2024 年（夏）课程设计 B

信号处理基础设计

结题报告

题　　目：信号频率估计算法仿真与分析

组　　别：第二大组 第 8 小组

姓　　名：石铭宇 万郁彬 梁展博 尹子昂 杨炫辉

学　　号：210210633

班　　级：通信 6 班

指导教师：李伊川　曲龙跃

开课学院：电子与信息工程学院

地　　点：K524

时　　间：2024 年 7 月 28 日

结题评分及标准

考察项目	考察内容		备注（评语）
A 口头答辩 100 分	PPT 精美清晰，内容完整，理解充分，表达清晰，问题回答全面准确（90-100 分）		准备 PPT，按组提交一份 PPT，分工合理
	PPT 清晰，内容完整，理解较充分，表达清晰，问题回答正确（80-89 分）		
	PPT 较清晰，内容完整，理解较充分，表达基本清晰，问题回答基本正确（60-79 分）		
	PPT 制作粗糙，内容较完整，表达模糊，问题回答不正确（60 分以下）		
B 结题报告内容（必选）100 分	格式规范；内容充实；分析深刻，见解独到（90-100 分）		按模板撰写，每人提交，并源程序代码,以工程文件形式按组提交一份
	格式规范；内容完整；分析正确，有个人见解（80-89 分）		
	格式较规范；内容较完整；分析基本正确（60-79 分）		
	格式不规范；内容不完整（60 分以下）		
综合评分	λ_1A+ λ_2B 式中 $\Sigma\ \lambda_i$=1		

报告正文

一、研究的目的与意义

对淹没在噪声中的正弦波信号进行频率估计是信号处理的一个经典课题，在雷达和通信等领域有广泛应用。本课程设计旨在通过对信号频率估计算法的仿真与分析，深入探究信号处理领域中频率估计技术的核心原理与应用价值。

频率估计应用的最早的领域就是军事领域。一方面，有助于电子对抗能力提升：通过精确估计敌方通信信号的频率，为军事电子对抗提供关键信息，增强对敌方信号的干扰和侦察能力。另一方面应用于实时信号处理：在复杂的战场环境中，实现对快速变化信号的实时频率估计，提高军事指挥和作战的响应速度。

雷达系统中也应用广泛。如目标识别与跟踪：通过雷达回波信号的频率估计，提高目标的识别精度和跟踪稳定性，为雷达系统提供更准确的目标状态信息。又如抗干扰能力：在雷达信号处理中，有效抑制噪声和干扰信号，提高雷达系统的抗干扰能力，确保目标检测的可靠性。

随着移动互联网和物联网的快速发展，频率估计在无线通信系统中的应用需求也日渐增多，精确的频率估计算法有助于提升无线通信系统的服务质量和用户体验，为相关产业注入新的动力和活力。

综上所述，该研究不仅具有重要的理论价值，还在实际应用中展现出广泛的前景和深远的影响。通过深入研究和仿真分析信号频率估计算法，可以为军事、雷达、无线通信等领域的发展提供有力支持。

二、国内外研究现状概述

在通信、雷达、电子对抗等领域，对淹没在噪声中的正弦波信号频率进行快速估计是参数估计中的经典问题，目前国内外已有学者提出了不少方法。针对正弦波频率的快速估计，最常用的是基于离散傅里叶变换（DFT）的各类频率估计算法。

采用快速离散傅里叶变换（FFT）直接谱估计法进行正弦波频率估计，计算量小，工程上得到了广泛的应用，但 FFT 算法精度依赖于采样长度，容易出现栅栏效应，导致信号频率无法准确地落在离散频点上，于是当被估计频率位于量化频率附近时估计误差较大，所以需要研究利用已知谱线恢复出目标信号频率的细估计方法。常见的有 Rife 算法、Quinn 算法，这样直接通过 DFT 系数构造插值从而得到修正因子的直接细估计；也有牛顿迭代算法这样通过反复迭代得到最终目标频率的迭代细估计方法。

三、研究内容

由于栅栏效应的影响，当被估计频率位于量化频率附近时估计误差较大，我们主要通过研究经典的直接细估计方法，如 Rife 算法和 Quinn 算法，研究在不同条件下突破栅栏效应的性能，比较总结各自的优缺点，并研究其改进方案，探索各个算法的优点，并综合起来，更好地解决栅栏效应的问题。

四、研究方案

1. 算法综述

1.1 直接估计方法

对信号进行 FFT 后，搜索幅度最大的频率谱线，并以此作为得到的估计信号频率，但当出现栅栏效应甚至频谱泄露的情况时，估计误差大，因此作为同改良算法的比较参考。

1.2 Rife 算法（双线幅度法）

Rife 算法给出的频率估计公式为

$$\widetilde{f_0} = \Delta f(k_0 + \delta) = \Delta f(k_0 + \alpha \frac{|X_{k_0+\alpha}|}{|X_{k_0+\alpha}| + |X_{k_0}|}) \tag{1}$$

其中，δ 为信号频率与量化频率的频差，修正方向α=±1，当$|X_{k_0+1}| > |X_{k_0-1}|$时α=1；$|X_{k_0+1}| \le |X_{k_0-1}|$时α=−1；修正因子$\Delta k = |X_{k_0+\alpha}|/(|X_{k_0+\alpha}| + |X_{k_0}|)$，$\Delta k$ 满足 $0 \le \Delta k \le 0.5$。因此，被估计位于$k_0\Delta f$ 与$(k_0 + 0.5\alpha)\Delta f$ 之间。Rife 算法利用两根相邻谱线$|X_{k_0+\alpha}|$和$|X_{k_0}|$，故又称为双线幅度法。

文献中给出的 Rife 算法频率估计方差的计算公式为

$$\mathrm{Var}(\widetilde{f_0}) = (f_s/N)^2\{\frac{(1-|\delta|)^2[(1-|\delta|)^2+\delta^2]}{N(SNR)\mathrm{sinc}^2(\delta)} + 2\,\delta^2\,\mathrm{erfc}[\frac{\delta|\sin(\pi\delta)|}{\pi(1-\delta^2)}\sqrt{N(SNR)}]\} \tag{2}$$

由式（2）可知，在适度信噪比条件下，当$|\delta| = 0.5$ 时，即信号真实频率 f0 接近最谱线和次大谱线中间区域时，Rife 算法的估计性能很好，然而当信噪比较低且$|\delta| \le 0.1$ 时，估计误差很大。

1.3 Quinn 算法

Quinn 算法给出的频率估计为

$$\widetilde{f_0} = \Delta f(k_0 + \delta) \tag{3}$$

其中，δ 为频率修正项，表示为

$$\delta = \begin{cases} \delta_2, \delta_1 > 0, \delta_1 > 0 \\ \delta_1, \text{others} \end{cases} \tag{4}$$

式中 δ_1 和 δ_2 表示为

$$\begin{cases} \delta_1 = \beta_1/(1-\beta_1) \\ \delta_2 = \beta_2/(\beta_2 - 1) \end{cases} \tag{5}$$

β_1 和 β_2 表示为

$$\begin{cases} \beta_1 = \mathrm{Re}\{X_{k_0-1}/X_{k_0}\} \\ \beta_2 = \mathrm{Re}\{X_{k_0+1}/X_{k_0}\} \end{cases} \tag{6}$$

其中 Re{•}表示实部。

文献中给出了 Quinn 算法的频率估计方差为

$$\mathrm{Var}(\widetilde{f_0}) = (f_s/N)^2 \frac{(1-|\delta|)^2[(1-|\delta|)^2+\delta^2]}{N(SNR)\mathrm{sinc}^2(\delta)} \tag{7}$$

由式（7）可知，当 δ 接近 ± 0.5 时，Quinn 算法的频率估计精度很高，但是当 δ 接近零时，Quinn 算法的估计误差较大。

1.4 对 Rife 和 Quinn 算法的改进

Rife 和 Quinn 算法弥补了直接估计的弊端，但仍有不足，需要继续研究其改进算法。

Rife 算法在较小时，DFT 最大谱线左右两侧谱线的幅度受噪声影响较大，估计误差较大；Quinn 算法则充分利用了最大谱线左右两侧谱线的相位值来判断插值方向，从而避免了 Rife 算法在 δ 较小时频率估计误差激增的现象。但依然会出现 δ 较小时，误差较大的情况，针对这种情况，文献中提出采用频谱细化、频谱搬移等技术，以实现对算法的改良，降低频率估计的误差。

1.5 Rife 的改进算法

一、I-Rife 算法：

由于被估计频率 f_0 实质上是以 $|X_{k_0+0.5}|$ 和 $|X_{k_0-0.5}|$ 两谱线为界，且 $|X_{k_0\pm0.5}|$ 相比 $|X_{k_0\pm1}|$ 要大，噪声免疫力更强，因此选取 $|X_{k_0\pm0.5}|$ 代替 $|X_{k_0\pm1}|$ 作为修正方向 α 的判据更为合理。

于是，I-Rife 算法的频率估计公式为。

$$\widetilde{f_0} = \frac{f_s}{N} \cdot \left(k_0 - \alpha\delta_k + \alpha \cdot \frac{|X_{k_0-\alpha\delta_k+\alpha}|}{|X_{k_0-\alpha\delta_k+\alpha}| + |X_{k_0-\alpha\delta_k}|}\right) \tag{8}$$

$$\delta_k = \frac{1}{2} - \Delta_k \approx \frac{1}{2} - \frac{|X_{k_0+\alpha}|}{|X_{k_0+\alpha}| + |X_{k_0}|} \tag{9}$$

其中α=±1，当$|X_{k_0+0.5}| > |X_{k_0-0.5}|$时α=1；$|X_{k_0+0.5}| \le |X_{k_0-0.5}|$时α=−1；式（8）中$X_{k_0+0.5}$、$X_{k_0-0.5}$、$X_{k_0-\alpha\delta_k}$和$X_{k_0-\alpha\delta_k+\alpha}$由频谱细化技术得到。如采用 Zoom-FFT 方法进行频谱细化。

二、A-I-Rife 算法：

为使噪声对估计结果的影响最小，修正 I-Rife 算法使用$|X_{k_0}|$和$|X_{k_0+0.5\alpha}|$进行比较来判断 真实谱线位置，当$|X_{k_0}| > |X_{k_0+0.5\alpha}|$时，真实谱线更接近峰值谱线，此时用$|X_{k_0\pm0.5}|$进行插 值计算；当$|X_{k_0}| \le |X_{k_0+0.5\alpha}|$，真实谱线更接近细化谱线，此时用$|X_{k_0}|$和$|X_{k_0+\alpha}|$进行插 值计算。频移因子$\delta_k$可以表示为

$$\delta_k = \frac{1}{2} - \Delta_k = \begin{cases} \frac{1}{2} - \frac{|X(k_0+a)|}{|X(k_0+a)| + |X(k_0)|} & |X(k_0)| \le |X(k_0+0.5a)| \\ \frac{|X(k_0-0.5a)|}{|X(k_0-0.5a)| + |X(k_0+0.5a)|} & |X(k_0)| > |X(k_0+0.5a)| \end{cases} \quad (10)$$

1.6 Quinn 的改进算法

将 Quinn 算法和 Aboutanios 迭代算法相结合。Aboutanios 迭代算法的复杂度较高，通过 Quinn 算法代替第一次迭代，可大大降低 Aboutanios 迭代算法的复杂度，同时通过 Aboutanios 迭代算法是的 Quinn 算法的估计频率更精确。

2. 条件概述

2.1 考虑噪声的有无

无噪声时，则仿真结果失去了随机性，意味着多次蒙特卡洛实验将是无意义的，我们直接将仿真结果与实际频率的差值作为参考指标。

有噪声时，由于噪声具有一定随机性，仿真结果也必然会出现随机性，故我们采用多次蒙特卡洛实验的方法，采用 RMS 或者均值等作为指标来衡量算法的性能。

2.2 考虑窗函数的有无

我们使用了不同的窗函数，如汉明窗、汉宁窗、布莱克曼窗、布莱克曼-哈里斯窗、凯赛窗等，将其对信号进行处理，研究窗函数的有无和种类对算法性能的影响。

2.3 考虑干扰信号的有无

我们考虑将不同频率但相近的正弦波信号作为干扰信号，同有用信号叠加，实现对干扰信号影响的研究。我们计划主要考虑单频干扰的情况，并在研究过程

中尝试研究多频干扰的影响。

3. 仿真方案概述

3.1 服从不同分布函数噪声或不同干扰时的算法性能

我们计划考虑高斯分布、泊松分布、瑞利分布、均匀分布等白噪声，以及不同频率干扰或是单频、多频干扰的影响。

3.2 使用不同窗函数时的算法性能

我们计划引入汉明窗、汉宁窗、布莱克曼窗、布莱克曼-哈里斯窗等，并研究其对算法性能的影响。

3.3 不同信噪比/信干比的影响下的算法性能

我们计划研究不同 SNR/SIR 下，各种算法的性能，探索各个算法在低信噪比、高信噪比或是低信干比、高信干比下的表现及差异。

3.4 目标频率落入采样间隔不同位置时的算法性能

我们计划研究当目标频率落在谱线间隔不同位置时，不同算法的表现和差异。

3.5 不同算法频率估计的性能比较

对复正弦波信号，在相位、幅度和频率三个参数均未知的情况下，频率估计方差的克拉美-罗下界（CRLB）为

$$\mathrm{Var}\left(\widetilde{f_0}\right)=\frac{6{f_s}^2}{4\pi^2(\mathrm{SNR})N(N^2-1)} \tag{11}$$

我们通过多个指标来衡量算法的性能，一是通过多次蒙特卡洛实验的频率估计 RMS 值同频率的准确值进行比较，判断算法的准确性；通过计算多次蒙特卡洛实验的频率估计方差，同式（8）给出的频率估计方差的克拉美-罗下界（CRLB）进行比较，判断算法的稳定性。

