修士学位論文

日立局・高萩局を結合した
二素子干渉計の立ち上げ

2015年度
（平成27年度）

茨城大学大学院理工学研究科

理学専攻

14NM155L
加古琳一

概 要

茨城大学では、国立天文台茨城観測局内にある 2 台の 32m 電波望遠鏡を運用している。両局ともに 6-9GHz 帯、22GHz 帯受信機が導入されているが、両局を結合した干渉計観測は行われていない。この 2 局を結合した二素子干渉計として立ち上げることで、高感度な観測が実現できる。若い星や活動銀河核などを高感度モニター観測することは、メーザーの強度変動現象やジェットの仕組みの解明に貢献すると考えられる。

本研究は、この 2 台の電波望遠鏡を結合した二素子干渉計の立ち上げを行った。具体的には、以前より利用されていた K5/VSSP32 システムと、2014 年度末に導入されたハードウェア相関器での試験観測を行った。

まず初めに、すでに導入されているシステムである K5/VSSP32 と相関処理ソフトウェアを用いたシステムでの干渉計観測を行った。その結果、フリンジの検出には成功したものの、相関強度とフリンジ位相に周期的な変動が見られた。また、メーザーでは、クロススペクトルの位相が互いに 180 度離れた 2 つの成分に分裂し、強度も周波数方向に広がった成分と狭い成分が重なっていた。周期変動の周期が天体の位置に依存していた。相関処理に用いる三角関数の値を少ないビット数で表している影響が、短い基線長での観測で顕著に現れる現象であるとみられる。両局の局部発振器の周波数をずらすと変動が平均化されるため、観測への影響を抑えることは可能である。一方、クロススペクトルの異常は、プログラムの計算処理に起因していた。現在は細い線スペクトルを処理するためのオプションが追加されている。

次に、新しく導入された結合型ハードウェア相関器での観測を行った。フリンジの検出に成功したものの、以下の問題点が見つかった。

- フリンジ位相が時間とともに変化していく
- 6GHz 帯の 1800ch 付近に雑音の混入
- 6GHz 帯の不規則な強度変動

このうち、フリンジ位相の問題は内部処理の不具合で、ファームウェアの更新により解消された。1800ch 付近の雑音は、帯域外の人工的な雑音が混入していることがわかった。また、この雑音の影響により、不規則な強度変動が発生していた。これらの問題は、必要な帯域のみを通すバンドパスフィルターを追加することで改善が見られた。積分時間の短い観測では、ハードウェア相関器でも相関強度と位相に周期的な変動が見られた。局部発振器の周波数をずらした観測はまだ行っていない。

ハードウェア相関器ではアラン分散の測定も行った。また、同時に計算用のプログラムも作成した。仰角が 85 度、30 度、15 度と電波吸収体で約 8 時間の観測をし、システムの連続波に対する安定時間は 30 秒程度であった。この値は単一鏡のアラン分散よりもよく、連続波源の観測は、単一鏡より干渉計での観測が適しているといえる。

目 次

1　電波天文学について

1.1　電波天文学

宇宙では様々な波長の電磁波が飛び交っており、現在はすべての波長帯で観測が行われている。そのなかでも宇宙からの電波を観測し、宇宙の仕組みを解明する学問を電波天文学という。宇宙観測における電波には、ほかの波長と比べ、以下のような特徴がある。

1. 他波長では見えないものが観測できる。特に極低温の分子ガスや高密度領域からのメーザー放射など、星形成過程を調べるためには、電波による観測が必須である。

2. 波長が長く、ダストによる吸収を受けにくい。そのため、銀河面の深くまで見ることができる。

3. 地上で観測できる (図 1.1)。したがって観測装置の大型化が可能となり、精度の高い観測を行うことができる。

4. 電気的に干渉技術が容易である。そのため複数のアンテナ間で電波を干渉させることができる。

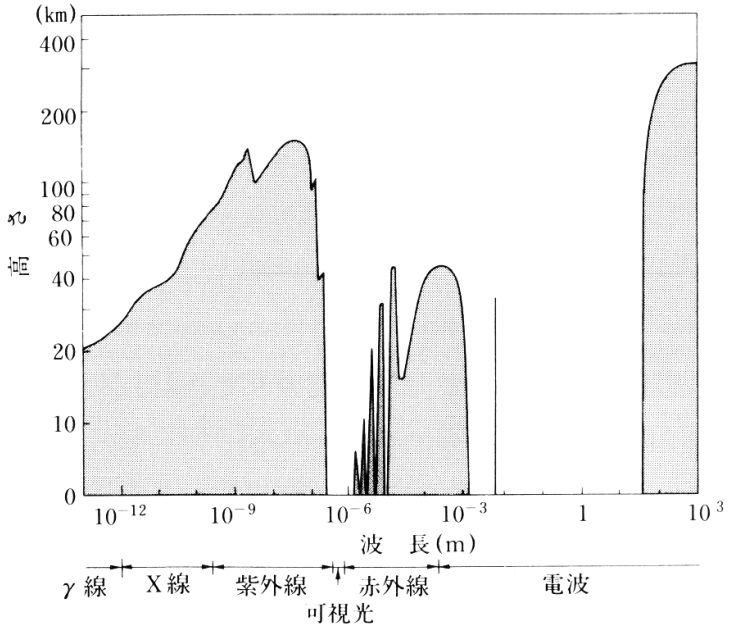

図 1.1: 電磁波の地球大気の透過度。宇宙からの信号の強度が半分になる高度を示す。(尾崎洋二『宇宙科学入門』[2]、5 ページ)

1.2 電波望遠鏡

電波望遠鏡は大きく分けてアンテナ、フロントエンド、バックエンド（検波器または分光器）、そしてそれらを制御しデータ収集をする計算機によって構成される。

アンテナは天体からの電波を集める役割を担う。微弱な電波を効率よく受信するためには指向精度や鏡面精度などが求められる。これらは重力や熱、風などによる変形に左右される。また、感度や角度分解能はアンテナの口径に大きく左右される。一般的に口径が大きいほど感度はよくなり、角度分解能は高くなる。

フロントエンドは天体からの電波を、増幅と周波数変換を行い扱いやすい電気信号に変換する。天体からの信号は微弱なため、装置雑音が低いことが要求される。

バックエンドは、入力された電波情報を電力信号として取り出す。検波器は電波全体の強さを、分光器は周波数ごとの電波の強度を出力する。

1.3 干渉計型望遠鏡

干渉計とは、複数の電波望遠鏡で得た電波信号間の相互相関を相関器でとる観測装置である。各アンテナで得られた信号を相関することで、天体の強度分布などが得られる。

干渉計は大きく分けて、結合型干渉計と超長基線干渉計（VLBI）の 2 つのタイプがある。結合型干渉計は、アンテナ同士が光ケーブルなどで接続されたものを指す。共通信号を周波数標準から各アンテナへケーブルで分配し、観測した受信信号もケーブルで相関器へ入力する（図 1.2）。主な結合型干渉計に、VLA や ALMA がある。これらは多くの素子アンテナを有し、有効面積を広げることで高感度観測を実現している。

VLBI は各アンテナでの受信電圧信号を保存し、後日データを集めてから相関処理を行う。主にアンテナ同士がケーブルで接続できないほど離れているときに利用される。主な超長基線干渉計に VLBA や VERA がある。これらは 1000km を優に超える基線長で、非常に高い角度分解能を実現している。また、人工衛星として打ち上げられたアンテナとの干渉計観測も成功している。

1.4 干渉計の原理

2 つの電波望遠鏡からなる二素子干渉計を考える（図 1.3）。両望遠鏡が同じ天体を同時に観測したとき、同じ波面が到達する時刻に差が生じる。この時間差を遅延時間といい、波面がアンテナに届くまでの時間差を幾何学的遅延 τ_g という。ここで、両局での受信信号を考える。右側の望遠鏡は τ_g だけ早く波面が到達しているので、受信信号はそれぞれ複素数で

$$V_1(t) = V_1 e^{i(2\pi\nu t+\phi)} \tag{1.1a}$$

$$V_2(t) = V_2 e^{i(2\pi\nu(t-\tau_g)+\phi)} \tag{1.1b}$$

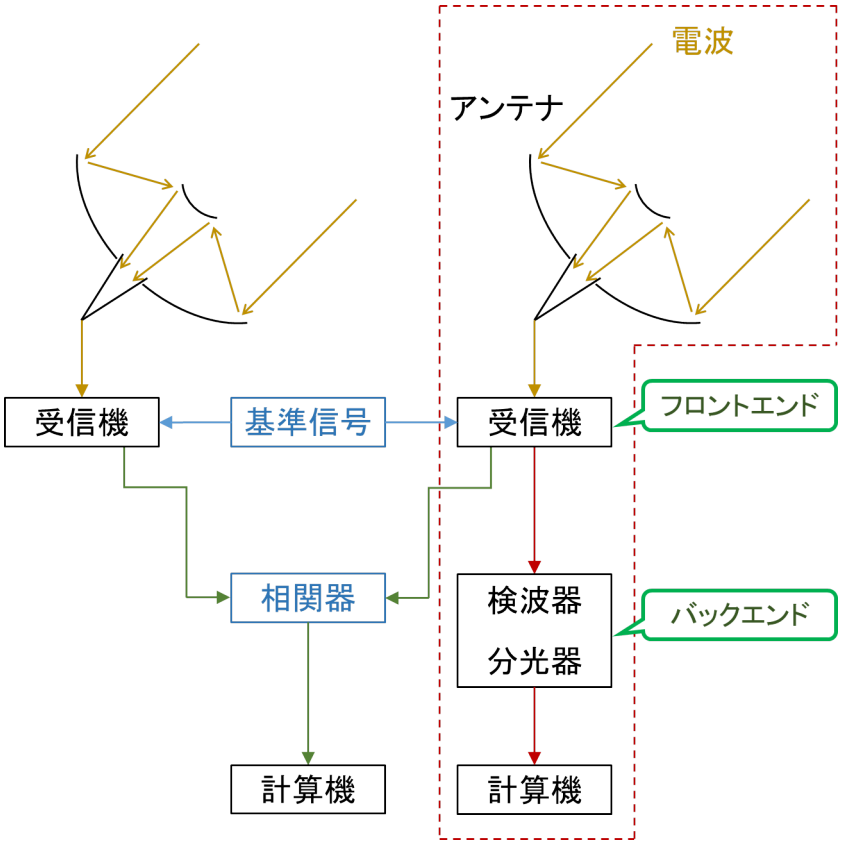

図 1.2: 結合型干渉計の構成。基準信号は共通のものが用いられる。また、点線で囲った部分は単一鏡の構成を表している。

となる。また、これを用いて相互相関関数 $C_{1,2}(\tau)$ を定義する。

$$C_{1,2}(\tau) = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} V_1(t) V_2^*(t+\tau) dt \tag{1.2}$$

また、これをフーリエ変換した $\hat{C}_{1,2}$ をクロススペクトルと呼ぶ。

$$\hat{C}_{1,2}(\nu) = \int_{-\infty}^{\infty} C_{1,2}(\tau) e^{-2\pi i \nu \tau} d\tau \tag{1.3}$$

なお、各局の信号を先にフーリエ変換をしてスペクトルを求め、スペクトル同士を掛け合わせてもクロススペクトルを求めることができる。相関器からは、相互相関関数かクロススペクトルが出力される。あるアンテナが捉える大気放射などの雑音は、他のアンテナ起源の信号や雑音とは相関しないので、相互相関関数やクロススペクトルを求める際に取り除くことができる。

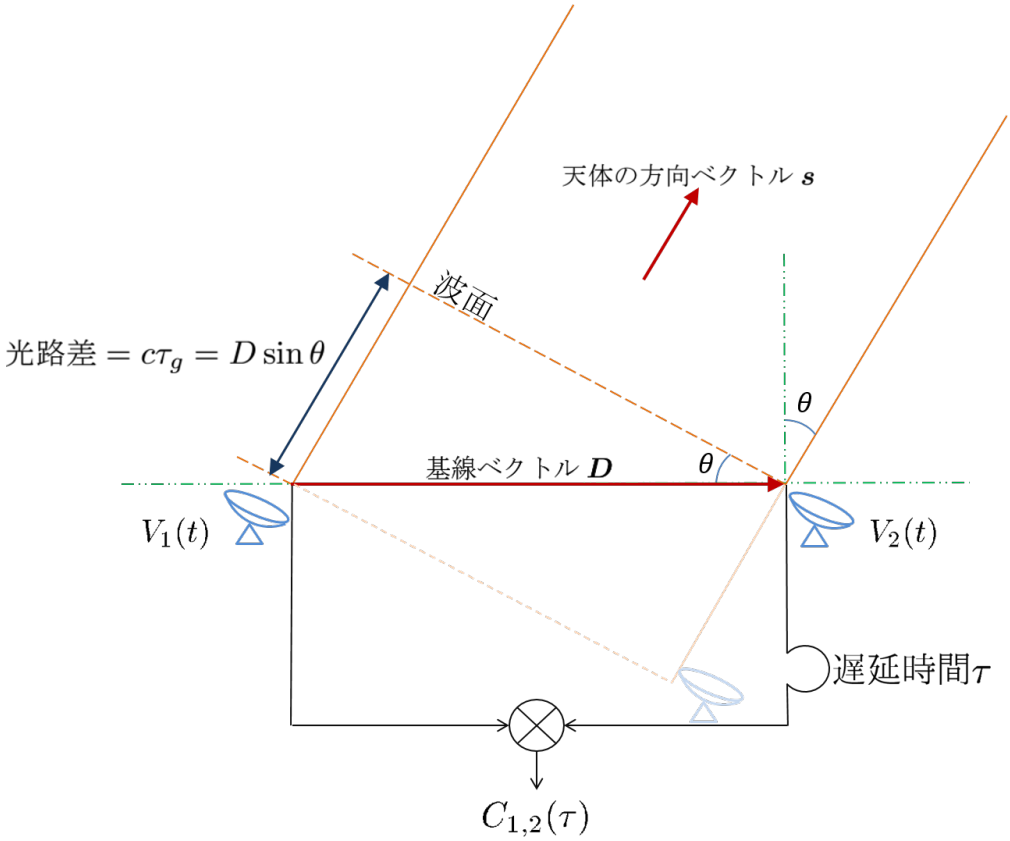

図 1.3: 二素子干渉計の概略図。

1.5 単一鏡と干渉計の比較

単一鏡による観測と干渉計の観測について、利点と欠点を簡単に述べる。

単一鏡の利点

- 空間的に広がった天体の観測が可能。
- 全強度の測定や絶対強度較正が容易。

単一鏡の欠点

- 角度分解能に限界がある（分解能はアンテナ口径に反比例する）。口径を大きくするほど、鏡面を高精度に保つのが難しくなる。
- 特に連続波観測で、大気の影響を大きく受ける。大気の影響を取り除くためにOFF 点を観測する必要がある。

