第 5 章

CHAPTER 5

OFDM 同步技术


源代码

移动通信的地面射频信号受到障碍物和信道界面的影响，通常会以多径方式传播，使得信号经由不同路径到达，产生多径效应，增加了空间自由度的同时也造成了时延拓展，导致不同路径到达信号的错位叠加，产生码间干扰。同时由于高速移动本身信道的多普勒效应，会引起时变特征和频谱扩展问题。为了克服移动通信中固有的多径效应和多普勒效应问题，同步和信道估计在接收端是必不可少的。

OFDM 由于实现简单、可抵抗信道多径衰落的优势，在全世界范围获得了极为成功的商业应用。但也存在诸多亟须解决的问题，例如 OFDM 系统对同步误差和载波频偏极其敏感，调制信号存在高峰均比引起的功放失真等缺陷。同步是 OFDM 系统中的关键性技术问题。随着 5G 通信的发展，超高频、大带宽得以广泛使用，通信系统对时间和频率偏差更为敏感，精确地实现载波同步、时钟同步与定时同步，是系统优越性和先进性的重要保障和先决条件。所谓同步是指收发双方在时间、频域甚至空域上步调一致，故又称为时间、频率、空间同步。任何通信系统都会配置完整的信号发射和接收过程，而接收信号实际上就是从噪声干扰与信道畸变中提取有效信号，获取完整的发送信息。提取信号就是估计信号的某个或数个特征参数，如振幅、频率、相位与时间等。同步的主要过程就是信号参数的估值过程。本章主要从 OFDM 同步信号的性能要求、同步信号的提取方法、OFDM 数字通信的同步系统结构及同步信号对通信系统误码率的影响等方面对 OFDM 系统中的同步技术进行介绍。

5.1 OFDM 同步系统结构及同步偏差分析

5.1.1 OFDM 同步系统结构

如图 5-1 所示，OFDM 接收机通常分为内接收机和外接收机两部分，其中内接收机主要包含同步、信道估计和均衡极大模块，广义上整个内接收机信号恢复的过程都可定义为同步，狭义的同步主要指时间、频域的资源片精确定位；而外接收机主要包含信道译码和信源解码，实现误比特的校正、解复用和信源呈现。

如图 5-2 所示，OFDM 采用了频域复用和时间复用技术，发射机的调制过程完成了时间和频域的二维资源分配，提升了资源利用率和信道容量，并可支持相关通道的正交隔离，获得了极大的鲁棒性优势。保证上述优势的前提是在解调的过程中实现精确的资源界定和识别，即时间窗、带宽范围的精准同步。在接收机对子载波进行解调之前必须进行两项同步

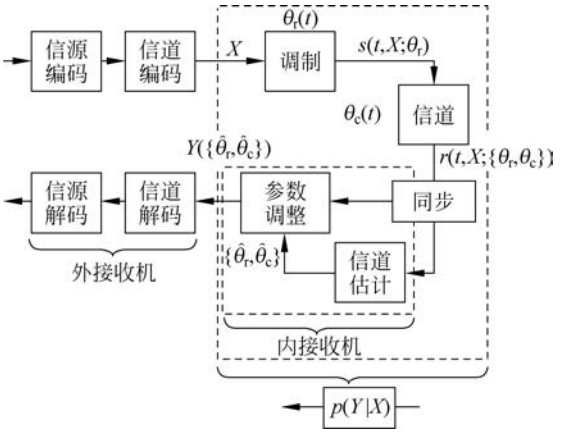


图 5-1　OFDM 系统结构图

工作：找出符号边界的位置和最佳的时间间隔(最佳的时间间隔一般是一个符号帧的长度)，以规避信道间干扰(ICI)和符号间干扰(ISI)；估计和补偿接收信号的载波频率偏移，因为任何偏移都会引入子载波间干扰和符号间干扰。尽管 OFDM 系统相对于单载波系统来说对相位噪声和频率偏差更为敏感，但是事实证明，利用循环前缀和加入特殊的 OFDM 训练符号等方法，可以获得较好的时间同步和频率同步。

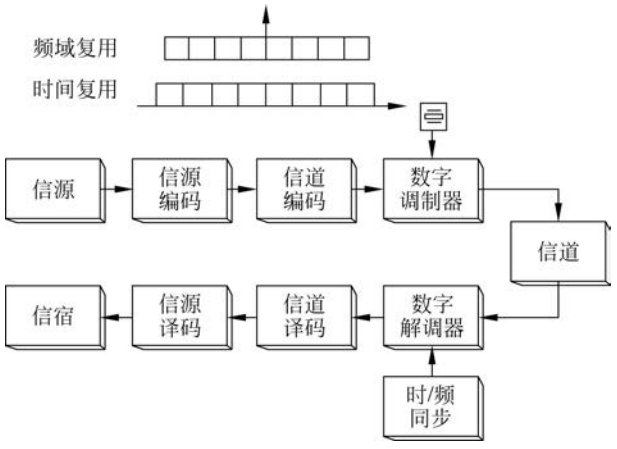


图 5-2　OFDM 系统同步模块

OFDM 系统数学模型中，通过在 IFFT/FFT 窗口内进行载波调整，完成时间、频域的资源分配和信息携带。连续的 OFDM 基带调制和解调的数学表达式为

$$s(t)=\sum_{k=0}^{N-1}X_k\,\mathrm{rect}(t)\,\mathrm{e}^{\mathrm{j}2\pi f_k t} \tag{5-1}$$

式中，N 为载波数；$\mathrm{rect}(t)$为宽度为 T_s 的 IFFT/FFT 时间窗口；f_k 为第 k 个载波的调制频率；X_k 为 OFDM 符号第 k 个子载波所承载的频域符号。

$$\widetilde{X}_m=\int_{t=0}^{T_s}\left[\sum_{k=0}^{N-1}X_k\,\mathrm{rect}(t)\,\mathrm{e}^{\mathrm{j}2\pi f_k t}\right]\mathrm{e}^{-\mathrm{j}2\pi f_m t}\,\mathrm{d}t$$

$$= \begin{cases} X_m T_s, & k = m \\ 0, & k \neq m \end{cases} \tag{5-2}$$

式(5-2)所示解调过程需要精确的传输参数加以修正，如载波偏移、采样定时偏差、符号定时偏差等，亟须这些估计量补偿接收信号，辅助完成解调。

在模拟解调过程中，解调之前需要对 OFDM 窗口进行精确定界，完成符号定界的功能，具体地就是同步 OFDM 符号积分区间，积分区间起止时刻和 IFFT/FFT 窗口位置一致；载波同步，接收端本地振荡器频率与发送端载波频率一致。

而离散的 OFDM 调制和解调结构为

$$s(n) = \sum_{k=0}^{N-1} X_k \mathrm{e}^{\mathrm{j}2\pi \frac{n}{N} k} \tag{5-3}$$

$$\begin{aligned} \widetilde{X}_k &= \sum_{k=0}^{N-1} r(k\Delta t)\, \mathrm{e}^{-\mathrm{j}2\pi f_m k\Delta t} \Delta t \\ &= T_s \sum_{n=0}^{N-1} \left[\frac{1}{N} \sum_{k=0}^{N-1} X_k \mathrm{e}^{\mathrm{j}2\pi \frac{n-m}{N} k} \right] \\ &= \begin{cases} X_k T_s, & n = m \\ 0, & n \neq m \end{cases} \end{aligned} \tag{5-4}$$

对应地，离散信号解调过程的同步主要完成：载波同步，接收端本地振荡器频率与发送端载波频率一致，值得注意的是数字信号的频偏估计、补偿与模拟信号的同步有着本质区别；定时同步，精确定时 OFDM 窗口开始的位置；采样同步，接收端 A/D 和发送端 D/A 时钟频率一致。通过定时同步、采样同步以及 OFDM 周期采样点数，可以确定 OFDM 窗口范围。符号同步的目的是找到 FFT 窗口的起始位置，可以采用特殊的训练序列来进行符号定时，也可以利用双同步头的相关特性进行定时。采样时钟同步的目的是使接收机的采样时钟频率和发射机的一致，采样时钟频率误差会引起 ICI，而且采样时钟误差还会导致符号定时的漂移，而使符号定时性能恶化。

图 5-3 为数字同步系统的结构，A/D 转换在处理器的最前端。同步可以通过多次迭代实现，首先通过相位估计和频域估计获得频率偏差，用以完成载波恢复，进而执行定时同步和采样同步，完成时间偏差补偿。通常时间同步需要频率偏差的补偿作为前提，因此频偏补偿先于时间补偿。其中采样时钟周期与符号周期无关。

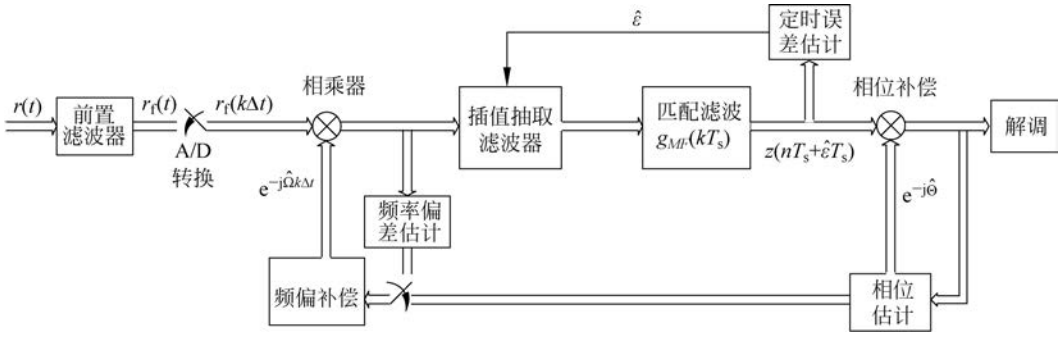


图 5-3　数字同步系统的结构

5.1.2 OFDM 同步偏差

OFDM 系统的同步技术分为时间同步和频率同步。下面将对 OFDM 系统的时间同步偏差和频率同步偏差加以分析。

1）频域偏差

在 OFDM 系统中，发射机基带调制的数字信号经 D/A 转换变换成模拟信号，经 RF 中心频率 f_c 上变频调制后通过无线信道传输，接收端 RF 接收解调模拟信号，进行下变频从 RF 搬移到基带。再经 A/D 数字化，最后送到 OFDM 解调器。由于发送端和接收端的载波频率存在偏差，每一个对 t 的信号采样都包含未知的相位因子 $e^{j2\pi\Delta f_c t}$，其中 Δf_c 是未知的载波频率偏差。为了不破坏子载波之间的正交性，在接收端进行 FFT 变换之前，必须对这个未知的相位因子进行估计和补偿。