四、实验数据及结果分析

本次仿真采样频率F_s设置为 2×10^3Hz，信号长度 L 为 128，选取的一个落在 DFT 频点的频率f_0为 5×10^2Hz。

选取信号功率为 1，相位为 0 的正弦波进行仿真，蒙特卡洛实验次数 n 设为 5×10^4

1. 无噪声情况

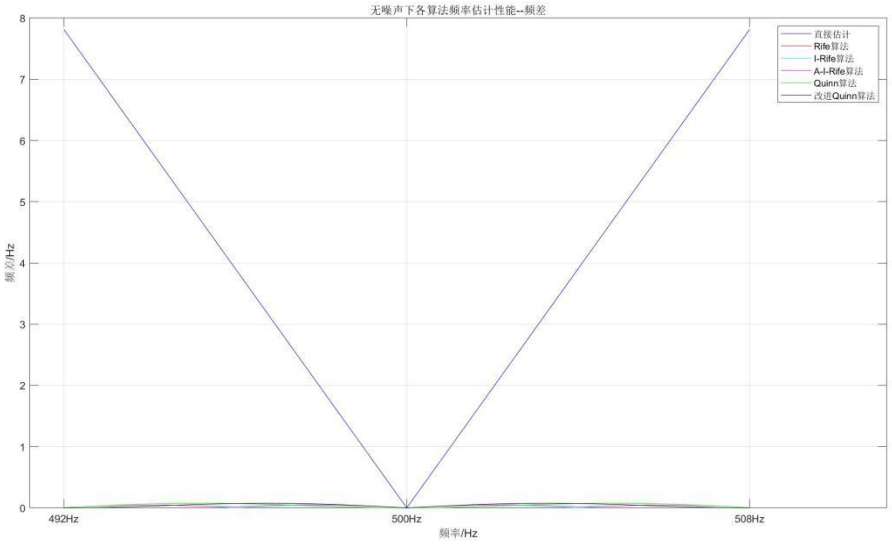

图 1-1 无噪声：估计偏差-信号频率

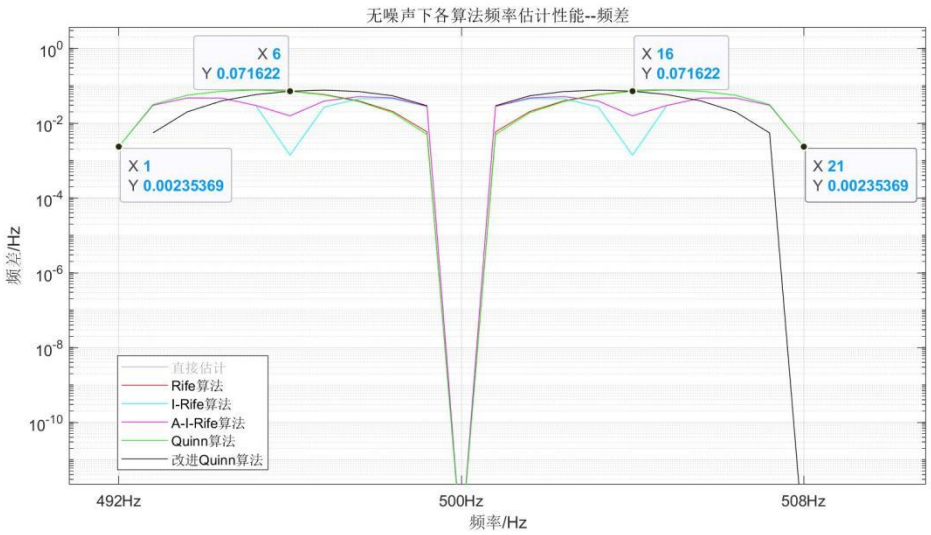

图 1-2 无噪声：估计偏差-信号频率（放大）

分析：①直接估计由于栅栏效应，虽然如果待测频率落在 DFT 的频点上，可以准确的估计出频率，但是当待测频率落在谱线间的时候，有明显的频率估计误差，而使用算法后的频率估计性能得到明显提高；

②相比原 Rife 算法，改进后的 Rife 算法在保证 Rife 算法性能的同时，在某些频率上得到了更好的提升。

2. 有噪声情况

本次仿真使用高斯白噪声

2.1 高信噪比（10dB）下：频率估计 RMS 值的频差Δf和频率估计方差 $\mathbf{Var}(\widetilde{\mathbf{f_0}})$

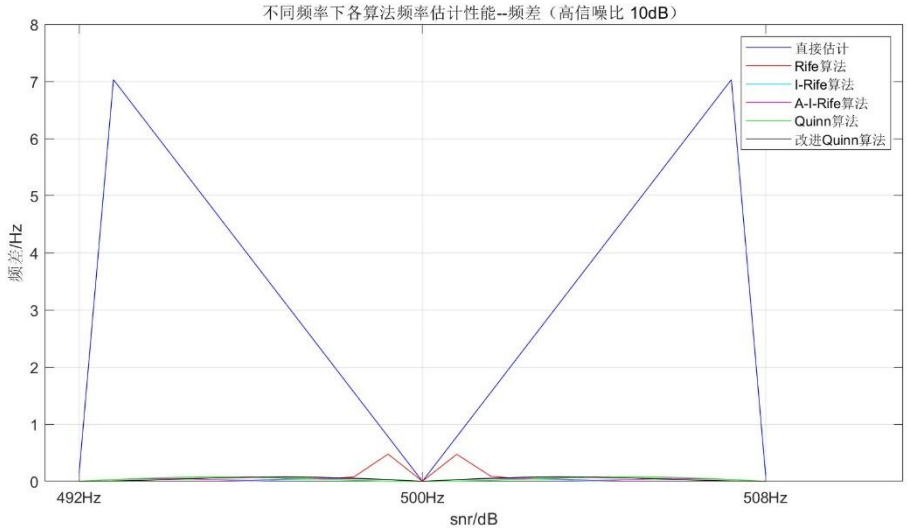

图 2-1 高信噪比时 频差Δf

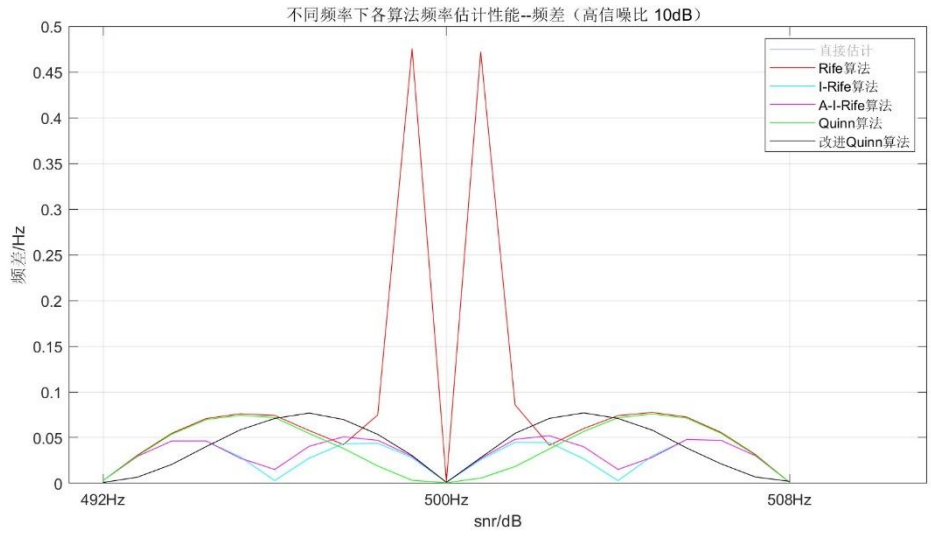

图 2-2 高信噪比时 频差Δf(放大)

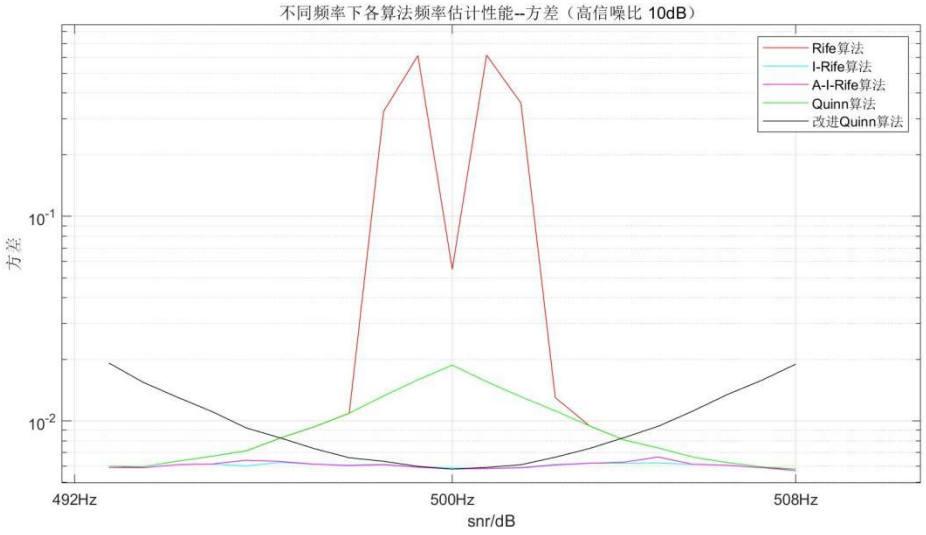

图 2-3 高信噪比时 信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

分析：高信噪比下：

①所有算法在谱线的点上，均表现良好，能准确估测出频率，但在谱线附近，Rife 算法表现出误差激增的现象，这一现象在改进后得到明显改善；

②直接估计在非 DFT 谱线处，表现差，而其他算法均解决的这一状况

③Rife 算法和 Quinn 算法改进后在 DFT 的谱线附近附近方差明显减小，即二者经改进后，在谱线附近频率估计不稳定的问题的到改善。

2.2 低信噪比（0dB）下：频率估计 RMS 值的频差Δf和频率估计方差$\mathbf{Var}\left(\widetilde{f_0}\right)$

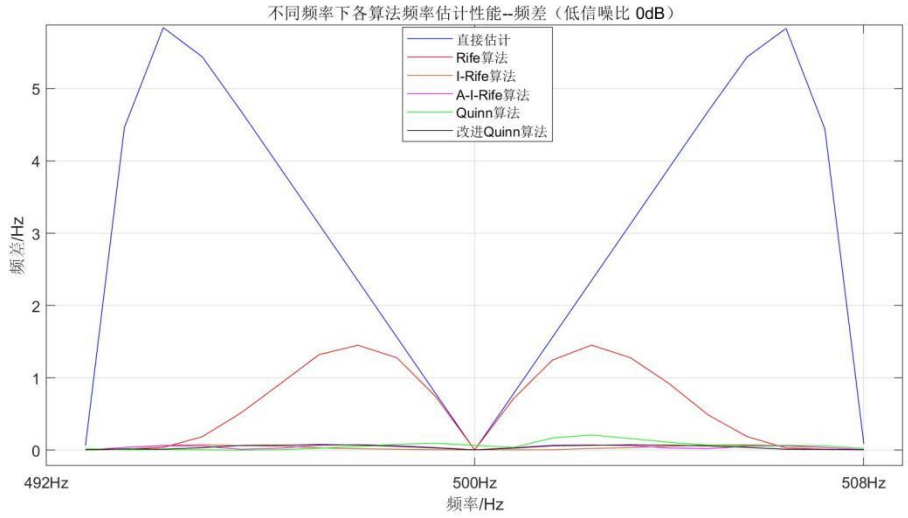

图 2-4 低信噪比下 信号频差Δf

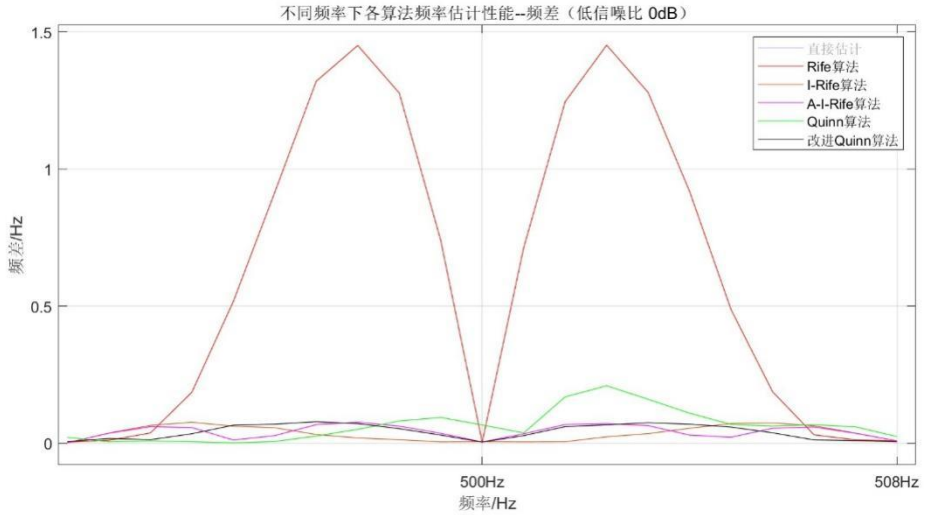

图 2-5 低信噪比下 信号频差Δf(放大)