干渉計の利点

- 大気放射はアンテナ間で相関しないので、その影響を受けにくい。
- 高い角度分解能が実現できる。分解能は基線長（アンテナ間の距離）に反比例する。

干渉計の欠点

- 視野が狭く、空間的に広がった天体の観測が苦手。
- 単一鏡と比べ、装置やデータ処理が複雑。

1.6 本研究の目的

日立局、高萩局どちらにも 6-9GHz 帯受信機と 22GHz 帯受信機が導入されている。しかし、この 2 局を結合しての干渉計観測は行われていない。若い星や活動銀河核などの連続波の強度の高感度な測定は単一鏡では難しい。これらを干渉計でモニター観測することは、メーザーの強度変動現象やジェットの仕組みの解明に貢献すると考えられる。また、VERA などの大規模な観測へのフィードバックも期待される。

本研究の目的は、日立局と高萩局を結合した二素子干渉計を立ち上げ、連続波の高感度観測を実現することである。そのために、K5/VSSP32 システムと、新たに導入されたハードウェア相関器で試験観測を行った。また、ハードウェア相関器ではアラン分散の測定も行った。本論文ではそれらの結果を報告する。

2　茨城局の干渉計システム

2.1　茨城局 32m 望遠鏡

茨城局は、日立市と高萩市に位置している 2 台の口径 32m 電波望遠鏡を擁する観測所である。KDDI から国立天文台に譲渡されたパラボラアンテナが電波望遠鏡として立ち上げられた。日立市にあるものを日立局、高萩市のものを高萩局という。

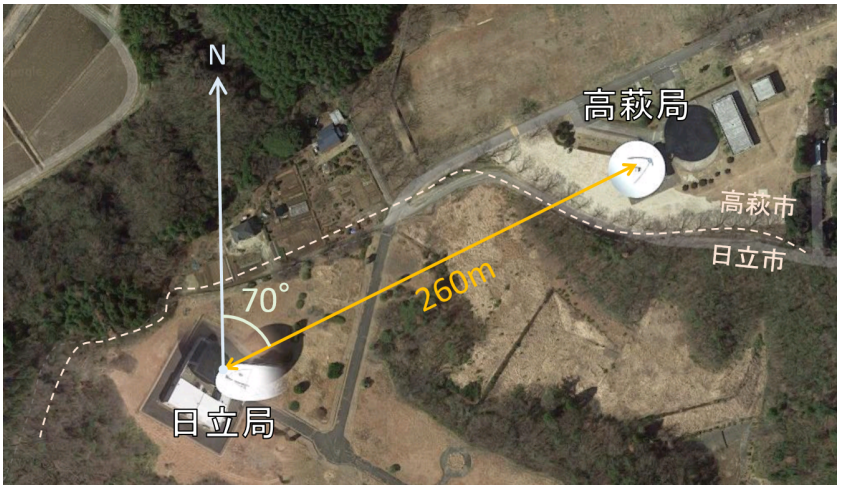

図 2.1: 茨城局望遠鏡の位置関係。（衛星画像は google Earth より。©2016 Google）

表 2.1: 日立局と高萩局の位置 [11]

	日立局	高萩局
東経	140°41′44″	140°41′54″
北緯	36°41′40″	36°41′43″
標高	60m	77m
距離	約 260m	

望遠鏡は互いに約 260m 離れており、高萩局は日立局から見て方位角約 70 度にある。天体を観測するときの典型的な遅延時間は 10^{-7}sec、遅延時間変化率は 10^{-11}sec/sec である。両局間は坑道でつながっており、光ファイバケーブルが敷設されている。信号はそのケーブルを利用してどちらかの局に集約される。この転送に伴う遅延は約 2.45μsec である。周波数標準には水素メーザーが使われている。両局ともに 6-9GHz 帯受信機と 22GHz 帯受信機が導入されている。（受信機の性能については滝沢 2011 年度修士論文 [6]、田中 2011 年度修士論文 [7]、森 2013 年度修士論文 [8] を参照）

2.2 K5/VSSP32 を利用したソフトウェア相関器システム

このシステムは、情報通信研究機構（NICT）が開発した、K5/VSSP32 サンプラーと相関処理ソフトウェア群を利用したもので、測地 VLBI 観測などに利用されている。この干渉計はデータを一度保存し、後日ソフトウェア上で再生しながら相関処理を行う。ソフトウェア上では、時系列データをフーリエ変換し（F)、遅延補正をして掛け合わせて（X) クロススペクトルを求める。このタイプの相関システムを FX 相関器という [*1]。なお、掛け合わせた後にフーリエ逆変換をし、時間方向のデータ列に戻してから出力している。相関処理時にフリンジサーチの範囲や単位積分時間などを設定できる。

このシステムでは生データは別に保存されているため、積分時間やラグ幅などを変えて何度でも相関処理をすることが出来る。また、ラグ幅はフーリエ変換の点数、つまり分光点と関係する。そのため、分光点を自由に変更することもできる。ただし、観測から相関結果を得るまでにタイムラグがある。

K5/VSSP32 の主なスペックについて、表 2.2 にまとめた。なお、その利便性から、K5/VSSP32 は単一鏡観測でも利用され、単一鏡観測用のソフトウェアもある [7][9]。

表 2.2: K5/VSSP32 の主なスペック [9]

サンプリング周波数 [MHz]	0.04, 0.1, 0.2, 0.5, 1, 2, 4, 8, 16, 32, 64
内蔵デジタルローパスフィルター [MHz]	2, 4, 8, 16, through
AD 変換器アナログ周波数 [MHz]	300
AD 分解能 [bit]	1, 2, 4, 8
分光点数	2048, 4096, 8192, 16384, 32768, 2097152 [*2]
ボードの最大データレート [MHz]	256
PC とのインターフェース	USB 2.0

2.2.1 相関処理の流れ

K5/VSSP32 には、専用の相関処理ソフトウェア群が用意されている。このシステムでの相関処理の流れを図 2.2 に示す。また、このソフトウェア群の各プログラムによるデータ処理の流れを以下に示す。

1. apri_calc で遅延時間と遅延時間変化率を計算する。
2. K5/VSSP32 で記録された生データを fx_cor で相関処理する。
3. sdelay でフリンジサーチやクロススペクトルの出力を行う。

[*1] 掛け合わせてからフーリエ変換をしてクロススペクトルを求める相関器を XF 相関器という。

[*2] K5/VSSP32 自体の分光点数は 2K 点と 2M 点から選択できる。2M 点で観測した場合、専用ソフトウェアにて 32K 点以下に変更することができる [7]。なお、本文で述べたように、干渉計観測では分光点数は相関処理時に指定できる。

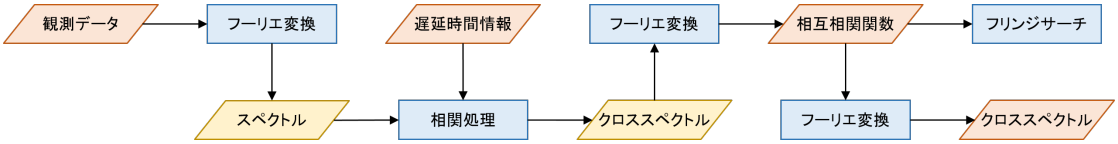

図 2.2: K5/VSSP32 での相関処理の流れ図。青四角は計算処理を、平行四辺形はデータを表す。このうち黄色のものは処理中の一時データである。また、時間方向のデータを上段、周波数方向のものは下段に配置している。

2.2.2 ソフトウェアの詳細

プログラムの各機能について説明をする。

apri_calc

観測指示書（skd ファイル）から観測時間、アンテナ座標、観測天体の座標を読み取り、遅延時間や遅延時間変化率などを計算する。出力されるファイルには生データの絶対パスなども記述されるが、場合によっては観測データ名の部分を書き替える必要がある [*3]。

fx_cor

apri_calc で出力された予測値ファイル（ape ファイル）から遅延時間などの情報とデータのパスを取得し、相互相関関数を計算しファイル出力する。実行時に総積分時間、単位積分時間、ラグ幅などを指定できる。引数に予測値ファイルではなく生データを指定すると、自己相関関数を計算する。

sdelay

fx_cor で出力したファイルをもとに、相互相関関数が最大となる遅延時間と遅延時間変化率の残差を求める。この作業をフリンジサーチという。また、単位積分ごとの相関強度と位相のプロットや、クロススペクトルのプロットを出力する。

2.3 結合型ハードウェア相関器

Elecs 社製の結合型ハードウェア相関器が 2014 年度末に導入された。この相関器は、各局で得たデータをフーリエ変換してスペクトルを求めてから掛け合わせクロススペクトルを求める FX 型相関器である。主なスペックを表 2.3 に示す。このシステムはサンプリング速度が K5/VSSP32 サンプラーより早い。また、フーリエ変換した後に位相傾斜を与え、遅延追尾をする Δw 補正をする。時間領域ではサンプリング速度の逆数刻みでの遅延追尾しかできないが、この方法ではさらに分光点の分だけ細かく遅延追尾することが出来る。そのため、より精密な遅延追尾をすることが出来る。

[*3] apri_calc のデフォルトでは sidDDDNNNN.dat（sid : アンテナ ID、DDD : 通算日、NNNN : その日の観測番号）である。apri_calc の起動設定で他に 5 つの設定が選べる。なお、茨城局では sidDDDHHMMSS.dat（HH : 時、MM : 分、SS : 秒）が使われており、起動時に設定できるもののどれにも当てはまらない。

表 2.3: ハードウェア相関器の主なスペック

ADC サンプリング周波数 [MHz]	1024, 2048, 4096, 8192
アナログ入力帯域 [MHz]	256 - 25500
ADC 量子化 [bit]	3
相関器入力サンプリング周波数 [MHz][*4]	1024, 2048, 4096
フリンジローテーション分解能 [bit]	12
FFT 点数 [*5]	4K, 32K
FFT 演算 [*6]	実部 16bit, 虚部 16bit　固定小数点
積分時間 [sec]	0.1024, 1.024

2.3.1 干渉計観測の流れ

この相関器で観測するためのプログラムが用意されている。これらを用いての観測の手順を簡単に説明する。

1. frprmgen で遅延時間や遅延時間変化率などが羅列されたファイルを作成
2. aprsend で羅列された遅延時間などを相関器に転送
3. 相関開始コマンドを相関器に送信（バッチファイルが用意されている）
4. 相関器から出力されたデータを cordata で保存

2.3.2 付属ソフトウェアの詳細

前項で述べたプログラムについて説明する。

frprmgen

SKD ファイル（K5/VSSP32 で利用する観測指示書とは別の物）から観測時間、アンテナ座標、観測天体の座標を読み取り、遅延時間や遅延時間変化率などを計算する。出力されるファイルには SKD ファイルから読み取った情報と、計算した値が羅列されている。時刻間隔が指定できるが、後述の aprsend との関係上、1.024 秒間隔にしなければならない。

aprsend

frprmgen で作成した予測値ファイルから、遅延時間や遅延時間変化率などを読み取り、相関開始と同時に相関器に転送する。転送間隔は 1.024 秒固定である。時刻情報は読み取らず、記述順に予測値を転送するため、相関開始時刻と SKD ファイルの観測時間を合わせなければならない。

[*4]現在は 1024MHz のみ対応
[*5]現在は 4K 点のみ対応
[*6]実際には整数が使われている模様

cordata

相関器から出力されたデータを VDIF フォーマットで保存をする。また、同時に相関強度と位相を一定間隔で画面上に表示する。保存をせず、画面への表示のみを行うこともできる。

2.3.3 作成したプログラムについて

保存したデータを解析するためのプログラムを作成したので、簡単に説明する。

vdif_datout

バイナリ形式で保存された VDIF ファイルをテキスト形式に変換して出力をする。出力する際に、複素数データをそのまま出力するか、振幅と位相に変換して出力するかを選択できる。

vdif_calc

vdif_datout で複素数のまま出力したファイルを基に、簡単な解析をする。現状では以下の機能がある。

- 全 CH を積分したときの単位積分ごとの相関強度と位相の出力
- 指定 CH を中心とした特定範囲内での単位積分ごとの最大強度とその時の位相の出力
- アラン分散の計算

3　試験観測と結果

3.1　K5/VSSP32 システムでの観測

二素子干渉計の立ち上げおよびハードウェア相関器の導入に先駆け、すでに導入されている K5/VSSP32 サンプラーを利用しての干渉計観測を行った。

3.1.1　連続波の観測（1）：2013 年 8 月 6 日の結果

まず、2013 年の 8 月 6 日に 1 分間の試験観測を行った。観測諸元を表 3.1 に示す。

表 3.1: 観測諸元

観測日	時刻 UT	観測天体	観測周波数	サンプリング周波数
2013/8/6 (218)	03:15 - 03:16	3C273B	8448 - 8480MHz	64MHz

フリンジ検出したものの、相関強度と位相がともに約 1 秒周期で変動していた（図 3.1、図 3.2）。3C273B はクエーサーで、明るさが変動することで知られているものの、その周期は数日以上のスケールである。そのため、周期約 1 秒での変動は観測システムに由来するものだと考えられる。

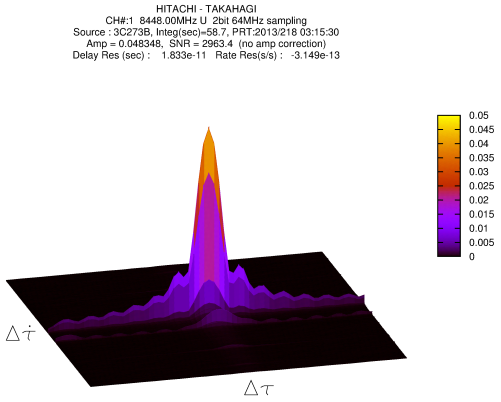

図 3.1: 単位積分 0.1 秒でのフリンジサーチの結果。横軸が遅延時間残差、奥行きが遅延変化率残差、高さが相関強度を表す。

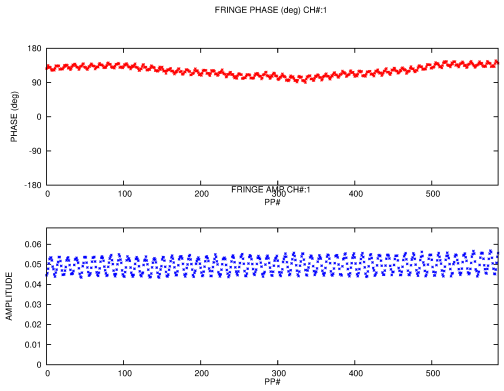

図 3.2: 単位積分 0.1 秒での、単位積分ごとの相関強度（青）と位相（赤）。横軸は単位積分の何番目かを表す。つまり、10PP が 1 秒に対応する。

3.1.2　連続波の観測（2）：2013 年 8 月 14 日の結果

前述の強度と位相の周期変動の原因を探るべく、3C273B を 1 日追尾観測した。

表 3.2: 観測諸元

観測日	時刻 UT	観測天体	観測周波数	サンプリング周波数
2013/8/14 (226)	00:00 - 09:00[*7]	3C273B	8448 - 8480MHz	64MHz

