卒業論文

3DCRアンテナのビームサイズの測定と
それを用いた天の川銀河の回転曲線の導出

指導教員　百瀬宗武 , 米倉覚則

茨城大学
理学部　理学科　物理学コース

22S2012A　川俣覚

概要

3DCR （ コーナーリフレクター ） アンテナは 3 枚の金属板とモノポールによって構成されているアンテナである。 3 枚の金属板をそれぞれ直交するように接続することで、特定の方向に強く電波を反射し、指向性を高めるとともに入射してくる電波を集め、設置したモノポールで受信する仕組みである。

本研究では BBQ 用の焼き網を使用した自作の 3DCR アンテナを用いて、太陽のドリフト観測と 21 cm 線の観測を行った。太陽のドリフト観測の目的はアンテナのビームサイズを測定して性能を評価することをであった。観測の結果、正確なビームサイズを求めることはできなかったが、ビーム中心の感度が最も高く中心から遠ざかるにつれて感度が減少していく様子が確認された。

21 cm 線の観測からはタンジェントポイント法を用いて天の川銀河の回転曲線を導出することを目的とした。その結果、銀河中心から半径 4 - 8 kpc で回転速度が大幅に変化せず、先行研究の結果と矛盾しない回転曲線を導出できることが確認された。

目次

1 **はじめに** 4

1.1 研究目的 4

1.2 21cm 線 4

1.3 回転曲線 5

1.4 タンジェントポイント法 5

2 **観測装置** 6

2.1 アンテナ部 6

2.2 受信部とデータ取得部 10

3 **太陽のドリフト観測** 10

3.1 観測手法 10

3.2 結果 12

3.3 考察 13

4 **21cm 線の観測** 13

4.1 観測手法 13

4.2 結果 14

4.3 考察 16

5 **今後の展望** 16

6 **謝辞** 16

7 **付録** 17

7.1 太陽のドリフト観測の観測コード 17

7.2 ドリフト観測の結果をガウシアンフィットするコード 21

7.3 銀河座標を地表座標に変換するコード 22

7.4 太陽の位置を調べるコード 23

7.5 21 cm 線の観測コード 23

7.6 21 cm 線の周波数スペクトルをベースラインフィットするコード 28

7.7 周波数スペクトルを視線速度スペクトルに変換するコード 29

7.8 局所静止基準に対する観測者の速度補正のコード 31

8 **参考文献** 33

1　はじめに

1.1　研究目的

3D コーナーリフレクターアンテナ（以後 3DCR アンテナと呼ぶ）は 3 枚の金属板とモノポールによって構成されているアンテナである。3 枚の金属板をそれぞれ直交するように接続することで、特定の方向に強く電波を反射し、指向性を高めるとともに入射してくる電波を集め、設置したモノポールで受信する仕組みである。

本研究の目的は大きく分けて 2 つある。1 つ目は 3DCR アンテナを用いて太陽のドリフト観測を行い、その結果からアンテナのビームサイズを求め、アンテナの性能を評価することを目的とした。

2 つ目の目的は 21 cm 線の観測を行い、その結果にタンジェントポイント法を適用して天の川銀河の回転曲線を導出することを目的とした。

1.2　21cm 線

21 cm 線とは中性水素原子の超微細構造によって現れる 2 つのエネルギー準位間の遷移により放出または吸収される波長 21.106114 cm、周波数 1420.40575 MHz のスペクトル線のことである。

中性水素原子は陽子（原子核） 1 個と電子 1 個が結合した状態であり、陽子と電子はともに大きさが $\frac{1}{2}$ のスピン角運動量を持っている。中性水素原子は原子核の磁気モーメントと電子の磁気モーメントとの相互作用により、中性水素原子の基底状態はわずかに異なる 2 つのエネルギー準位をもつ。これを超微細構造と呼ぶ。

原子核と電子のスピン角運動量の向きが同じ（平行）である状態の方がエネルギーが高く、両者の向きが反対（反平行）である状態の方がエネルギーが低い。つまり、原子核と電子のスピンの向きが平行から反平行に遷移した際に、そのエネルギー準位の差に相当する電磁波が放出される。その中性水素原子から放出された電磁波が 21 cm 線と呼ばれている。