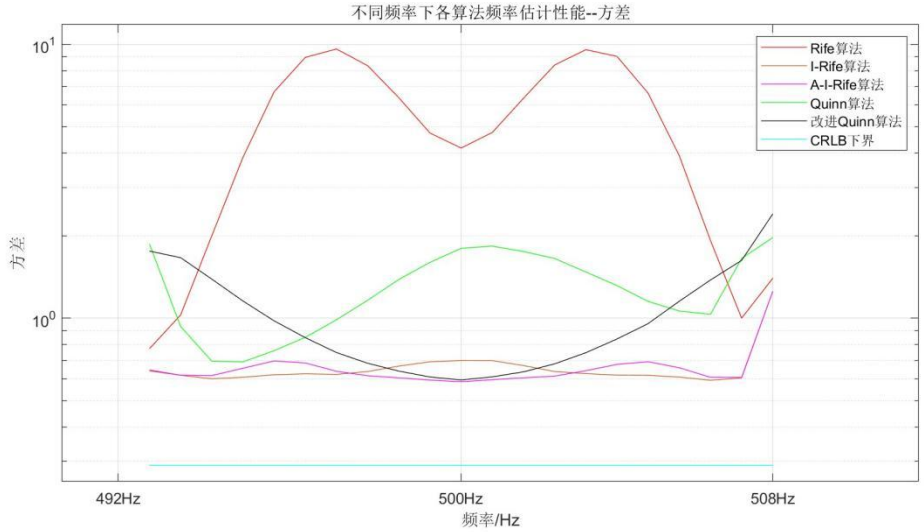

图 2-6 低信噪比下 信号方差$\mathrm{Var}\left(\tilde{f}_0\right)$

分析：低信噪比下：同高信噪比相比

① Rife 和 Quinn 算法的性能表现出不同程度的恶化，但改进后的算法依然较为精确；

② 由于噪声的增加，信号频率估计的稳定性，即方差也均遭到恶化，但基本情况仍同高信噪比类似，表现为 Rife＜Quinn≈改进 Quinn＜I-Rife≈A-I-Rife；