この観測でも周期変動があったが、時刻によって周期が変化していることがわかった（図 3.3、図 3.4）。そこで、各 10 分間の観測の最初の 1 分と最後の 1 分で変動周期を求め、それぞれ方位角（図 3.5）、仰角（図 3.6）、時刻（図 3.7）でプロットした。方位角が基線と垂直の方向になるときに、周期が一番短くなっていることから、天体の方向（方位角、仰角の両方）に依存していると考えられる。つまり、遅延時間 $\tau_g = \boldsymbol{D} \cdot \boldsymbol{s}/c$ もしくはその時間変化率 $\dot{\tau}_g = \boldsymbol{D} \cdot \dot{\boldsymbol{s}}/c$ に依存していると考えられる。

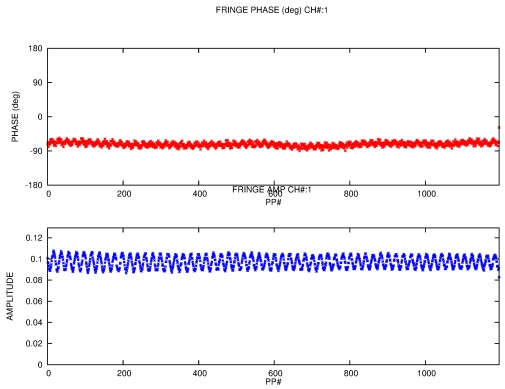

図 3.3: UT04:00 からの 1 分間を単位積分時間 0.05 秒で相関処理した際の、単位積分ごとの相関強度と位相。変動周期は約 1.01 秒。

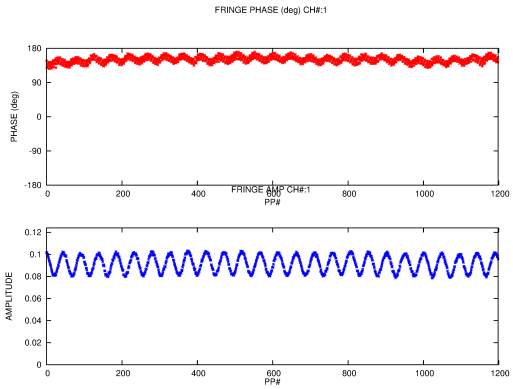

図 3.4: UT08:49 からの 1 分間を単位積分時間 0.05 秒で相関処理した際の、単位積分ごとの相関強度と位相。変動周期は約 2.40 秒。

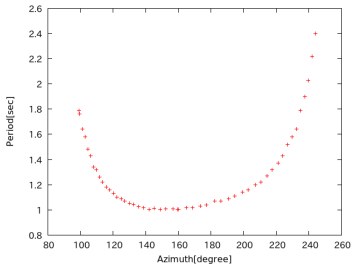

図 3.5: 方位角と変動周期の関係

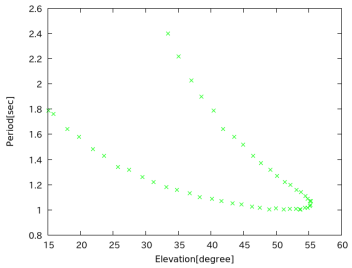

図 3.6: 仰角と変動周期の関係

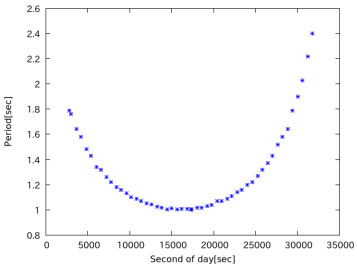

図 3.7: 時刻と変動周期の関係

[*7]10 分おきに 10 分間観測を繰り返した。つまり、00:00-00:10, 00:20-00:30, 00:40-00:50... というように観測をしている。

3.1.3 連続波の観測（3）：2014 年 12 月 26 日から 2015 年 1 月 14 日にかけての結果

前項の結果をふまえ、2014 年 12 月 26 日に再現性の確認、2015 年 1 月 8 日から 1 月 14 日にかけて両局で局部発振器（LO）の周波数をずらしての観測を行った。観測諸元を表 3.3 に示す。

表 3.3: 観測諸元

観測日	時刻 UT	観測天体	観測周波数	サンプリング周波数	備考
2014/12/26 (360)	16:00 - 翌 01:30[*7]	3C273B	6664 - 6696, 8448 - 8480MHz	64MHz	
2015/1/8 (008)	15:00 - 21:30[*7]	3C273B	6664 - 6696, 8448 - 8480MHz	64MHz	高萩 LO を 10kHz 低く設定
2015/1/9 (009)	14:56 - 21:26[*7]	3C273B	6664 - 6696, 8448 - 8480MHz	64MHz	高萩 LO を 100kHz 低く設定
2015/1/14 (014)	15:37 - 21:07[*7]	3C273B	6664 - 6696, 8448 - 8480MHz	64MHz	高萩 LO を 1MHz 低く設定

12 月 26 日の観測データを相関処理をした結果、再現性があることが確認された。また、6GHz 帯でも周期的な変動を確認した。1 月 8 日から 14 日のデータ処理したところ、LO 周波数をずらした観測では周期変動が平均化されていることがわかった。

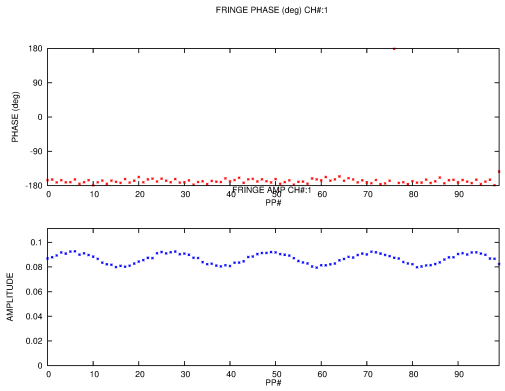

図 3.8: 6GHz 帯の 12/26 の観測データを、UT16:00 からの 10 秒間を単位積分時間 0.1 秒で相関処理した際の、単位積分ごとの相関強度と位相。

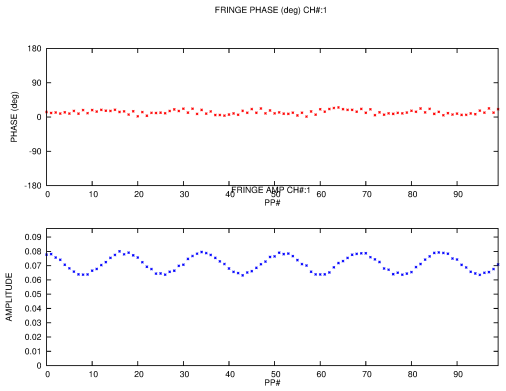

図 3.9: 8GHz 帯の 12/26 の観測データを、UT16:00 からの 10 秒間を単位積分時間 0.1 秒で相関処理した際の、単位積分ごとの相関強度と位相。

これらの結果から周期約 1 秒の変動は、この干渉計の基線長が短く、遅延時間の変化が遅いことが原因だと予想される。fx_cor は相関処理に用いる三角関数を 8 レベルで近似している。単位積分の範囲内での、三角関数のレベルの切り替えのタイミングによって相関強度に差が生じ、周期変動となって表れていたと考えられる（図 3.12）。両局間で LO 周

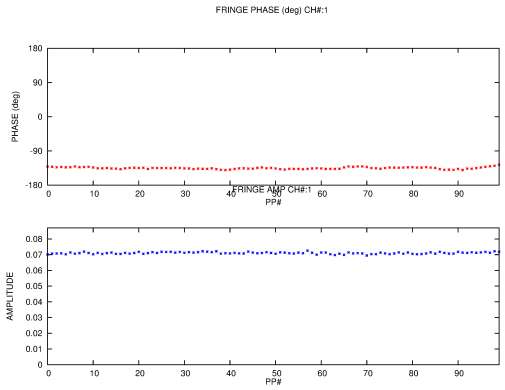

図 3.10: 8GHz 帯の 1/8 の観測データを、UT15:08:54 からの 10 秒間を単位積分時間 0.1 秒で相関処理した際の、単位積分ごとの相関強度と位相。LO をずらさず観測していたときは強度が 0.06 から 0.08 の間で振動していたが、0.07 付近で安定している。

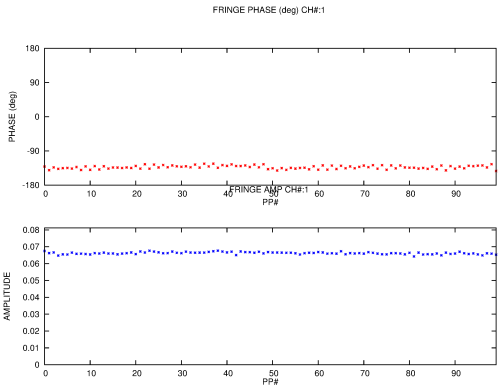

図 3.11: 8GHz 帯の 1/14 の観測データを、UT15:45:19 からの 10 秒間を単位積分時間 0.1 秒で相関処理した際の、単位積分ごとの相関強度と位相。

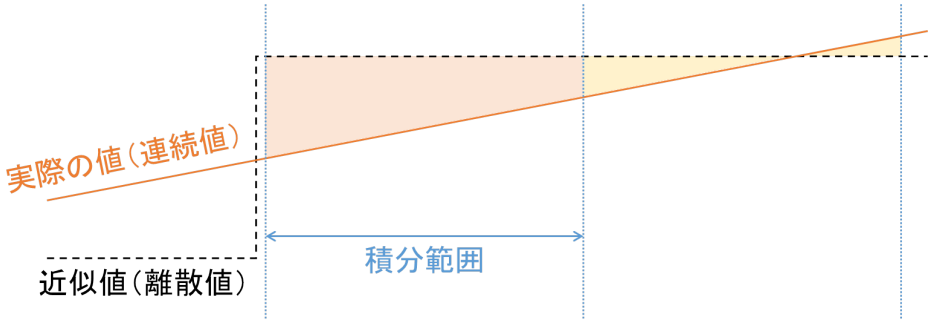

図 3.12: 三角関数の近似値と単位積分時間の関係。LO をずらした際に変動が見られなくなったのは、実際の値の傾きが加えられたためと考えられる。

波数をずらすと、それに伴ったフリンジを回転が発生する。このフリンジの回転が、積分時間内での切り替えタイミングの影響を軽減する。そのため、LO 周波数をずらした際に変動が見られなくなったと予想される。なお、基線長が長くなれば、その分遅延時間の変化率が大きくなるため、周期変動は発生しなくなると考えられる。

3.1.4 メーザーの観測：2015 年 1 月 13 日の結果

2015 年 1 月 13 日には、メタノールメーザー G188.95+0.89 の観測を行った。諸元を表 3.4 に示す。

相関処理をした結果、クロススペクトルの位相が互いに 180 度離れた成分に分裂し、強度が周波数方向に広がった成分と狭い成分が重なっていた（図 3.13）。

表 3.4: 観測諸元

観測日	時刻 UT	観測天体	観測周波数	サンプリング周波数
2015/1/13 (013)	07:14 - 19:04[*7]	G188.95+0.89	6664 - 6696MHz	64MHz

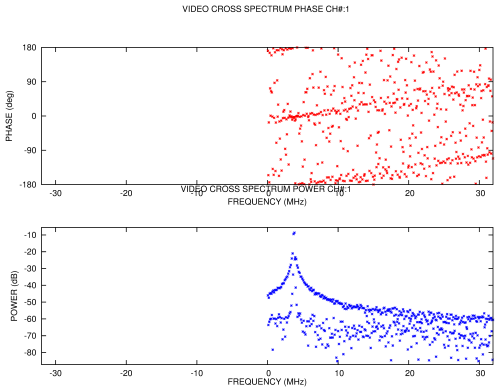

図 3.13: メーザー G188.95+0.89 のクロススペクトル。位相には 180 度離れた 2 成分が、強度には広がった成分と細い成分が確認できる。

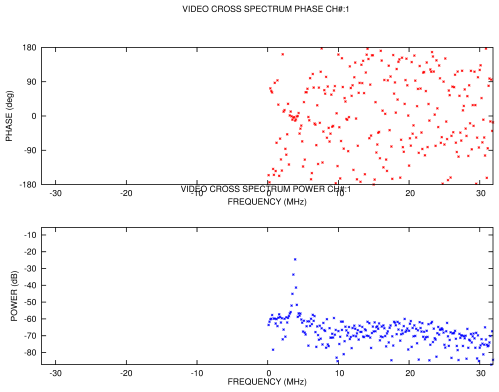

図 3.14: ラインスペクトル用のオプションを利用した時の G188.95+0.89 のクロススペクトル。強度、位相ともにもっともらしくなっている。

この現象は、sdelay の内部処理に起因するものであった。プログラム開発者の近藤氏（NICT 鹿島）によると、この現象は次のような理由で発生していた。

メーザーのように特定周波数に強い信号がある場合の相互相関関数は 正弦波関数に近い形となる。この相関関数をフーリエ変換する際、ラグ範囲のみを切り出すような関数（ゲート関数）をかけてからフーリエ変換している（図 3.15）。フーリエ変換後の値は、畳み込みの原理により、正弦波関数をフーリエ変換したデルタ関数とゲート関数をフーリエ変換した sinc 関数の畳み込みで表される（図 3.16）。