如图 5-4 所示，发送端和接收端之间存在频率偏差 Δf_c，导致所有子载波都会一定程度地偏离发射机的调制频率，频偏产生的失真随着时间的延长而线性增大。造成频偏的原因主要为：

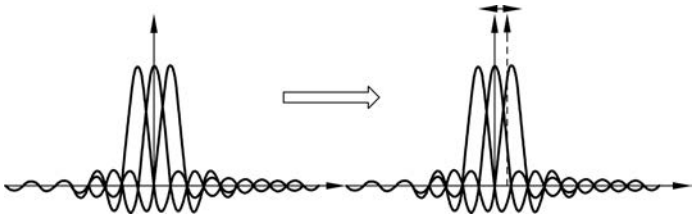


图 5-4 接收频率偏差

（1）发射机和接收机中振荡器存在相对偏差。振荡器由于元器件老化、环境电气参数变化和电路噪声等原因，引起频率精度偏差。

（2）发射台和接收台之间由于相对移动引起的多普勒频移。

为描述频偏产生的相位失真，设定通带信号为

$$\bar{s}(t)=\mathrm{Re}[s(t)e^{j2\pi f_c t}] \tag{5-5}$$

式中，$s(t)$为基带等效信号；f_c 为中心频率。

接收环节的本地载波频率为 f'_c，则接收机下变频及低通滤波后的信号为

$$\begin{aligned} y(t)&=s(t)e^{j2\pi f_c t}e^{-j2\pi f'_c t}\\ &=s(t)e^{j2\pi(f_c-f'_c)t} \end{aligned} \tag{5-6}$$

式中，$e^{-j2\pi f'_c t}$ 为本地载波。

假设收发两端频偏为 $\mathrm{d}f=f_c-f'_c$，则归一化的频偏为载波间隔的系数：

$$\varepsilon=\frac{\mathrm{d}f}{\Delta f} \tag{5-7}$$

式中，Δf 为 OFDM 载波间隔。

将接收信号进行数字化采样，一个 OFDM 周期 T_s 内包含 N 个子载波，N 个采样点，则离散形式可以表示为

$$y(n)=s(n)\mathrm{e}^{\mathrm{j}2\pi\mathrm{d}f\frac{T_s}{N}n} \tag{5-8}$$

式中，$t=\frac{T_s}{N}n=n\Delta t$，$n$ 为时间 t 对应于周期的采样点，Δt 为采样间隔。

进一步地，由于载波间隔和周期是倒数关系 $\Delta f=\frac{1}{T_s}$，可以得到：

$$\begin{aligned} y(n)&=s(n)\mathrm{e}^{\mathrm{j}2\pi\mathrm{d}f\frac{T_s}{N}n}\\ &=s(n)\mathrm{e}^{\mathrm{j}2\pi\varepsilon\Delta f\frac{T_s}{N}n}\\ &=s(n)\mathrm{e}^{\mathrm{j}2\pi\varepsilon\frac{n}{N}} \end{aligned} \tag{5-9}$$

载波频偏相当于时域采样序列增加了一个等效的指数因子 ε，也是采样失真产生的重要因素之一。载波频偏失真的影响随着时间延长不断增大。频偏对系统的影响可写为归一化频偏 ε 的函数。

考虑频偏对 OFDM 接收信号的影响，设定发射端基带信号的表示形式为

$$s(n)=\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{n}{N}k} \tag{5-10}$$

接收端存在频率偏移，此时信号应表示为

$$y(n)=\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{n}{N}(k+\varepsilon)}+w(n) \tag{5-11}$$

式中，$w(n)$为高斯白噪声。

接收信号经过 FFT 变换可得：

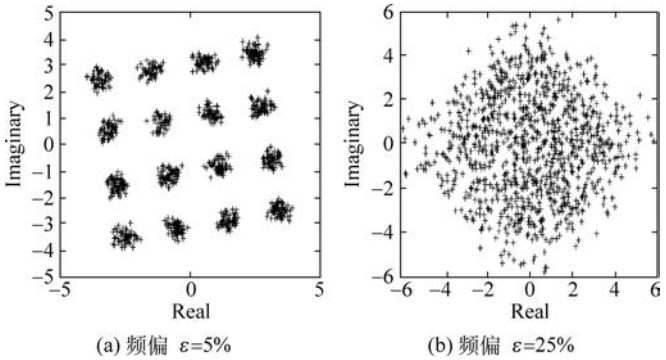
$$\begin{aligned} Y(l)&=\frac{1}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{n}{N}(k+\varepsilon)}\mathrm{e}^{-\mathrm{j}2\pi\frac{n}{N}l}+W(k)\\ &=\sum_{k=0}^{N-1}X_k\frac{\sin(\pi\varepsilon)\mathrm{e}^{\mathrm{j}\pi\varepsilon}}{N}\left(\cot\left(\frac{\pi}{N}(k+\varepsilon-l)\right)-\mathrm{j}\right)+W(k) \end{aligned} \tag{5-12}$$

式中，l 为接收端频域标号。

令 $S_{k-l}=\frac{\sin(\pi\varepsilon)\mathrm{e}^{\mathrm{j}\pi\varepsilon}}{N}\left(\cot(\frac{\pi}{N}(k+\varepsilon-l))-\mathrm{j}\right)$为频偏产生的相位失真，则有：

$$Y(l)=\sum_{k=0}^{N-1}X_kS_{k-l}+W(k) \tag{5-13}$$

如图 5-5 所示，由于频偏的存在，正交性被破坏，其他子载波都会对解调的频点产生干



(a) 频偏 ε=5%　　(b) 频偏 ε=25%

图 5-5　频偏引起的星座图解调失真

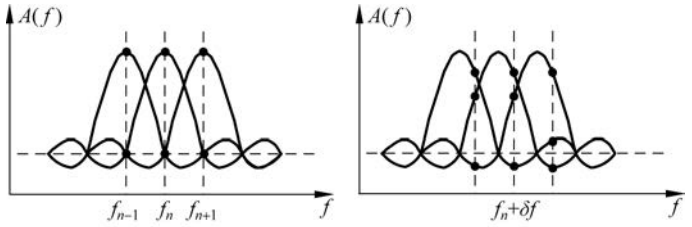
扰；S_{k-l} 为载波间干扰系数或 ICI 系数；当频偏存在时，接收星座图产生整体角度旋转，同时噪声引起信号星座图扩散。

频偏和频偏估计以载波间隔系数 ε 的形式描述和实现，具体可分为载波间隔的整数倍和小数倍的形式：

$$\mathrm{d}f = \varepsilon \Delta f = (D + d)\Delta f \tag{5-14}$$

式中，$D\Delta f$ 为整数倍载波间隔；$d\Delta f$ 为小数倍载波间隔。

如图 5-6(a)所示，当频偏为载波间隔的整数倍时，OFDM 信号在频域平移，等效于接收频域地址错位，并未引起正交性的破坏或产生载波间干扰和码间干扰，但是接收信号等效于错位乱码(误码率 50%)。



(a) 整数倍频偏　　(b) 小数倍频偏

图 5-6　频谱的整数倍和小数倍偏移

如图 5-6(b)所示，当频偏为载波间隔的小数倍时，每个子载波频域符号产生相位旋转和幅值衰减；另外子频点的偏移导致了正交性的破坏，其他子载波对当前频域信号产生载波间干扰 S_{k-l}。

尽管时间偏差会破坏子载波之间的正交性，但是通常情况下可以忽略不计。当抽样错误可以被校正时，就可以用内插滤波器来控制它在正确的时间进行抽样。

2）定时偏差

OFDM 系统定时恢复与单载波系统的定时恢复不同，单载波系统的定时恢复是找到眼图张开最大时刻为最佳抽样时刻。OFDM 符号沿时间轴顺序传输，OFDM 符号由循环前缀和 OFDM 数据体组成，因此 OFDM 同步就是要确定 OFDM 数据体的开始时刻，即确定 FFT 窗的开始时刻。抽样时钟同步主要是指接收机和发射机的抽样时钟频率保持一致，抽样时钟频率偏差将导致 ICI，还将影响同步，因此首先假设抽样时钟同步是理想的。

理想的符号同步就是选择最佳的 FFT 窗，使子载波保持正交，且 ISI 被完全消除或者降至最小。由于使用了循环前缀技术，在保护范围内 OFDM 系统能够容忍一定的符号定时误差而无性能损失。所以 OFDM 系统对定时偏差的鲁棒性相对于频率偏差更强。

如图 5-7 所示，当定时偏差小于保护间隔长度时，在保护间隔保护范围内，定时偏差可以看作多径时延的一部分，减少了 OFDM 对多径时延的容忍能力。如果是定时超前，循环前缀作为保护间隔可以保证 OFDM 的完整性，不会产生 ISI 和 ICI，根据 IFFT 圆周循环定理，子载波相位旋转。如果是定时滞后，由于没有后保护，定时偏差导致后一个 OFDM 干扰当前 OFDM 符号(即 ISI)，ISI 同时破坏了载波间的正交约束，引起 ICI；同时时延偏差导致子载波相位旋转。在移动多媒体相关标准中，通常设置循环后缀作为后保护，以避免 ISI 和 ICI。

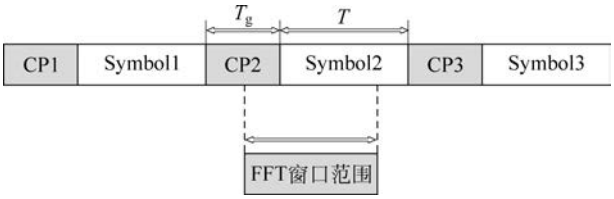


图 5-7　OFDM 定时同步偏差

如图 5-8 所示，在多径条件下，理想的定时并不一定是同步首径，因为移动场景中多径随机分布，并不存在直射径，能量最大径往往是中间的某一路径。定时同步算法往往是利用导频的相关性定时到最相关的一条路径，即能量最强径。相对于同步的主径而言，存在超前和滞后的其他路径，如果超前和滞后路径在保护间隔的保护范围以内，则不会产生 ISI 和 ICI。