2.3 不同信噪下：频率估计 RMS 值的频差Δf和频率估计方差Var $(\widetilde{f_0})$

2.3.1 高斯分布噪声

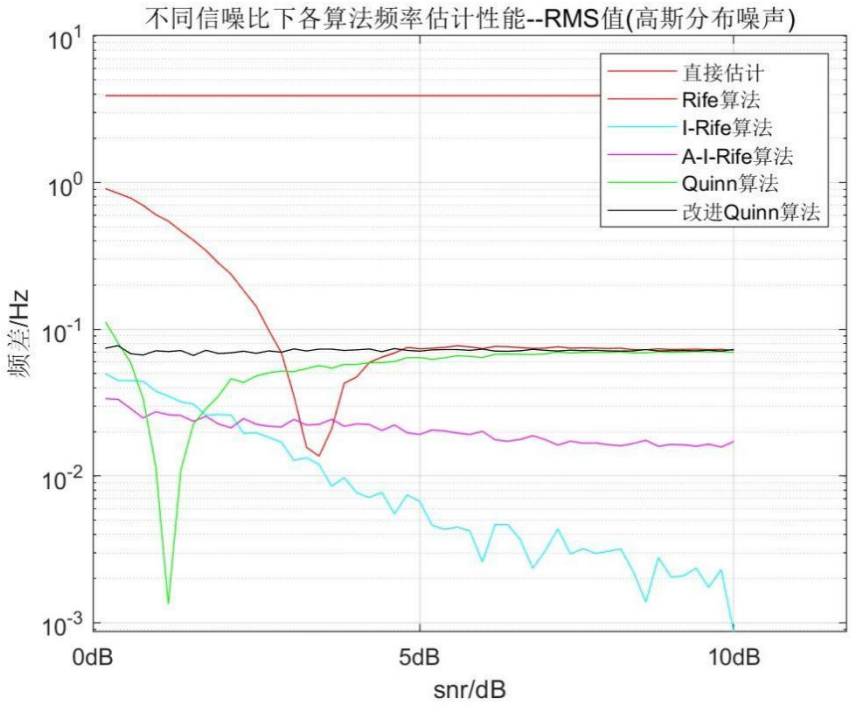

图 3-1 高斯噪声下 信号频差Δf

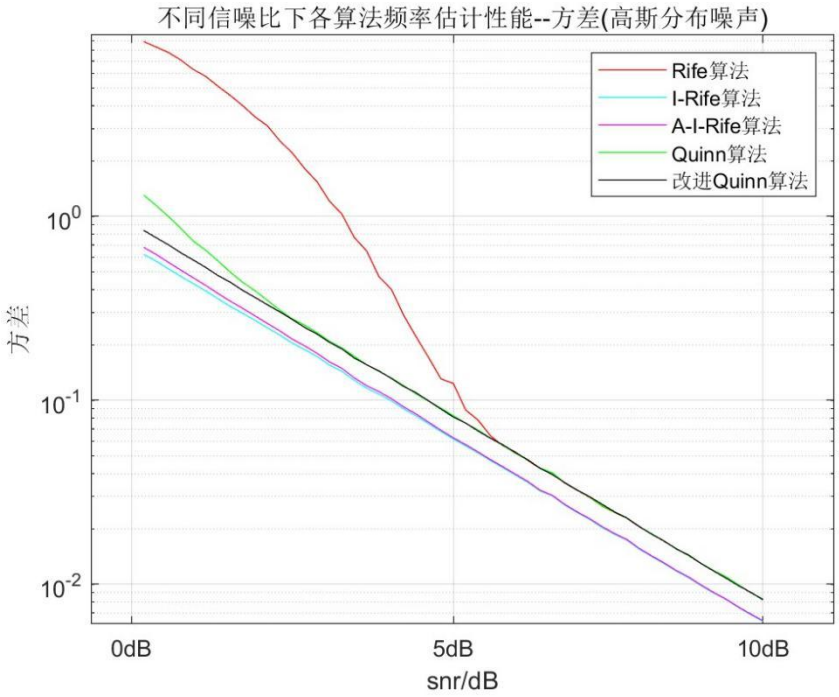

图 3-1 高斯噪声下 信号方差Var $(\widetilde{f_0})$

2.3.2 均匀分布噪声

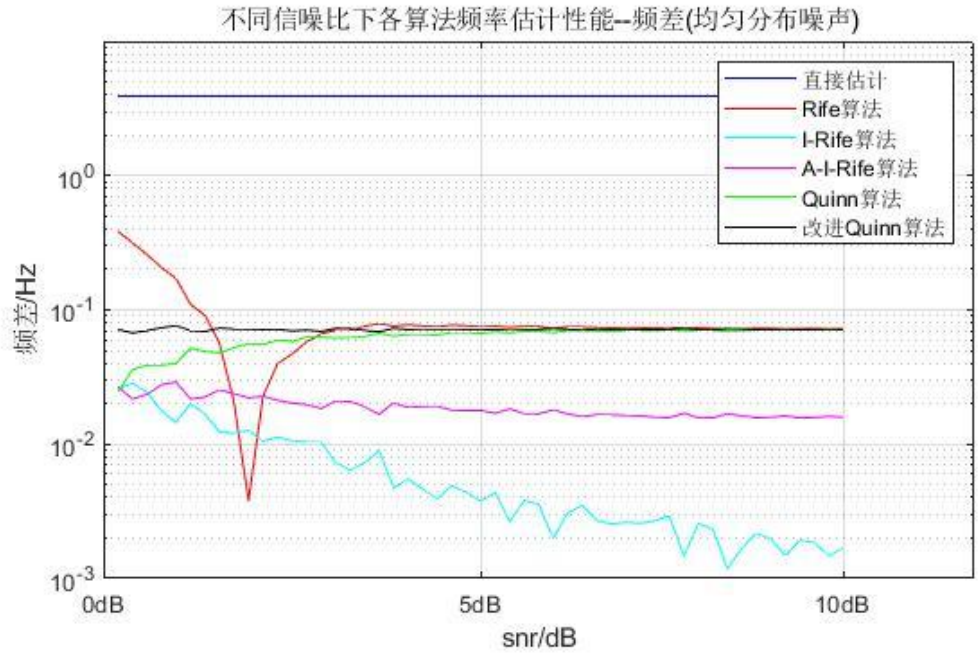

图 3-3 均匀分布噪声下 信号频差Δf

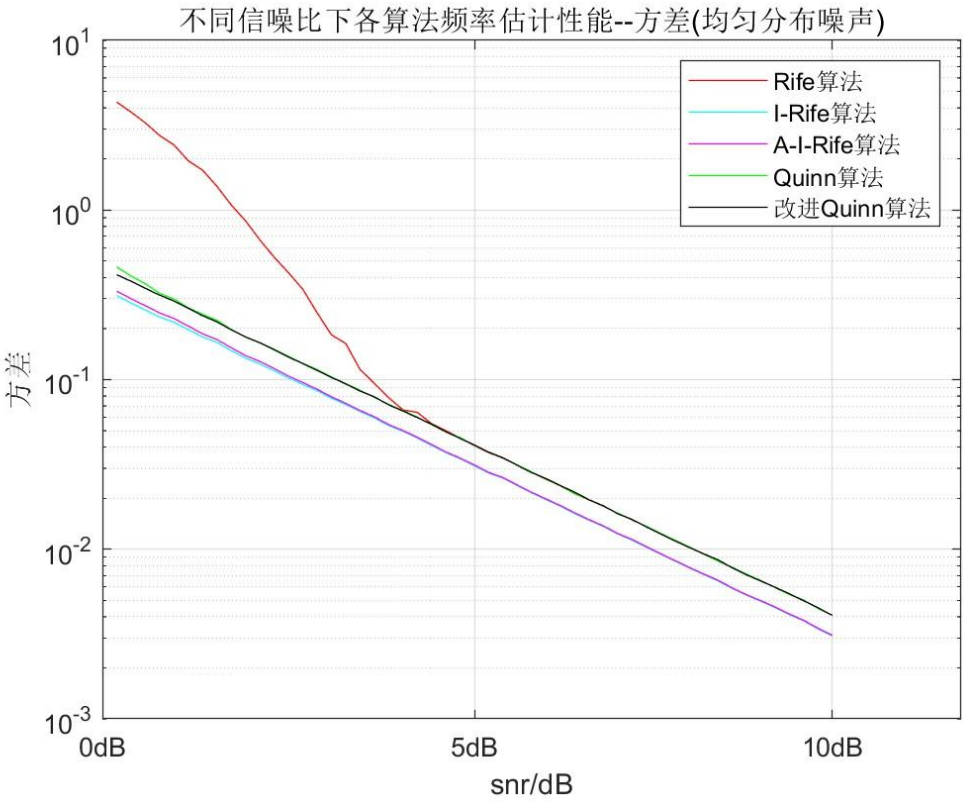

图 3-4 均匀分布噪声下 信号方差Var $(\widetilde{f_0})$

2.3.3 泊松分布噪声

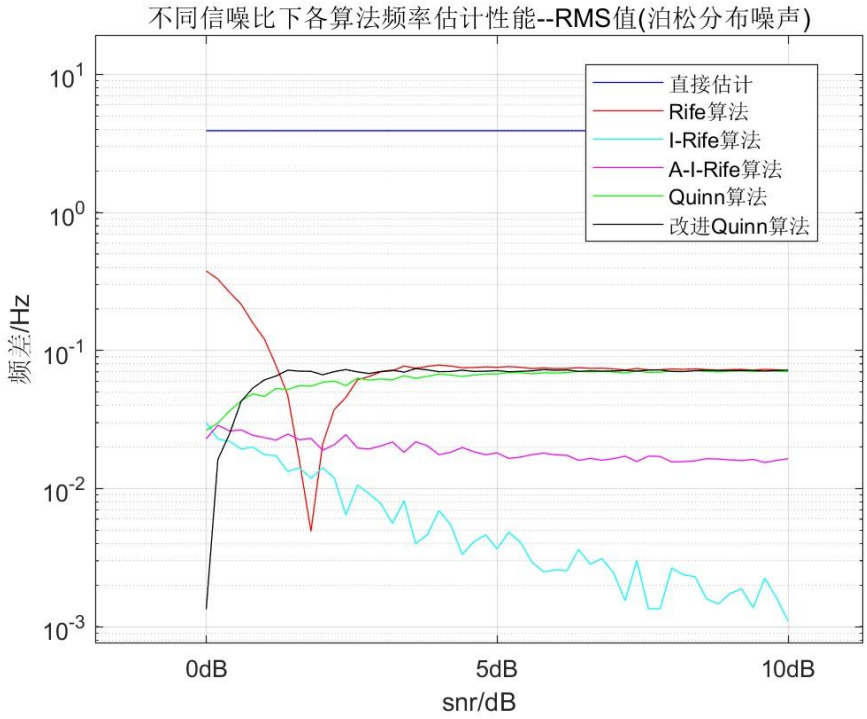

图 3-5 泊松分布噪声下 信号频差Δf

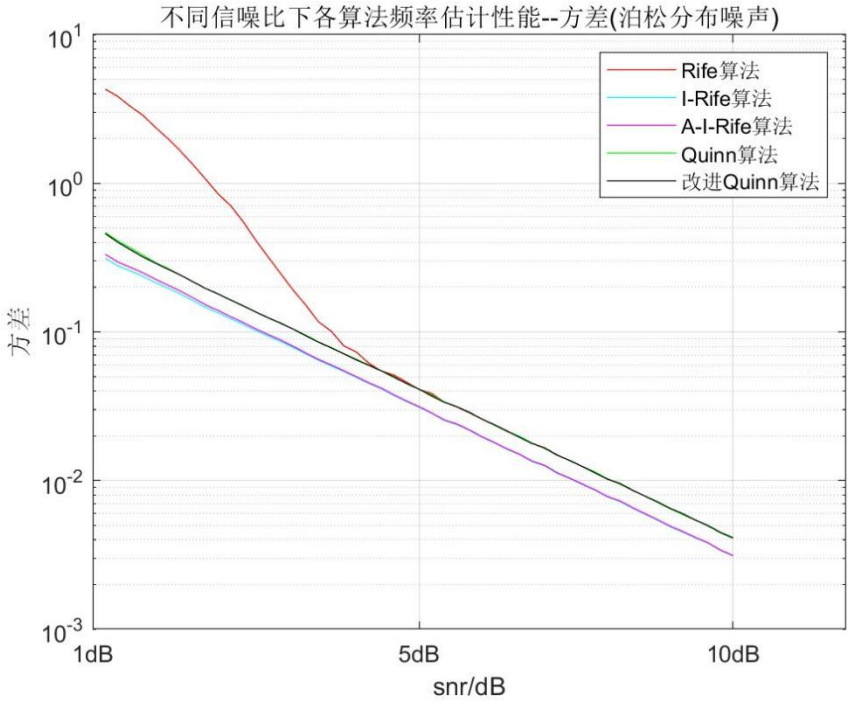

图 3-6 泊松分布噪声下 信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

2.3.4 瑞利分布噪声

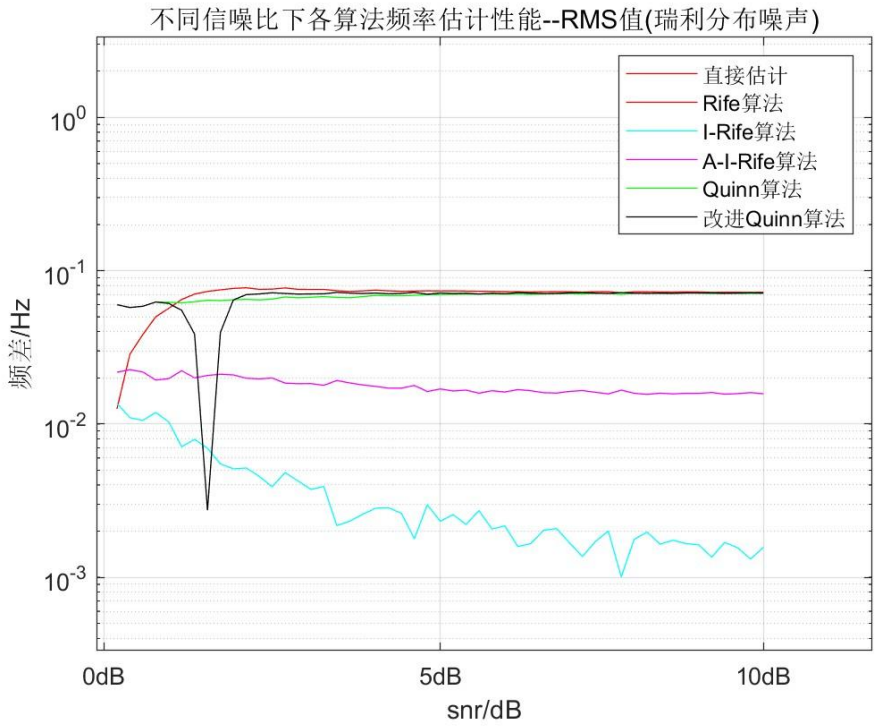

图 3-7 瑞利分布噪声下 信号频差Δf

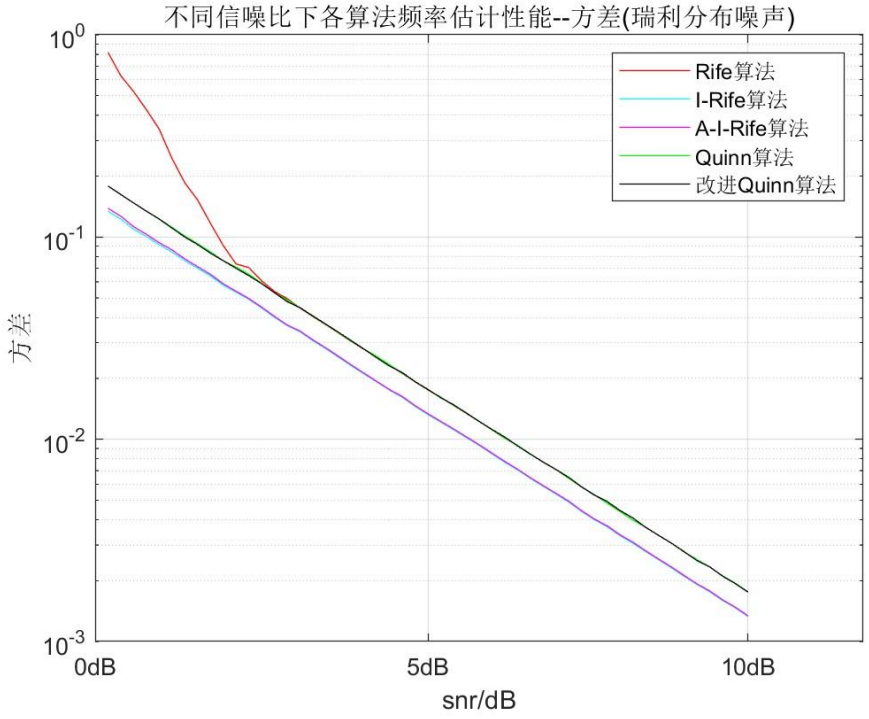

图 3-6 瑞利分布噪声下 信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

分析：① 直接估计与使用算法后的效果差别较大。不同噪声类型下，频差大小与方差差别总体差别较小，其中在瑞利分布下方差最小，下降较快，效果最好；

② Rife 算法在低信噪比下（除瑞利分布）较改进后的频率估计波动较剧烈，频率估计方差较大，改进后的 I-Rife 和 A-I-Rife 改善了这一点，方差更小，频率估计性能更稳定。其中 A-I-Rife 在的情况下变化较 I-Rife 频差更稳定，二者虽方差相近，但 A-I-Rife 算法受信噪比的影响更小，其估计频差保持在一个较为稳定的值。

③ Quinn 算法在低信噪比的高斯噪声下，改进后的 Quinn 算法在低信噪比的瑞利噪声下，频率估计的波动也较大。

④ 改进 Quinn 算法的在噪声这一条件上的改进并不明显，但 Rife 的改进算法无论是低信噪比还是高信噪比均对 Rife 有显著提升。具体表现为：

（低信噪比）频率估计误差：Rife＞Quinn≈改进 Quinn＞I-Rife≈A-I-Rife；
（高信噪比）频率估计误差：Rife≈Quinn≈改进 Quinn＞A-I-Rife＞I-Rife；
（低信噪比）频率估计方差：Rife＞Quinn≈改进 Quinn＞I-Rife≈A-I-Rife；
（高信噪比）频率估计方差：Rife≈Quinn≈改进 Quinn＞A-I-Rife≈I-Rife；

2.4 单频干扰下：频率估计 RMS 值的频差Δf和频率估计方差Var $(\widetilde{f_0})$

2.4.1 半个采样间隔

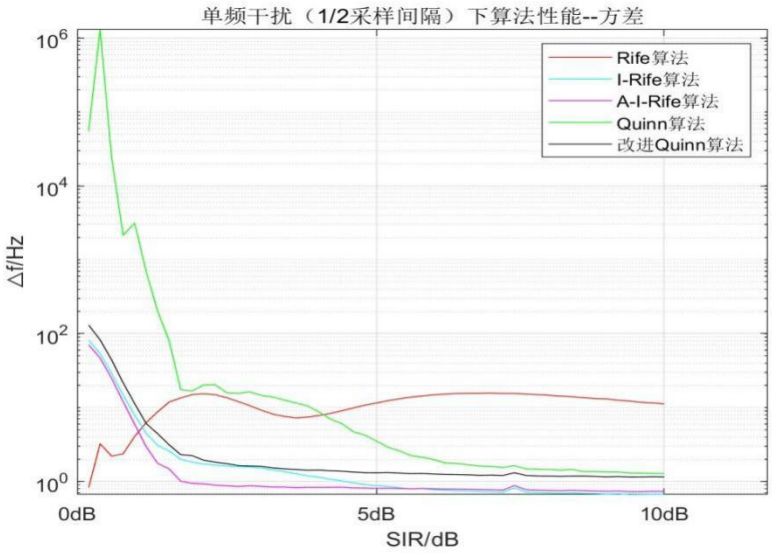

图 4-1 单频干扰（半个采样周期）信号频差Δf

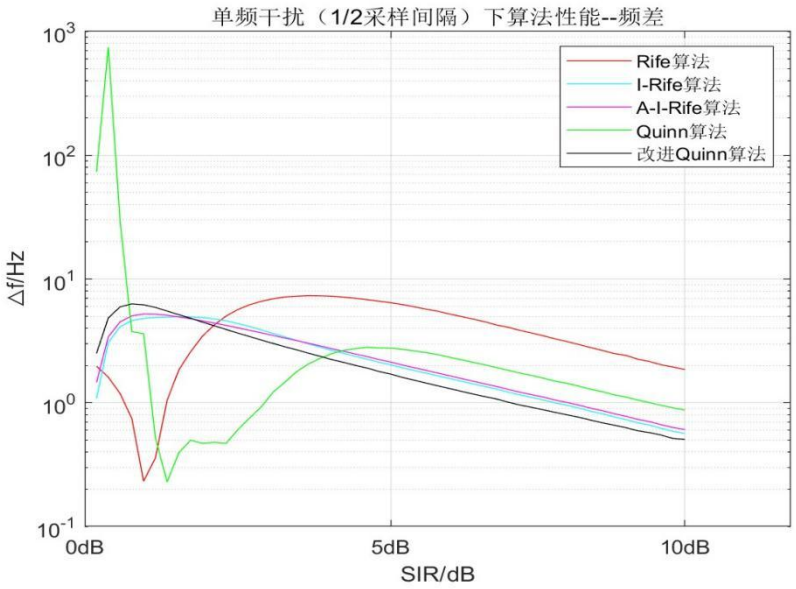

图 4-2 单频干扰（半个采样周期）信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

2.4.2 一个采样间隔

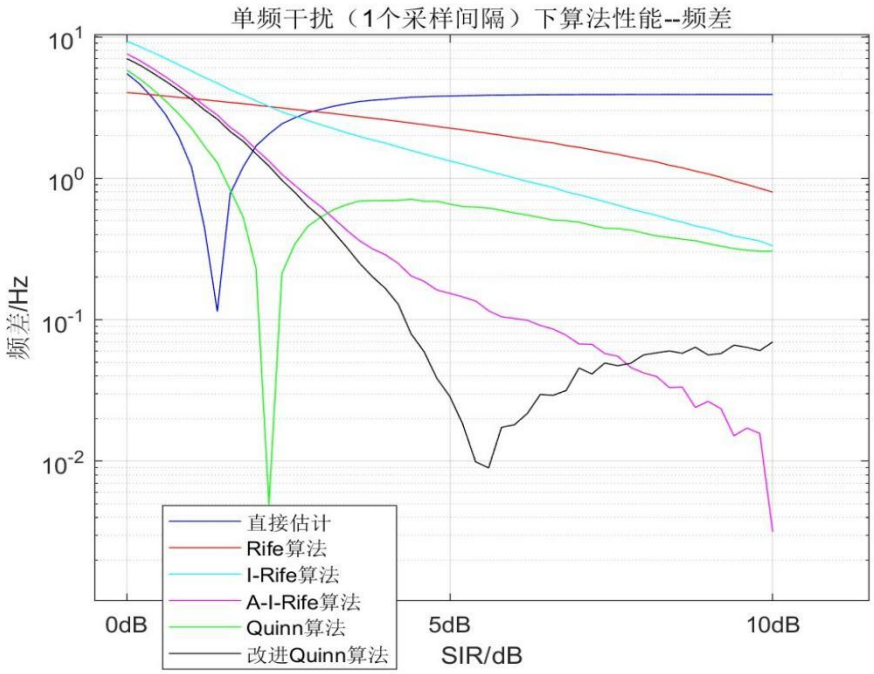

图 4-3 单频干扰（一个采样周期）信号频差Δf

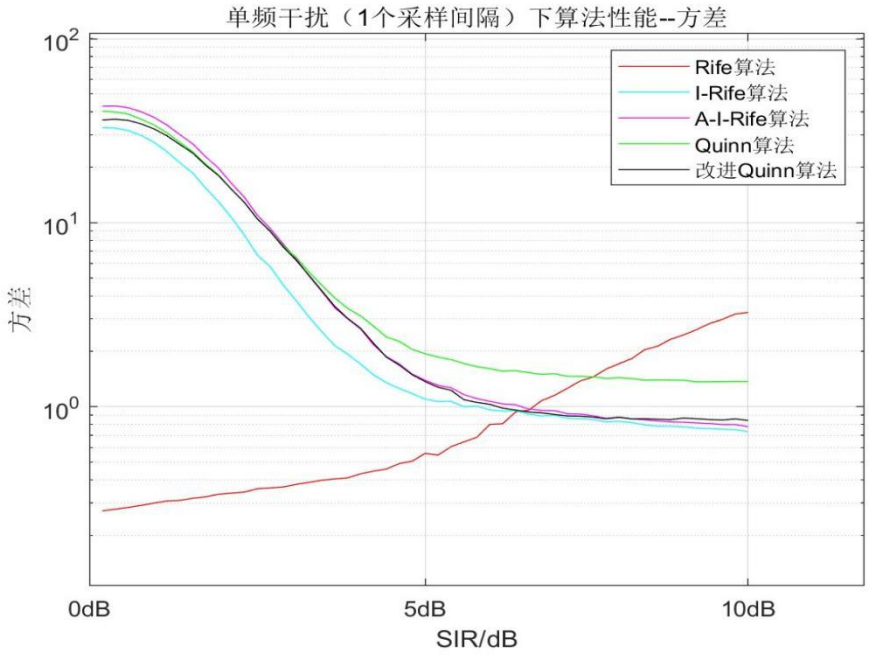

图 4-4 单频干扰（半个采样周期）信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

分析：①单频干扰下， Quinn 受干扰影响较大，其性能随信干比的波动较大；Rife 算法次之。

②Quinn 的改进算法和 A-I-Rife 算法抗干扰能力最强，其性能并不会随着信干比而剧烈变化，I-Rife 算法则较为次之，但也在 Rife 的基础上做出的改善。

