算法进行交叉得到后代

```
function new_pop = crossover_nomal(dad,mom,pc)
%dad,mom 进行交叉的两个父辈种群
%pc 交叉概率
%new_pop 交叉得到的新种群
[pop_size gen_length]=size(dad);
for i=1:pop_size
    r=rand;
    if pc>r
        cpoint=randi([1 gen_length-1]);
        new_pop(i,:) = [dad(i,1:cpoint) mom(i,cpoint+1:end)];%单点交叉
    else
        new_pop(i,:)=dad(i,:);
```

```
    end
end
end
```

程序 3-33 “bianyi”,GA-PTS-minPAPR 优化方案中的变异操作,采用单点变异操作增加后代基因的多样性

```
function new_pop = bianyi(pop,pm)
%pop 交叉后的种群
%pm 编译概率
%new_pop 变异后的新种群
[pop_size gen_length]=size(pop);
new_pop = zeros(pop_size,gen_length);
for i=1:1:pop_size
    r=rand;
    if pm>r
        mpoint=randi([1 gen_length-1]);
        new_pop(i,:)=pop(i,:);
        new_pop(i,mpoint)=~pop(i,mpoint);%单点变异
    else
        new_pop(i,:)=pop(i,:);
    end
end
end
```

为了分析验证基于 BPSO-PTS 优化的 OFDM 功放优化方案,建立以下仿真模型:程序 3-21 为主程序,程序 3-22 和程序 3-33 为主程序调用的子程序。该实验平台考虑 1024 点 IFFT 变换的 OFDM 系统,调制方式采用 16-QAM,并采用相邻分割划分频域数据块,设定相位旋转因子 $\boldsymbol{b}_v \in \{\pm 1\}$,载波分块数 $V=16$。BPSO 算法的相关参数设置如下:粒子群大小 P 取 50 个个体,初始惯性权重 $\bar{\omega}_{\max}=0.9$,最终权重 $\bar{\omega}_{\min}=0.4$,学习因子 $c_1=c_2=2$,最大速度限制 $V_{\max}=0.2$,以及最大迭代次数 $t_{\max}=20$。

图 3-25 仿真了 Rapp 功放模型下不同智能算法优化的 PTS 信号的 IBO-MER 曲线。表 3-5 为图 3-25 的观测数据,结论如下:MER=40dB 时,OFDM-BPSO-PTS-minIBO 方案相较于原始 OFDM 信号有 4.2dB 的 IBO 增益,相较于 OFDM-BPSO-PTS-minPAPR 方案也有 0.25dB 的 IBO 增益,且比 OFDM-GA-PTS-minPAPR 方案有 1dB 的提升。表 3-6 为传统 PTS 方案和所提 PTS 方案的搜索算法复杂度分析。由表 3-6 可知,所提三种 PTS 方案的搜索复杂度相近,OFDM-BPSO-PTS-minIBO 方案相比传统 PTS 方案提升了 98.4% 的计算效率。

表 3-5　MER = 40dB 时,对比各种算法与未处理的 OFDM 信号 IBO 增益

测 试 方 案	IBO	IBO 增益
原始 OFDM 方案	9.2	0
OFDM-BPSO-PTS-minIBO 方案	5.0	4.2
OFDM-BPSO-PTS-minPAPR 方案	5.25	3.95
OFDM-GA-PTS-minPAPR 方案	6.0	3.2

表 3-6 MER_PTS = 40dB 时，对比各种算法的计算复杂度

方　　案	搜索复杂度	相较于 PTS 方案的计算效率的提升
OFDM-BPSO-PTS-minIBO 方案	$O(P\times t_{\max}=1000)$	98.4%
OFDM-BPSO-PTS -minPAPR 方案	$O(P\times t_{\max}=1000)$	98.4%
OFDM-GA-PTS -minPAPR 方案	$O(P\times t_{\max}=1000)$	98.4%
传统 PTS 方案	$O(W^V=2^{16})$	0

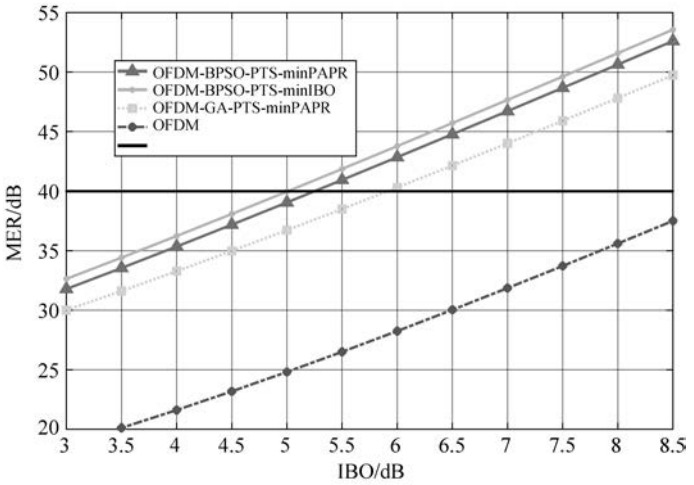


图 3-25 Rapp 功放模型下不同智能算法优化的 PTS 信号的 IBO-MER 曲线

图 3-26 分析了在不同优化准则下，BPSO-PTS 方案通过功率放大器的 IBO-MER 曲线。从表 3-5 和表 3-6 可以看出，在相同的搜索复杂度下，OFDM-BPSO-PTS-minIBO 方案相较于 OFDM-BPSO-PTS-minPAPR 方案有 0.25dB IBO 增益。

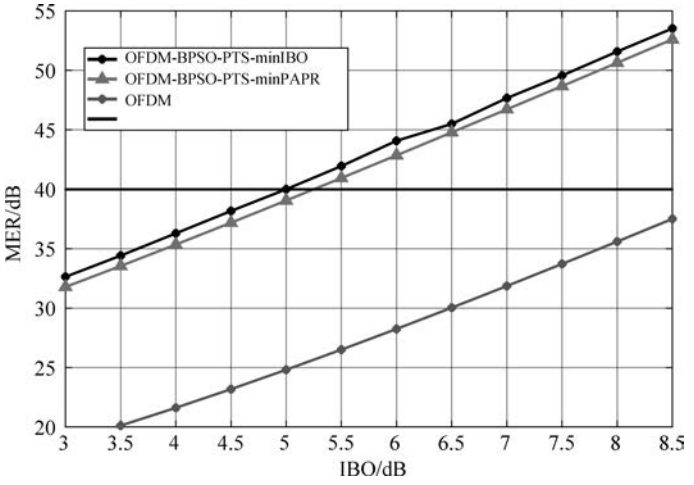


图 3-26 Rapp 功放模型下不同优化准则的 IBO-MER 曲线

图 3-27 给出了高斯信道条件下 OFDM-BPSO-PTS-minIBO 优化方案和原始 OFDM 信号在 Rapp 功率放大 MER=40dB 时的 SNR-BER 仿真结果。实验结果显示，相较于原始