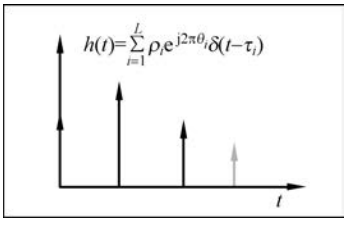


图 5-8　多径条件下的定时同步

考虑频偏对 OFDM 接收信号的影响，设定发射端基带信号的表示形式为

$$s(t)=\sum_{k=0}^{N-1}X_k e^{j2\pi k\Delta f t} \tag{5-15}$$

多径信道模型简单描述为

$$h(t)=\sum_{i=1}^{L}h_i(t)\delta(t-\tau_i)+w(t) \tag{5-16}$$

OFDM 经过多径信道，此时信号应表示为

$$y(t)=\sum_{i=1}^{L}h_i(t)s(t-\tau_i)+w(t) \tag{5-17}$$

3）采样同步偏差

如图 5-9 所示，采样频率的同步是指发射端的 D/A 转换器和接收端的 A/D 转换器的工作频率保持一致。一般地，收发两端的转换器之间的偏差较小，相对于载波频移的影响来说也较小，而一帧的数据如果不太长的话，只要保证了帧同步，可以忽略采样时钟不同步造成的漏采样或多采样，而只需要在一帧数据中补偿由于采样偏差造成的相位噪声。

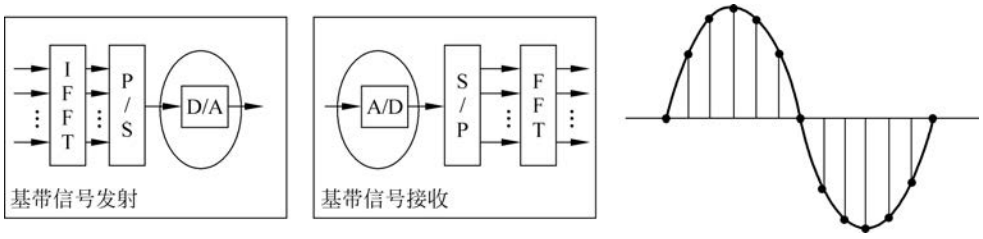


图 5-9　采样频率同步

在 D/A 和 A/D 模块中采样频率和采样时间间隔成反比：

$$\begin{cases}\dfrac{1}{f_{\text{samp}}}=\Delta t\\ T_{\text{s}}=\Delta t N\end{cases} \tag{5-18}$$

如果接收机的 A/D 模块采样频率偏大，会造成采样间隔偏小，导致积分区间缩短，在固定的 FFT 窗口周期内，OFDM 数据不完整，定时逐渐提前；如果 A/D 模块采样频率偏小，会造成采样间隔偏大，导致积分区间延长，OFDM 时延扩展到下一个 OFDM 符号，定时逐渐滞后。两种情况都会造成 ISI，并且破坏正交性，引起 ICI。

发射机基带时域信号的模拟形式为 $s(t)=\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi k\Delta f t}$，没有偏差的采样时刻为 $t=\dfrac{T_{\text{s}}}{N}n=n\Delta t$。当接收机 A/D 模块数字化的过程中存在时间偏差：

$$r(t+\mathrm{d}t)=r\left(\frac{T_{\text{s}}}{N}n'\right) \tag{5-19}$$

采样频率偏差也分为整数部分和小数部分的采样偏差：

$$n'=u+\zeta \tag{5-20}$$

式中，u 为整数倍频率偏差；ζ 为小数倍频率偏差。

采样频率偏差等效为频偏的相位抖动，造成的影响也是相位旋转和噪声离散。理想的采样时钟的时间间隔为 $\Delta t=\dfrac{T_{\text{s}}}{N}$，此时采样点和接收机匹配滤波器的最大点相吻合。当采样时间间隔存在偏差，表述为

$$\Delta t'=\Delta t+\delta \tag{5-21}$$

式中，δ 为设定采样时刻的绝对偏差。

为了方便分析偏差的影响，考虑采样偏差和采样时间间隔的倍数关系，设定归一化的采样偏差时刻为

$$\xi=\frac{\delta}{\Delta t} \tag{5-22}$$

采样偏差导致实际采样点偏离理想采样点，且随着时间推移和样本点的增加，漂移越来越大。

设定离散形式的发射机基带信号为 $s(n)=\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{n}{N}k}$，在采样时钟偏差条件下，接收信号可以表示为

$$y(n)=\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{(n+n\xi)}{N}k} \tag{5-23}$$

随着取样频率系数 k 的增加，取样误差也随之增大为 $(1+\xi)k$。接收信号 $y(n)$ 经过 FFT 变换，转换为频域表达式为

$$\begin{aligned}Y(l)&=\frac{1}{N}\sum_{n=0}^{N-1}\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{(n+n\xi)}{N}k}\mathrm{e}^{-\mathrm{j}2\pi\frac{n}{N}l}+W(k)\\&=\sum_{k=0}^{N-1}X_k\frac{\sin(\pi(k+k\xi-l))\mathrm{e}^{\mathrm{j}\pi\varepsilon}}{N\sin\left(\dfrac{\pi}{N}(k+k\xi-l)\right)}\mathrm{e}^{\mathrm{j}\pi\left(1-\frac{1}{N}\right)(k+k\xi-l)}+W(k)\end{aligned} \tag{5-24}$$

式中，l 为接收端频域标号。

令 $S'_{k-l}=\dfrac{\sin(\pi(k+k\xi-l))\mathrm{e}^{\mathrm{j}\pi\varepsilon}}{N\sin\left(\dfrac{\pi}{N}(k+k\xi-l)\right)}\mathrm{e}^{\mathrm{j}\pi\left(1-\frac{1}{N}\right)(k+k\xi-l)}$ 为采样偏差产生的相位失真，则有：

$$Y(l)=\sum_{k=0}^{N-1}X_k S'_{k-l}+W(k) \tag{5-25}$$

因此采样偏差也会导致 ICI，且 ICI 的大小与子载波的频率系数 l 相关。

本章不专门分析采样同步偏差的补偿方法，而将其看成频率偏差的一部分进行估计和补偿，由此频偏估计应对的频率偏差包含 A/D 偏差、多普勒频移等因素。

5.2 定时同步

同步对于任何数字通信系统来说都是重要的任务，没有精确的同步就不能对传送的数据进行可靠的恢复。可以说，同步是任何通信接收机实现的基础。OFDM 既可以用于广播类型的通信系统，又可以用于突发数据传输的通信系统，在同步的问题上二者可以采取的途径不尽相同。广播类型的系统传输的是连续的数据，因此最初需要经过较长的一段时间获得信息（同步捕获），之后转换成跟踪模式。突发通信系统通常采用分组的方式，需要在分组开始发送之后的很短时间内完成同步。由于 OFDM 信号结构特殊，使得很多为单载波系统设计的同步算法不能被采用，因此，必须从 OFDM 信号本身的角度出发来设计同步算法。

在数字电视/广播和电路交换蜂窝系统为代表的 OFDM 系统中，OFDM 往往执行日帧、分帧、秒帧的划分，最基本的帧单元和 OFDM 周期是连续规则的数据格式，且满足循环结构，时分和频分复用是静态的划分方式，因此结构和帧长都是恒定的格式，所以精确的定时同步很容易完成定界，且允许用较长时间重新捕获调整同步精度，减少一定时间内的同步次数。同时，广播类节目对于同步的时延误差要求相对较低，可以允许较长的同步捕获时间，因此对于同步时间消耗的约束不大，通常允许用较长的时间进行多次同步以获得估计精度的提升。

5.2.1 滑动窗口法定时同步

以 DVB 系列为例，如图 5-10 所示在 DVB 的时频二维结构中，设定一个或多个符号为空符号，后续帧的 OFDM 符号为正常的帧结构。接收端在接收连续信号时通过监测空符号标识的上升沿与下降沿，监测能量的连续过程来进行定界。

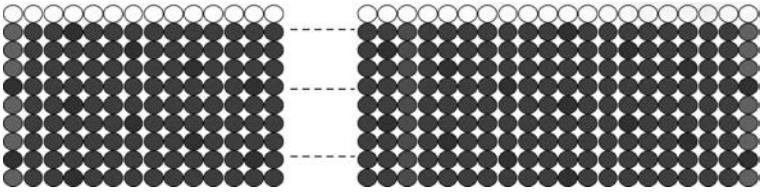


图 5-10 DVB 时频二维结构

如图 5-11 所示，采用滑动窗口法设定一个能量监测的窗口，包含 L 个采样点，进行自相关处理。

$$m_n = \sum_{k=0}^{L-1} r_{n-k} r_{n-k}^* = \sum_{k=0}^{L-1} | r_{n-k} |^2 \tag{5-26}$$

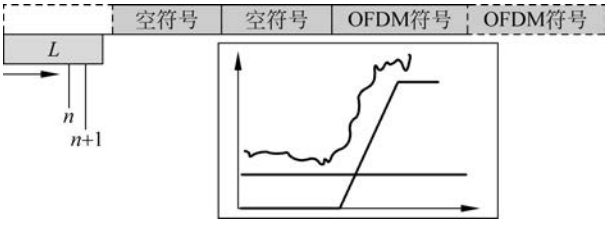


图 5-11　滑动窗口法

滑动的过程可以表示为窗前去掉第一个采样点，窗尾增加一个采样点，即所有采样点 $n+1$ 移位。

$$m_{n+1} = \sum_{k=0}^{L-1} r_{n+1-k} r_{n+1-k}^* = \sum_{k=0}^{L-1} | r_{n+1-k} |^2 \tag{5-27}$$