②半个采样间隔下与一个采样间隔对比下，频差更大，但方差更小。其距估计频率的点更近，也符合事实。

2.5 不同窗函数下：频率估计 RMS 值的频差Δf和频率估计方差Var $(\widetilde{f_0})$

2.5.1 汉宁窗

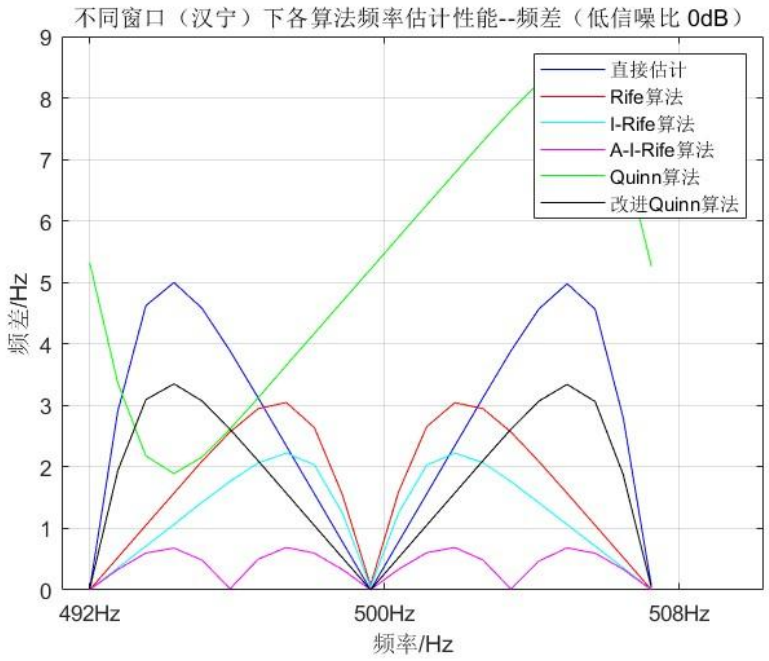

图 5-1 汉宁窗 信号频差Δf

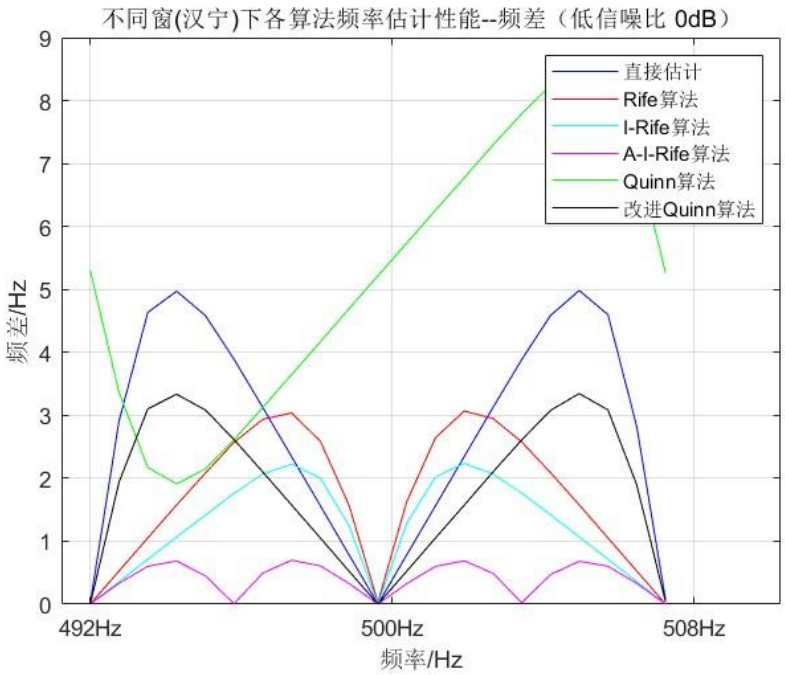

图 5-2 汉宁窗 信号方差Var $(\widetilde{f_0})$

2.5.2 哈明窗

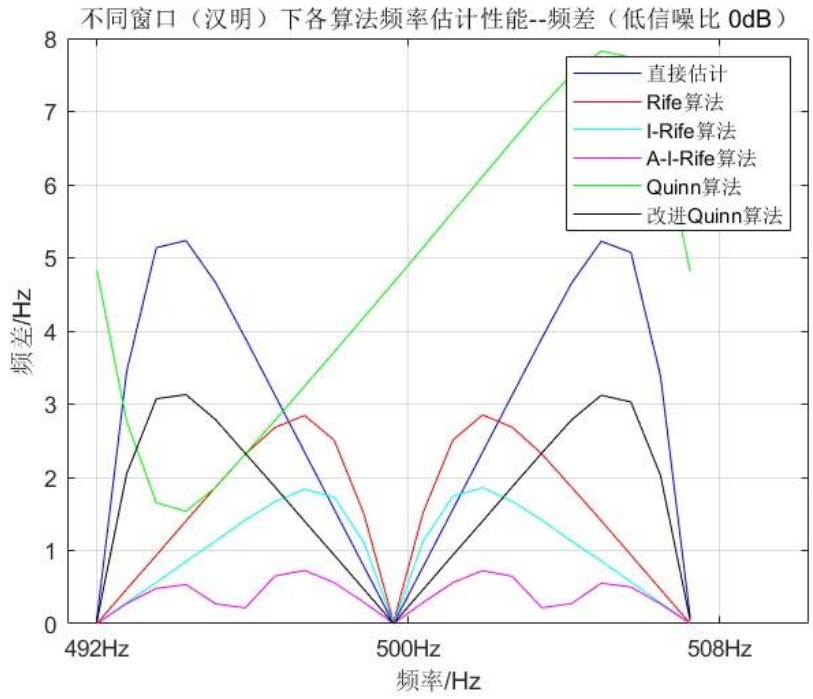

图 5-3 哈明窗　信号频差Δf

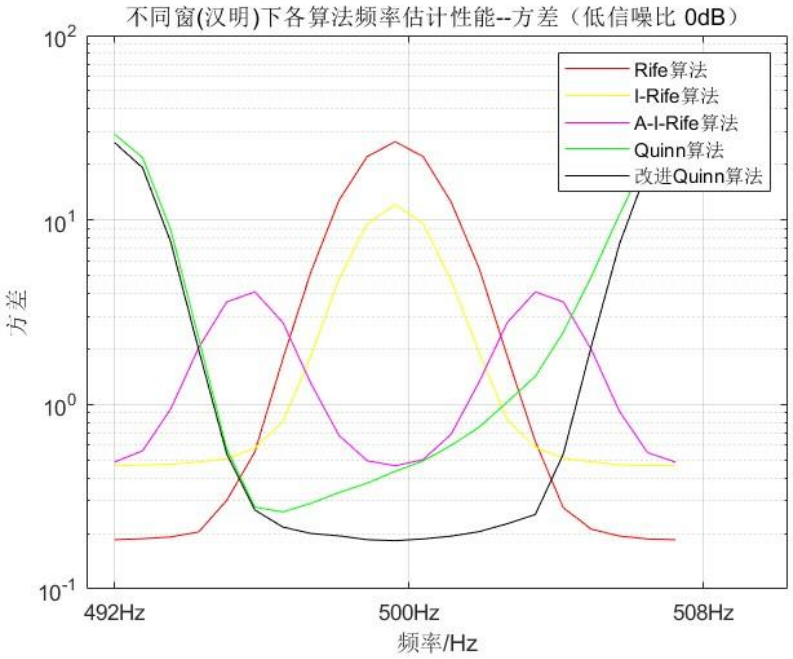

图 5-4 哈明窗　信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

2.5.3 布莱克曼窗

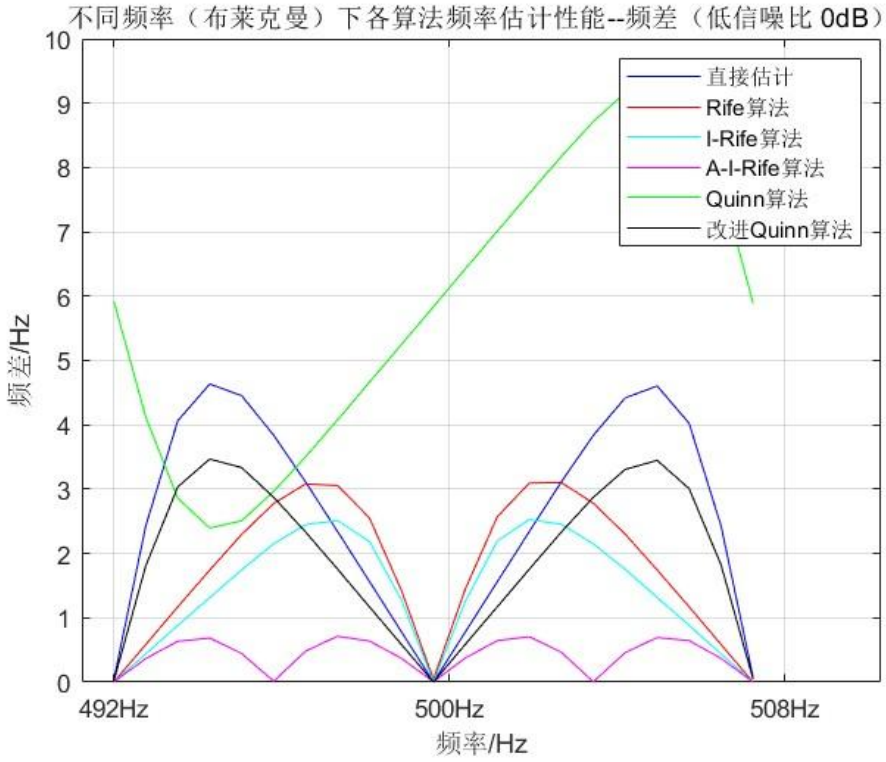

图 5-5 布莱克曼窗　信号频差Δf

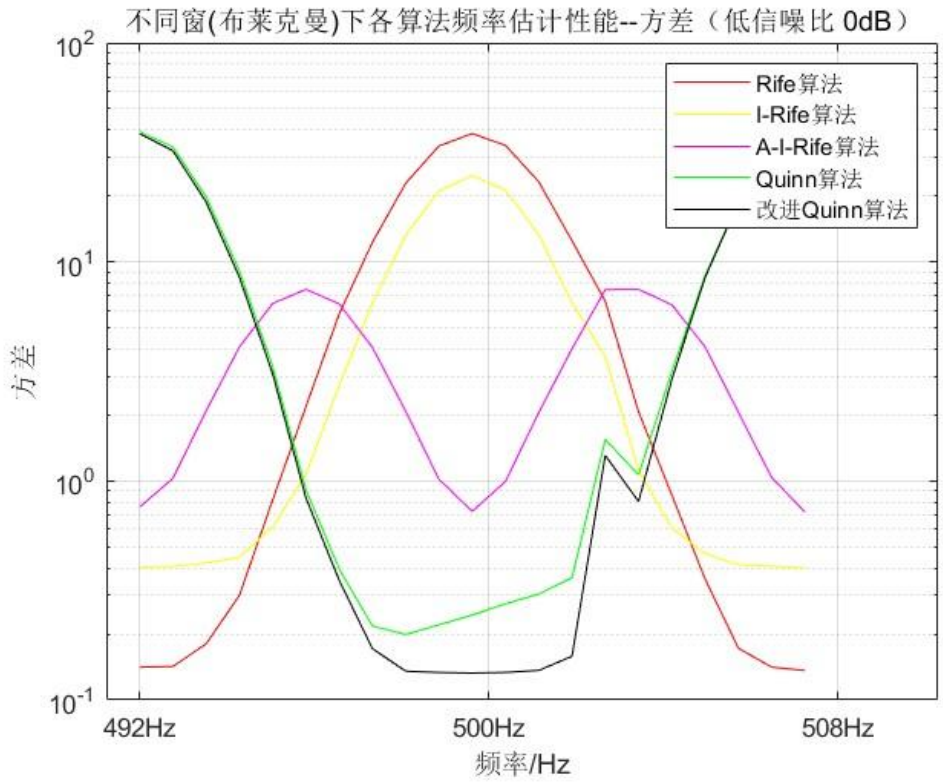

图 5-6 布莱克曼窗　信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

2.5.4 布莱克曼哈里斯窗

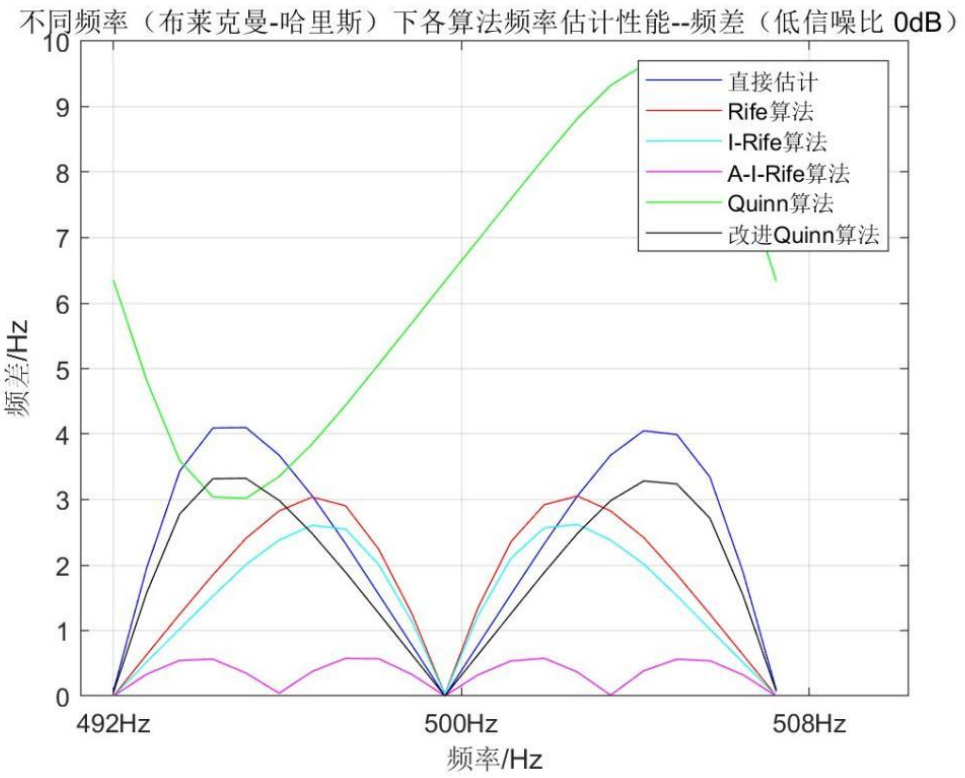

图 5-7 布莱克曼哈里斯窗　信号频差Δf

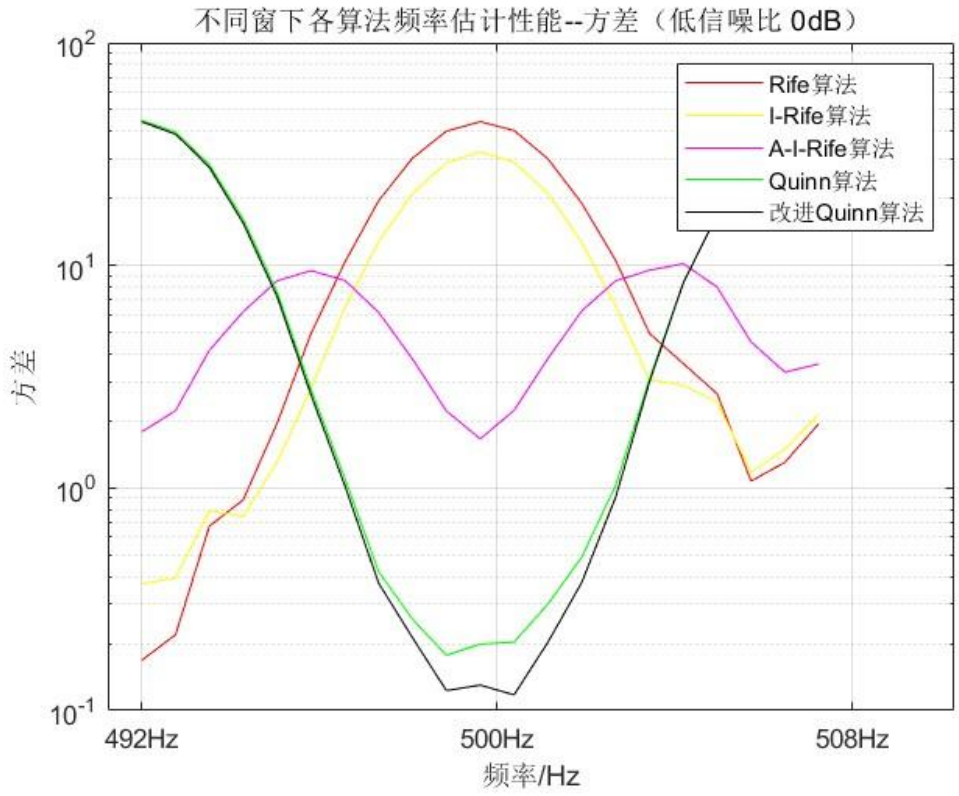

图 5-8 布莱克曼哈里斯窗　信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

2.5.5 凯赛窗

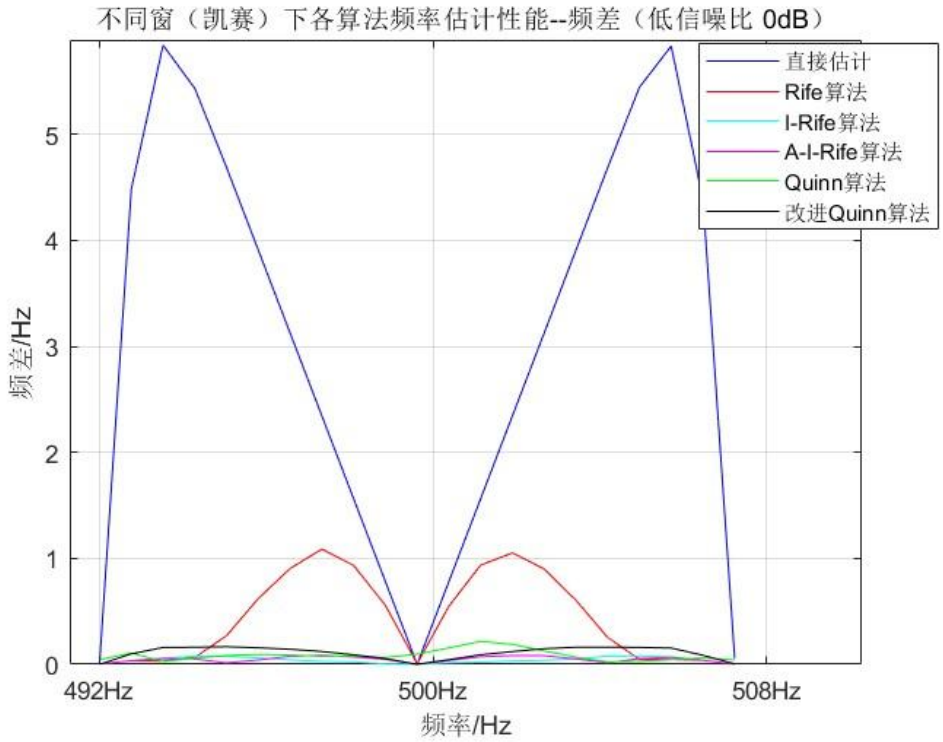

图 5-9 凯赛窗　信号频差Δf

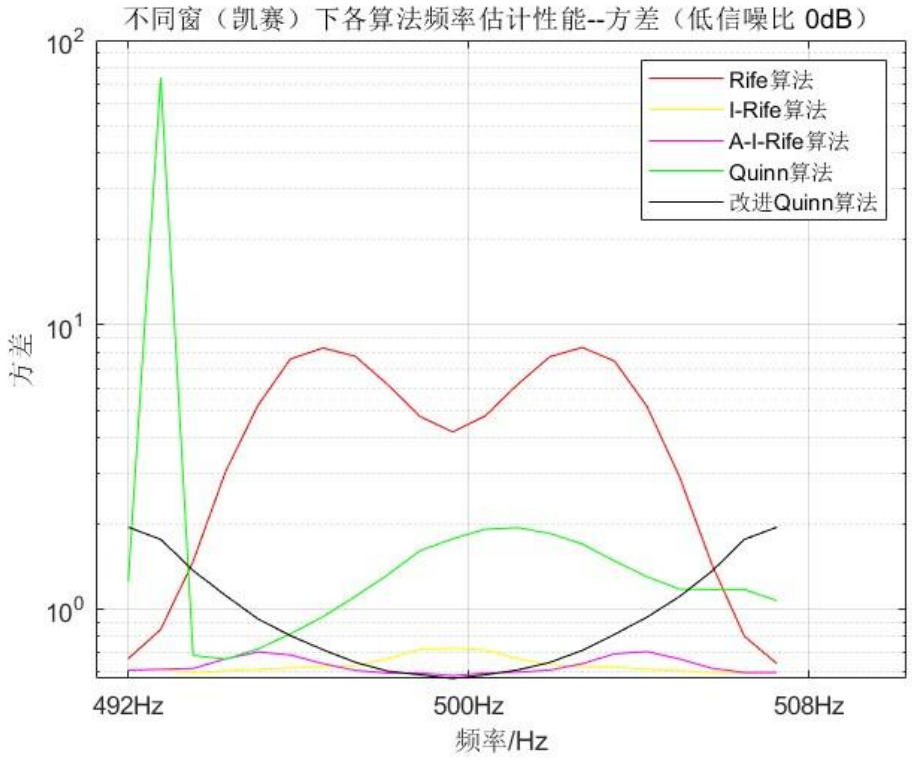

图 5-10 凯赛窗　信号方差$\mathrm{Var}\left(\widetilde{f_0}\right)$

分析：①加窗都会带来误差。其中凯赛窗的影响最小；

②低信噪比情况下，加窗（除凯赛窗）对 Quinn 算法的影响最为剧烈，它的频率在加窗条件下估计性能误差较大，对于落于不同点的频率都有不同程度的误差。这一原因我们猜想是由于 Quinn 是加入了相位信息的，而加窗改变了相位信息。改进 Quinn 算法显著的改善了这一缺点；

③Rife 算法窗函数的加入，误差明显增大，I-Rife 算法略微改进了这一点，A-I-Rife 算法则则表现出优异的性能。

③在加窗这一条件上，改进算法的频率估计误差和方差均做出了改善，表现为：（低信噪比）频率估计误差：Quinn＞直接估计＞改进 Quinn＞Rife＞I-Rife＞＞A-I-Rife。

五、结论

总体来说，随着信噪比/信干比的增大，RMS 误差都会减小。而不同的噪声类型和窗函数也会影响估计误差。同时对于不同的算法，频率落在间隔里的位置不同所对应的误差也不一样。同时改进的算法也在不同方面对原算法做出了不同程度的改进，各自有着各自的优点。

具体来说，A-I-Rife 做出的提升，主要体现在稳定性，其估计性能并不会随着信噪比/信干比的变化，而出现明显波动，同时窗函数也不会造成多大影响，这依靠其强大的抗干扰能力；I-Rife 做出的提升主要体现在频率估计误差上，其在高信噪比条件下表现出优异的性能；Quinn 算法在 Rife 算法的基础上，加入了相位信息，使得靠近 DFT 谱线的频率估计误差不会激增，但其在窗函数和干扰条件下仍有欠缺，性能波动较大，而改进 Quinn 算法解决了这一点。