OFDM 信号，OFDM-BPSO-PTS-minIBO 方案在 BER 为 10^{-3} 处获得了 7.5dB 的信噪比增益。

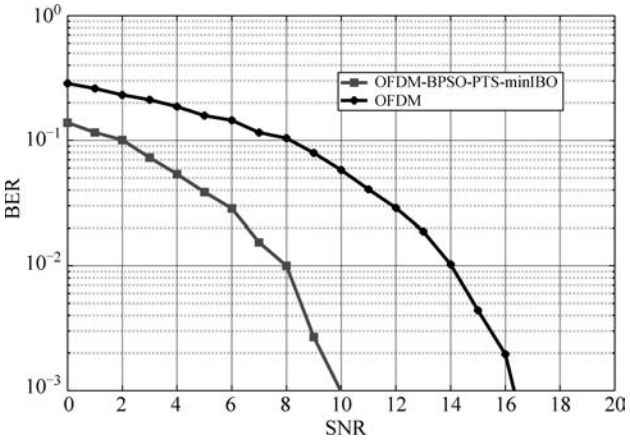


图 3-27　高斯信道下的 SNR-BER 曲线(SNR/dB)

3.3　多载波系统线性化技术

3.3.1　多项式模型

3.2.2 节对功放模型进行了分类分析，本节着重介绍功放模型的多项式表达形式。任何功放失真函数都可以用泰勒分解(或牛顿分解等)展开成多项式的形式，泰勒级数展开式中，项数越多越精确。

多项式模型并不是一种功率放大器的模型，而是一种表达式的模型，任何功率放大器的模型函数都可以分解为多项式的形式，通常以泰勒级数来描述功率放大器模型。泰勒级数的一般形式为

$$f(x)=\frac{f(x_0)}{0!}+\frac{f'(x_0)}{1!}(x-x_0)+\frac{f''(x_0)}{2!}(x-x_0)^2+\cdots+\frac{f^{(n)}(x_0)}{n!}(x-x_0)^n+R_n(x) \tag{3-31}$$

程序 3-34 “taylorex”，Rapp 模型的泰勒分解

```
function taylorex
x = sym('x','real');
% syms x;
syms a b;
% x=a*exp(j*b)
% y=x+0.5*x^3+0.1*x^5+0.2*x^7;
n=40;
pp=1;
sat=1;
f(x)=x./((1+(x/sat).^(2*pp)).^(1/2/pp));
y(x)=f(0);
```

```
g(x)=f(x);
for i=1:n
 g(x)=diff(g(x));
 an=factorial(i);
 y=y+g(0)/an*(x^i);
end
end
```

程序 3-34 以 Rapp 模型为例，进行泰勒分解。仿真结果如图 3-28 所示，将 Rapp 函数分解为泰勒级数的形式，多项式的阶数越高，多项式级数表达式越接近于 Rapp 原函数，同时高阶分量的多项式系数越来越小，高阶分量的功率占比也越来越低。

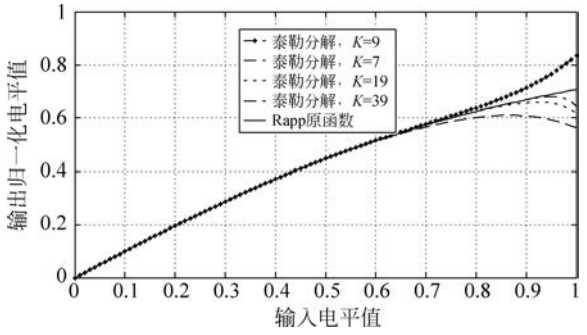


图 3-28　Rapp 函数的泰勒分解

以多项式模型来统一表达功放模型：

$$z(n)=\sum_{k=1}^{K}a_k\,|u(n)|^{k-1}u(n) \tag{3-32}$$

式中，每一项都表示其中的一个频率分量，每一项的系数为该频率分量的大小。随着阶数的增加，干扰频率分量离基波频率越来越远，失真产生的影响也越来越小。离中心频率较远的干扰频率分量可以用低通滤波器滤除；偶次项产生的失真分量，离基波较远，也可以用低通滤波器滤除；且功率放大之后存在滤除邻频干扰的低通滤波器，原则上分析功率放大器模型时，可以忽略偶次项的影响，因此只剩下中心频率附近的干扰频率分量即奇次项，简化为如下多项式模型：

$$\begin{aligned}z(n)&=\sum_{k=0}^{K}a_{2k+1}\,|u(n)|^{2k}u(n)\\&=a_1u(n)+a_3\,|u(n)|^2u(n)+a_5\,|u(n)|^4u(n)+\cdots+a_K\,|u(n)|^{2K}u(n)\end{aligned} \tag{3-33}$$

以双音信号功率放大为例，将双音信号输入到组成元件会产生非线性效应的系统之中，其中两个信号的频率分别为 ω_1 和 ω_2，且满足二者频率之差 $\Delta\omega=|\omega_1-\omega_2|\ll\dfrac{\omega_1+\omega_2}{2}$，双音信号可表示为

$$d(t)=A(\cos\omega_1 t+\cos\omega_2 t) \tag{3-34}$$

假定多项式模型为三阶多项式，得到：

$$a(t)=c_1d(t)+c_2d^2(t)+c_3d^3(t) \tag{3-35}$$

将双音信号代入式(3-35)中，得到：

$$
\begin{aligned}
a(t)=&\underbrace{c_1A(\cos\omega_1t+\cos\omega_2t)}_{T_1(t)}+\underbrace{c_2A^2(\cos\omega_1t+\cos\omega_2t)^2}_{T_2(t)}+\underbrace{c_3A^3(\cos\omega_1t+\cos\omega_2t)^3}_{T_3(t)}\\
=&c_1A\cos\omega_1t+c_1A\cos\omega_2t+\frac{1}{2}c_2A^2(1+\cos2\omega_1t)+\frac{1}{2}c_2A^2(1+\cos2\omega_2t)\\
&+c_2A^2\cos[(\omega_1-\omega_2)t]+c_2A^2\cos[(\omega_1+\omega_2)t]\\
&+c_3A^3\left(\frac{3}{4}\cos\omega_1t+\frac{1}{4}\cos3\omega_1t\right)+c_3A^3\left(\frac{3}{4}\cos\omega_2t+\frac{1}{4}\cos3\omega_2t\right)\\
&+c_3A^3\left\{\frac{3}{2}\cos\omega_1t+\frac{3}{4}\cos[(2\omega_2-\omega_1)t]+\frac{3}{4}\cos[(2\omega_2+\omega_1)t]\right\}\\
&+c_3A^3\left\{\frac{3}{2}\cos\omega_2t+\frac{3}{4}\cos[(2\omega_1-\omega_2)t]+\frac{3}{4}\cos[(2\omega_1+\omega_2)t]\right\}
\end{aligned}
\tag{3-36}
$$