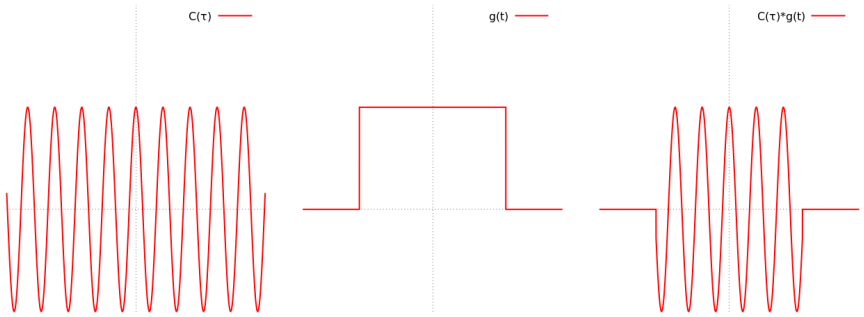

図 3.15: 正弦波的な相関関数（左）をゲート関数（中）で切り出した値（右）が実際には出力されている。

ところが、実際の相関処理では離散的な値を使っている。この時、切り出したデータ数と同じ点数で離散フーリエ変換をすればゲート関数の影響は表れない。しかし、sdelay で

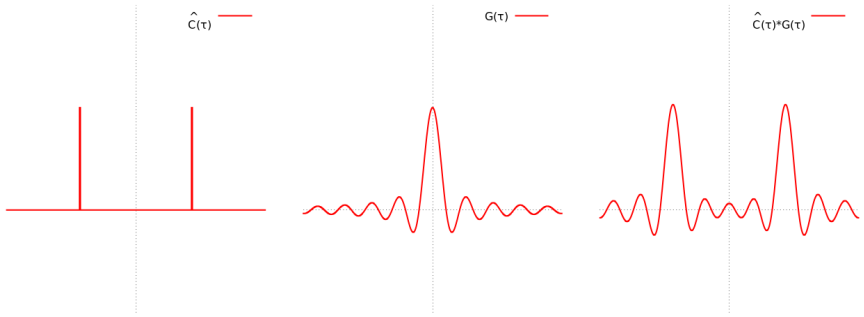

図 3.16: 相関関数のフーリエ変換（左）とゲート関数のフーリエ変換（中）の畳み込み（右）が、クロススペクトルとして出力される。

はデータ数の 2 倍の点数でフーリエ変換をしていたため、広がった成分が重なってしまった（図 3.17）。また、正負で位相が 180 度違うため、位相も 2 つの成分が見えてしまっていた。

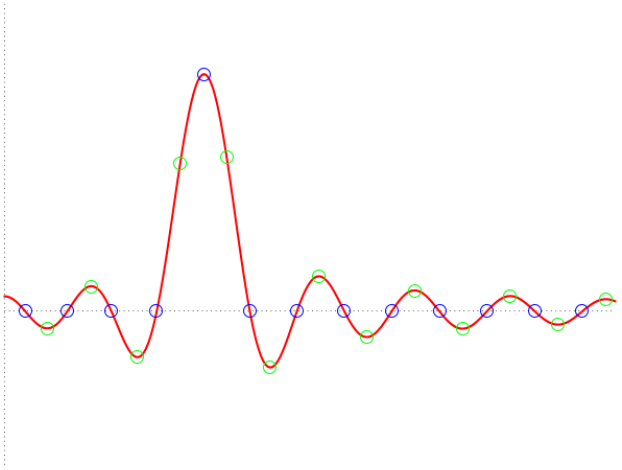

図 3.17: データ数でフーリエ変換した場合は青色の点の値のみが出力されるが、2 倍の点数でフーリエ変換したために緑色の点の値も出力されていた。

この結果を踏まえ、現在はラインスペクトル用のオプションが追加されている。このオプションによりデータ数と同じ点数でフーリエ変換ができるようになり、上記の現象は解消された（図 3.14）。なお、データ点の 2 倍の点数でフーリエ変換していた背景は、初期の測地 VLBI では連続波源に対する 8 点での相関処理だったことに起因する。遅延分解関数という関数を計算する際に、周波数方向への平滑化のために倍の点数でフーリエ変換をしていたとのことである。

以上の結果から、K5/VSSP32 ソフトウェア相関器システムにより連続波源、メーザー源ともに正しい相関処理が可能であることを確認した。

3.2　結合型ハードウェア相関器での観測

2014 年度末にハードウェア相関器が設置され観測が可能となったため、試験観測を行った。相関器用のサンプラーが両局にそれぞれ 2 つずつ設置されている。すべての観測で 6-9GHz 帯受信機を利用し、6GHz 帯と 8GHz 帯を同時に受信、処理している。

3.2.1　試験観測（1）: 2015 年 3 月 3 日から 3 月 6 日の結果

初期設定を行い、その後連続波源 4C39.25 を観測した。フリンジの検出に成功したが、位相の回転と、6GHz 帯で不規則な強度変動が見られた（図 3.18、図 3.19）。なお、観測周波数帯のスペクトルは 2047-0ch に表示される。2048-4095ch はその折り返しで、同等の情報が表示されている。

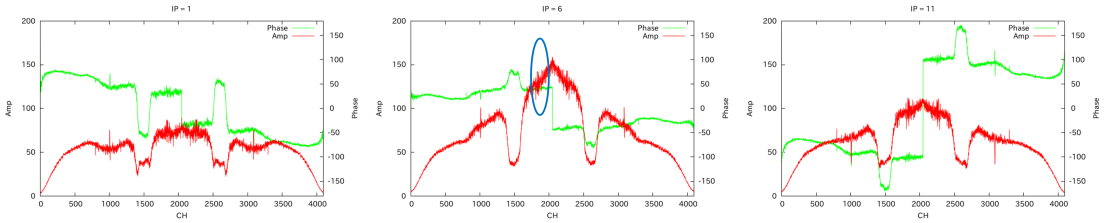

図 3.18: 4C39.25 の 6GHz 帯のクロススペクトル。観測周波数は 6600-7112MHz である。単位積分時間は 1.024 秒で、左から約 5 秒ごとのクロススペクトルを並べた。1000-2047ch（及び 2048-3095ch）の強度がつぶれるように変動しているのがわかる。また、位相が回転している様子も見える。

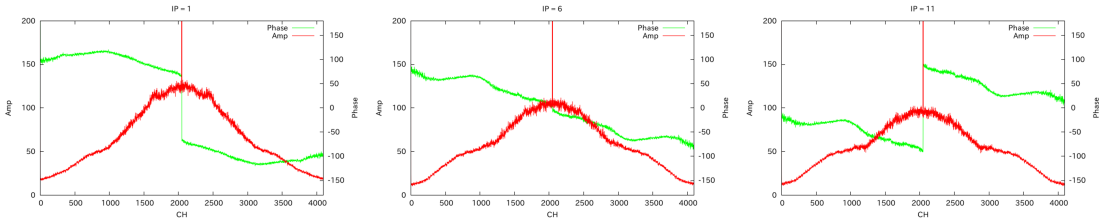

図 3.19: 4C39.25 の 9GHz 帯のクロススペクトル。観測周波数は 8192-8704MHz。8GHz 帯でも位相が回転している。2048ch の信号は ADC によるものとのこと。

位相回転の原因は、ファームウェア上の不具合であった。サンプラーでは 8GHz でサンプリングしているが、相関器入力は 1GHz のため、間引く作業を行っている。この間引くことによる位相のずれの補正がされていなかったとのことであった [*8]。

強度変動は、人工的な雑音の混入が原因であった。かねてより 6550MHz 付近に雑音が確認されていたが、この雑音が混入したことによって相関が崩れてしまっていた。茨城局のシステムでは、LO6088MHz を用いて 6600-7112MHz を IF512-1024MHz として観測し

[*8] この相関器と同系統のものが先に韓国に導入されているが、そちらは ADC と相関器入力のサンプリング速度が等しいため、この位相補正が不必要であった。

ている。そのため、6550MHz の雑音が折り返して 1800ch 付近に現れていた（図 3.18 の青丸）*9。

3.2.2　試験観測（2） 2015 年 4 月 7 日から 4 月 10 日の結果

前項の問題点について、次のような対策を行った。位相が回転する問題は、位相補正が修正されたファームウェアの導入した。強度変動および雑音混入については、IF 帯に 550-950MHz のバンドパスフィルタを高萩局に追加した *10。これらの効果を確認するため、4 月 7 日から 10 日にかけて、4C39.25 とメタノールメーザー W3OH の観測を行った。

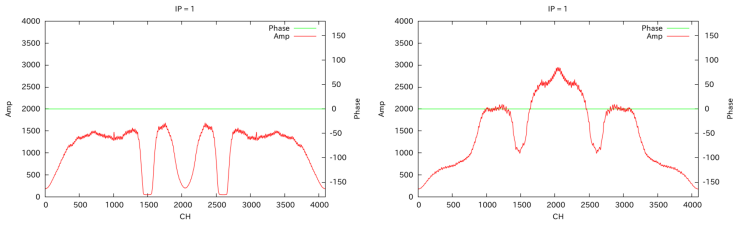

図 3.20: 6GHz 帯の 4C39.25 の高萩局自己相関（左）と日立局自己相関（右）。高萩局では、追加したフィルタによって雑音の混入が抑えられている。

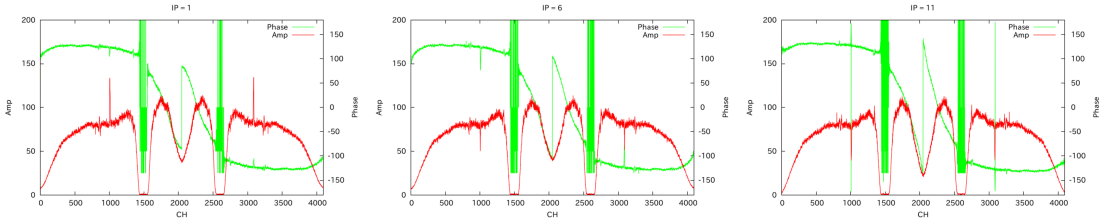

図 3.21: 6GHz 帯の 4C39.25 のクロススペクトルを左から約 5 秒ごとに並べた。大きな強度変動が見られなくなったことが分かる。また、位相回転も補正されている。

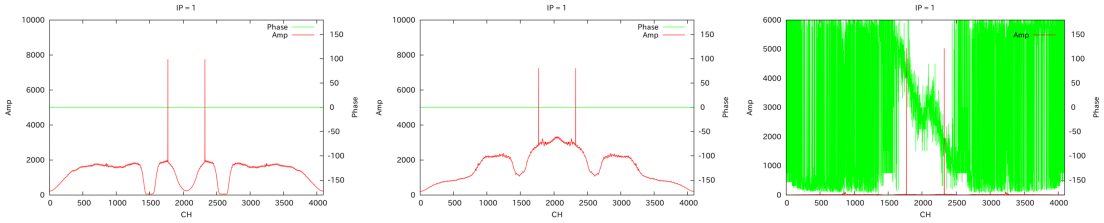

図 3.22: 6GHz 帯のメタノールメーザー W3OH の、高萩局自己相関（左）と日立局自己相関（中）、クロススペクトル（右）。メーザー信号が 1750ch に表示されている。

バンドパスフィルタの追加により、雑音の混入が抑制され、強度変動がなくなった。また、ファームウェアの更新により、位相の回転も補正された（図 3.21）。W3OH の観測に

*9 K5/VSSP32 でこの雑音が現れないのは、LO が 6088+576MHz のためだと思われる。また、K5/VSSP32 自体にデジタルフィルタがある。

*10 1500ch 付近の信号を抑制するノッチフィルタは、相関器導入以前から使用されているものである。

より、新しく追加したバンドパスフィルタは、メタノールメーザーの観測にはほとんど影響がないことが確認された（図 3.22）。

3.2.3　試験観測（3）：2016 年 1 月 7 日の結果

K5/VSSP32 ソフトウェア相関器システムでみられた周期変動が、ハードウェア相関器でも発生するか確認をするため、2016 年 1 月 7 日に連続波源 3C273B を単位積分時間 0.1024 秒で観測をした。単位積分ごとに複素相関スペクトルを 0-2047ch で積分し、両局の自己相関で正規化したものを図 3.23 に示す。K5/VSSP32 ソフトウェア相関システムと比べると崩れた形をしているものの、6GHz 帯、8GHz 帯どちらにも正弦波的な周期変動がみられた。また、位相には周期の異なった矩形波的な変動も見られる。

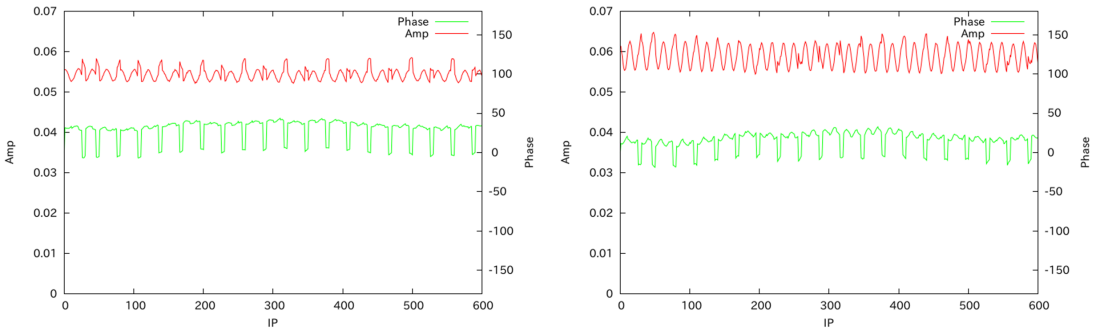

図 3.23: 単位積分 0.1024 秒で観測した際の、単位積分ごとの相関強度と位相。左が 6GHz 帯、右が 8GHz 帯である。

K5/VSSP32 ソフトウェア相関器では、遅延時間の変化が遅い事と三角関数を離散値に近似している関係から周期変動が現れると予想した。しかし、ハードウェア相関器では 12bit（4096 レベル）で近似している。また、Δw 補正を行っているため、遅延時間残差による誤差も生じにくい。ただし、適切なビットの分布で相関処理が行われていなければ意味がないため、確認する必要がある [*11]。一方、受信機のシステム雑音温度は 30K 以下と非常に感度がよい。そのため、わずかなポインティングのずれなどが周期変動として出てしまっている可能性もある。

位相の矩形波的な変動は 1.024 秒積分の観測のデータでは見られなかった。図 3.23 で低いレベルにある時間が 0.5 秒と短いため、積分時間内で均されている可能性が高い。

3.2.4　アラン分散の測定

雑音には、大きく分けてランダムな雑音と系統的な雑音がある。ランダムな雑音は積分時間を長くすれば小さくなるが、系統的な雑音は必ずしもそうではない。これらを評価す

[*11] なお、サンプラー出力のビット分布については適正な状態になるよう調整してから観測をしている。

る指標として、アラン分散がある。アラン分散は、時間に対し、ランダムな雑音と系統的な雑音のどちらが卓越するかを表す。ランダムな雑音が卓越している時間を安定時間という。安定時間内では、積分することで信号雑音比が向上する。