个人收获方面：本人主要负责算法编写这一块，难点主要在于如何将所学的知识，应用于算法编写中，如何将学到的理论知识，转为计算机读得懂的语言，特别是 Debug 阶段，有时，看似算法写对了，可以运行，但其中总是藏着些潜在的错误，可能会导致算法的性能达不到想象的高度，而这也是最难熬的阶段。通过本次实验，本人认识到，一个好的项目，往往要不断地精细加工，才能得到。

六、项目分工

石铭宇：算法编写，开题报告，结题报告
万郁彬：开题答辩，仿真，开题 PPT
梁展博：仿真，结题 PPT
尹子昂：结题报告，论文查找
杨炫辉：开题 PPT，论文查找，结题 PPT
所有组员参与实验思路的设计。

七、参考文献

[1] 严静,周胜文,周云生,等.正弦波频率估计的改进 Quinn 算法[J].遥测遥控,2017,38(02):7-12.DOI:10.13435/j.cnki.ttc.002830.

[2] 邓振森刘渝王志忠等.正弦波频率估计的修正 Rife 算法[J].数据采集与处理,2006.12

[3] Barry G.Quinn. Estimation of Frequency, Amplitude, and Phase from the DFT of a Time Series[J]. IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45, NO. 3, MARCH 1997

[4] 王宏伟,赵国庆.正弦波频率估计的改进 Rife 算法[J].信号处理,2010,26(10):1573-1576.

[5] 齐国清.几种基于 FFT 的频率估计方法精度分析[A].大连海事大学.2006

[6] DAVID C. RIFE, ROBERT R. BOORSTYN. Single-Tone Parameter Estimation from Qiscrete-Time Observations[J].IEEE TRANSACTIONS ON INFORMATION THEORY, VOL. IT-20, NO. 5, SEPTEMBER 1974

[7] 王哲文,许晖,易辉跃,等.用于正弦波频率估计的修正 I-Rife 算法[J].数据采集与处理,2024,39(02):471-480.DOI:10.16337/j.1004-9037.2024.02.019.

[8] 赵强,侯孝民,廉昕.基于 Zoom-FFT 的改进 Rife 正弦波频率估计算法[J].数据采集与处理,2017,32(04):731-736.DOI:10.16337/j.1004-9037.2017.04.009.

[9] D. C. Rife and G. A. Vincent, "Use of the discrete fourier transform in the measurement of frequencies and levels of tones," in The Bell System Technical Journal, vol. 49, no. 2, pp. 197-228, Feb. 1970, doi: 10.1002/j.1538-7305.1970.tb01766.x.

[10] G. Yue, Z. Xiong and S. Jun, "Modified algorithm of sinusoid signal frequency estimation based on Quinn and Aboutanios iterative algorithms," 2016 IEEE 13th International Conference on Signal Processing (ICSP), Chengdu, China, 2016, pp. 232-235, doi: 10.1109/ICSP.2016.7877830.