式中，$T_1(t)$部分的频率分量项是期望得到的线性放大的信号，即线性分量；$T_2(t)$部分的频率分量项是系统的二阶项非线性失真产物；$T_3(t)$部分的频率分量项是系统的三阶项非线性失真产物。经过频率合成的信号中存在基波 ω_1 和 ω_2、二次谐波项 $2\omega_1$ 和 $2\omega_2$、三次谐波项 $3\omega_1$ 和 $3\omega_2$，还有其他基波组合项 $m\omega_1\pm n\omega_2(m,n=0,\pm1,\pm2,\cdots)$的频率成分。其中，由基波频率新组成的频率成分被称之为互调(Inter Modulation，IM)分量，$|m|+|n|$是互调阶数。新产生的频率分量对系统造成了非线性失真等不利影响时，就产生了互调干扰。

单独提取出中心频率及其附近无法滤除的频率分量项，即 $\cos\omega_1t$、$\cos\omega_2t$、$\cos[(2\omega_1-\omega_2)t]$、$\cos[(2\omega_1-\omega_2)t]$，得到基波和干扰项：

$$
\begin{aligned}
\tilde{a}(t)=&\left(c_1A+\frac{9}{4}c_3A^3\right)(\cos\omega_1t+\cos\omega_2t)\\
&+\frac{3}{4}c_3A^3\{\cos[(2\omega_1-\omega_2)t]+\cos[(2\omega_2-\omega_1)t]\}
\end{aligned}
\tag{3-37}
$$

其中，$2\omega_1-\omega_2$ 和 $2\omega_2-\omega_1$ 是滤波器无法滤除的部分，成为功放系统中非线性失真的主要来源。而其他远离中心频率的分量($\cos[(\omega_1-\omega_2)t]$、$\cos[(\omega_2-\omega_1)t]$、$\cos2\omega_1t$、$\cos[(\omega_1+\omega_2)t]$、$\cos3\omega_1t$、$\cos[(2\omega_1+\omega_2)t]$、$\cos[(2\omega_2+\omega_1)t]$、$\cos3\omega_2t$)都落在通带之外，可以被滤波器完全滤除。

图 3-29 可以直观地展示中心频率附近的频率分量，以及远离中心频率的可被滤波器滤除的频率分量。同理，再将双频率信号代入到五阶多项式得到：

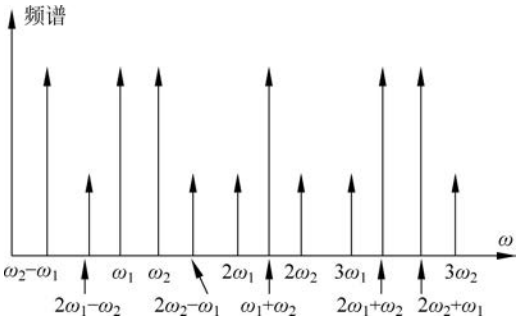


图 3-29　双频率信号的三阶互调失真

$$
\begin{aligned}
a_5(t) &= c_1A(\cos\omega_1 t+\cos\omega_2 t)+c_3A^3(\cos\omega_1 t+\cos\omega_2 t)^3+c_5A^5(\cos\omega_1 t+\cos\omega_2 t)^5\\
&= c_1A\cos\omega_1 t+c_1A\cos\omega_2 t\\
&\quad+c_3A^3\left(\frac{3}{4}\cos\omega_1 t+\frac{1}{4}\cos 3\omega_1 t\right)+c_3A^3\left(\frac{3}{4}\cos\omega_2 t+\frac{1}{4}\cos 3\omega_2 t\right)\\
&\quad+c_3A^3\left\{\frac{3}{2}\cos\omega_1 t+\frac{3}{4}\cos[(2\omega_2-\omega_1)t]+\frac{3}{4}\cos[(2\omega_2+\omega_1)t]\right\}\\
&\quad+c_3A^3\left\{\frac{3}{2}\cos\omega_2 t+\frac{3}{4}\cos[(2\omega_1-\omega_2)t]+\frac{3}{4}\cos[(2\omega_1+\omega_2)t]\right\}\\
&\quad+c_5A^5\left\{\frac{25}{8}\cos[(2\omega_2-\omega_1)t]+\frac{25}{8}\cos[(\omega_1+2\omega_2)t]+\frac{25}{8}\cos[(2\omega_1+\omega_2)t]\right\}\\
&\quad+c_5A^5\left\{\begin{array}{l}\frac{5}{16}\cos[(\omega_1-4\omega_2)t]+\frac{5}{16}\cos[(\omega_1+4\omega_2)t]+\frac{5}{16}\cos[(\omega_2-4\omega_1)t]+\\ \frac{5}{16}\cos[(4\omega_1+\omega_2)t]\end{array}\right\}\\
&\quad+c_5A^5\left(\frac{25}{16}\cos 3\omega_1 t+\frac{25}{16}\cos 3\omega_2 t+\frac{1}{16}\cos 5\omega_1 t+\frac{1}{16}\cos 5\omega_2 t\right)\\
&\quad+c_5A^5\left\{\frac{25}{8}\cos[(2\omega_1-\omega_2)t]+\frac{5}{8}\cos[(3\omega_2-2\omega_1)t]+\frac{5}{8}\cos[(2\omega_1+3\omega_2)t]\right\}\\
&\quad+c_5A^5\left\{\frac{5}{8}\cos[(3\omega_1-2\omega_2)t]+\frac{5}{8}\cos[(3\omega_1+2\omega_2)t]\right\}\\
&\quad+c_5A^5\left(\frac{25}{4}\cos\omega_1 t+\frac{25}{4}\cos\omega_2 t\right)
\end{aligned}
\tag{3-38}
$$

取出基波频率及其附近的频率分量项，得到：

$$
\begin{aligned}
\tilde{a}_5(t) &= \left(c_1A+\frac{9}{4}c_3A^3+\frac{25}{4}c_5A^5\right)(\cos\omega_1 t+\cos\omega_2 t)\\
&\quad+\left(\frac{3}{4}c_3A^3+\frac{25}{8}c_5A^5\right)\{\cos[(2\omega_2-\omega_1)t]+\cos[(2\omega_1-\omega_2)t]\}\\
&\quad+\frac{5}{8}c_5A^5\{\cos[(3\omega_2-2\omega_1)t]+\cos[(3\omega_1-2\omega_2)t]\}
\end{aligned}
\tag{3-39}
$$

式中无法滤除的 $2\omega_2-\omega_1$、$2\omega_1-\omega_2$、$3\omega_1-2\omega_2$、$3\omega_2-2\omega_1$ 频率分量是非线性失真的主要来源；其他诸如直流分量、二次谐波项等距离中心频率比较远的频率分量可被滤波器直接滤除。