程序 5-1“energy_detection”，滑动窗口能量检测法定时

```
function SyncPos=energy_detection(chips_channel,sync_signal)
Recv1thOFDMzeros= chips_channel;
L=length(sync_signal);%窗的长度 L
len=length(Recv1thOFDMzeros);
M=zeros(1,len);
for i=1:len-L
    M(i)=Recv1thOFDMzeros(i:i+L-1)*(Recv1thOFDMzeros(i:i+L-1))';
end
M1=(M - min(M))/(max(M) - min(M));
%M2=M1(20000,100000);
% plot(M1);
% xlim([20000,100000]);
% title('Sliding window energy detection');
% hold on;
% Y1=ones(1,100000).*((1/2)*(max(M1)-min(M1)));%0.5
% plot(Y1,'r');
M2=M1(20001:100000);
[~,SyncPos]=min(abs(M2-0.5));
clear M M1 Y1 len;
end
```

程序 5-1 为滑动窗口实现过程。图 5-12 为滑动窗口法定时同步时域能量监测遍历窗口，能量的变化可以作为有效信号的定界。利用滑动窗口的信号自相关方法监测能量，当没有接收到数据或滑动到空符号时，接收信号中只有噪声，能量明显小于有数据传输的时刻；当滑动到有效数据符号时，接收信号中包含有效信号成分，因此当接收能量值发生变化时可

以检测到有效 OFDM 符号。

滑动窗口法只能用于监测能量，因此监测的结果与信息内容无关，且监测过程搜索速度快，实现复杂度小。但同时存在一定的误差瓶颈：滑动窗口法仅用以区分有效信号和噪声，因此无法判定有效信号具体是什么标准格式和什么业务类型，无法执行帧同步功能，解决的方法是在授权频段内通过中频滤波保证监测信号的唯一性；当传输信号处于覆盖边缘，传播损耗较大或者噪声较强时，信号和噪声功率无法通过阈值区分，导致无法精确定界，并且定界的精度受噪声的影响较大，无法执行定时同步；当固定频段受到同频干扰时，无法通过噪声和信号的阈值区分边界和确定标准信号。因此滑动窗口法最大的缺陷是同步偏差过大。

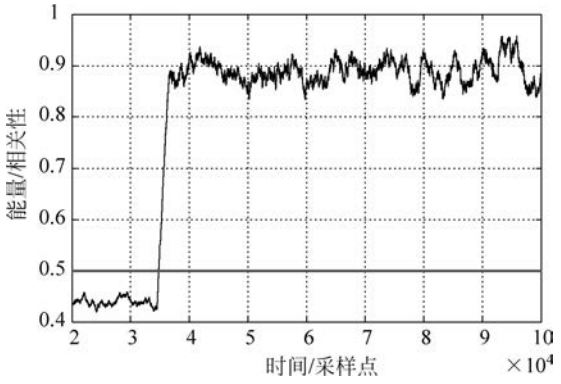


图 5-12　滑动窗口法定时同步时域能量监测遍历窗口

5.2.2　SC 定时同步

随着通信技术的发展和变革，以无线通信为代表的 OFDM 系统，如 WiFi(802.11a/b/n/ac/ad)，4G/5G 通信标准，面临着更复杂、更高精度需求的同步场景。相对于广播电视直播场景，OFDM 通信系统传输的分组数据报具有随机接入和高速传输的特点，因此数据传输格式并不是连续的、时频二维静态分片的，而是需要接收端实时监测，在呼叫连接和数据传输的过程，需要实时确定分组窗口定界，获取分组的头部和尾部、分组的帧数。同步需要在极短的时间内捕获，且在一次性捕获的过程中精确定时，同时因分组长度不统一，也需要对帧尾进行精确的同步定界。传统互联网在有线通道传输时通过监测曼彻斯特编码的信号能量变化就可以获取精确的定界，但此种方法在无线衰落信道中并不可行，因为需要在分组首部、尾部设置同步信号帧，因为分组精度造成的失真或者残帧，会被丢弃。

在分组检测过程中，可通过设计相关性很强的同步头进行快速捕获，比如采用扩频序列作为 OFDM 同步头进行帧/定时同步时，可以通过优化扩频序列的正交特性提升相关性和抗噪能力，可以通过增加同步头的长度增加相关性和抗噪能力，还可以通过降低同步头的失真和噪声污染来提升相关性，以此获得同步精度的增强。

作为经典的同步结构，移动多媒体标准在帧头设计了两个相同的同步符号，用于快速同步和辅助信道估计。同步头由 $N/2$ 点的 IFFT 变换得到，长度为 OFDM 数据体的一半，载波间隔为 OFDM 数据体的 2 倍，频域信号为 BPSK 调制的扩频码，具备很强的相关性。

基于双同步结构的同步算法中比较经典的是 SC(Schmidl and Cox)定时同步法。

由图 5-13 可以看出两个同步符号是完全相同的 OFDM 同步序列，OFDM 同步符号由相关性很强的扩频码 IFFT 调制完成，且 OFDM 同步头长度为 OFDM 数据体周期的一半，即 OFDM 数据体的载波数为 N 时，OFDM 同步头的载波数为 $N/2$。另外双同步头结构用作频偏估计，可以同时解决时间和频域同步问题。频率偏差造成信号的相位偏差，而 SC 算法利用双同步头的互相关，近似于单同步头的自相关，而相关性对频率偏差不敏感，因此测量的精度与频偏无关。SC 定时同步的算法度量函数为

$$M(d)=\frac{|P(d)|^2}{R^2(d)} \tag{5-28}$$

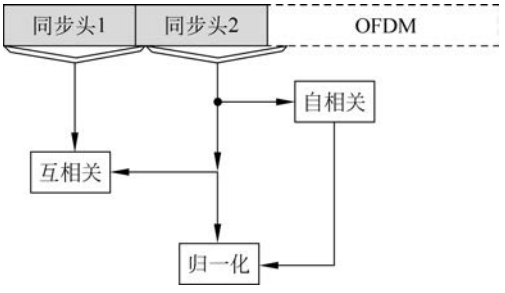


图 5-13　SC 定时估计算法

其中，$|P(d)|^2$ 表示将双同步头中所有的样本前同步与后同步对应的数据共轭相乘相加，即

$$P(d)=\sum_{m=0}^{N/2-1} r_{d+m}^{*} r_{d+m+\frac{N}{2}} \tag{5-29}$$

式中，d 是估计到的滑动窗的起始位置；$P(d)$是前同步头的数据能量。基于信号能量的判决也基于滑动窗机制，其表达式为

$$R(d)=\sum_{m=0}^{N/2-1} | r_{d+m+\frac{N}{2}} |^2 \tag{5-30}$$

在 SC 算法中，多径衰落不会影响度量函数。所以在 $M(d)$取得最大值时，d 就是符号起始位置，记作 $\hat{d}$，基于 SC 算法可以估计 FFT 的起始位置为

$$\hat{d}=\arg\max_{d}[M(d)] \tag{5-31}$$

程序 5-2“S_C”，SC 定时估计

```
function SyncPos=S_C(sync_signal,chips_channel)
samples = length(sync_signal);%2048
Ns=samples * 2;
P1=zeros(1,Ns * 2);
R1=zeros(1,Ns * 2);
recv= chips_channel;
for d = 1+Ns/2:1:2 * Ns
    for m=0:1:Ns/2-1
        P1(d-Ns/2) = P1(d-Ns/2) + conj(recv(d+m)) * recv(d+Ns/2+m);
        R1(d-Ns/2) = R1(d-Ns/2) + power(abs(recv(d+Ns/2+m)),2);
```

```
        end
    end
    M=power(abs(P1),2)./power(abs(R1),2);
    M1 = (M - min(M))/(max(M) - min(M));%归一化
    % d=1:1:Ns;
    % figure(1)
    % plot(d,M1(d));
    % grid on;
    % title('S&C');
    % xlabel('Time (sample)');
    % ylabel('Time');
    SyncPos = find(M==max(M(10:samples)));
    clear M M1;
    end
```

程序 5-2 为 SC 定时同步的实现过程。图 5-14 为 SC 定时估计遍历窗口，选择峰值采样点作为定时起始位置。基于双同步头的自相关性在受到噪声干扰的情况下会使 SC 算法产生明显的平台效应，而且信噪比越低平台效应就越明显，以至于 SC 算法在同步高精度要求下难以胜任；SC 算法的优点是 SC 这种对称结构和扩频比较容易生成，而且有着较强的稳定性，同时适用于时间和频域同步。

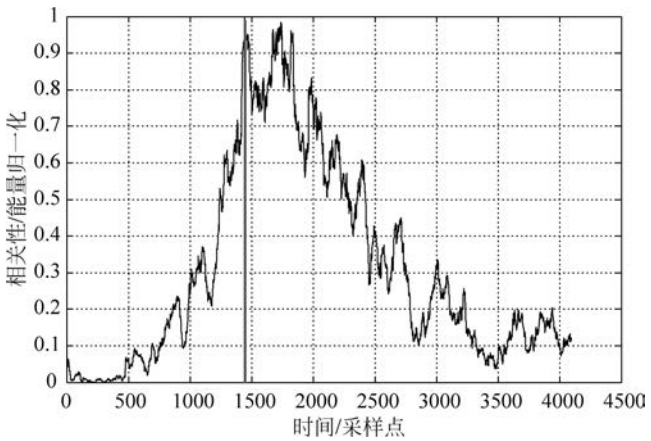


图 5-14　SC 定时估计遍历窗口

5.2.3　双滑动窗口法定时同步

如图 5-15 所示，双滑动窗口法定时同步算法类似于 SC 算法，信号帧由双同步头和 OFDM 数据体组成，类似于 SC 算法发送的信号帧由双同步头和 OFDM 数据体组成，为了进行时域同步，接收机存储无损的双同步头来进行定时估计。接收机利用本地的双同步头同接收信号进行互相关，当双同步窗口滑动到接收信号帧同步头的位置可以获得相关性最强的遍历值。同步过程中，设定 $\boldsymbol{r}_1$ 为接收数据，$\boldsymbol{r}_2$ 为接收器存储的同步头。利用本地无损的双同步头对接收信号进行相关性遍历，互相关公式为