天の川銀河には星間物質と呼ばれるダストやガスが存在している。その中でも中性水素原子ガスは最も豊富に存在するガスである。本研究では 21cm 線 を観測する際に天の川銀河の中性水素原子ガス（HI ガス）から放出される 21 cm 線を対象とする。

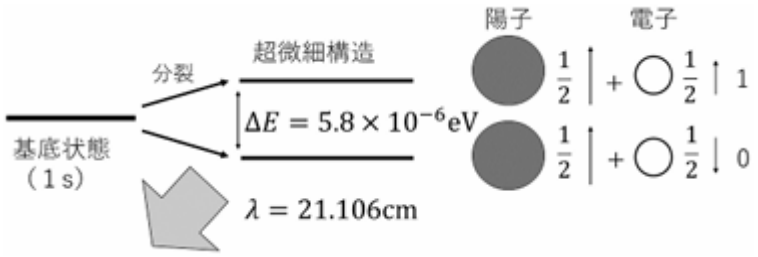

図 1: 21cm 線の放射機構 ([1] 谷敷怜空, 2024, 天文月報, 117, 315 より引用)

1.3 回転曲線

円盤銀河において、銀河円盤内の星や星間物質は銀河中心の周りを同じ方向に回転している。回転曲線とは銀河中心からの距離と回転速度の関係を表す曲線のことである。多くの円盤銀河の回転曲線では、銀河中心から剛体回転のように立ち上がって数 kpc の距離で最大値となり、それより遠くでは円盤の端までほぼ水平を保っている。

天の川銀河も円盤銀河の一つであり、回転曲線の平坦部では回転速度が約 200 km/s のまま保たれていることが知られている。

1.4 タンジェントポイント法

タンジェントポイント法とは内銀河の 21 cm 線スペクトルから求められる最大視線速度 $v_{\max}$ を用いて内銀河の回転速度を測定する方法である。

天の川銀河には中性水素原子ガスが豊富に存在しており、天の川銀河の中性水素原子ガスからは常に 21 cm 線が放出されている。

図 2 は天の川銀河面を上から見た様子である。図 2 のように、天の川銀河の太陽系にいる観測者より内側の領域では、ある視線方向において銀河中心を中心に持ち、かつ視線上に接点を持つ円が描ける。このとき、銀河中心から接点までの距離、すなわち円の半径を R とおき、銀河中心から太陽系までの距離を R_0 とおく。また、観測者の視線方向の銀経を l とおくと、図 2 から次の式 (1) が求まる。

$$R = R_0 \sin l \cdots (1)$$

また、接点にある HI ガスの回転速度を v とおき、太陽系の回転速度を v_0 とおく。観測者が銀経 l の方向を向いて観測するとき、視線上に多くの HI ガスが存在するが、接点上にある HI ガスは他の視線上の HI ガスより回転速度が視線上に最も強く投影されるので、観測される視線速度の中で値が最も大きい速度が接点の HI ガスの回転速度に対応すると考えられる。

このとき観測される最大視線速度を $v_{\max}$ をおくと相対速度の関係から次の式 (2) が求まる。

$$v = v_{\max} + v_0 \sin l \cdots (2)$$

観測した銀経 l と最大視線速度 $v_{\max}$ を式 (1) と式 (2) に代入すると v と R の関係を求めることができる。その結果、天の川銀河の回転曲線を導出することができる。このように 21 cm 線スペクトルの観測から内銀河の回転曲線を導出する手法がタンジェントポイント法と呼ばれる。

一方で、天の川銀河の $90^\circ < l < 270^\circ$ の外銀河では、視線上に接点を持つ円を描くことができず、最大視線速度と接点を対応させることができないため、タンジェントポイント法では外銀河の回転曲線を導出することはできない。