3.3.2 多项式功放模型估计算法

以泰勒级数三阶展开多项式为例，在给定功放模型，即对应多项式的系数是定值的情况下，将双音信号（双频率信号的幅值 A 已知）代入多项式，将时域函数转化为频域的频谱图，即可测得主信号和干扰分量各个频率点的幅值，如式(3-37)所示，可以从高阶分量到低阶分量逐项推算出功放多项式的系数。计算过程如下：

$\cos[(2\omega_1-\omega_2)t]$ 和 $\cos[(2\omega_2-\omega_1)t]$ 分量的幅度一样，系数只有 c_3，测得该频率点处

的幅值为 M_1，则由 $\frac{3}{4}c_3A^3=M_1$ 得到：

$$c_3=\frac{M_1}{\frac{3}{4}A^3} \tag{3-40}$$

进而再根据 $\cos\omega_1 t$（此时 $\cos\omega_2 t$ 的分量幅度和 $\cos\omega_1 t$ 的一样）的频谱幅值 M_2，由 $c_1A+\frac{9}{4}c_3A^3=M_2$ 得到：

$$c_1=\frac{M_2-\frac{9}{4}c_3A^3}{A} \tag{3-41}$$

同理，对于泰勒级数五阶展开式，设测得的各频率点的幅值（按互调失真阶数从高到低）分别为 N_1、N_2、N_3，则多项式各系数分别为

$$c_5=\frac{N_1}{\frac{5}{8}A^5} \tag{3-42}$$

$$c_3=\frac{N_2-\frac{25}{8}c_5A^5}{\frac{3}{4}A^3} \tag{3-43}$$

$$c_1=\frac{N_3-\frac{25}{4}c_5A^5-\frac{9}{4}c_3A^3}{A} \tag{3-44}$$

通过监测各频率分量，计算功放多项式系数，以此拟合出功率放大器多项式模型。

程序 3-35 “HPA_estimation”，基于双音信号的 HPA 多项式模型估计

```
function HPA_estimation
close all;
clear all;
fsig = 10e6;
fcarrier1 = 6e6;
fcarrier2 =7e6;
q = 8;
figureindex = 0;
N = 1;
fsample = fsig * q;
ts = 1/fsample;
t = -0.01:ts:0.01-ts;
fft_pnts = length(t)/q;
Fcs = fsig/fft_pnts;
A=0.5;
f1 = cos(2 * pi * fcarrier1 * t);
f2 = cos(2 * pi * fcarrier2 * t);
Dtmf=A * (f1+f2);
% Dtmf=[0:0.001:1];
```

```
pp=1;
sat=1;
% f(x)=x/((1-(x/sat)^(2*pp))^(1/2/pp));
Sig1=Dtmf-0.5*Dtmf.^3;
% Sig2=Dtmf-0.5*Dtmf.^3+3/8*Dtmf.^5-5/16*Dtmf.^7;
% sat=1;
% f(x)=x/((1+(x/sat)^(2*pp))^(1/2/pp));
if 1
    SignalEnsem = fft(Sig1,fft_pnts)/fft_pnts*2;
    disp('Calculating Spectrum...');
    figureindex = figureindex + 1;
    figure(figureindex);
    %plot_OFDMTimeSig = 20*log10(abs(SignalEnsem));
    f = (-fsig/2:Fcs:(fft_pnts/2-1)*Fcs)/1e6;
    plot(f,fftshift(SignalEnsem));
    %ylabe0l('magnitude (dB)');
    %title('OFDM Spectrum');
end
end
```

设定双频率信号幅值均为 $A=0.5$，利用程序 3-35 仿真平台测得三阶展开式对应的频率幅值如表 3-7 所示。

表 3-7　三阶幅值

M_1	M_2
−0.04687	0.3594

原始系数和式(3-40)和式(3-41)的估计系数如表 3-8 所示。

表 3-8　三阶原始系数和估计系数

	c_1	c_3
原始	1	−0.5
估计	1.00002	−0.499946667

由此拟合三阶功放多项式曲线，如图 3-30 所示。从图中可看出，原始系数和估计系数

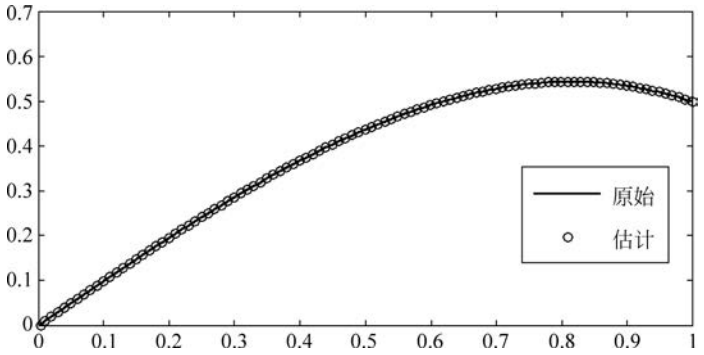


图 3-30　三阶原始、估计多项式曲线图

的误差可控制在0.01%以内，拟合曲线也近似重合。

五阶展开式对应频点幅值如表3-9所示。

表3-9　五阶幅值

N_1	N_2	N_3
0.007324	−0.01025	0.4326

原始系数和式(3-42)～式(3-44)的估计系数如表3-10所示。

表3-10　五阶原始系数和估计系数

	c_1	c_3	c_5
原始系数	1	−0.5	0.375
估计系数	0.99994	−0.499946667	0.3749888

由此拟合五阶功放多项式曲线，如图3-31所示，原始系数和估计系数的误差可以控制在0.01%左右，拟合曲线也近似重合。因此基于多项式的功放估计算法可以精确获得功放失真函数。

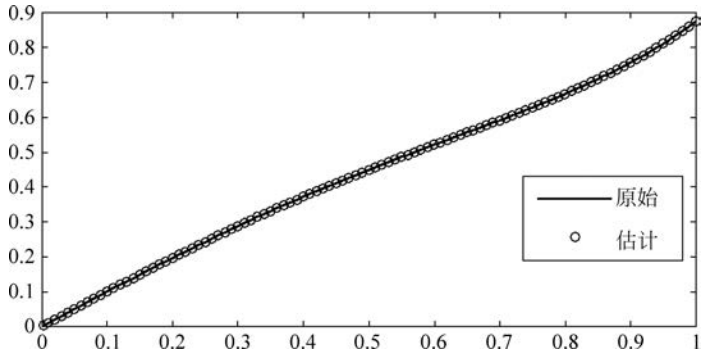


图3-31　五阶原始、估计多项式曲线图

3.3.3　基于多项式的非线性校正技术

在估计获得功放失真函数之后，可以基于反馈功放失真进行反函数补偿的预失真校正，以抵消功放失真的影响。预失真技术的基本思想如图3-32所示：输入信号功率为r_{i}，此时功放工作在非线性区，得到输出信号功率为r_{o}，而如果是线性放大则可以得到r_{op}，即理想功率放大。根据功放的特性曲线，如果对输入信号r_{i}进行预失真处理后得到r_{ip}，那么r_{i}在经过功放系统后虽然经历了非线性失真依然可以通过功率预失真得到理想放大效果的r_{op}。具体地说，对于每一个输入信号r_{i}，预失真器先进行预失真处理，得到对应的理想输出值，并代入功放特性函数的反函数，得到调整过的r_{ip}。将其作为功放输入信号，便可得到理想放大信号r_{op}。