$$\gamma(d)=\sum_{k=d}^{d+N} r_{1,k} r_{2,k}^{*} \tag{5-32}$$

执行接收同步头与本地无损同步头的能量统计：

$$\Phi(d)=\sum_{k=d}^{d+N} |r_{1,k}|^2+|r_{2,k}|^2 \tag{5-33}$$

基于式(5-33)进行归一化：

$$\hat{d}=\arg\max_{d}\left[\frac{|\gamma(d)|^2}{\Phi(d)^2}\right] \tag{5-34}$$

当 $\hat{d}$ 为先增后减的凸分布时，判定最大遍历值为 FFT 窗口的起始位置。

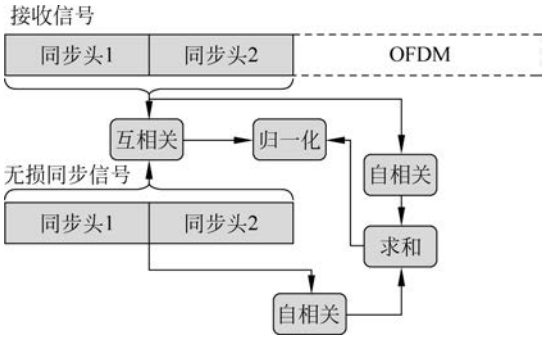


图 5-15 双滑动窗口法定时估计算法

程序 5-3“Double_sliding_window”，双滑动窗口定时估计

```
function SyncPos=Double_sliding_window(sync_signal,chips_channel)
q=4;%采样率
samples = length(sync_signal);
theta = zeros(1,samples * q);
r2 = sync_signal;
for i=1:samples * q
    r1 = chips_channel(i:q:i+q * (samples-1));
    gamma = r1 * r2';
    epsilon = r1 * r1'+r2 * r2';
    theta(i) = gamma * gamma'/epsilon/epsilon;
end
SyncPos = find(theta==max(theta(10:end)));
clear r2 r1 gamma epsilon theta;
end
```

程序 5-3 为双滑动窗口法定时同步过程。图 5-16 为双滑动窗口法定时估计算法遍历窗口，遍历结果表明双滑动窗口法可有效提升相关特性，容易通过峰值定位到精确的起始点。

双滑动窗口法相对于 SC 算法获得了精度的提升，且不存在平层效应。主要得益于以下几方面的改进：

(1) 实际参与相关性分析的同步头窗口长度变为原来的 2 倍，扩频序列构成 OFDM 同步头，增加同步头的长度可以提升相关性。

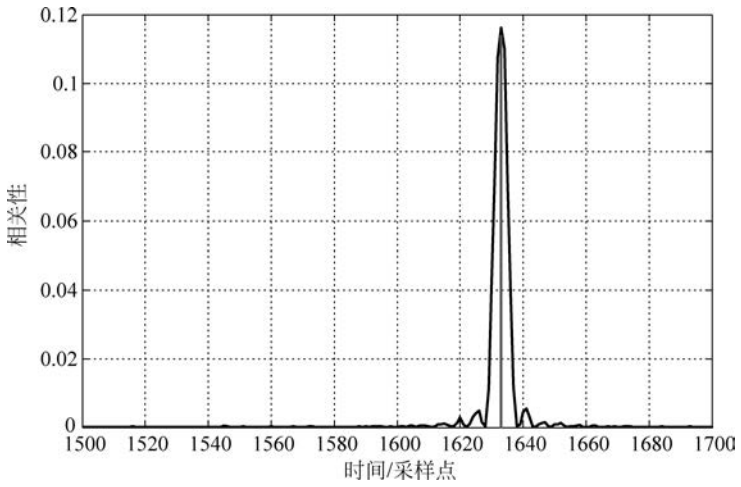


图 5-16　双滑动窗口法定时估计算法遍历窗口

(2) 通过香农极限分析,降低同步头的噪声干扰可以提升相关性,接收机通过引入无损的同步头与接收信号的同步头进行同步,可以有效提升同步精度。

5.3　频偏估计

式(5-12)分析了离散信号的频率偏差,可以归一化成载波间隔的系数。

$$y(n)=\left[\sum_{k=0}^{N-1}X_k\mathrm{e}^{\mathrm{j}2\pi\frac{n}{N}k}H_k\mathrm{e}^{\mathrm{j}2\pi\frac{n}{N}\varepsilon}\right]+w(n) \tag{5-35}$$

式中,$\varepsilon=\dfrac{\mathrm{d}f}{\Delta f}=\dfrac{\Delta\theta}{2\pi}$为相对频偏系数;$H_k$ 为第 k 个子载波的信道衰落系数。

多径信道和多普勒频移都会产生相位偏移,但是二者有本质的区别。

如图 5-17 所示,在时频二维结构中多径产生的失真分为两部分,一部分是恒定失真,包含相位失真和幅值衰落,恒定失真部分在时域维和频域维一定范围内具备平坦特征;另一部分为多径时延,所有 OFDM 符号入射角的相偏在时域维满足平坦特征,但是时延等效的相偏在频域维存在频率选择性衰落特征。

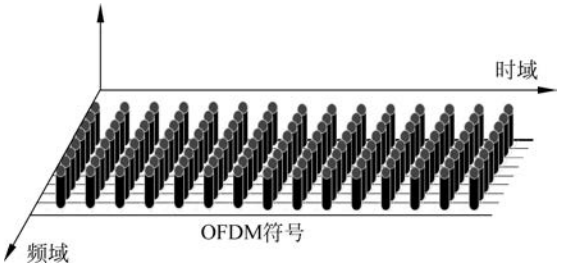


图 5-17　OFDM 时频二维结构

多普勒频移引起的相偏随着时间不断变化,但是相对值满足稳定特征。从时域角度,相邻 OFDM 的相位相对偏差满足平坦特征,固定频点相邻 OFDM 符号,子载波之间相位随着

时间不断变化，但是相对偏差满足平坦特征。基于相位偏差的频偏估计正是利用相偏相对值的稳定特征来进行测算的。

5.3.1　小数倍频偏估计

如图 5-18 所示，Moose 频率同步采用双同步头结构，理想情况下假设接收信号已经完成精确定时。设定两个接收 OFDM 同步符号的时域导频序列为 $\boldsymbol{y}_1=\{y_{1,n}\}$，$(n=0,1,2,\cdots,N)$和 $\boldsymbol{y}_2=\{y_{2,n}\}$，$(n=0,1,2,\cdots,N)$；将双同步信号进行 FFT 变换，FFT 变换后的前同步导频序列中的第 k 个子载波的信号是 $Y_{1k}=R_{1k}+W_{1k}$，后同步导频序列中第 k 个子载波的信号是 $Y_{2k}=R_{1k}\mathrm{e}^{\mathrm{j}2\pi\varepsilon}+W_{2k}$，其中，$R_{1k}=R_{2k}$ 为同步头第 k 个子载波频域导频，W_{1k} 和 W_{2k} 为加性噪声，$\mathrm{e}^{\mathrm{j}2\pi\varepsilon}$ 为固定子载波在相邻 OFDM 周期内的相对相偏。

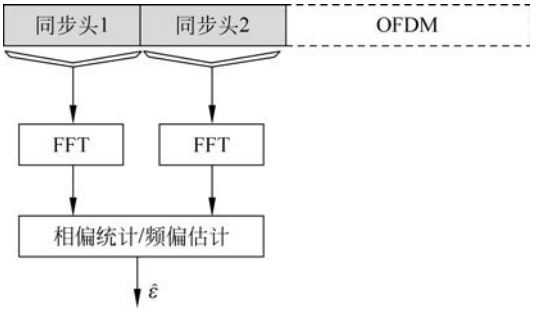


图 5-18　Moose 频率同步算法的训练符号结构

连续两个 OFDM 符号对应子载波的相对频率偏差引起的相位偏差由互相关函数获取：

$$\begin{aligned}Y_{2k}Y_{1k}^{*}&=(R_{1k}\mathrm{e}^{\mathrm{j}2\pi\varepsilon}+W_{2k})(R_{1k}+W_{1k})^{*}\\&=|R_{1k}|^{2}\mathrm{e}^{\mathrm{j}2\pi\varepsilon}+R_{1k}\mathrm{e}^{\mathrm{j}2\pi\varepsilon}W_{1k}^{*}+R_{1k}W_{1k}+W_{2k}W_{1k}^{*}\end{aligned}\tag{5-36}$$

式中，$R_{1k}\mathrm{e}^{\mathrm{j}2\pi\varepsilon}W_{1k}^{*}+R_{1k}W_{1k}+W_{2k}W_{1k}^{*}$ 为估计误差函数。

根据欧拉变换，最后可以根据最大似然法估计出载波频偏，即

$$\hat{\varepsilon}=\frac{1}{2\pi}\tan^{-1}\left\{\frac{\sum_{k=-K}^{K}\mathrm{Im}[Y_{2k}Y_{1k}^{*}]}{\sum_{k=-K}^{K}\mathrm{Re}[Y_{2k}Y_{1k}^{*}]}\right\}\tag{5-37}$$

由式(5-37)可知，$-1/2$ 到 $+1/2$ 的子载波间隔为 Moose 频率同步算法的最大频偏估计范围，可以通过所有子载波的加权平均统计得到。然而频域导频需要精确的时域同步，和复杂的 FFT 变换过程，限制了 Moose 频率同步算法的广泛应用。

既然频偏引起的相对相偏在时域维和频域维都存在相同的统计特性，那么，时域维的相偏检测将更为简单(规避了 FFT 变换的步骤)，所以时域频偏估计相对而言更为普及。如图 5-19 所示，根据多普勒频移的相偏在时域维的统计特征，通过时域的互相关性统计相位偏差为

$$\begin{aligned} P(d) &= \sum_{n=0}^{N-1} y_{1,n}^{*} y_{2,n} \\ &= \sum_{n=0}^{N-1} y_{1,n}^{*} y_{1,n} \mathrm{e}^{\mathrm{j}\varphi} \\ &= \left(\sum_{n=0}^{N-1} y_{1,n}^{*} y_{1,n} \right) \mathrm{e}^{\mathrm{j}\varphi} \end{aligned} \tag{5-38}$$