附：程序代码（注释不少于 50%）

1. 主函数

1.1 基本参数设置

```
clc; clear;
% 基本参数设置
Fs = 2e3; % 采样频率，可设置其他
T = 1/Fs;  % 采样周期
L = 128;  % 信号长度，可设置其他
t = (0:L-1)*T; % 时间向量
delta_f = Fs/L; %采样间隔
f0 = 5e2; %选定的一个落在采样点上的频率，也可设置其他
A = sqrt(2);  % 振幅，使得信号功率为 1，也可设置其他
phi_0 = 0;   % 相位，可设置其他
n = 10000;%蒙特卡洛实验次数，可设置其他
```

1.2 X 轴为不同频率的仿真

```
%x轴为不同频率的仿真（适用'高/低信噪比情况下不同频点的仿真'、'无噪声时仿真'、'窗函数'等基于频率的仿真）

f = f0-delta_f/2:delta_f/20:f0+delta_f/2;%更改待检测频率的范围，所示为从选定的采样点到两侧采样点中心，以 1/20 个采样间隔为单位等间距取点
x_f = ones(length(f),L); %预先设置待测信号矩阵的大小
for j = 1:length(f) %利用预先设置的参数生成待测信号
    x_f(j,:) = A*sin(2*pi*f(j)*t + phi_0);
end
P_signal = 0.5*A^2;%待测信号功率
%设置 SNR 的大小，可考虑 0dB 为低信噪比，10dB 为高信噪比情况
snr = 0;%若希望无噪声，则预先设置蒙特卡洛仿真次数为 n=1,设置 snr = Inf
```

```
    sigma = sqrt(P_signal*(10^(-snr/10))); %计算 sigma，方便后续生成高斯噪声
    %预先设置窗函数的类型
    window = hann(L);%汉宁窗
    % window = hamming(L);%哈明窗
    % window = blackman(L);%布莱克曼窗
    % window = blackmanharris(L);%布莱克曼-哈里斯窗
    % window = kaiser(L);%凯赛窗

    %预先设置频率估计 RMS 值的矩阵大小，用 RMS 值代表估计频率大小
    RMS_Dirc = ones(1,length(f));%直接估计算法
    RMS_Rife= ones(1,length(f));%Rife 算法
    RMS_Irife = ones(1,length(f));%Irife 算法
    RMS_AIrife = ones(1,length(f));%AIrife 算法
    RMS_Quinn = ones(1,length(f));%Quinn 算法
    RMS_Quinn_About = ones(1,length(f));%改进 Quinn 算法

    %预先设置频率估计方差的矩阵大小，用方差衡量频率估计的稳定性
    Var_Dirc = ones(1,length(f));%直接估计算法
    Var_Rife= ones(1,length(f));%Rife 算法
    Var_Irife = ones(1,length(f));%Irife 算法
    Var_AIrife = ones(1,length(f));%AIrife 算法
    Var_Quinn = ones(1,length(f));%Quinn 算法
    Var_Quinn_About = ones(1,length(f));%改进 Quinn 算法

    x_I = sin(2*pi*(f0+delta_f/2)*t + pi/2);

    for j = 1:length(f)
    x = ones(n,L);
    for k = 1:n
    %x(k,:) = x_f(j,:) + sigma^2*randn([1,L]); %选用高斯噪声
    x(k,:) = (x_f(j,:) + sigma^2*randn([1,L])).*window';%加窗函数,附加高斯
噪声
        end
        %预先设置好频率估计的矩阵大小
        f_est_Dirc = ones(1,n);
        f_est_Rife = ones(1,n);
        f_est_Irife = ones(1,n);
        f_est_AIrife = ones(1,n);
        f_est_Quinn = ones(1,n);
        f_est_Quinn_About = ones(1,n);
        %运行频率估计算法
        for m = 1:n
            f_est_Dirc(m) = Dirc_esti(x(m,:),t);
            f_est_Rife(m) = Rife_esti(x(m,:),t);
```

```
            f_est_Irife(m) = Irife_esti(x(m,:),t);
            f_est_AIrife(m) = AIrife_esti(x(m,:),t);
            f_est_Quinn(m) = Quinn_esti(x(m,:),t);
            f_est_Quinn_About(m) = Quinn_About_esti(x(m,:),t);
        end
        %计算 RMS 值
        RMS_Dirc(1,j) = sqrt(sum(f_est_Dirc.^2)/n);
        RMS_Rife(1,j) = sqrt(sum(f_est_Rife.^2)/n);
        RMS_Irife(1,j) = sqrt(sum(f_est_Irife.^2)/n);
        RMS_AIrife(1,j) = sqrt(sum(f_est_AIrife.^2)/n);
        RMS_Quinn(1,j) = sqrt(sum(f_est_Quinn.^2)/n);
        RMS_Quinn_About(1,j) = sqrt(sum(f_est_Quinn_About.^2)/n);
        %计算方差
        Var_Dirc(1,j) = var(f_est_Dirc);
        Var_Rife(1,j) = var(f_est_Rife);
        Var_Irife(1,j) = var(f_est_Irife);
        Var_AIrife(1,j) = var(f_est_AIrife);
        Var_Quinn(1,j) = var(f_est_Quinn);
        Var_Quinn_About(1,j) = var(f_est_Quinn_About);
    end
    Var_CRLB = 6*Fs^2/(4*pi^2*10^(snr/10)*L*(L^2-1));%计算频率估计的 CRLB
下界
    %！！！这个是信号幅度、频率、相位均未知时的公式，按理说不适用，所以可考虑不
用


    close all;
    %绘图，频差的仿真图象
    figure(1);
    %具体采用线性坐标还是对数坐标，要看实际的图像适合哪个，可在图窗界面自行设置
选项，以及，曲线的选项同理
    plot(1:length(f),abs(RMS_Dirc-f),'b');
    hold on;
    plot(1:length(f),abs(RMS_Rife-f),'r');
    hold on;
    plot(1:length(f),abs(RMS_Irife-f),'c');
    hold on;
    plot(1:length(f),abs(RMS_AIrife-f),'m');
    hold on;
    plot(1:length(f),abs(RMS_Quinn-f),'g');
    hold on;
    plot(1:length(f),abs(RMS_Quinn_About-f),'k');
    %以下参数是为了方便手动设置图像的部分选项，仅供参考，可直接忽略，后续可在图
窗界面设置
    xticks([1 1+length(f)/2 1+length(f)]);
```

```
    xticklabels({'492Hz','500Hz','508Hz'});
    xlabel('频率/Hz');
    ylabel('频差/Hz');
    title('不同频率下各算法频率估计性能--频差（低信噪比 0dB）');
    legend('直接估计','Rife 算法','I-Rife 算法','A-I-Rife 算法','Quinn 算法','改进 Quinn 算法');
    grid on;
    %绘图，方差的仿真图象
    figure(2);
    %具体采用线性坐标还是对数坐标，要看实际的图像适合哪个，可在图窗界面自行设置选项，以及，曲线的选项同理
    semilogy(1:length(f),Var_Rife,'r');
    hold on;
    semilogy(1:length(f),Var_Irife,'y');
    hold on;
    semilogy(1:length(f),Var_AIrife,'m');
    hold on;
    semilogy(1:length(f),Var_Quinn,'g');
    hold on;
    semilogy(1:length(f),Var_Quinn_About,'k');
    hold on;
    semilogy(1:length(f),Var_CRLB*ones(1,length(f)),'c');
    %以下参数是为了方便手动设置图像的部分选项，仅供参考，可直接忽略，后续可在图窗界面设置
    xticks([1 1+length(f)/2 1+length(f)]);
    xticklabels({'492Hz','500Hz','508Hz'});
    xlabel('频率/Hz');
    ylabel('方差');
    title('不同频率下各算法频率估计性能--方差（低信噪比 0dB）');
    legend('Rife 算法','I-Rife 算法','A-I-Rife 算法','Quinn 算法','改进 Quinn 算法','CRLB 下界'); %直接估计的方差不考虑
    grid on;
```

1.3 X 轴为不同信干比/信噪比的仿真

```
    %x 轴为不同信噪比/信干比的仿真（适用'窗函数类型'、'信干比'、'噪声类型'、'信噪比'等基于信噪比/信干比的仿真）
    f = f0 + delta_f/4; %选择距选定的落于采样点的频点 1/4 个采样间隔的频点仿真，也可设置其他
    % 这里，考虑靠近两谱线中心，各算法差距不大，落于谱线上，各算法相比直接估计性能不能得到明显提升，因此选用 1/4 个采样间隔点
    x_f = A*sin(2*pi*f*t + phi_0); %生成待测信号
```

```
    %x_I = sin(2*pi*(f+delta_f)*t + pi/2); %干扰信号，此处示例为设置干扰信号
距离待测频率一个采样间隔。
    snr = 0:0.2:10;%更改待信噪比 SNR 的范围，所示为从 0dB 到 10dB，以 0.2dB 为单
位等间距取点
    sigma = ones(size(snr));%预先设置 sigma 矩阵的大小
    P_signal = 0.5*A^2;%待测信号功率
    for j = 1:length(snr) %通过 SNR 计算 sigma 值，以便于后续生成噪声。
        sigma(j) = sqrt(P_signal*(10^(-snr(j)/10)));
    end
    %预先设置窗函数的类型
    % window = hann(L);%汉宁窗
    % window = hamming(L);%哈明窗
    % window = blackman(L);%布莱克曼窗
    % window = blackmanharris(L);%布莱克曼-哈里斯窗
    % window = kaiser(L);%凯赛窗
    %预先设置频率估计 RMS 值的矩阵大小，用 RMS 值代表估计频率大小
    RMS_Dirc = ones(1,length(snr));%直接估计算法
    RMS_Rife= ones(1,length(snr));%Rife 算法
    RMS_Irife = ones(1,length(snr));%Irife 算法
    RMS_AIrife = ones(1,length(snr));%AIrife 算法
    RMS_Quinn = ones(1,length(snr));%Quinn 算法
    RMS_Quinn_About = ones(1,length(snr));%改进 Quinn 算法
    %预先设置频率估计方差的矩阵大小，用方差衡量频率估计的稳定性
    Var_Dirc = ones(1,length(snr));%直接估计算法
    Var_Rife= ones(1,length(snr));%Rife 算法
    Var_Irife = ones(1,length(snr));%Irife 算法
    Var_AIrife = ones(1,length(snr));%AIrife 算法
    Var_Quinn = ones(1,length(snr));%Quinn 算法
    Var_Quinn_About = ones(1,length(snr));%改进 Quinn 算法
    for j = 1:length(sigma)
        x = ones(n,L);
        for k = 1:n
            %选用不同的噪声类型，已预先设置好，使得 snr = -20*log10(sigma)
            x(k,:) = x_f + sigma(j)^2*randn([1,L]);%高斯噪声
            %x(k,:) = x_f + sqrt(0.5)*sigma(j)^2*raylrnd(1,[1,L]); %瑞利噪
声
            %x(k,:)                =                x_f                +
sqrt(0.5)*sigma(j)^2*sqrt(12)*(rand([1,L])-0.5); %平均分布噪声
            %x(k,:)                =                x_f                +
sqrt(0.5)*sigma(j)^2*random('Poisson',1,[1,L]); %泊松分布噪声

            %x(k,:) = (x_f + sigma(j)^2*randn([1,L])).*window';%加窗函数，
附加高斯噪声
            %x(k,:) = x_f + sqrt(2)*sigma(j)*x_I+randn([1,L]); %加干扰信号，
```

```
附加 0dB 高斯白噪声
            %同理，已预先设置好使得 SIR = 20*log10(sigma)
        end
        %预先设置好频率估计的矩阵大小
        f_est_Dirc = ones(1,n);
        f_est_Rife = ones(1,n);
        f_est_Irife = ones(1,n);
        f_est_AIrife = ones(1,n);
        f_est_Quinn = ones(1,n);
        f_est_Quinn_About = ones(1,n);
        %运行频率估计算法
        for m = 1:n
            f_est_Dirc(m) = Dirc_esti(x(m,:),t);
            f_est_Rife(m) = Rife_esti(x(m,:),t);
            f_est_Irife(m) = Irife_esti(x(m,:),t);
            f_est_AIrife(m) = AIrife_esti(x(m,:),t);
            f_est_Quinn(m) = Quinn_esti(x(m,:),t);
            f_est_Quinn_About(m) = Quinn_About_esti(x(m,:),t);
        end
        %计算 RMS 值
        RMS_Dirc(1,j) = sqrt(sum(f_est_Dirc.^2)/n);
        RMS_Rife(1,j) = sqrt(sum(f_est_Rife.^2)/n);
        RMS_Irife(1,j) = sqrt(sum(f_est_Irife.^2)/n);
        RMS_AIrife(1,j) = sqrt(sum(f_est_AIrife.^2)/n);
        RMS_Quinn(1,j) = sqrt(sum(f_est_Quinn.^2)/n);
        RMS_Quinn_About(1,j) = sqrt(sum(f_est_Quinn_About.^2)/n);
        %计算方差
        Var_Dirc(1,j) = var(f_est_Dirc);
        Var_Rife(1,j) = var(f_est_Rife);
        Var_Irife(1,j) = var(f_est_Irife);
        Var_AIrife(1,j) = var(f_est_AIrife);
        Var_Quinn(1,j) = var(f_est_Quinn);
        Var_Quinn_About(1,j) = var(f_est_Quinn_About);
    end
    %绘图，RMS 值的仿真图象
    close all;
    figure(3);
    %具体采用线性坐标还是对数坐标，要看实际的图像适合哪个，可在图窗界面自行设置
选项，以及，曲线的选项同理
    semilogy(1:length(snr),abs(RMS_Dirc-f),'b');
    hold on;
    semilogy(1:length(snr),abs(RMS_Rife-f),'r');
    hold on;
    semilogy(1:length(snr),abs(RMS_Irife-f),'c');
```

```
    hold on;
    semilogy(1:length(snr),abs(RMS_AIrife-f),'m');
    hold on;
    semilogy(1:length(snr),abs(RMS_Quinn-f),'g');
    hold on;
    semilogy(1:length(snr),abs(RMS_Quinn_About-f),'k');
    %以下参数是为了方便手动设置图像的部分选项，仅供参考，可直接忽略，后续可在图窗界面设置
    xticks([1 1+length(snr)/2 1+length(snr)]);
    xticklabels({'0dB','5dB','10dB'});
    xlabel('snr/dB');
    ylabel('频差/Hz');
    title('不同信噪比下算法频率估计性能--RMS 值');
    legend('Rife 算法','I-Rife 算法','A-I-Rife 算法','Quinn 算法','改进 Quinn 算法');
    grid on;
```