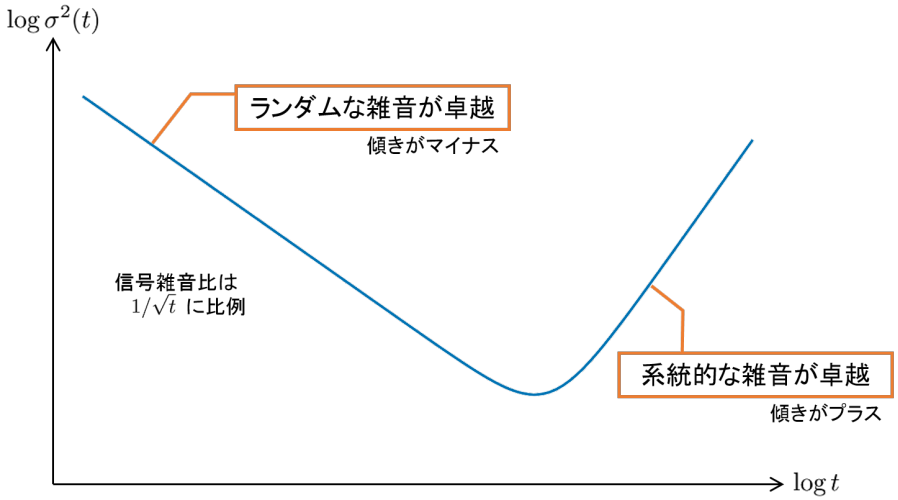

図 3.24: 典型的なアラン分散

まず、各チャンネルでの出力がどの程度の時間で安定しているかを求める。あるチャンネルの単位時間ごとの出力を N 番目まで記したデータ列がある。このデータ列の i 番目のデータを $x_i (i = 1, 2, \cdots, N)$ とおく。x_i を K 個ずつ平均したデータ列

$$R_n(K) = \frac{1}{K} \sum_{i=1}^{K} x_{nK+i} \tag{3.1}$$

を作る。ただし、$n = 0, 1, 2, \cdots, M$ とし、$M = N/K - 1$ である。次に、$R_n(K)$ の分散 $\sigma^2(K)$ を

$$\sigma^2(K) = \frac{1}{2M} \sum_{n=1}^{M} \{R_n(K) - R_{n-1}(K)\}^2 \tag{3.2}$$

と定義する。この分散をアラン分散という。式 (3.2) は選択したチャンネルの安定性を示す。

次に、チャンネル同士の出力の差がどの程度の時間で安定しているかを求める。同時に取得した 2 つのチャンネルの単位時間ごとの出力を N 番目まで記録したデータ列がある。それぞれのチャンネルでの i 番目のデータを $x_i, x'_i (i = 1, 2, \cdots, N)$ とおく。$x_i - x'_i$ を K 個ずつ平均したデータ列

$$R'_n(K) = \frac{1}{K} \sum_{i=1}^{K} (x_{nK+i} - x'_{nK+i}) \tag{3.3}$$

を作る。ただし、$n = 0, 1, 2, \cdots, M$ とし、$M = N/K - 1$ である。次に、$R'_n(K)$ の分散 $\sigma^2(K)$ を

$$\sigma^2(K) = \frac{1}{2M} \sum_{n=1}^{M} \left\{ R'_n(K) - R'_{n-1}(K) \right\}^2 \tag{3.4}$$

と定義する。この分散を分光アラン分散といい、帯域通過特性の安定性を示す。

どちらの分散も、横軸を時間としてプロットすると図 (3.24) のようになる。分散が時間に反比例して減少している時間スケールが安定時間となる。

システムの安定時間を評価するため、遅延追尾を行わない設定の下、次の条件で観測を行った。

- 仰角 85 度（大気の影響が最小となる角度）
- 仰角 30 度（観測時の典型的な角度）
- 仰角 15 度（観測時のおおよその下限値で、大気の影響を最も受ける角度）
- 電波吸収体（ホーン上部に電波吸収体を設置し、大気の影響を取り除いた状態）

なお、方位角はいずれも 130 度である。また、各条件での観測諸元を表 3.5 に示す。

表 3.5: アラン分散測定時の諸元

	仰角 85 度	仰角 30 度	仰角 15 度	電波吸収体 [*12]
日時	2015/12/28	2016/1/8	2016/1/9	2016/1/13
天気	晴れ	曇り	曇り一時雨	—
観測時間	7 時間半	8 時間	8 時間	8 時間
観測周波数	6600 - 7112, 8192 - 8704MHz			
相関器設定	1Gsps, 4096 点 FFT, 単位積分時間 1.024 秒			

6GHz 帯、8GHz 帯の両方で、1010 番目のチャンネル（以下 ch1010 とする）のアラン分散と 7 組のチャンネル（ch1010-1110, ch910-1110, ch910-1210, ch810-1210, ch710-1210, ch510-1510, ch10-2010）の分光アラン分散を求め、プロットした。なお、横軸はデータ点数であるが、単位積分 1.024 秒なので、1 データを 1 秒として扱う。

[*12]電波吸収体の測定時、高萩局はデバイドを停止した。日立局はメンテナンスを行っていた関係上、デバイドの停止はしなかったが、空気抜き弁を解放した。また、大気を見た時と入力レベルが同じになるよう、ダウンコンバータ前にアッテネータを 10dB 追加した。

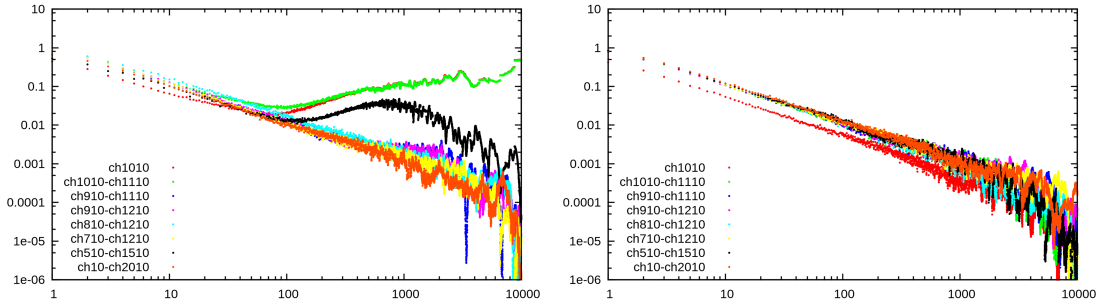

図 3.25: 仰角 85 度のアラン分散。左が 6GHz、右が 8GHz である。

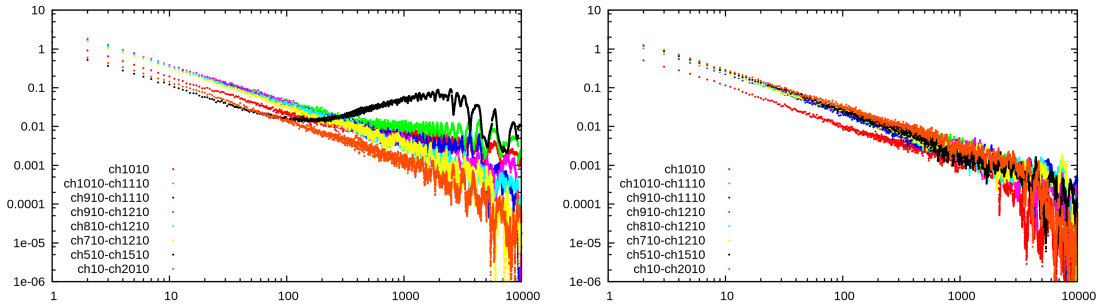

図 3.26: 仰角 30 度のアラン分散。左が 6GHz、右が 8GHz である。

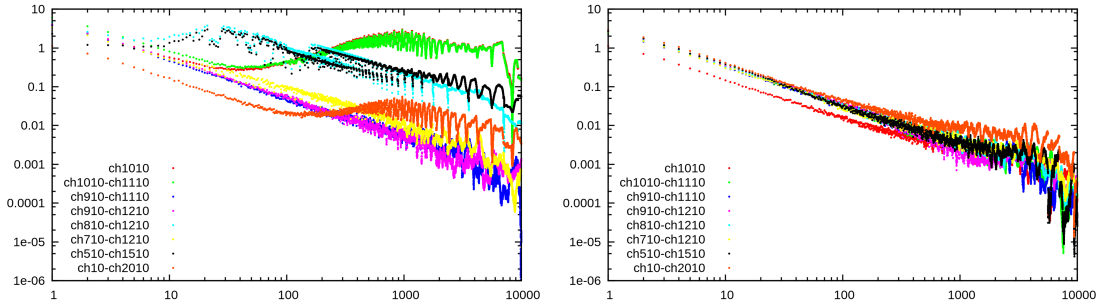

図 3.27: 仰角 15 度のアラン分散。左が 6GHz、右が 8GHz である。

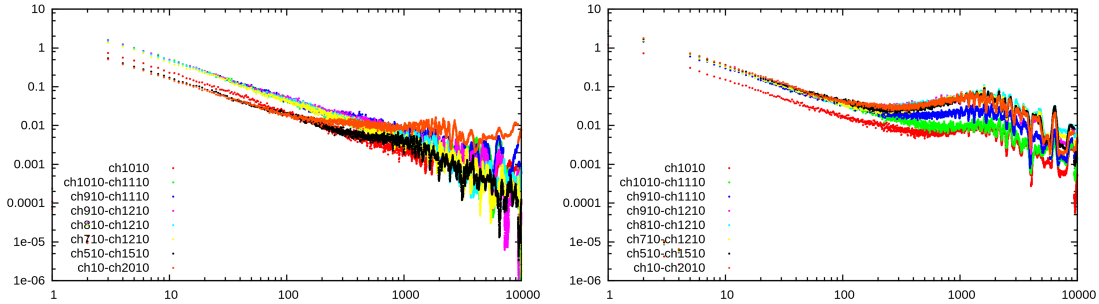

図 3.28: 電波吸収体設置時のアラン分散。左が 6GHz、右が 8GHz である。

それぞれの安定時間を、表 3.6 にまとめた。全体的に、8GHz 帯より 6GHz 帯の安定時間が短いことが明らかになった。ch1010 のアラン分散は、どちらの帯域でも仰角 15 度で安定時間が最短となっていた。分光アラン分散では、6GHz 帯では仰角 15 度のときの安定時間は 5 秒という結果になった。

表 3.6: 各条件でのおおよその安定時間

		仰角 85 度	仰角 30 度	仰角 15 度	電波吸収体
アラン分散の安定時間	6GHz	70 秒	300 秒	30 秒	1000 秒
	8GHz	1000 秒	>1000 秒	>1000 秒	400 秒
分光アラン分散の安定時間	6GHz	100 秒	100 秒	5 秒	200 秒
	8GHz	>1000 秒	600 秒	800 秒	200 秒

6GHz 帯では、電波吸収体に比べ、空を見た時の安定性が悪くなっていた。特に仰角 15 度のときに悪化している。気象条件が悪かったのもあるだろうが、分光アラン分散が特定のチャンネルの組みでよくなかったことから、地上からの人工的な雑音が混入してしまっている可能性がある。また、1510ch はノッチフィルタによって信号が抑制されている周波数にあたる。フィルタによって信号が抑制されると、大気由来のランダムな雑音信号が減り、装置由来の系統的な雑音信号の割合が相対的に高くなる。そのために、510-1510ch の分光アラン分散（図 3.25 から 3.28 では黒色にあたる）での安定時間が短くなっている可能性も考えられる。

8GHz 帯では、全体的に安定度が良いという結果が得られた。空を見た時と比べ、電波吸収体での安定時間が短いのは、日立局のデバイドを切ることができなかったことが原因だと考えられる。

単一鏡の結果 [*13] と比較すると、1 つのチャンネルでの安定時間は長くなったが、2 つのチャンネル間の安定時間は短くなった。 ただし、1ch あたりの帯域を比べると、ハードウェア相関器が K5/VSSP32 より 16 倍広い [*14]。ランダムな雑音のノイズレベルは、帯域の広さの平方根に反比例する。仮に、帯域が広い分、ハードウェア相関器のランダムな雑音のノイズレベルが低いとすると、統計的な雑音が卓越する時間は早くなる（図 3.29）。そのため、連続波観測に対しては、単一鏡よりもハードウェア相関器での干渉計観測が適しているといえるが、分光観測に対して判断するには、さらなる検証が必要である。

[*13] K5/VSSP32 サンプラーを用いての測定で、特定の ch の安定時間は 10-20 秒、2 つの ch 間の安定時間は 400-1000 秒である。[7]

[*14] ハードウェア相関器は 512MHz を 2048 点分光、K5/VSSP32 では 32MHz を 2048 点分光している。1ch あたりの帯域はそれぞれ 250kHz、15.6kHz である。

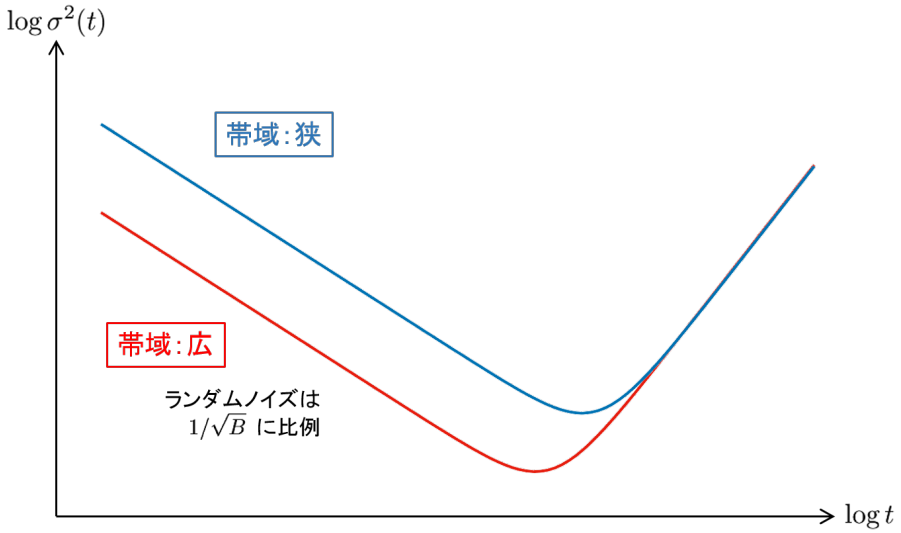

図 3.29: 1ch あたりの帯域とアラン分散の関係。帯域が広いとランダムな雑音が減り、系統的な雑音が卓越する時間スケールに早く到達する。

4 まとめ

4.1 結論

本研究では、茨城局の 2 台の 32m 電波望遠鏡を結合した二素子干渉計の立ち上げを行い、K5/VSSP32 を用いたソフトウェア相関システムと結合型ハードウェア相関システムで試験観測をした。