注：预失真器对输入信号的范围有一定要求，即在稳态的非线性区，如图3-33所示。

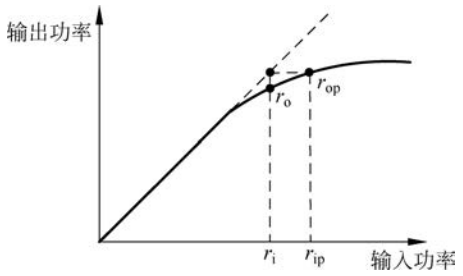


图 3-32 预失真技术的基本思想

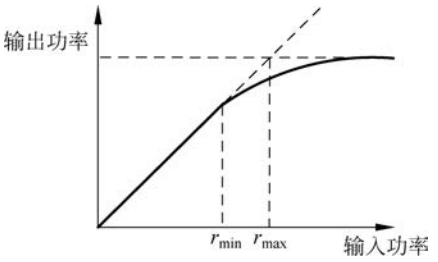


图 3-33 预失真器工作范围

如果输入信号功率小于 r_{min}，功放特征为线性放大，无须采取预失真处理。如果输入信号功率大于 r_{max}，则功放工作在饱和截止区，此种情况下功放失真无法预测，即无法执行预失真校正。因此，通常功放非线性校正的输入电压应在 r_{min} 和 r_{max} 之间。

1. 基于查表法的预失真技术

查表法就是构建基于功放模型的预失真查询表，根据这个查询表进行映射预处理。查询表包括：地址索引和查询表内容。功放输入信号通过地址索引找到查询表中的预失真修正值，完成离散条件下的映射和预失真，而映射关系由反函数校正自动生成。

图 3-34 是基于查表法对输入信号的幅度和相位校正的具体流程：设输入信号的幅度和相位分别为 ρ_n、φ_n，分别经过幅度预失真器和相位预失真器进行补偿，然后送到功放。对于幅度修整，输入信号经过 AM/AM 预失真器中已得到的功放幅度失真函数的反函数进行功率放大，整个预失真器-功放级联系统便可满足 $F(|V_m|)\cdot G(|V_d|)=K$，所以功放最终的输出幅度为 $K\rho_n$。对于相位预失真，输入信号经过 AM/PM 预失真器得到新相位 $\theta_n=\varphi_n+\gamma[\rho_n]$，进行功率放大得到 $\Psi_n=\Phi[r_n]+\theta_n$。校正的结果满足线性抵消 $\Psi_n=\varphi_n$，所以令 $\gamma[\rho_n]=-\Phi[r_n]$，则相位预失真满足 $\theta_n=\varphi_n-\Phi[r_n]$。

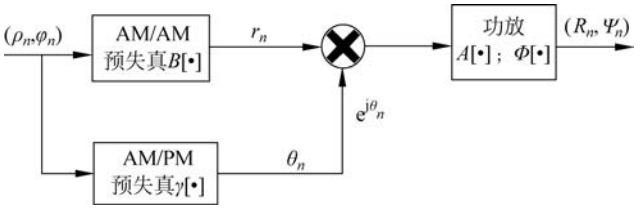


图 3-34 基于查表法的预失真线性化技术示意图

查表法中地址索引非常重要，不同的索引技术会影响查询表内数据的分布和效率，最终影响到硬件的复杂度和实施可能性。通常做法是根据输入信号功率进行校正映射，即映射到目标功率值。此方法电路简单，可依据输入信号直接映射。功率索引技术中，预失真表的分布特点是随输入信号的功率均匀分布。在输入信号幅度较小时表项内容少、分布稀疏，输入信号幅度较大时表项内容多、分布紧密。但是功放特性曲线正好相反，输入信号功率小时功放处在线性区，输出功率变化范围大，输入信号功率大时功放处在非线性区甚至是在饱和截止区，输出功率变化范围小，所以查询表时效率较低。查表法除了时延问题之外，另一个明显缺点就是需要特定的存储空间，预失真查询表作为一个本地数据库，必须足够大，以保证每一个输入信号都能顺利映射到对应的校正信号。如果是比较复杂的信号和变化比较快

的功放模型，匹配系统还要预留存储空间来存放匹配运算数据。并且现有的查询表需要不断更新，以适应功率放大器的动态变化，这成为制约它广泛应用的主要缺陷。

2. 基于多项式的预失真线性化技术

假设功放的整体非线性是由多项式表示的各阶非线性叠加而成的，则对各阶非线性失真进行校正，就能对功放整体进行优化。

以三阶多项式功放模型为例，将表示功放特性函数的泰勒级数展开成三项，即表明功放有三阶非线性失真，只有奇次项才会产生无法滤除的互调失真，并且假设预失真器-功放级联系统能完全消除非线性失真，如图 3-35 所示。

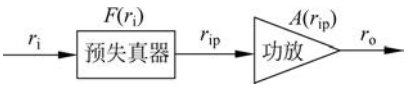


图 3-35　预失真系统

将功放增益因子和预失真器增益因子均归一化，得到 $r_o=r_{ip}+c_3r_{ip}^3$，满足预失真和功放失真相互抵消，$r_o=r_i$。通过变形得到 $r_{ip}=r_i-c_3r_{ip}^3$。令初始时 $r_i=r_{ip}$，进行迭代计算可得：

$$\begin{aligned} r_{ip1}&=r_i-c_3r_i^3\\ r_{ip2}&=r_i-c_3(r_i-c_3r_i^3)^3\\ r_{ip3}&=r_i-c_3\left[r_i-c_3(r_i-c_3r_i^3)^3\right]^3\\ &\vdots \end{aligned} \tag{3-45}$$

由式(3-45)可知，通过迭代次数的增加，r_{ip} 的值越来越接近理想情况，但是却引入了更多的高阶项。

功放的特性函数用泰勒级数五阶展开式表示，此时功放的输入是预失真器的输出：

$$r_o=A(r_{ip})\approx\beta_1r_{ip}+\beta_3r_{ip}^3+\beta_5r_{ip}^5 \tag{3-46}$$

与此功放对应的五阶预失真器模型为

$$r_{ip}=F(r_i)\approx\alpha_1r_i+\alpha_3r_i^3+\alpha_5r_i^5 \tag{3-47}$$

式中，多项式的系数均为复数。

由式(3-46)和式(3-47)可以得到：

$$\begin{aligned} r_o&=A\left[F(r_i)\right]=\eta_1r_i+\eta_3r_i^3+\eta_5r_i^5+\cdots\\ &=\beta_1(\alpha_1r_i+\alpha_3r_i^3+\alpha_5r_i^5)+\beta_3(\alpha_1r_i+\alpha_3r_i^3+\alpha_5r_i^5)^3\\ &\quad+\beta_5(\alpha_1r_i+\alpha_3r_i^3+\alpha_5r_i^5)^5+\cdots \end{aligned} \tag{3-48}$$