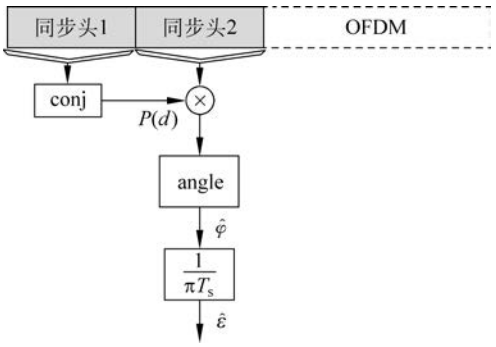


图 5-19 小数倍频偏估计

连续两个 OFDM 符号的时域相偏由频偏引起，即 $\mathrm{e}^{\mathrm{j}\varphi} = \mathrm{e}^{\mathrm{j}2\pi\varepsilon}$，由此可知相位偏移为

$$\hat{\varphi} = \mathrm{angle}\left[P(d)\right] \tag{5-39}$$

式中，同步头的长度为 OFDM 数据体的 1/2，即 $T_s/2$。

基于相位偏移得到频偏系数：

$$\varphi = 2\pi \frac{T_s}{2} \varepsilon = \pi T_s \varepsilon \tag{5-40}$$

$$\varepsilon = \frac{\varphi}{\pi T_s} \tag{5-41}$$

因为观测到的相位偏差在$[0, 2\pi]$，即无法统计整数倍 2π 的相位偏移。利用相位偏差估算频偏的算法只能获取小数倍频偏，无法估计整数倍频偏。

5.3.2 整数倍频偏估计

完成对小数倍频偏估计之后，可以利用同步信号中频域导频的相关性来完成整数倍频偏的估计。整数倍频偏需要通过统计频域频谱搬移获得：

设定两个相同的同步头作为频偏估计的导频，每个同步头载波数为 $2N$ 的 OFDM 信号，即载波间隔为 $\Delta f/2$，载波数为 $2N$，在有效频域导频数 N 的基础上进行频域间隔插零获得。

图 5-20 为整数倍频偏估计的原理图。在时域定时双同步的基础之上，首先补偿接收到的同步信号的小数倍频偏 V_k，并基于接收频域同步信号进行相关性扫描，互相关公式为

$$\widetilde{\gamma}(d) = \sum_{k \leqslant N} X_{1,k+2g}^{*} V_k^{*} X_{2,k+2g} \tag{5-42}$$

统计其中一个接收同步头的能量：

$$\widetilde{\Phi}(d) = \sum_{k \leqslant N} \left| X_{2,k} \right|^2 \tag{5-43}$$

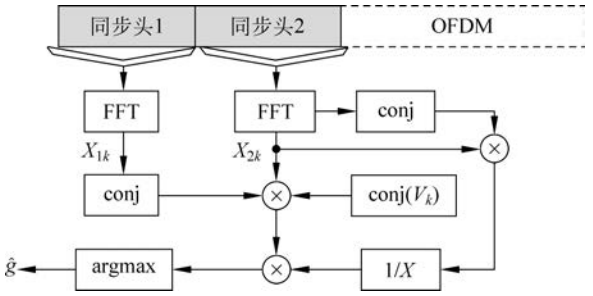


图 5-20 整数倍频偏估计的原理图

基于式(5-43)进行归一化：

$$\hat{g}=\arg\max_{g}\left[\frac{|\widetilde{\gamma}(d)|^2}{2\Phi(d)^2}\right] \tag{5-44}$$

当存在整数倍频偏时，通过相关性的扫描可以统计得到存在相关性的载波数，相对频偏的载波相关性近似为 0，通过统计 $\hat{g}$ 与 N 的关联度，计算出整数倍频偏的数值。