1.4 方差绘图

```
    %绘图，方差的仿真图象
    figure(4);
    %具体采用线性坐标还是对数坐标，要看实际的图像适合哪个，可在图窗界面自行设置选项，以及，曲线的选项同理
    semilogy(1:length(snr),Var_Rife,'r');
    hold on;
    semilogy(1:length(snr),Var_Irife,'c');
    hold on;
    semilogy(1:length(snr),Var_AIrife,'m');
    hold on;
    semilogy(1:length(snr),Var_Quinn,'g');
    hold on;
    semilogy(1:length(snr),Var_Quinn_About,'k');
    %以下参数是为了方便手动设置图像的部分选项，仅供参考，可直接忽略，后续可在图窗界面设置
    xticks([1 1+length(snr)/2 1+length(snr)]);
    xticklabels({'0dB','5dB','10dB'});
    xlabel('snr/dB');
    ylabel('方差');
    title('不同信噪比下算法频率估计性能--方差');
    legend('Rife 算法','I-Rife 算法','A-I-Rife 算法','Quinn 算法','改进 Quinn 算法');  %直接估计的方差不考虑
    grid on;
```

2. DFT 函数

```
function [X,P1,P2] = DFT(x,t)
    % 计算 DFT
    L = length(t); %计算信号长度
    X = fft(x); %fft
    P2 = abs(X/L); %计算谱线对应幅度大小
    P1 = P2(1:L/2+1); %由于对称，取前半
    P1(2:end-1) = 2*P1(2:end-1); %由于折叠，2 倍
end
```

3. 直接估计算法

```
function [esti_freq, P1]= Dirc_esti(x,t)%esti_freq 为估计出的频率，P1 为 DFT 频谱，x 为待测信号，t 为时间信号
     L = length(t); %计算信号长度
    Fs = (L-1)/(t(L)-t(1)); %计算采样频率
    [~,P1,~]  = DFT(x,t); %做 FFT，得出 DFT 频谱
    [~, idx] = max(P1); %搜索找到最大谱线位置

    % 修正索引，如果峰值在第一个或最后一个点
    if idx == 1
        idx = 2;
    elseif idx == L/2+1
        idx = L/2;
    end

    esti_freq = (idx-1)*Fs/L; %直接估计得出的频率，考虑 matlab 向量起点为 1，idx 要-1
end
```

4. Rife,改进算法及其相关函数

1.1 Rife

```
function [esti_freq, P1]  = Rife_esti(x,t)%esti_freq 为估计出的频率，P1 为 DFT 频谱，freq 为细化后 DFT 频谱，x 为待测信号，t 为时间信号
    L = length(t); %计算信号长度
    Fs = (L-1)/(t(L)-t(1)); %计算采样频率
    [~,P1,~]  = DFT(x,t); %做 FFT，得出 DFT 频谱
    [~, idx] = max(P1); %搜索找到最大谱线位置

    % 修正索引，如果峰值在第一个或最后一个点
    if idx == 1
        idx = 2;
```

```
        elseif idx == L/2+1
            idx = L/2;
        end

        % Rife 算法
        if P1(idx+1)<=P1(idx-1) %计算得到频率修正方向
            r = -1;
        else
            r = 1;
        end
        delta = P1(idx+r)/(P1(idx)+P1(idx+r));%delta 为频率修正项
        correction = r*delta*Fs/L;

        % 估计的频率
        esti_freq = (idx-1)*Fs/L + correction; %考虑 matlab 向量起点为 1，idx
要-1
    end
```

1.2 I-Rife

```
    function [esti_freq, P1, freq]  = Irife_esti(x,t)%esti_freq 为估计出的
频率，P1 为 DFT 频谱，freq 为细化后 DFT 频谱，x 为待测信号，t 为时间信号
        L = length(t); %计算信号长度
        Fs = (L-1)/(t(L)-t(1)); %计算采样频率
        [~,P1,~]  = DFT(x,t); %做 FFT，得出 DFT 频谱
        [~, idx] = max(P1); %搜索找到最大谱线位置

        % 修正索引，如果峰值在第一个或最后一个点
        if idx == 1
            idx = 2;
        elseif idx == L/2+1
            idx = L/2;
        end

        fe = (idx-1)*Fs/L; %直接估计得出的频率，考虑 matlab 向量起点为 1，idx
要-1

        % 简单的频谱细化技术（线性插值）（较简单）
        %alpha=simple_zfft(idx,L,P1);

        % 通过 Zoom-FFT 算法细化 FFT(已预先设置好参数,是的能取到 idx±0.5 的频点)
        [y,freq] = zoom_fft(x,fe,Fs);

        %以下是 I-Rife 算法
```

```
        [~,q] = find(freq == fe); %确定细化后频谱原最大谱线的位置

        if abs(y(q+1)) > abs(y(q-1)) %计算频率修正方向
            a = 1;
        else
            a = -1;
        end

        delta_k = 0.5 - abs(y(q+2*a))/(abs(y(q))+abs(y(q+2*a))); %计算频率修正因子

        %单独计算频移后的两谱线
        i = sqrt(-1);%虚数 i，防止冲突
        X1 = 0; %初始值
        for l = 1:L
           X1 = X1 + x(l)*exp(-i*(l-1)*2*pi*(idx-a*delta_k+a-1)/L); %考虑 matlab 向量起点为 1，idx 要-1
        end
        X2 = 0; %初始值
        for l = 1:L
           X2 = X2 + x(l)*exp(-i*(l-1)*2*pi*(idx-a*delta_k-1)/L); %考虑 matlab 向量起点为 1，idx 要-1
        end

        alpha = a*abs(X1)/(abs(X2)+abs(X1))-a*delta_k; %频率修正项

        esti_freq = fe + alpha*Fs/L;

    end
```

1.3 A-I-Rife

```
    function [esti_freq, P1, freq]  = AIrife_esti(x,t)%esti_freq 为估计出的频率，P1 为 DFT 频谱，freq 为细化后 DFT 频谱，x 为待测信号，t 为时间信号
        L = length(t); %计算信号长度
        Fs = (L-1)/(t(L)-t(1)); %计算采样频率
        [~,P1,~]  = DFT(x,t); %做 FFT，得出 DFT 频谱
        [~, idx] = max(P1); %搜索找到最大谱线位置

        % 修正索引，如果峰值在第一个或最后一个点
        if idx == 1
            idx = 2;
        elseif idx == L/2+1
            idx = L/2;
```

```
        end

        fe = (idx-1)*Fs/L; %直接估计得出的频率，考虑 matlab 向量起点为 1，idx
要-1
        % 简单的频谱细化技术（线性插值）（较简单）
        %alpha=simple_zfft(idx,L,P1);

        % 通过 Zoom-FFT 算法细化 FFT(已预先设置好参数,是的能取到 idx±0.5 的频点)
        [y,freq] = zoom_fft(x,fe,Fs);

        %以下是 A-I-Rife 算法
        [~,q] = find(freq == fe); %确定细化后频谱原最大谱线的位置

        if abs(y(q+1)) > abs(y(q-1))%计算频率修正方向
            a = 1;
        else
            a = -1;
        end

        if abs(y(q))<=abs(y(q+a)) %在 I-rifed 的基础上做出的改进，即分情况计
算频率修正因子
            delta_k = 0.5 - abs(y(q+2*a))/(abs(y(q))+abs(y(q+2*a)));
        else
            delta_k = abs(y(q-a))/(abs(y(q-a))+abs(y(q+a)));
        end

        %单独计算频移后的两谱线
        i = sqrt(-1);%虚数 i，防止冲突
        X1 = 0; %初始值
        for l = 1:L
           X1 = X1 + x(l)*exp(-i*(l-1)*2*pi*(idx-a*delta_k+a-1)/L); %考虑
matlab 向量起点为 1，idx 要-1
        end
        X2 = 0; %初始值
        for l = 1:L
           X2 = X2 + x(l)*exp(-i*(l-1)*2*pi*(idx-a*delta_k-1)/L); %考虑
matlab 向量起点为 1，idx 要-1
        end

        alpha = a*abs(X1)/(abs(X2)+abs(X1))-a*delta_k; %频率修正项

        esti_freq = fe + alpha*Fs/L;

    end
```

1.4 简单插值函数

```
function alpha=simple_zfft(idx,L,P)%简单插值细化 FFT，用于改进 Rife 算法
    if idx > 1 && idx <= L/2
        leftidx = idx - 1;
        rightidx = idx + 1;
        % 确保索引在有效范围内
        if leftidx < 1
            leftidx = idx;
        end
        if rightidx > L/2
            rightidx = idx;
        end

        alpha                                                    =
(P(rightidx)-P(leftidx))/(2*P(idx)-(P(leftidx)+P(rightidx))/2); % 计算插
值系数
    else
        alpha = 0;
    end
end
```

1.5 频谱细化函数

```
%%待完善
function [y,freq] = zoom_fft(x,fe,Fs)%细化 FFT，用于改进 Rife 算法
    %x 被测信号，fe 是细化区间的中心频率，m 是细化倍数
    m = 16;
    L=length(x);           % 计算读入数据长度
    L_fft = 2*L/m;
    fi=fe-Fs/m/2;           % 计算细化截止频率下限
    fa=fi+Fs/m;             % 计算细化截止频率上限
    La=round(0.5*L/m+1); % 确定低通滤波器截止频率对应的谱线条数
    % 频移
    idx=0: L-1;             % 序列索引号
    b=idx*pi* (fi+fa)/Fs;     % 设置单位旋转因子
    y=x.*exp(-1i*b);        % 进行频移
    X= fft(y, L);          % FFT
    % 低通滤波和下采样
    a(1: La) =X(1: La);     % 取正频率部分的低频成分
    a(L-La+2 : L) =X(L-La+2 : L); % 取负频率部分的低频成分
    X=ifft(a, L);
    c=X(1 : m: L);         % 下采样
    % 求细化频谱
    y=fft(c, L_fft) * 2/L_fft; % 再一次 FFT
```

```
    y=fftshift(y);          % 重新排列
    freq=fi+(0:L_fft-1)*Fs/m/L_fft; % 频率设置
end
```

5. Quinn 及其改进算法

1.1Quinn 算法

```
function [esti_freq, P1] = Quinn_esti(x,t)%esti_freq 为估计出的频率，P1
为 DFT 频谱，x 为待测信号，t 为时间信号
    L = length(t); %计算信号长度
    Fs = (L-1)/(t(L)-t(1)); %计算采样频率
    [X,P1,~] = DFT(x,t); %做 FFT，得出 DFT 频谱
    [~,idx] = max(P1); %搜索找到最大谱线位置

    % 修正索引，如果峰值在第一个或最后一个点
    if idx == 1
        idx = 2;
    elseif idx == L/2+1
        idx = L/2;
    end

    %以下是 Quinn 算法
    beta1 = real(X(idx-1)/X(idx)); %加入相位信息，取实部
    beta2 = real(X(idx+1)/X(idx));
    delta1 = beta1/(1-beta1);
    delta2 = beta2/(beta2-1);
    if delta1>0 && delta2>0 %beta 为频率修正项
        delta = delta2;
    else
        delta = delta1;
    end
    esti_freq = (idx-1)*Fs/L+delta*Fs/L; %考虑 matlab 向量起点为 1，idx
要-1
end
```

1.2 Aboutanios 算法

```
function [esti_freq, P1] = Quinn_About_esti(x,t)%esti_freq 为估计出的
频率，P1 为 DFT 频谱，x 为待测信号，t 为时间信号
    L = length(t); %计算信号长度
    Fs = (L-1)/(t(L)-t(1)); %计算采样频率
    [X,P1,~] = DFT(x,t); %做 FFT，得出 DFT 频谱
    [~,idx] = max(P1); %搜索找到最大谱线位置

    % 修正索引，如果峰值在第一个或最后一个点
```

```
        if idx == 1
            idx = 2;
        elseif idx == L/2+1
            idx = L/2;
        end

        %Quinn 算法
        beta1 = real(X(idx-1)/X(idx)); %加入相位信息，取实部
        beta2 = real(X(idx+1)/X(idx));
        delta1 = beta1/(1-beta1);
        delta2 = beta2/(beta2-1);
        if delta1>0 && delta2>0 %beta 为频率修正项
            delta = delta2;
        else
            delta = delta1;
        end
        %改进部分
        %单独计算频移后的三谱线
        i = sqrt(-1); %虚数 i，防止冲突
        X05 = 0;
        for l = 1:L
           X05 = X05 + x(l)*exp(-i*(l-1)*2*pi*(idx-1+0.5)/L); %考虑 matlab
向量起点为 1，idx 要-1
        end
        X_05 = 0;
        for l = 1:L
           X_05 = X_05 + x(l)*exp(-i*(l-1)*2*pi*(idx-0.5-1)/L); %考虑 matlab
向量起点为 1，idx 要-1
        end
        X0 = 0;
        for l = 1:L
           X0 = X0 + x(l)*exp(-i*(l-1)*2*pi*(idx\-1)/L); %考虑 matlab 向量
起点为 1，idx 要-1
        end
        a = abs(X_05)/abs(X0);
        b = abs(X05)/abs(X0);
        if a>1 || b>1 %判断频率修正项的取值
            delta = 0.5*real((X05+X_05)/(X05-X_05));
        end
        esti_freq = (idx-1)*Fs/L+delta*Fs/L; %考虑 matlab 向量起点为 1，idx
要-1
    end
```