忽略偶次项，得到剩余奇次项的系数：

$$\begin{cases} \eta_1=\beta_1\times\alpha_1\\ \eta_3=\beta_1\times\alpha_3+\beta_3\times\alpha_1^2\times\bar{\alpha}_1\\ \eta_5=\beta_5\times\alpha_1^3\times\bar{\alpha}_1^2+\beta_3\times\alpha_1^2\times\bar{\alpha}_3+2\times\beta_1\times\alpha_3\times\alpha_1\times\bar{\alpha}_1+\beta_1\times\alpha_5\\ \quad\vdots \end{cases} \tag{3-49}$$

式中，$\bar{\alpha}_i$ 为 α_i 的共轭。为满足 $r_o=r_i$，必须消除三次项、五次项和高阶项，即 η_3、η_5 为零。假设功放的增益倍数为 1，$\eta_1=\beta_1=1$，将 β_1、η_1、η_3、η_5 的值代入式(3-49)得到：

$$\begin{cases} \alpha_1=1\\ \alpha_3=-\beta_3\\ \alpha_5=-(2\alpha_3\beta_3+\bar{\alpha}_3\beta_3+\beta_5) \end{cases} \tag{3-50}$$

同样，可以获得更高阶的系数，即

$$\alpha_7 = -(2\alpha_5\beta_3 + \bar{\alpha}_5\beta_3 + 2|\alpha_3|^2\beta_3 + (\alpha_3)^2\beta_3 + 3\alpha_3\beta_5 + 2\bar{\alpha}_3\beta_5) \tag{3-51}$$

$$\alpha_9 = -\frac{\begin{pmatrix} 2\beta_5\bar{\alpha}_5\alpha_1^3\bar{\alpha}_1 + \beta_5\alpha_1^3\bar{\alpha}_3^2 + 6\beta_5\alpha_1^2|\alpha_3|^2\bar{\alpha}_1 + 3\beta_5\alpha_5|\alpha_1|^4 \\ +\beta_3\bar{\alpha}_7\alpha_1^2 + 3\beta_5\alpha_1\alpha_3^2\bar{\alpha}_1^2 + 2\beta_3\bar{\alpha}_5\alpha_3\alpha_1 + 2\beta_3\alpha_7|\alpha_1|^2 \\ +2\beta_3\alpha_5\bar{\alpha}_3\alpha_1 + \beta_3\alpha_3^2\bar{\alpha}_3 + 2\beta_3\alpha_5\alpha_3\bar{\alpha}_1 \end{pmatrix}}{\beta_1} \tag{3-52}$$

如图 3-36 所示，当反馈得到功放的多项式形式的非线性拟合曲线时，可以基于多项式反函数运算得到预失真器的多项式校正模型。

输入信号由预失真器进行预失真处理后经过 D/A 转换后进行滤波，预失真器的多项式的高阶被滤波器滤除。功放对预失真处理过的信号进行放大，放大后输出信号的高阶项同样被滤除。因此进行多项式预失真校正时，也无法执行高阶多项式的补偿。