K5/VSSP32 ソフトウェア相関システムでは、相関強度と位相の周期変動やクロススペクトルの分裂があった。周期変動については、局部発信機の周波数をずらすことで平均化された。クロススペクトルの分裂はソフトウェアに対策が施され、解消した。このシステムでは、連続波源、メーザー源ともに正しい相関処理が可能であることを確認した。

ハードウェア相関システムでは帯域外の雑音の折り返しや位相回転、振幅の周期変動があった。雑音の折り返しによる問題は、帯域外をカットするようなバンドパスフィルターの追加によって改善された。位相回転はファームウェアの修正により解消した。また、ハードウェア相関システムではアラン分散の測定を行い、連続波に対しては単一鏡よりも安定度がよいことを確認した。

4.2 今後の課題

1. K5/VSSP32 ソフトウェア相関システムで、干渉計としてのアラン分散の測定を行い、ハードウェア相関システムとの比較をする。また、ハードウェア相関システムとともに、実際に安定時間内で雑音が減少するかを確かめる。

2. ハードウェア相関システムでの周期的な変動が、K5/VSSP32 ソフトウェア相関システムと同様に局部発振器の周波数をずらすことで解消されるか確認する。また、周期変動の原因を特定し、可能なら取り除く。

謝辞

本研究を進めるにあたり、多くの方にお世話になりました。指導教官の百瀬宗武教授には、電波天文学の基礎から研究内容、論文の添削まで、広くご指導いただきました。米倉覚則准教授には、観測装置の基礎から観測方法の指導から、観測の補助までしていただきました。研究員の杉山孝一郎氏、澤田-佐藤聡子氏には観測日程の調整をしていただいたり、質問、相談に答えていただきました。宮本祐介氏、古川尚子氏には、ゼミで干渉計の基礎を教えていただきました。心から感謝を申し上げます。

K5/VSSP32 システムでの相関処理に、NICT のソフトウェアを使用させていただきました。また、NICT 次世代時空計測グループ技術員の近藤哲朗氏には、観測データの検証などをしていただきました。エレックス工業株式会社の原田健一氏、大貫博文氏、林由記氏には、ハードウェア相関システムでの観測方法の指導からデータの検証までお世話になりました。心よりお礼申し上げます。

研究室の皆さまとは時に議論を交わし、時にふざけ合い、充実した研究室生活を送ることが出来ました。飽き性の私でも、3 年間とても楽しく研究に打ち込めました。ありがとうございました。

最後に、大学院への進学を快く許可してくれた両親と、支えてくれた家族に感謝いたします。

参考文献

[1] A. R. Thompson, J. M. Moran, G. W. Swenson Jr. ”Interferometry and Synthesis in Radio Astronomy. Second Edition” WILEY-VCH Verlag GmbH & Co. KGaA, 2004

[2] 尾崎洋二『宇宙科学入門』東京大学出版会 2010

[3] 中井直正ほか編『宇宙の観測 II-電波天文学 シリーズ現代の天文学 第 16 巻』日本評論社 2009

[4] 高橋冨士信ほか『ウェーブサミット講座 VLBI 技術』オーム社 1997

[5] 国立天文台編『干渉計サマースクール 2005 教科書』

[6] 滝沢美里『茨城 32m 電波望遠鏡用 6.5-8.8GHz 帯低雑音広帯域受信機システムの開発』茨城大学大学院理工学研究科 2011 年度修士論文

[7] 田中智明『茨城 32m 電波望遠鏡の単一鏡分光観測システム開発』茨城大学大学院理工学研究科 2011 年度修士論文

[8] 森智彦『茨城局 32m 電波望遠鏡搭載用 22GHz 帯冷却受信機の開発』茨城大学大学院理工学研究科 2013 年度修士論文

[9] ”K5/VSSP VLBI SYSTEM”, http://www2.nict.go.jp/aeri/sts/stmg/K5/VSSP/, 2016 年 1 月アクセス

[10] ”天体測定学 II 2008-10”, http://veraserver.mtk.nao.ac.jp/VERA/honma/lecture/note2008-10.pdf, 2016 年 1 月アクセス

[11] ”VERA プロジェクト”, http://veraserver.mtk.nao.ac.jp/index-J.html, 2016 年 1 月アクセス

A　ハードウェア相関器用解析プログラム

vdif_datout

ソースコード **1:** vdif_datout_makefile

```
# make -f vdif_datout_makefile
# vdif_datout.c, vdifreader.c から vdif_datout を作成する Makefile
# 更新があった部分のみ再コンパイルするように設定してある。
# 2015/12/22 v0.1 仮作成

# gcc コンパイラを使用, オプション-lm を付与
CC = gcc
LFLAGS = -lm

.PHONY: all
all: vdif_datout

vdif_datout: vdif_datout.o vdifreader.o
	$(CC) -o $@ vdif_datout.o vdifreader.o $(LFLAGS)

vdif_main.o: vdif_datout.c
	$(CC) -c vdif_datout.c $(LFLAGS)

vdifreader.o: vdifreader.c
	$(CC) -c vdifreader.c $(LFLAGS)

.PHONY: clean
clean:
	rm -f vdif_datout.o vdifreader.o
```

ソースコード **2:** vdif_datout.c

```
/*
  vdif_datout.c
  VDIF から dat(txt)を出力

  2015/12/22 v0.01 作成
  2015/12/26 v0.03 諸元部分の表示を微調整
  2015/12/27 v0.04 TotalIP 出力部作成
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "vdifreader.h"

#define FFT 4096 // FFT 点数
```

```c
#define datamax 1000000 // datamax = IP * FFT / 256

int main(int argc, char *argv[]){
  FILE *vdif; // VDIF ファイルポインタ
  FILE *vout; // 出力ファイル用

  int *vdata; // VDIF データ保管用

  char *v32bit; // bit 用 -> データ変換中継用

  char *vfile; // VDIF ファイル名
  char *outfile;

  char opt[2]; // 出力オプション

  fpos_t fpos; // カーソル位置みたいなの
  int ip;

  int i;


  /* 動的メモリ確保 */
  vdata = (int*)malloc(sizeof(int)*328 * datamax);
  v32bit = (char*)malloc(sizeof(char)*32);
  vfile = (char*)malloc(sizeof(char)*256);
  outfile = (char*)malloc(sizeof(char)*256);
  if(vdata == NULL || v32bit == NULL || vfile == NULL || outfile == NULL){
    fprintf(stdout, "malloc error\n");
    goto END;
  }

  /* VDIF ファイルの指定とパイプ */

  /* argv にファイル指定があった場合はそちらを */
  /* なければ入力するよう求め、バイナリーを開く */
  printf("VDIF filename : ");
  putin_filename(argv[1], vfile);
  vdif = fopen(vfile, "rb");
  if(vdif == NULL){
    fprintf(stdout, "%s FILE NOT FOUND\n", vfile);
    // exit(1);
    goto END;
  }


  printf("Output file name : ");
  putin_filename(argv[2], outfile);
  vout = fopen(outfile, "w");
  if(vout == NULL){
```

```
66    printf("Fail to make %s file \n", outfile);
67    // exit(1);
68    goto END;
69  }
70
71  /* VDIF 読み込み、配列に int 型で保管 */
72  /* vdifreader.c, vdifreader.h を参照 */
73  memset(vdata, '\0', sizeof(vdata)); // vdata 初期化
74  binread(vdif, vdata);
75
76  /* VDIF ファイルから読み取れる観測諸元を表示 */
77  /* 観測開始時刻 */
78  fobsdate(vdata, vout);
79
80  /* IP 長 */
81  keepbit(v32bit, vdata[4]);
82  fprintf(vout, "#IP_length : ");
83  if(decimal(v32bit, 7, 0) == 1){
84    fprintf(vout, "0.1024 sec\n");
85  }else if(decimal(v32bit, 7, 0) == 10){
86    fprintf(vout, "1.024 sec\n");
87  }else{
88    fprintf(vout, "Undefined\n");
89  }
90
91  /* FFT 長 */
92  fprintf(vout, "#FFT_points : ");
93  if(decimal(v32bit, 15, 8) == 0){
94    fprintf(vout, "4096\n");
95  }else if(decimal(v32bit, 15, 8) == 1){
96    fprintf(vout, "32768\n");
97  }else{
98    fprintf(vout, "Undefined\n");
99  }
100
101 /* サンプリング周波数 */
102 keepbit(v32bit, vdata[5]);
103 fprintf(vout, "#Sampling_rate : ");
104 if(decimal(v32bit, 31, 24) == 1){
105   fprintf(vout, "1024 Msps\n");
106 }else if(decimal(v32bit, 31, 24) == 2){
107   fprintf(vout, "2048 Msps\n");
108 }else if(decimal(v32bit, 31, 24) == 3){
109   fprintf(vout, "4096 Msps\n");
110 }else{
111   fprintf(vout, "Undefined\n");
112 }
113
114 /* サンプルビット数 */
```

```
115   keepbit(v32bit, vdata[3]);
116   fprintf(vout, "#Sample_bits : %d bits\n", decimal(v32bit, 30, 26)+1);
117
118   /* TotalIP 用の場所確保と位置保存 */
119   fprintf(vout, "#Total_IP : ");
120   fgetpos(vout, &fpos);
121   fprintf(vout, " \n");
122
123   /* 出力タイプ設定 */
124   printf("Choose output type, r(Re/Im) or a(Amp/Phase) : ");
125   fgets(opt, 2, stdin); // 末尾に'\0'が自動入力される
126   /* stdin のバッファ消去 */
127   if(strchr(opt, '\n') == NULL){
128     while(getchar() != '\n');
129   }
130
131   if(strchr(opt, 'r') != NULL){
132     fprintf(vout, "#Outputtype : r\n");
133     ip = vdifoutput_r(vdata, vout);
134   }else if(strchr(opt, 'a') != NULL){
135     fprintf(vout, "#Outputtype : a\n");
136     ip = vdifoutput_a(vdata, vout);
137   }else{
138     printf("Undefined option\n");
139     goto END;
140   }
141
142   /* Total IP の入力 */
143   fsetpos(vout, &fpos); // サンプルビット数の直後に移動
144   fprintf(vout, "%d", ip);
145
146
147  END: // エラーしたらここに飛んでくる
148         // 本当はgoto 文は使うべきではないらしい
149
150   /* メモリ解放 */
151   /* malloc で確保した順と逆に開放すること */
152   free(outfile);
153   free(vfile);
154   free(v32bit);
155   free(vdata);
156
157   /* パイプを閉じる */
158   fclose(vout);
159   fclose(vdif);
160
161   return EXIT_SUCCESS;
162 }
```

ソースコード **3:** vdifreader.h

```
/*
  vdifreader.h
  vdifreader.c 中心のヘッダーファイル

  2015/09/15 v0.01 仮作成
  2015/12/22 v0.02 datout に対応
 */

#include <stdio.h>
#include <stdlib.h>

#ifndef READER_H
#define READER_H

/* 32bit 分を int で取り出し、ビット表示に変換後、buf に格納 */
extern void keepbit(char *buf, int x);

/* 64bit 分取り出し、double 型に変換 */
extern double print_tau(int *buf, int line);

/* 16bit の数を符号付きに拡張 */
extern int signed16bit(int x);

/* 32bit の配列から、指定部分のみ取り出し 10 進数として表す */
extern int decimal(char *buf, int x, int y);

/* strcat(配列の結合)をint 型に適用したもの */
/* buf の後ろに buf2 を結合 */
extern void intcat(int *buf, int *buf2);

/* fp(VDIFfile)を 1312Byte ごと読み出し、buf に保存 */
/* VDIF は 1312Byte が 1 セットで書かれている */
extern void binread(FILE *fp, int *buf);

/* argv に入力があればそれを str に、、なければ入力を求める */
extern void putin_filename(char *arg, char *fname);

/* Sec from Ref Epoch と Ref Epoch から観測開始時刻を求める */
/* うるう年までは考慮。それ以上は適用外 */
extern void obsdate(int *data);
extern void fobsdate(int *data, FILE *out);

/* VDIF データの出力 */
extern int vdifoutput_r(int *data, FILE *out);
extern int vdifoutput_a(int *data, FILE *out);

#endif
```