5.4 OFDM 同步系统仿真和性能分析

5.4.1 OFDM 同步系统实验仿真模型


仿真视频

程序 5-4“OFDM_synchronization”，OFDM 同步系统仿真分析模型

```
function OFDM_synchronization
format long;
warning off
clear all;
clc;
OFDM_SimNum=2;
BER=zeros(OFDM_SimNum,41);
for SimNum=1:OFDM_SimNum
    %参数设置%
    Ensemble_SimNum = 1;
    Sys_OFDM_MODE = 8;%8MHz 模式
    %modulationMode ='16QAM';%'QPSK';
    modulationMode ='QPSK';
    bitwidth = 18;
    xid_8M_file = 'xid_8M.dat';%TS 流文件
    xid_2M_file = 'xid_2M.dat';
    prbs_mode = 2;
    txid_region_code = 100;%0～127
    txid_txer_code = 228;%128～255
    fs = 1e7;%采样频率
    q=4; %采样倍数
    fover = q*fs;
    %固定参数设置%
```

```
Fcs_id = 39062.5;%发射机识别标识 TXID 载波间隔, 39.0625kHz
Tu_id = 0.0000256; %TXID OFDM 周期, 25.6μs
Tcp_id = 0.0000104;%TXID 循环前缀, 10.4μs
Fcs_sync = 4882.8125; %同步头载波间隔, 4.8828125kHz
Tu_sync = 0.0002048;%同步头 OFDM 周期, 204.8μs
Fcs = 2441.40625;%数据 OFDM 载波间隔, 2.44140625kHz
Tu = 0.0004096;%OFDM 周期, 409.6μs
Tcp = 0.0000512;%OFDM 循环前缀, 51.2μs
Tgi = 0.0000024;%保护间隔 GI, 2.4μs
%参数生成%
fft_pnts = 512 * Sys_OFDM_MODE;%IFFT 点数
fft_pnts_txid = 32 * Sys_OFDM_MODE;
fft_pnts_sync = 256 * Sys_OFDM_MODE;
T = 1/fs;
Tofdm = Tu+Tcp+Tgi;%OFDM 符号总长度
Tid = Tu_id+Tcp_id+Tgi;%TXID OFDM 符号总长度
Tsync = 2 * Tu_sync+Tgi;%同步头总长度
Tensemble = Tofdm * 53+Tsync+Tid;%时隙长度
NumEsemble = round(Tensemble/T);
prbs_init_regs =[...
    0 0 0 0 0 0 0 0 0 0 0 0 1;
    0 0 0 0 1 0 0 1 0 0 1 1;
    0 0 0 0 0 1 0 0 1 1 0 0;
    0 0 1 0 1 0 1 1 0 0 1 1;
    0 1 1 1 0 1 0 0 0 1 0 0;
    0 0 0 0 0 1 0 0 1 1 0 0;
    0 0 0 1 0 1 1 0 1 1 0 1;
    0 0 1 0 1 0 1 1 0 0 1 1];
if Sys_OFDM_MODE == 8
    ValidCarrierNum = 3076;%有效载波数
    DataCarrierNum = 2610;%数据载波数
    ContinualPilotNum = 82;%连续导频数
    ScatteredPilotNum = 384;%离散导频数
    ZeroPad_carriers = 1020;
    ZeroPad_Ensemble = 90;
    xid_file = xid_8M_file;
    txid_num = 191;
    sync_num = 1536;
    DirectorContinualPilotCarriers =[...
        22   78   92   168  174  244  274  278...
        344  382  424  426  496  500  564  608...
        650  688  712  740  772  846  848  932...
        942  950  980  1012 1066 1126 1158 1214...
        1860 1916 1948 2008 2062 2094 2124 2132...
        2142 2226 2228 2302 2334 2362 2386 2424...
        2466 2510 2574 2578 2648 2650 2692 2730...
        2796 2800 2830 2900 2906 2982 2996 3052];
```

```
    ZeroContinualPilotCarriers = [0 1244 1276 1280 1326 1378 1408 ...
        1508 1537 1538 1566 1666 1736 1748 1794 1798 1830 3075];
elseif Sys_OFDM_MODE == 2
    ValidCarrierNum = 628;
    DataCarrierNum = 522;
    ContinualPilotNum = 18;
    ScatteredPilotNum = 78;
    ZeroPad_carriers = 396;
    ZeroPad_Ensemble = 18;
    xid_file = xid_2M_file;
    txid_num = 37;
    sync_num = 314;
    DirectorContinualPilotCarriers = [20 32 72 88 128 146 154 156 ...
        470 472 480 498 538 554 594 606];
    ZeroContinualPilotCarriers = [0 216 220 250 296 313 314 330 ...
        388 406 410 627];
end
N=53;
SignalContinualPilotCarriers=zeros(1,64);
ext_iter = zeros(1,4);
cnt_signal=zeros(53,3076);
tic
fcsrrc = Fcs * (ValidCarrierNum+2)/2;%4e6;
figureindex = 0;
%生成连续导频
load a a
for p=1:16
    for i=1:16:64
        SignalContinualPilotCarriers(i+p-1)=BPSK_Mapping(a(p));
    end
end
%生成 TXID 信号
if 1
    disp('Generating signal of tx id...');
    txid_code = txid_region_code;
    txid = load(xid_file);
    txid = txid(txid_code+1,:);
    txid = [0 1-2 * txid(1:floor(txid_num/2)) ...
    zeros(1,fft_pnts_txid-txid_num-1) 1-2 * txid(floor(txid_num/2)+1:txid_num)];
    %txid = ifft(txid,fft_pnts_txid) * sqrt(fft_pnts_txid);
    txid = ifft([txid zeros(1, fft_pnts-fft_pnts_txid)], fft_pnts) * fft_pnts/sqrt(fft_pnts_txid);
    txid = txid(1:fft_pnts/fft_pnts_txid:fft_pnts);
    Num = Tu_id/T;
    times = Num/fft_pnts_txid;
    txid_resample = zeros(1, Num);
    for cnts = 0:times-1
        txid_resample(cnts+1:times:(fft_pnts_txid-1) * times+cnts+1) = txid;
```

```
        end
        txid_even = txid_resample;
        txid_code = txid_txer_code;
        txid = load(xid_file);
        txid = txid(txid_code+1,:);
        txid =[0 1-2 * txid(1:floor(txid_num/2)) ...
        zeros(1,fft_pnts_txid-txid_num-1) 1-2 * txid(floor(txid_num/2)+1:txid_num)];
        txid = ifft([txid zeros(1, fft_pnts-fft_pnts_txid)], fft_pnts) * fft_pnts/sqrt(fft_pnts_txid);
        txid = txid(1:fft_pnts/fft_pnts_txid:fft_pnts);
        for cnts = 0:times-1
            txid_resample(cnts+1:times:(fft_pnts_txid-1) * times+cnts+1) = txid;
        end
        txid_odd = txid_resample;
        clear txid_resample;
        save txid txid_even txid_odd;
    else
        load txid;
    end
    %生成同步头%
    if 1
        disp('Generating signal of sync...');
        prbs_sync = [0 1 1 1 0 1 0 1 1 0 1];
        sync_signal = zeros(1, sync_num);
        for i=1:sync_num
            sync_signal(i) = prbs_sync(1);
            temp = mod(prbs_sync(1)+prbs_sync(3), 2);
            prbs_sync = [prbs_sync(2:11) temp];
        end
        sync_signal =[0 1-2 * sync_signal(1:floor(sync_num/2)) ...
        zeros(1,fft_pnts_sync-sync_num-1) 1-2 * sync_signal(floor(sync_num/2)+1:sync_num)];
        sync_signal=ifft([sync_signal zeros(1, fft_pnts-fft_pnts_sync)], ...
        fft_pnts) * fft_pnts/sqrt(fft_pnts_sync);
        sync_signal = sync_signal(1:fft_pnts/fft_pnts_sync:fft_pnts);
        Num = Tu_sync/T;
        times = Num/fft_pnts_sync;
        sync_signal_resample = zeros(1, Num);
        for cnts = 0:times-1
            sync_signal_resample(cnts+1:times:(fft_pnts_sync-1) * times+cnts+1) = sync_
signal;
        end
        sync_signal = sync_signal_resample;
        clear sync_signal_resample;
        save sync_signal sync_signal;
    else
        load sync_signal;
    end
    %生成扰码序列%
```

```
prbs_init_reg = prbs_init_regs(prbs_mode, :);
Si = zeros(1,ValidCarrierNum * 53);
Sq = zeros(1,ValidCarrierNum * 53);
for PcCounter = 1:ValidCarrierNum * 53
    Si(PcCounter) = prbs_init_reg(1);
    Sq(PcCounter) = prbs_init_reg(4);
    RegNew = mod(sum(prbs_init_reg([1 2 5 7])),2);
    prbs_init_reg = [prbs_init_reg(2:12) RegNew];
end
Pc = complex(1-2 * Si,1-2 * Sq)/sqrt(2);
save prbs_pc Pc;
clear Pc;
%成帧%
NumGI=Tgi/T;
Wt=0.5+0.5 * cos(pi+pi * (0:T:(Tgi-T))/Tgi);
Pos=1;
ValidDataPos = 0;
SignalLastFrameForGI = zeros(1, NumGI);
SignalLastFrameForGI_noclip = zeros(1, NumGI);
ValidCarrier=[];
ValidCarrier1=[];
if 1
    disp('Generating ensemble signal...');
    for TimeSlot_Var = 0:Ensemble_SimNum-1
        disp(['The ' num2str(TimeSlot_Var+1) 'th time slot generating...']);
        assert(Pos == NumEsemble * TimeSlot_Var+1);
        %TXID%
        if mode(TimeSlot_Var,2) == 0
            txid = txid_even;
        else
            txid = txid_odd;
        end
        Num = Tu_id/T;
        NumCP = Tcp_id/T;
        SignalEnsemble(Pos:Pos+NumGI-1)=txid(Num-NumCP-NumGI+1:Num- ...
        NumCP. * Wt + SignalLastFrameForGI . * (1-Wt);
        SignalEnsemble_nocode(Pos:Pos+NumGI-1)=txid(Num-NumCP-...
        VNumGI+1:Num-NumCP) . * Wt + SignalLastFrameForGI_noclip . * (1-Wt);
        Pos = Pos+NumGI;
        SignalEnsemble(Pos:Pos+NumCP-1) = txid(Num-NumCP+1:Num);
        SignalEnsemble_nocode(Pos:Pos+NumCP-1) = txid(Num-NumCP+1:Num);
        Pos = Pos+NumCP;
        SignalEnsemble(Pos:Pos+Num-1) = txid;
        SignalEnsemble_nocode(Pos:Pos+Num-1) = txid;
        Pos = Pos+Num;
        SignalLastFrameForGI = txid(1:NumGI);
        SignalLastFrameForGI_nocode = txid(1:NumGI);
```

```
%同步头%
Num = Tu_sync/T;
SignalEnsemble(Pos:Pos+NumGI-1) = sync_signal(Num-NumGI+1:Num) .* Wt...
+ SignalLastFrameForGI .* (1-Wt);
SignalEnsemble_nocode(Pos:Pos+NumGI-1)=sync_signal(Num- ...
NumGI+1:Num). * Wt + SignalLastFrameForGI_nocode .* (1-Wt);
Pos = Pos+NumGI;
SignalEnsemble(Pos:Pos+Num-1) = sync_signal;
SignalEnsemble_nocode(Pos:Pos+Num-1) = sync_signal;
Pos = Pos+Num;
SignalEnsemble(Pos:Pos+Num-1) = sync_signal;
SignalEnsemble_nocode(Pos:Pos+Num-1) = sync_signal;
Pos = Pos+Num;
SignalLastFrameForGI = sync_signal(1:NumGI);
SignalLastFrameForGI_nocode = sync_signal(1:NumGI);
%OFDM 信号%
for framecnt = 0:52
    disp(['The ' num2str(framecnt+1) 'th ofdm signal generating...']);
    if mod(framecnt,2) == 0
        ScatteredPilotInit = 1;
    else
        ScatteredPilotInit = 5;
    end
    %OFDM 频域信号%
    ScatteredPilotCarriers = zeros(1,ScatteredPilotNum);
    ScatteredPilotCarriers(1:ScatteredPilotNum/2) =...
    8 * (0:ScatteredPilotNum/2-1)+ScatteredPilotInit;
    ScatteredPilotCarriers(ScatteredPilotNum/2+1:ScatteredPilotNum) =...
    8 * (ScatteredPilotNum/2:ScatteredPilotNum-1)+ScatteredPilotInit+2;
    ContinualPilotNum=0;
    ValidCarrierSig = zeros(1, ValidCarrierNum);
    for ValidCarrierCnt = 0:ValidCarrierNum-1
        if find(ZeroContinualPilotCarriers==ValidCarrierCnt)
            ValidCarrierSig(ValidCarrierCnt+1) = BPSK_Mapping(0);
        elseif find(DirectorContinualPilotCarriers==ValidCarrierCnt)
            ContinualPilotNum=ContinualPilotNum+1;
            ValidCarrierSig(ValidCarrierCnt+1)=...
            SignalContinualPilotCarriers(ContinualPilotNum);
        elseif find(ScatteredPilotCarriers==ValidCarrierCnt)
            ValidCarrierSig(ValidCarrierCnt+1) = 1;
        else
            ValidDataPos = ValidDataPos+1;
            switch modulationMode
                case 'BPSK'
                    ValidData(ValidDataPos) = randint();
                    ValidCarrierSig(ValidCarrierCnt+1)=...
                    BPSK_Mapping(ValidData(ValidDataPos));
```

```
                    case 'QPSK'
                        ValidData(ValidDataPos) = randi([0 3],1,1);
                        %ValidData(ValidDataPos) = randint(1,1,4);
                        ValidCarrierSig(ValidCarrierCnt+1)=...
                        QPSK_Mapping(ValidData(ValidDataPos));
                    case '16QAM'
                        ValidData(ValidDataPos) = randi([0 15],1,1);
                        ValidCarrierSig(ValidCarrierCnt+1)=...
                        QAM16_Mapping(ValidData(ValidDataPos));
                    otherwise
                        error('Modulation not supported');
                end
            end
        end
        %加扰%
        load prbs_pc;
        ValidCarrierSig=ValidCarrierSig.*...
        Pc(framecnt*ValidCarrierNum+1:(framecnt+1)*ValidCarrierNum);
        ValidCarrier=[ValidCarrier ValidCarrierSig];
        %IFFT 变换%
        OFDMSig=[zeros(1, 1) ValidCarrierSig(1:ValidCarrierNum/2) zeros(1,1019) ...
        ValidCarrierSig(ValidCarrierNum/2+1:ValidCarrierNum)];
        OFDMSig = ifft(OFDMSig, fft_pnts)*sqrt(fft_pnts);
        %OFDM 组帧%
        NumCP = Tcp/T;
        Num = Tu/T;
        times = Num/fft_pnts;
        OFDMSig_resample = zeros(1, Num);
        OFDMSig_nocode = zeros(1, Num);
        for cnts = 0:times-1
            OFDMSig_resample(cnts+1:times:(fft_pnts-1)*times+cnts+1)= ...
            OFDMSig;
        end
        SignalEnsemble(Pos:Pos+NumGI-1)=OFDMSig_resample(Num-NumCP-...
        NumGI+1:Num-NumCP).*Wt + SignalLastFrameForGI.*(1-Wt);
        Pos = Pos+NumGI;
        SignalEnsemble(Pos:Pos+NumCP-1)=OFDMSig_resample(Num-...
        NumCP+1:Num);
        Pos = Pos+NumCP;
        SignalEnsemble(Pos:Pos+Num-1) = OFDMSig_resample;
        Pos = Pos+Num;
        SignalLastFrameForGI = OFDMSig_resample(1:NumGI);
        clear OFDMSig_resample  ValidCarrierSig ValidDataRev;
    end
end
save ValidData ValidData;
save SignalEnsemble SignalEnsemble;
```

```
else
    load ValidData ValidData;
    load SignalEnsemble SignalEnsemble;
end

if 1
    disp('Oversample and filter...');
    Hdlow = filter1;%(fs/2, fover);
    %Upconverter (incomplished)
    %chips = [SignalEnsemble;zeros(q-1,L)];
    %chips = reshape(chips, 1, L*q);
    %升采样%
    OFDM_Signall = upsample(SignalEnsemble, q);
    OFDM_Signall=complex(conv(real(OFDM_Signall), ...
    Hdlow.numerator*q),conv(imag(OFDM_Signall), Hdlow.numerator*q));
    OFDM_Signall=OFDM_Signall((length(Hdlow.numerator)-1)/2+1:end-...
    (length(Hdlow.numerator)-1)/2);
    save OFDM_Signall OFDM_Signall;
else
    load OFDM_Signall OFDM_Signall;
end
%Rayleigh 信道%
p1=[ 0.057662 0.576809  0.407163  0.303585 0.258782 0.061831 0.150340 0.051534
  0.185074 0.400967 0.295723 0.350825 0.262909 0.225894 0.170996 0.149723 0.240140
  0.116587 0.221155 0.259730];
q1=[4.885121 3.419109 5.864470 2.215894 3.758058 5.430202 3.952093 1.093586 5.775198
  0.154459 5.928383 3.053023 0.628578 2.128544 1.099463 3.462951 3.664773 2.833799
  3.334290 0.393889];
ti=[1.003019 5.422091 0.518650 2.751772 0.602895 1.016585 0.143556 0.153832 3.324866
   1.935570 0.429948 3.228872 0.848831 0.073883 0.203952 0.194207 0.924450 1.381320
   0.640512 1.368671]*100;
ti1=fix(ti/0.025);
K=1/(sqrt(sum(p1.^2)));
li=sqrt(-1);
Signal_channel=zeros(size(OFDM_Signall));
p2=circshift(p1,1);
q2=circshift(q1,5);
ti2=circshift(ti,8);
for i=1:20
    Signalstest=  K*p1(i)*exp(-2*pi*li*q1(i))*circshift(OFDM_Signall,ti1(i));
    Signal_channel=Signal_channel+Signalstest;
end
Signal_channel1=Signal_channel;
%高斯信道接收解调和 BER 分析%
SNRS = [0:0.5:30];
for SNRindex = 1:length(SNRS)
    chips_channel = awgn(Signal_channel1,SNRS(SNRindex),'measured');
```

```
%同步%
sync_type=2;
disp('sync searching...');
% %单滑动窗口能量检测%
%if sync_type==1%未设置空符号
%      SyncPos=energy_detection(chips_channel,sync_signal);
%end
%S&C 同步算法%
if sync_type==2
    SyncPos=S_C(sync_signal,chips_channel);
end
%双滑动窗口同步算法%
if sync_type==3
    SyncPos=Double_sliding_window(sync_signal,chips_channel);
end
samples = length(sync_signal);
%降采样
chips_channel=[chips_channel chips_channel(1:10000)];
Signal_rev = chips_channel(SyncPos:q:end);
clear r2 r1 gamma epsilon theta;
disp('Receiving and demodulating ofdm signal...');
Signal_rev = Signal_rev(2*samples+1+round((Tgi+Tcp)*fft_pnts/Tu):end);
OFDM_Signall = zeros(53,fft_pnts);
for i=1:53
    offset = (i-1)*round((Tgi+Tcp+Tu)*fft_pnts/Tu)+1;
    OFDM_Signall(i,:) = Signal_rev(offset:offset+fft_pnts-1);
end
OFDM_Signall = fft(OFDM_Signall.',fft_pnts)/sqrt(fft_pnts);
OFDM_Signall=OFDM_Signall([2:ValidCarrierNum/2+1 fft_pnts-...
ValidCarrierNum/2+1:fft_pnts], :);
OFDM_Signall = reshape(OFDM_Signall, 1, 53*ValidCarrierNum);
disp('Descramble...');
load prbs_pc;
%Pc1=Pc(1:3076*53);
OFDM_Signall = OFDM_Signall./Pc;
clear Pc;
OFDM_Signall = reshape(OFDM_Signall, ValidCarrierNum, 53).';
% scatterplot(OFDM_Signall);
ValidDataRev = zeros(53, DataCarrierNum);
disp('Get off pilots carriers...');
ScatteredPilotCarriersEven=[1:8:8*(ScatteredPilotNum/2-1)+1 ...
8*(ScatteredPilotNum/2)+3:8:8*(ScatteredPilotNum-1)+3];
ScatteredPilotCarriersOdd=[5:8:8*(ScatteredPilotNum/2-1)+5...
8*(ScatteredPilotNum/2)+7:8:8*(ScatteredPilotNum-1)+7];
ContinualPilotCarriers = [ZeroContinualPilotCarriers DirectorContinualPilotCarriers];
k0=1;
k1=1;
```

```
%LS 信道估计%
if 1
    ValidDataRev_ScatteredPilot=zeros(size(OFDM_Signall));
    for i=1:ValidCarrierNum
        if (find([ScatteredPilotCarriersEven]==i-1))
            ValidDataRev_ScatteredPilot(1:2:53, i) = OFDM_Signall(1:2:53, i);
        elseif (find([ScatteredPilotCarriersOdd]==i-1))
            ValidDataRev_ScatteredPilot(2:2:53, i) = OFDM_Signall(2:2:53, i);
        end
    end
    D1=find(ValidDataRev_ScatteredPilot(1,:));
    D2=find(ValidDataRev_ScatteredPilot(2,:));
    Hp=ValidDataRev_ScatteredPilot;
    HL=Hp;
    % %linear interpolation in time
    HL(2:2:52,D1)=(HL((2:2:52)-1,D1)+HL((2:2:52)+1,D1))/2;
    HL(1,D2)=HL(2,D2)-(HL(4,D2)-HL(2,D2))/2;
    HL(3:2:51,D2)=(HL((3:2:51)-1,D2)+HL((3:2:51)+1,D2))/2;
    HL(53,D2)=HL(52,D2)-(HL(50,D2)-HL(52,D2))/2;
    Hl=HL.';%%LMMS
    %linear interpolation in frequency
    HL(:,1)=HL(:,2)-(HL(:,6)-HL(:,2))/2;
    HL(:,3076)=HL(:,3072)-(HL(:,3072-4)-HL(:,3072))/2;
    for k=0:(384-2)
        for t=1:3
            HL(:,2+k*4+t)=(1-t/3)*HL(:,2+k*4)+(t/3)*HL(:,2+(k+1)*4);
        end
    end
    for t=1:5
        HL(:,1534+t)=(1-t/5)*HL(:,1534)+(t/5)*HL(:,1540);
    end
    for k=0:(384-1)
        for t=1:3
            HL(:,1538+2+k*4+t)=(1-...
            t/3)*HL(:,1538+2+k*4)+(t/3)*HL(:,1538+2+(k+1)*4);
        end
    end
    OFDM_Signall=OFDM_Signall./HL;
end
%判决%
k0=1;
k1=1;
for i=1:ValidCarrierNum
    if (find([ContinualPilotCarriers ScatteredPilotCarriersEven]==i-1))
        assert(k0<=DataCarrierNum+1);
    else
        ValidDataRev(1:2:53, k0) = OFDM_Signall(1:2:53, i);
```

```
                k0=k0+1;
            end
            if (find([ContinualPilotCarriers ScatteredPilotCarriersOdd]==i-1))
                assert(k1<=DataCarrierNum+1);
            else
                ValidDataRev(2:2:53, k1) = OFDM_Signall(2:2:53, i);
                k1=k1+1;
            end
        end
        disp('Demodulating...');
        switch modulationMode
            case 'BPSK'
                Const = BPSK_Mapping(0:1);
            case 'QPSK'
                Const = QPSK_Mapping(0:3);
            case '16QAM'
                Const = QAM16_Mapping(0:15);
            otherwise
                error('Modulation not supported');
        end
        ValidDataRev = reshape(ValidDataRev.', Ensemble_SimNum*53*DataCarrierNum, 1).';
        ValidDataRev1=ValidDataRev(1:2610*53);
        for i=1:length(ValidDataRev)
            [c x] = min(abs(Const-ValidDataRev(i)));
            ValidDataRev(i) = x-1;
        end
        %SER 和 BER 分析%
        disp('Calculating BER...');
        ValidData=ValidData(1:2610*53);
        diff = ValidDataRev-ValidData;
        %catterplot(ValidDataRev,1,0,'rx');
        SER(SNRindex) = sum(abs(diff)>1e-6)/length(ValidDataRev);
        bit_recev_noclip = dec2bin(ValidDataRev,2);
        bit_recev_noclip = reshape(bit_recev_noclip.',1,[]);
        bit_ori = dec2bin(ValidData,2);
        bit_ori = reshape(bit_ori.',1,[]);
        diff = bit_recev_noclip - bit_ori;
        BER(SimNum,SNRindex) = sum(abs(diff)>1e-6)/length(diff);
        clear diff;
    end
end
ber=mean(BER,1);
figure;
hold on;
semilogy(SNRS,ber,'r')
grid on;
hold off;
```

```
title('BER of transimited data')
xlabel('SNR/dB')
ylabel('BER')
end
```