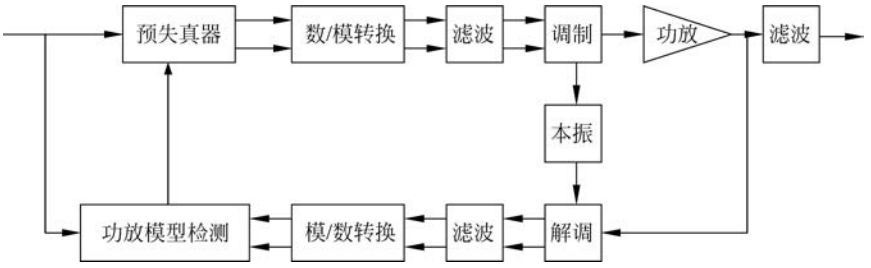


图 3-36　预失真器-功放系统原理框图


仿真视频

3.3.4　非线性校正的 OFDM 发射机系统功放效率优化仿真平台

程序 3-36 “nonlinear_correction”，OFDM 发射机非线性校正模型

```
function nonlinear_correction
warning off;
global DVBTSETTINGS;
global FIX_POINT;
global papr_fig;
global log_fid;
%参数设置%
DVBTSETTINGS.mode = 2; %模式选择: 2 for 2K, 8 for 8K
DVBTSETTINGS.BW = 8; %带宽
DVBTSETTINGS.level = 2; %星座映射选项: 2 for QPSK, 4 for 16QAM, 6 for 64QAM
DVBTSETTINGS.alpha = 1;
DVBTSETTINGS.cr1 = 1; %编码码率: cr/(cr+1)
DVBTSETTINGS.cr2 = 7;
nframe = 2;%帧数
nupsample = 4; %采样率
DVBTSETTINGS.GI=1/32; %保护间隔选项:1/4, 1/8, 1/16, and 1/32
wv_file = 'C:\Documents and Settings\Administrator\dvbtQPSK2kGI32_papr.wv';%测试流存储
ACE_ENABLED = true;
FIX_POINT = false;
skip_step = 1;
```

```
tstart=tic;
fig=1;
log_file='';
papr_fig = 1;
close all;
DVBTSETTINGS.T = 7e-6 / (8 * DVBTSETTINGS.BW);%基带基本周期
DVBTSETTINGS.fftpnts = 1024 * DVBTSETTINGS.mode;
DVBTSETTINGS.Tu = DVBTSETTINGS.fftpnts * DVBTSETTINGS.T;%OFDM 数据体周期
DVBTSETTINGS.Kmin = 0;%最小载波数值
DVBTSETTINGS.Kmax = 852 * DVBTSETTINGS.mode;%最大载波数值
NTPSPilots = 17;
NContinualPilots = 44;
DVBTSETTINGS.NData = 1512 * DVBTSETTINGS.mode/2;
DVBTSETTINGS.NTPSPilots = NTPSPilots * DVBTSETTINGS.mode/2;
DVBTSETTINGS.NContinualPilots = NContinualPilots * DVBTSETTINGS.mode/2;
DVBTSETTINGS.N = DVBTSETTINGS.Kmax - DVBTSETTINGS.Kmin + 1;%载波数
if DVBTSETTINGS.mode ~= 2 && DVBTSETTINGS.mode ~= 8
    error('mode setting error');
end
%delta=DVBTSETTINGS.GI * DVBTSETTINGS.Tu; %保护间隔长度
%Ts=delta+DVBTSETTINGS.Tu; %OFDM 完整符号周期
fsymbol = 1/DVBTSETTINGS.T;
fsample=nupsample * fsymbol; %中心载频
%固定参数%
SYMBOLS_PER_FRAME = 68;
if isempty(log_file)
    log_fid= 1;
else
    log_fid = fopen(log_file, 'w+');
end
fprintf(log_fid, 'Generating %d frames...\n', nframe);
%生成 OFDM 频域信号%
if skip_step == 1
    load signal0 signal0;%提取生成原始 OFDM 信号,见程序 3-1
    load signal signal;%提取生成 ACE 峰均比抑制 OFDM 信号,见程序 3-1 和程序 3-20
    load data data
end
toc;
%HPA 功率放大器模型%
pp=10;%平滑因子
data_real = real(data);
data_imag = imag(data);
IBO =[5:0.1:13];
signal1=signal;
data_real1=data_real;
data_imag1=data_imag;
PA_rms = signal0 * signal0'/length(signal0);
PA_rms1=signal * signal'/length(signal);
```

```
aa=10*log10(PA_rms1/PA_rms)
%
if 1%ACE 峰均比抑制 OFDM 信号非线性校正
    for var = 1:length(IBO)
        signal=signal1;
        data_real=data_real1;
        data_imag=data_imag1;
        IBO_level = IBO(var);
        max_clip_HPA = sqrt(PA_rms*(10^(IBO_level/10)));
        signal=signal/max_clip_HPA;
        a1=1;a3=-0.1-j*0.1;a5=-0.05-j*0.05;%多项式功放模型系数
        %预失真非线性校正%
        b1=1;b3=-a3; b5=-(2*b3*a3+conj(b3)*a3+a5);
        b7=-(2*b5*a3+conj(b5)*a3+2*(abs(b3)^2)*a3+(b3^2)*a3+3*b3*a5+2*
        conj(b3)*a5);
        % b9=-(2*a5*conj(b5)*(b1^3)*conj(b1)+a5*(b1^3)*(conj(b3))^2+6*a5);
        %signal=b1*signal+b3*signal.^3+b5*signal.^5+b7*signal.^7;
        signal=b1*signal+b3*signal.^3+b5*signal.^5;
        % HPA 非线性失真%
        signal=a1*signal+a3*signal.^3+a5*signal.^5;
        point = find((abs(signal) >= 1));
        signal(point)=signal(point)./abs(signal(point));
        signal=signal*max_clip_HPA;
        if 0
            figure(fig);%fig=fig+1;
            [Pxx, f]=pwelch(signal(1:SYMBOLS_PER_FRAME*DVBTSETTINGS.fftpnts*
            (1+DVBTSETTINGS.GI)),[],[],[],fsample);
            Pxx = 10*log10(fftshift(Pxx));%
            plot(f,Pxx);
        end
        %if var==1
        %   data_ACE=signal;
        %else
        %   datao=signal;
        %end
        %HPA 失真信号解调%
        signal_rx = downsample(signal, nupsample);
        signal_rx=reshape(signal_rx, DVBTSETTINGS.fftpnts* (1+DVBTSETTINGS.GI),
        SYMBOLS_PER_FRAME, []);
        signal_rx=signal_rx(1+DVBTSETTINGS.fftpnts*
        DVBTSETTINGS.GI:DVBTSETTINGS.fftpnts* (1+DVBTSETTINGS.GI), :, :);
        data_rx = fft(signal_rx) / sqrt(DVBTSETTINGS.fftpnts);

        data_rx=[data_rx(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.
        Kmin)/2+1:DVBTSETTINGS.fftpnts, ...
        :, :); data_rx(1:DVBTSETTINGS.fftpnts-...
        (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
        data_rx = data_rx(1:DVBTSETTINGS.N, :, :);
```

```
        %修改 MER 算法%
        data_rx_real = real(data_rx);
        data_rx_imag = imag(data_rx);
        pos=find(abs(data_rx_real)> abs(data_real));
        data_rx_real(pos) = 0;
        data_real(pos)=0;
        pos=find(abs(data_rx_imag)> abs(data_imag));
        data_rx_imag(pos) = 0;
        data_imag(pos)=0;
        data_rx=complex(data_rx_real,data_rx_imag);
        data=complex(data_real,data_imag);
        err = data_rx - data;
        ap_data = average_power(reshape(data, 1, []));
        ap_err = average_power(reshape(err, 1, []));
        mer(var) = 10 * log10(ap_data/ap_err);%(clip_time)
    end
end
if 1%原始 OFDM 信号非线性校正
    for var = 1:length(IBO)
        signal=signal0;
        data_real=data_real1;
        data_imag=data_imag1;
        IBO_level = IBO(var);
        max_clip_HPA = sqrt(PA_rms * (10^(IBO_level/10)));
        signal=signal/max_clip_HPA;
        a1=1;a3=-0.1-j * 0.1;a5=-0.05-j * 0.05;%多项式功放模型系数
        %预失真非线性校正%%
        b1=1;b3=-a3; b5=-(2 * b3 * a3+conj(b3) * a3+a5);
        b7=-(2 * b5 * a3+conj(b5) * a3+2 * (abs(b3)^2) * a3+(b3^2) * a3+3 * b3 * a5+2 *
        conj(b3) * a5);
        % b9=-(2 * a5 * conj(b5) * (b1^3) * conj(b1)+a5 * (b1^3) * (conj(b3))^2+6 * a5);
        % signal=b1 * signal+b3 * signal.^3+b5 * signal.^5+b7 * signal.^7;
        signal=b1 * signal+b3 * signal.^3+b5 * signal.^5;
        % HPA 非线性失真%%
        signal=a1 * signal+a3 * signal.^3+a5 * signal.^5;
        point = find((abs(signal) >= 1));
        signal(point)=signal(point)./abs(signal(point));
        signal=signal * max_clip_HPA;
        if 0
            figure(fig);%fig=fig+1;
            [Pxx, f]=pwelch(signal(1:SYMBOLS_PER_FRAME * DVBTSETTINGS.fftpnts *
            (1+DVBTSETTINGS.GI)),[],[],[],fsample);
            Pxx = 10 * log10(fftshift(Pxx));%
            plot(f,Pxx);
        end
        %HPA 失真信号解调%
        signal_rx = downsample(signal, nupsample);
        signal_rx = reshape(signal_rx, DVBTSETTINGS.fftpnts * (1+DVBTSETTINGS.GI),
```

```
            SYMBOLS_PER_FRAME, [ ]);
            signal_rx=signal_rx(1+DVBTSETTINGS.fftpnts*
            DVBTSETTINGS.GI:DVBTSETTINGS.fftpnts* (1+DVBTSETTINGS.GI), :, :);
            data_rx = fft(signal_rx) / sqrt(DVBTSETTINGS.fftpnts);

            data_rx=[data_rx(DVBTSETTINGS.fftpnts-(DVBTSETTINGS.Kmax+DVBTSETTINGS.
            Kmin)/2+1:DVBTSETTINGS.fftpnts…
            , :, :); data_rx(1:DVBTSETTINGS.fftpnts-…
            (DVBTSETTINGS.Kmax+DVBTSETTINGS.Kmin)/2, :, :)];
            data_rx = data_rx(1:DVBTSETTINGS.N, :, :);
            %修改 MER 算法%
            data_rx_real = real(data_rx);
            data_rx_imag = imag(data_rx);
            pos=find(abs(data_rx_real)> abs(data_real));
            data_rx_real(pos) = 0;
            data_real(pos)=0;
            pos=find(abs(data_rx_imag)> abs(data_imag));
            data_rx_imag(pos) = 0;
            data_imag(pos)=0;
            data_rx=complex(data_rx_real,data_rx_imag);
            data=complex(data_real,data_imag);
            err = data_rx - data;
            ap_data = average_power(reshape(data, 1, [ ]));
            ap_err = average_power(reshape(err, 1, [ ]));
            mer0(var) = 10 * log10(ap_data/ap_err);%(clip_time)
        end
    end
    if 1
        figure
        plot(IBO,mer,'r')
        hold on;
        plot(IBO,mer0,'b')
    end
    end
```