ソースコード **4:** vdifreader.c

```
/*
  vdifreader.c
  VDIF ファイルの読み込みを担当

  2015/09/15 v0.01 仮作成
  2015/12/22 v0.02 datout に対応
  2015/12/26 v0.03 gnuplot block 用記述追加
 */

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include "vdifreader.h"

/* 32bit 分を int で取り出し、ビット表示に変換後、buf に格納 */
void keepbit(char *buf, int x){
  int i;
  for(i=31; i>=0; i--){
    buf[i] = (x >> i) & 1;
  }
}


/* 32bit の配列から、指定部分のみ取り出し 10 進数として表す */
int decimal(char *buf, int x, int y){
  int i, j, n;
  j=0;
  n=0;
  for(i=y; i<=x; i++){
    n = n + (buf[i] * pow(2,j));
    j++;
  }
  return n;
}

/* 64bit 分取り出し、double 型に変換 */
/*
  double 型
  符号 指数 11bit 仮数 52bit
  0 01111111011 1001100110011001100110011001100110011001100110011001100110011010
  指数は-1023から 1024、2^n の n 部を表す
  仮数は 1.nnnnn を示す
  */
double print_tau(int *buf, int line){
  int i;
  int *bit;
  int s = 0; // 符号、0:+ 1:-
```

```
49    int e = 0; // 指数
50    double f = 0; // 仮数
51    double d;
52
53    bit=(int*)malloc(sizeof(int)*64); // bit 配列保存用
54
55    /* bit 格納 */
56    for(i=63; i>=32; i--){
57      bit[i] = (buf[line + 1] >> (i-32)) & 1;
58    }
59    for(i=31; i>=0; i--){
60      bit[i] = (buf[line] >> i) & 1;
61    }
62
63    /* double 型へ */
64    s = 1 - 2*bit[63];
65    for(i=62; i>=52; i--){
66      e += bit[i]*pow(2, i-52);
67    }
68    for(i=51; i>=0; i--){
69      f += bit[i]*pow(2, i-52);
70    }
71
72    d = s*(1+f)*(pow(2, e-1023));
73    free(bit);
74    return d;
75  }
76
77  /* 16bit の数を符号付きに拡張 */
78  int signed16bit(int x){
79    int i;
80    if(x < pow(2,15)){
81      i = x;
82      return i;
83    }else{
84      i = x - pow(2,16);
85      return i;
86    }
87  }
88
89  /* buf の後ろに buf2 を結合 */
90  /* このプログラム群専用。他にコピペしても多分動きません */
91  /* buf の大きさをオーバーした時はどうなるかわかりません */
92  void intcat(int *buf, int *buf2){
93    int i, j;
94    i = 0;
95    /* buf の最後(\0)を検出 */
96    while(buf[i*328] != '\0'){
97      i++;
```

```
98    }
99    j = 0;
100   /* buf2 の最後まで buf の後ろに追加 */
101   while(j < 328){
102     buf[i*328 +j] = buf2[j];
103     j++;
104   }
105   /* buf2 リセット */
106   memset(buf2, '\0', sizeof(buf2));
107 }
108
109
110 /* VDIF を 1312Byte ごと読み出し、配列 buf2 に保存 */
111 /* その後、buf の最後に連結して一つの配列にする。 */
112 /* VDIF は 1312Byte が 1 セットで書かれている */
113 void binread(FILE *fp, int *buf){
114   int i = 328;
115   int *buf2;
116   buf2 = (int*)malloc(sizeof(int) * 328);
117   if(buf2 == NULL){
118     printf("DEBUG vdifreader.c \"binread\" malloc error");
119     exit(EXIT_FAILURE);
120   }
121   while(i == 328){
122     i = fread(buf2, sizeof(int), 328, fp);
123     intcat(buf, buf2);
124   }
125   free(buf2);
126 }
127
128 /* argv に入力があればそれを返し、なければ入力を求める */
129 void putin_filename(char *arg, char *fname){
130   if(arg == NULL){
131     fgets(fname, 256, stdin);
132   }else{
133     strcpy(fname, arg);
134     printf("%s\n", fname);
135   }
136   return;
137 }
138
139 /* Sec from Ref Epoch と Ref Epoch から観測開始時刻を求める */
140 /* うるう年までは考慮。それ以上は適用外 */
141 void obsdate(int *data){
142   int year, day, hour, min, sec;
143   int sec_from_epoch, ref_epoch;
144   char *header32bit;
145
146   header32bit = (char*)malloc(sizeof(char)*32);
```

```
147   keepbit(header32bit, data[0]);
148   sec_from_epoch = decimal(header32bit, 29, 0);
149   sec = sec_from_epoch % 60;
150   min = ((sec_from_epoch - sec) % 3600) / 60;
151   hour = ((sec_from_epoch - sec - (min * 60)) % (24 * 3600)) / 3600;
152   day = 1 + (sec_from_epoch - sec - (min * 60) - (hour * 3600)) / (24 *
          3600);
153
154   keepbit(header32bit, data[1]);
155   ref_epoch = decimal(header32bit, 29, 24);
156   if(ref_epoch % 2 == 0){
157     year = 2000 + (ref_epoch / 2);
158   }else{
159     year = 2000 + ((ref_epoch - 1)/2);
160     if(year % 4 == 0){
161       day = day + 182;
162     }else{
163       day = day + 181;
164     }
165   }
166
167   printf("ObsDate(UT) : %dy %dd %dh%dm%ds\n", year, day, hour, min, sec);
168 }
169
170 /* obsdate を fprintf に対応 */
171 void fobsdate(int *data, FILE *out){
172   int year, day, hour, min, sec;
173   int sec_from_epoch, ref_epoch;
174   char *header32bit;
175
176   header32bit = (char*)malloc(sizeof(char)*32);
177   keepbit(header32bit, data[0]);
178   sec_from_epoch = decimal(header32bit, 29, 0);
179   sec = sec_from_epoch % 60;
180   min = ((sec_from_epoch - sec) % 3600) / 60;
181   hour = ((sec_from_epoch - sec - (min * 60)) % (24 * 3600)) / 3600;
182   day = 1 + (sec_from_epoch - sec - (min * 60) - (hour * 3600)) / (24 *
          3600);
183
184   keepbit(header32bit, data[1]);
185   ref_epoch = decimal(header32bit, 29, 24);
186   if(ref_epoch % 2 == 0){
187     year = 2000 + (ref_epoch / 2);
188   }else{
189     year = 2000 + ((ref_epoch - 1)/2);
190     if(year % 4 == 0){
191       day = day + 182;
192     }else{
193       day = day + 181;
```

```
194     }
195   }
196
197   fprintf(out, "#ObsDate(UT) : %dy %dd %dh%dm%ds\n", year, day, hour, min,
          sec);
198 }
199
200
201 /* VDIF データの出力 */
202 /* ReIm で出力 */
203 int vdifoutput_r(int* data, FILE *out){
204   int ip = 0;
205   int ch = 0;
206   int chmax;
207   char *v32bit;
208
209   int Re, Im;
210
211   int i;
212
213   v32bit = (char*)malloc(sizeof(char)*32);
214
215   /* FFT 点 = CH 数を検出 */
216   keepbit(v32bit, data[4]);
217   if(decimal(v32bit, 15, 8) == 0){
218     chmax = 16;
219   }else if(decimal(v32bit, 15, 8) == 1){
220     chmax = 128;
221   }else{
222     printf("FFTpoints Undefined\n");
223     exit(1);
224   }
225
226   /* データ吸い出しと出力 */
227   while(data[ip*chmax*328 + 2] != 0){ // データの有無を確認
228     fprintf(out, "#IPNo %d\n", ip+1);
229     fprintf(out, "#Tau %.15e\n", print_tau(data, ip*chmax*328 + 17));
230     fprintf(out, "#Taudot %.15e\n", print_tau(data, ip*chmax*328 + 19));
231     fprintf(out, "#Taudotdot %.15e\n", print_tau(data, ip*chmax*328 +
          21));
232     fprintf(out, "#CH\tRe\tIm\n");
233     for(ch=0; ch<chmax; ch++){
234       for(i=72; i<328; i++){
235         keepbit(v32bit, data[(ip*chmax*328) + (ch*328) + i]);
236
237         Re = signed16bit(decimal(v32bit, 15, 0));
238         Im = signed16bit(decimal(v32bit, 31, 16));
239
240         fprintf(out, "%d\t%d\t%d\n", ch*256+i-72, Re, Im);
```

```
241
242       }// for(i)
243       if(ch == chmax-1){
244         fprintf(out, "\n"); // gnuplot 使用時のブロック用
245       }
246     }// for(ch)
247     ip++;
248   }// while
249   free(v32bit);
250   return ip;
251 }
252
253 /* AmpPhase で出力 */
254 int vdifoutput_a(int* data, FILE *out){
255   int ip = 0;
256   int ch = 0;
257   int chmax;
258   char *v32bit;
259
260   double Amp, Phase;
261
262   int i;
263
264   v32bit = (char*)malloc(sizeof(char)*32);
265
266   /* FFT 点 = CH 数を検出 */
267   keepbit(v32bit, data[4]);
268   if(decimal(v32bit, 15, 8) == 0){
269     chmax = 16;
270   }else if(decimal(v32bit, 15, 8) == 1){
271     chmax = 128;
272   }else{
273     printf("FFTpoints Undefined\n");
274     exit(1);
275   }
276
277   /* データ吸い出しと出力 */
278   while(data[ip*chmax*328 + 2] != 0){ // データの有無を確認
279     fprintf(out, "#IPNo %d\n", ip+1);
280     fprintf(out, "#Tau %.15e\n", print_tau(data, ip*chmax*328 + 17));
281     fprintf(out, "#Taudot %.15e\n", print_tau(data, ip*chmax*328 + 19));
282     fprintf(out, "#Taudotdot %.15e\n", print_tau(data, ip*chmax*328 +
          21));
283     fprintf(out, "#CH\tRe\tIm\n");
284     for(ch=0; ch<chmax; ch++){
285       for(i=72; i<328; i++){
286         keepbit(v32bit, data[(ip*chmax*328) + (ch*328) + i]);
287
288         Amp = sqrt(pow(signed16bit(decimal(v32bit, 15, 0)),2) + pow(
```

```
            signed16bit(decimal(v32bit, 31, 16)),2));
289         Phase = atan2(signed16bit(decimal(v32bit, 31, 16)), signed16bit(
                decimal(v32bit, 15, 0))) *180 / acos(-1);
290
291         fprintf(out, "%d\t%.4lf\t%.4lf\n", ch*256+i-72, Amp, Phase);
292
293       }// for(i)
294       if(ch == chmax-1){
295         fprintf(out, "\n"); // gnuplot 使用時のブロック用
296       }
297     }// for(ch)
298     ip++;
299   }// while
300   free(v32bit);
301   return ip;
302 }
```