程序 5-4 为双同步头 OFDM 同步系统仿真分析模型实验平台主程序，第 2 章程序 2-22～程序 2-25 和第 5 章程序 5-1～程序 5-3 为程序 5-4 调用的子程序模块。

5.4.2 实验结果分析

表 5-1 给出了实验平台(程序 5-4)OFDM 系统各项参数，无线信道选择瑞利信道模型及参数。

表 5-1 OFDM 实验平台参数

带宽模式	8M
IFFT/FFT 载波个数	4K
有效子载波	3076
数据子载波	2610
离散导频/连续导频	384/82
调制模式	QPSK/16QAM

如图 5-21 所示，信号经过瑞利信道后，因为多径时延的影响，导致信号混叠。滑动窗口法能量检测不适合定时同步，由此本实验选择 SC 定时同步和双滑动窗口同步算法进行同步性能检验，基于 BER 分析同步的影响和精度。由图 5-21 可知，双滑动窗口同步算法的同步更为精确，验证了前续分析结论。SC 定时同步在估计精度要求下，受衰落和噪声影响较大，很难达到高度精确的估计效果。

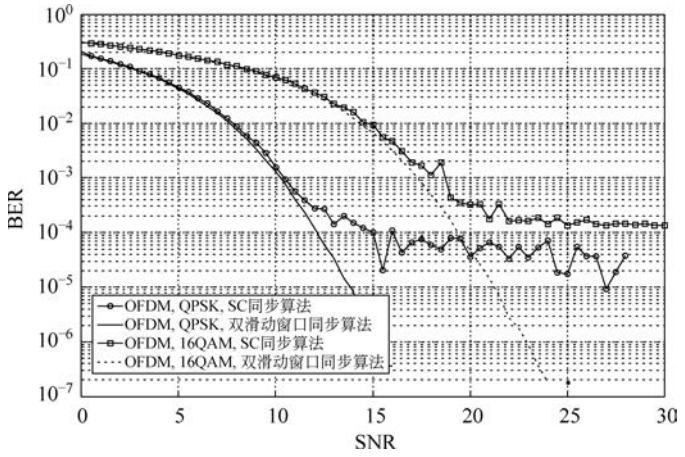


图 5-21 OFDM 同步系统 BER 分析(SNR/dB)