通过程序 3-36 搭建了预失真非线性校正的 OFDM 发射机功放优化仿真平台，其中前续章节程序 2-2～程序 2-12 为程序 3-36 调用的子程序模块。实验平台设置 8MHz 频谱带宽，2048 点的 IFFT/FFT 变换，有效载波数是 1512 点，星座映射方式为 QPSK 调制。设定功放模型采用五阶多项式形式，多项式系数设置为

$$\begin{cases}\beta_1 = 1\\ \beta_3 = -0.1 - 0.1\mathrm{j}\\ \beta_5 = -0.05 - 0.05\mathrm{j}\end{cases} \tag{3-53}$$

通过图 3-36 分析可知，输入信号的高阶非线性失真会被滤波器滤除，输出信号的高阶校正量也会被滤除，所以高阶预失真补偿无法完成高阶校正。因此，实验选取三阶、五阶和七阶非线性校正，整理、分析校正结果，并对失真补偿程度进行评估。评估以 IBO-MER 作为功放效率和信号质量指标。

如图 3-37 所示，IBO-MER 曲线中，对原始 OFDM 信号进行未校正、三阶校正、五阶校正、七阶校正四种情况进行分析。选取 MER＝40dB 作为质量约束，可得未校正、三阶校正、五阶校正、七阶校正时 IBO 的值分别为 15dB、11dB、9dB、9dB。由此可知利用非线性校正可以获得极大的 IBO 增益和功放效率优化，同时随着非线性校正阶数的增加，功率占比逐渐减小，校正增益也随之减少，甚至于在五阶校正就已经收敛。

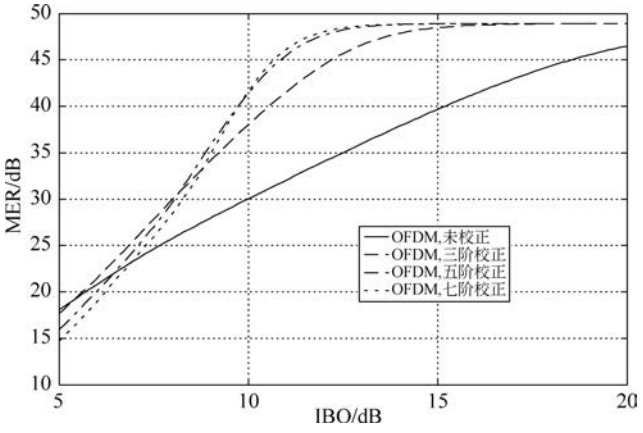


图 3-37　IBO-MER 仿真结果图 1

如图 3-38 所示，IBO-MER 曲线中，对 ACE 峰均比抑制的 OFDM 信号进行未校正、三阶校正、五阶校正、七阶校正四种情况进行分析，即在峰均比抑制的基础上考虑非线性校正的叠加增益。MER＝40dB 时，得到 ACE 峰均比抑制 OFDM 未校正、三阶校正、五阶校正、七阶校正时 IBO 的值分别为 15dB、9dB、7.5dB、7dB。由此可知 ACE 峰均比抑制的信号随着非线性校正的应用也可以获得极大的 IBO 增益和功放效率优化，同时随着校正阶数的增加，校正增益也随之减少，但是相较于原始信号，七阶校正依然可以获得功放优化增益。对比图 3-37 和图 3-38，通过峰均比抑制和非线性校正的叠加可以获得功率的双重增益。

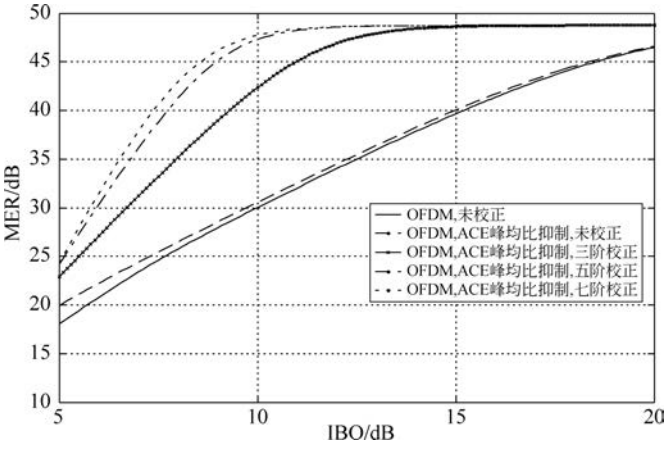


图 3-38　IBO-MER 仿真结果图 2

从上述对校正效果的分析可知，基于多项式的预失真线性化技术的校正效果增益明显。如图 3-39 所示，可将图 3-37 的校正效果直观反映在 Rapp 模型中的功放特性曲线上，进行

高阶失真校正时，校正阶数越高失真补偿越接近线性。未校正的功放特性曲线上有占比很大的非线性区。进行三阶校正后，功放在饱和截止功率不变的情况下，特性曲线的非线性区有一定程度的扩展，在相同输入信号的情况下相比于未校正时功率增益得到提升。进行五阶校正后，特性曲线的非线性区有所提升。三阶校正相比于未校正时饱和区更窄，五阶校正相比于三阶校正时饱和区有所缩短。说明校正对于改善功放效率起到了明显的作用。

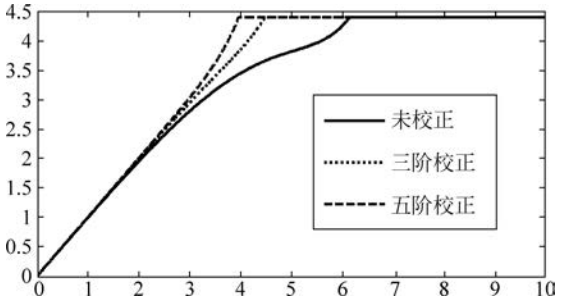


图 3-39　功放特性曲线