vdif_calc

ソースコード 5: vdif_calc_openmp.c

```
1  /*
2    gcc -fopenmp -lgomp vdif_calc_openmp.c -o vdif_calc -lm
3
4    datout で作成したファイルを元に計算するプログラム
5
6    v0.01 2015/12/26
7    v0.02 2016/01/12 アラン分散の計算方法変更と並列化
8    v0.05 2016/01/29 -f, -l でもデータをメモリ上に記録するように変更
9                     -x, -y 有効化
10   v0.06 2016/02/01 コアダンプエラー対策
11  */
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <string.h>
16 #include <math.h>
17 #include <unistd.h>
18 #include <omp.h>
19 // #include <errno.h>
20
21 #define CHMAX 4096
22 #define IPMAX 30000
23
24 typedef struct vdifdata_str{
25   int *Re;
26   int *Im;
```

```
27 }Vdifdata;
28
29 int main(int argc, char *argv[]){
30
31   int opt, opt_l, range, loopmax, j;
32   char optstr[10]; // コマンドラインオプションの保存用
33
34   FILE *vdat, *odat, *xdat, *ydat;
35
36   char *datname, *outname, *xname, *yname; // datout 出力ファイル名,
         calc 出力ファイル名, X 局の datout ファイル名, Y 局の
37   char *row, *notuse; // 1 行分のデータ保存用, 数値以外のデータを入れるためだ
         けの箱
38   char *err, rs[6];
39   int i = 0;
40   int loop = 0; // アラン分散用､読み直し回数
41   int n = 0; // アラン分散用､
42   int ip, ch, ipmax, chmax;
43   int Re1, Im1;
44   double A = 0; // 計算の経由地
45   double B = 0; // 計算経由地
46   double Ax= 0;
47   double Ay= 0;
48   double Amp, Phase;
49   double R1[10], R2[10], Sigma[8]; // アラン分散用
50   // 10, 510, 710, 810, 910, 1010, 1110, 1210, 1510, 2010
51
52   Vdifdata *datastr, *xstr, *ystr;
53
54   // #pragma omp threadprivate(i, n, j, R1, R2, Sigma, ip, ch)
55
56   /* メモリ確保 */
57   row = (char*)malloc(sizeof(char)*256);
58   notuse = (char*)malloc(sizeof(char)*256);
59   datname = (char*)malloc(sizeof(char)*256);
60   outname = (char*)malloc(sizeof(char)*256);
61   xname = (char*)malloc(sizeof(char)*256);
62   yname = (char*)malloc(sizeof(char)*256);
63   datastr=(Vdifdata*)malloc(sizeof(Vdifdata)*IPMAX);
64   for(i=0;i<IPMAX;i++){
65     datastr[i].Re = (int*)malloc(sizeof(int)*CHMAX);
66     datastr[i].Im = (int*)malloc(sizeof(int)*CHMAX);
67   }
68   xstr=(Vdifdata*)malloc(sizeof(Vdifdata)*IPMAX);
69   for(i=0;i<IPMAX;i++){
70     xstr[i].Re = (int*)malloc(sizeof(int)*CHMAX);
71     xstr[i].Im = (int*)malloc(sizeof(int)*CHMAX);
72   }
73   ystr=(Vdifdata*)malloc(sizeof(Vdifdata)*IPMAX);
```

```
74  for(i=0;i<IPMAX;i++){
75    ystr[i].Re = (int*)malloc(sizeof(int)*CHMAX);
76    ystr[i].Im = (int*)malloc(sizeof(int)*CHMAX);
77  }
78
79
80  while((opt = getopt(argc, argv, "fva:l:x:y:o:")) != -1){
81    switch(opt){
82    case 'f':// fringe
83      strcat(optstr, "f");
84      break;
85    case 'v':// vspe 未実装
86      strcat(optstr, "v");
87      break;
88    case 'a':// arran
89      strcat(optstr, "a");
90      loopmax = strtol(optarg, &err, 10);
91      if(strlen(err) != 0){
92        fprintf(stdout, "Option %c error\n", opt);
93        fprintf(stdout, "Set natural number in option a\n");
94        fprintf(stdout, "Program stop running\n");
95        goto END;
96      }
97      break;
98    case 'l':// line
99      strcat(optstr, "l");
100     opt_l = strtol(optarg, &err, 10);
101     if(strlen(err) != 0){
102       fprintf(stdout, "Option %c error\n", opt);
103       fprintf(stdout, "Set natural number in option l\n");
104       fprintf(stdout, "Program stop running\n");
105       goto END;
106     }
107     break;
108   case 'x':// x 正規化用
109     strcpy(xname, optarg);
110     strcat(optstr, "x");
111     xdat = fopen(xname, "r");
112     if(xdat == NULL){
113       fprintf(stdout, "Xdat fopen failed\n");
114       goto END;
115     }
116     break;
117   case 'y':// y 正規化用
118     strcpy(yname, optarg);
119     strcat(optstr, "y");
120     ydat = fopen(yname, "r");
121     if(ydat == NULL){
122       fprintf(stdout, "Ydat fopen failed\n");
```

```
123         goto END;
124       }
125       break;
126     case 'o':// output
127       strcpy(outname, optarg);
128       strcat(optstr, "o");
129       break;
130     case ':':
131       fprintf(stdout, "Option %c needs value\n", opt);
132       fprintf(stdout, "Program stop running\n");
133       goto END;
134       break;
135     case '?':
136       fprintf(stdout, "Unknown option\n");
137       fprintf(stdout, "Program stop running\n");
138       goto END;
139       break;
140     }
141   }
142
143   strcpy(datname, argv[optind]);
144   if(strchr(optstr, 'o') == NULL){
145     strcpy(outname, "output.dat");
146   }
147
148   vdat = fopen(datname, "r");
149   odat = fopen(outname, "w");
150
151
152   /* 読み込み */
153   while(fgets(row, 256, vdat) != 0){ // fgets は改行記号("\n")まで読み込み、最
          後に"\0"を入れる
154     if(strstr(row, "#") != NULL){
155       // fprintf(stdout, "%s", row);
156       if((strstr(row, "#CH") == NULL && strstr(row, "#IPNo") == NULL) &&
              strstr(row, "Tau") == NULL){
157         fprintf(odat, "%s", row);
158       }
159
160       /* 諸元から各パラメータ取得 */
161       if(strstr(row, "FFT_points") != NULL){
162         sscanf(row, "%[^:]:%d", notuse, &chmax);
163         fprintf(stdout, "DEBUG FFT_Points\n");
164         fprintf(stdout, "chmax = %d\n", chmax);
165       }
166       if(strstr(row, "Total_IP") != NULL){
167         sscanf(row, "%[^:]: %d", notuse, &ipmax);
168         fprintf(stdout, "ipmax = %d\n", ipmax);
169       }
```

```
170       if(strstr(row, "IPNo") != NULL){
171         sscanf(row, "%s %d", notuse, &ip);
172       }
173
174       /* 出力タイプと設定の書き込み */
175       if(strstr(row, "Outputtype") != NULL){
176
177         if(strchr(optstr, 'f') != NULL){
178           fprintf(odat, "#CalcType : f\n");
179         }else if(strchr(optstr, 'v') != NULL){
180           fprintf(odat, "#CalcType : v\n");
181         }else if(strchr(optstr, 'a') != NULL){
182           fprintf(odat, "#CalcType : a arranplot loopmax=%d\n", loopmax);
183           fprintf(odat, "#IPpoints ch1010\tch1010-ch1110\tch910-ch1110\
                tch910-ch1210\tch810-ch1210\tch710-ch1210\tch510-ch1510\tch10
                -ch2010\n");
184
185         }else if(strchr(optstr, 'l') != NULL){
186           fprintf(stdout, "Peak searching range (+- CH) : ");
187
188           /* ピークを追いかけるCH幅を入力 */
189           fgets(rs, 6, stdin);
190           if(strchr(rs, '\n') == NULL){
191             while(getchar() != '\n');
192           }
193           fprintf(stdout, "rs = %s\n", rs);
194           strcpy(err, "\0");
195           range = strtol(rs, &err, 10);
196           if(strcmp(err, "\n") != 0 && strlen(err) != 0){
197             fprintf(stdout, "Set natural number in range\n");
198           }
199
200           /* 計算の条件とかを出力 */
201           fprintf(odat, "#CalcType : l %d %d\n", opt_l, range);
202         }// if(optstr)
203
204       }// if(Outputtype)
205
206     }else if(strcmp(row, "\n") == 0){
207       /* 改行のみを読み飛ばす */
208     }else if(strstr(row, "#") == NULL){
209       sscanf(row, "%d %d %d", &ch, &Re1, &Im1);
210       /* 保管 */
211
212         datastr[ip].Re[ch] = Re1;
213         datastr[ip].Im[ch] = Im1;
214
215     }// else(row, #)
216   }// while(fgets)
```

```
217
218   /* 計算 */
219   if(strchr(optstr, 'a') != NULL){
220     fprintf(stdout, "num max procs: %d %d %d\n", omp_get_num_threads(),
          omp_get_max_threads(), omp_get_num_procs());
221
222       /* 平均を求める */
223 #pragma omp parallel private(i, n, j, R1, R2, Sigma, ip, ch)
224     {
225       i=0;
226 #pragma omp for schedule(dynamic)
227       for(loop=0;loop<loopmax;loop++){
228         for(ip=0;ip<ipmax;ip++){
229           if(i <= loop){
230             for(ch=0;ch<CHMAX;ch++){
231
232               switch(ch){
233               case 10:
234                 R1[0] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch10
235                 break;
236               case 510:
237                 R1[1] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch510
238                 break;
239               case 710:
240                 R1[2] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch710
241                 break;
242               case 810:
243                 R1[3] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch810
244                 break;
245               case 910:
246                 R1[4] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch910
247                 break;
248               case 1010:
249                 R1[5] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch1010
250                 break;
251               case 1110:
252                 R1[6] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch1110
253                 break;
254               case 1210:
255                 R1[7] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch1210
256                 break;
```

```
257             case 1510:
258               R1[8] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch1510
259               break;
260             case 2010:
261               R1[9] += (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(
                      datastr[ip].Im[ch], 2));// ch2010
262               break;
263             }// switch(ch)
264           }// for(ch)
265           i++;
266         }// if(ip==loop)
267
268         /* 分散を求める */
269         if(i > loop){
270           if(R2[0] != 0){
271             Sigma[0] += pow((R1[5] - R2[5])/(loop+1), 2); // ch1010
272
273             Sigma[1] += pow(((R1[5] - R1[6]) - (R2[5] - R2[6]))/(loop+1),
                    2);// ch1010 - ch1110
274             Sigma[2] += pow(((R1[4] - R1[6]) - (R2[4] - R2[6]))/(loop+1),
                    2);// ch910 - ch1110
275             Sigma[3] += pow(((R1[4] - R1[7]) - (R2[4] - R2[7]))/(loop+1),
                    2);// ch910 - ch1210
276             Sigma[4] += pow(((R1[3] - R1[7]) - (R2[3] - R2[7]))/(loop+1),
                    2);// ch810 - ch1210
277             Sigma[5] += pow(((R1[2] - R1[7]) - (R2[2] - R2[7]))/(loop+1),
                    2);// ch710 - ch1210
278             Sigma[6] += pow(((R1[1] - R1[8]) - (R2[1] - R2[8]))/(loop+1),
                    2);// ch510 - ch1510
279             Sigma[7] += pow(((R1[0] - R1[9]) - (R2[0] - R2[9]))/(loop+1),
                    2);// ch10 - ch2010
280           }
281           for(j=0; j<10; j++){
282             R2[j] = R1[j];
283             R1[j] = 0;
284           }
285           i = 0;
286           n++;
287         }// if(i>loop)
288
289       }// for(ip)
290
291       fprintf(stdout, "loop i n : %d %d %d\n", loop, i, n);
292       fprintf(odat, "%d\t%.9e\t%.9e\t%.9e\t%.9e\t%.9e\t%.9e\t%.9e\t%.9e\n
              ", loop+1, Sigma[0]/n, Sigma[1]/n, Sigma[2]/n, Sigma[3]/n,
              Sigma[4]/n, Sigma[5]/n, Sigma[6]/n, Sigma[7]/n);
293       i = 0;
294       n = 0;
```

```
295         for(j=0; j<10; j++){
296           R1[j] = 0;
297           R2[j] = 0;
298           if(j<8){
299             Sigma[j] = 0;
300           }
301         }
302       }// for(loop)
303     }// parallel
304   }// if(optstr,a)
305 
306   else if(strchr(optstr, 'f') != NULL){
307     /* IP ごとの周波数積分強度 */
308     // sscanf(row, "%d %d %d", &ch, &Re, &Im);
309 
310     /* 正規化用自己相関データの取り出し */
311     if(strchr(optstr, 'x') != NULL && strchr(optstr, 'y') != NULL){
312       while(fgets(row, 256, xdat) != 0){
313         if(strstr(row, "#") != NULL){
314           if(strstr(row, "IPNo") != NULL){
315             sscanf(row, "%s %d", notuse, &ip);
316           }
317         }else if(strcmp(row, "\n") == 0){
318           /* 改行のみを読み飛ばす */
319         }else if(strstr(row, "#") == NULL){
320           sscanf(row, "%d %d %d", &ch, &Re1, &Im1);
321           xstr[ip].Re[ch] = Re1;
322           xstr[ip].Im[ch] = Im1;
323         }// strstr(#)
324       }// while(fgets)
325 
326       while(fgets(row, 256, ydat) != 0){
327         if(strstr(row, "#") != NULL){
328           if(strstr(row, "IPNo") != NULL){
329             sscanf(row, "%s %d", notuse, &ip);
330           }
331         }else if(strcmp(row, "\n") == 0){
332           /* 改行のみを読み飛ばす */
333         }else if(strstr(row, "#") == NULL){
334           sscanf(row, "%d %d %d", &ch, &Re1, &Im1);
335           ystr[ip].Re[ch] = Re1;
336           ystr[ip].Im[ch] = Im1;
337         }// strstr(#)
338       }// while(fgets)
339 
340     }// if(x!=NULL && y!=NULL)
341 
342     for(ip=0;ip<ipmax;ip++){
343       for(ch=1;ch<(CHMAX/2)-1;ch++){
```

```
344         A += datastr[ip].Re[ch];
345         B += datastr[ip].Im[ch];
346 
347         if(strchr(optstr, 'x') != NULL && strchr(optstr, 'y') != NULL){
348           Ax += xstr[ip].Re[ch];
349           Ay += ystr[ip].Re[ch];
350         }
351       }// for(ch)
352 
353       if(strchr(optstr, 'x') != NULL && strchr(optstr, 'y') != NULL){
354         Amp = (sqrt(pow(A/(CHMAX/2), 2) + pow(B/(CHMAX/2), 2))) / (sqrt(abs
              (Ax)/(CHMAX/2)) * sqrt(abs(Ay)/(CHMAX/2)));
355         Phase = atan2(B, A)*180/acos(-1);
356         fprintf(odat, "%d\t%f\t%f\n", ip, Amp, Phase);
357         A = 0;
358         B = 0;
359         Ax= 0;
360         Ay= 0;
361       }else{
362         Amp = sqrt(pow(A/(CHMAX/2), 2) + pow(B/(CHMAX/2), 2));
363         Phase = atan2(B, A)*180/acos(-1);
364         fprintf(odat, "%d\t%f\t%f\n", ip, Amp, Phase);
365         A = 0;
366         B = 0;
367       }
368     }// for(ip)
369 
370   }else if(strchr(optstr, 'v') != NULL){
371 
372   }else if(strchr(optstr, 'l') != NULL){
373     for(ip=0;ip<ipmax;ip++){
374       for(ch=0;ch<CHMAX;ch++){
375         if(ch >= opt_l-range && ch <= opt_l+range){
376           A = (double)sqrt(pow(datastr[ip].Re[ch], 2) + pow(datastr[ip].Im[
                ch], 2));
377           B = (double)atan2(datastr[ip].Im[ch], datastr[ip].Re[ch])*180/
                acos(-1);
378           if(Amp < A){
379             Amp = A;
380             Phase = B;
381           }
382         }
383       }// for(ch)
384       fprintf(odat, "%d\t%f\t%f\n", ip, Amp, Phase);
385       A = 0;
386       B = 0;
387       Amp = 0;
388       Phase = 0;
389     }// for(ip)
```

```
390   }// if(optstr, l)
391
392   fprintf(stdout, "END LOOP ipmax: %d %d\n", loop, ipmax);
393
394
395
396  END:
397
398   for(i=IPMAX-1;i>=0;i--){
399     free(ystr[i].Im);
400     free(ystr[i].Re);
401   }
402   free(ystr);
403   for(i=IPMAX-1;i>=0;i--){
404     free(xstr[i].Im);
405     free(xstr[i].Re);
406   }
407   free(xstr);
408   for(i=IPMAX-1;i>=0;i--){
409     free(datastr[i].Im);
410     free(datastr[i].Re);
411   }
412   free(datastr);
413
414   free(yname);
415   free(xname);
416   free(outname);
417   free(datname);
418   free(notuse);
419   free(row);
420   fclose(odat);
421   fclose(vdat);
422   if(strchr(optstr, 'x') != NULL){
423     fclose(xdat);
424   }
425   if(strchr(optstr, 'y') != NULL){
426     fclose(ydat);
427   }
428   return 0;
429 }
```

B 出力データプロット用バッチファイル

ソースコード **6:** gnu.txt

```
1 # datout を連続表示するためのもの。
2 # 適宜数値やファイル名を変えてお使いください。
```

```
3
4  if(exists("start")==0 || start<0) start=0  # 開始IP
5  if(exists("stop")==0 || stop<start) stop=10  # 終了IP
6  if(exists("name")==0)  name="test.dat" # ファイル名
7  if(exists("xmax")==0)  xmax=4096  # X 軸(CH)最大値
8  if(exists("ymax")==0)  ymax=3000  # 左Y 軸(Amp)最大値
9  if(exists("i")==0)  i=0  # 繰り返し回数
10 if(exists("ptime")==0)  ptime=0.1  # 停止時間
11
12 if(i==0 && exists("outname")!=0) set term pngcairo enhanced size 800,480
13 #if(exists("outname")==0) outname="gnuplot"
14 if(exists("outname")!=0) set out sprintf("%s_%d.png", outname, start+i)
15
16
17 if(i==0) set xr[0:xmax];\
18         set ytics nomirror;\
19         set y2tics;\
20         set yr[0:ymax];\
21         set y2r[-180:180];\
22         set xl "CH";\
23         set yl "Amp";\
24         set y2l "Phase"
25
26 set title sprintf("IP = %d", start+i)
27 plot name u 1:3 every :::start+i::start+i axis x1y2 title "Phase" w l lc
       2,\
28      name u 1:2 every :::start+i::start+i title "Amp" w l lc 1;\
29 pause ptime      # 停止時間 処理速度の関係上、一定より短くならなさそう
30
31 if(start+i<stop) i=i+1;\
32        reread
33
34 set term wxt
35 #set term x11
36
37 undefine i
38 #undefine start
39 #undefine stop
40 undefine name
41 undefine outname
42 #undefine xmax
43 #undefine ymax
```

ソースコード 7: arranplot.txt

```
1  # gnuplot で arranplot
2
3  if(exists("name")==0) name="../arran/arranplot_el15_6.dat" # ファイル名
4
```

```
set logscale xy
set key left bottom
set yr[1e-6:10]
p name u 1:2 w p pt 7 ps 1 lc 1 title "ch1010"
rep name u 1:3 w p pt 7 ps 1 lc 2 title "ch1010-ch1110"
rep name u 1:4 w p pt 7 ps 1 lc 3 title "ch910-ch1110"
rep name u 1:5 w p pt 7 ps 1 lc 4 title "ch910-ch1210"
rep name u 1:6 w p pt 7 ps 1 lc 5 title "ch810-ch1210"
rep name u 1:7 w p pt 7 ps 1 lc 6 title "ch710-ch1210"
rep name u 1:8 w p pt 7 ps 1 lc 7 title "ch510-ch1510"
rep name u 1:9 w p pt 7 ps 1 lc 8 title "ch10-ch2010"

if(exists("outname")!=0) set term pngcairo enhanced size 1920,1080 font "
    Arial,32";\
                      set border lw 4
                      set output outname;\
                      rep;\
                      set term wxt

unset logscale xy
set border lw 1
undefine name
undefine outname
```