最大視線速度を求めるためには、銀河面上の銀経 l を観測した時に得られる 21 cm 線の周波数スペクトルをドップラーシフトの式を用いて視線速度スペクトルに変換する。変換するための式は

$$v = c\frac{f_0 - f}{f} \cdots (3)$$

となる。このとき、v は観測対象の視線速度、c は光速、f は観測された周波数、f_0 は静止周波数である。

また、21 cm 線を地球上から観測した際、地球の自転と公転、太陽の固有運動を考慮した補正が必要になる。 v_{corr} を局所静止基準に対する観測者の速度を考慮したときの補正値としたとき、(2) は実際には

$$v = v_{\mathrm{max}} + v_0 \sin\theta + v_{\mathrm{corr}} \cdots (4)$$

と書き換えられる。

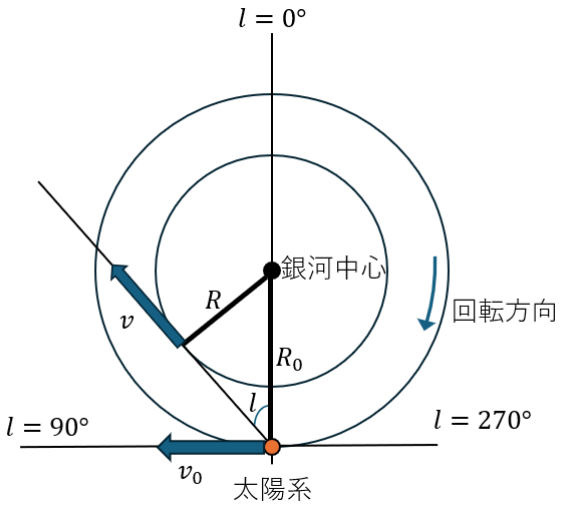

図 2: 円盤銀河の回転の様子

2　観測装置

本研究では BBQ 用の焼き網を用いた 3D コーナーリフレクターアンテナと観測に用いるための枠と台を作成した。本節ではそれらの設計や作成方法について説明する。

2.1　アンテナ部

2.1.1　3DCR アンテナ

3DCR アンテナは 3 枚の金属板とモノポールによって構成されているアンテナである。本研究で作成した 3DCR アンテナの設計は根本靖彦 ほか.(2026) [5] の設計を参考にした。本研究でも先行研究と同様に BBQ 用の焼き網を用いて反射面とした。はじめに 1 枚の大きさが 40 × 40 cm で網目の大きさが約 1.2 cm の BBQ 用の焼き網 12 枚を用意して、 4 枚の焼き網を 1 枚の正方形になるように結束バンドで接続した。すると 80 × 80 cm の反射面ができる。この反射面を 3 枚作成し、各辺を図 3 の形になるように結束バンドで接続する。次に反射面のそれぞれの辺に補強のため園芸用の支柱を取り付ける。側面となる部分を直立させ側面同士が接続する部分を紐で固定すると、大きさが 80 × 80 × 80 cm となるコーナーリフレクターアンテナとなる。

受信用のモノポールは外径 3 mm、長さが 15.5 cm の真鍮丸管を用いた。SMA 同軸コネクタを各側面から 12.5 cm 離れた底面に結束バンドで取り付け、モノポールを SMA 同軸コネクタに挿入する。

反射面に穴が開いていても、穴の大きさが観測する波長より十分小さければ電波を反射することができるという性質がある。本研究で観測する波長は 21 cm であり、網目の大きさが 1.2 cm であるため、観測する電磁波の波長に比べて穴の大きさは十分小さいと考えられる。

3DCR アンテナを組み立てる際は、反射面同士の接続部に隙間ができていないかどうかを確認する。隙間ができていた場合は結束バンドや紐を使って隙間を閉じることで、反射面同士が電気的に導通するようにする。

また、底面の反射面と SMA 同軸コネクタの接続に隙間ができていないかどうかを確認し、加えて SMA 同軸コネクタとモノポールが確実に接続しているかを確認する。できていない場合はアルミ箔などで隙間を埋めて、底面の反射面とモノポールを電気的に導通させる必要がある。実際に作成した 3DCR アンテナを図 4 に示した。

3DCR アンテナのメインビームの感度が最大になる向き、すなわちビームの中心方向は各反射面に対して 45° 傾いた向きになる [5]。この 3DCR アンテナのビームサイズは円対称ではないものの平均して約 25° 程度である [5]。

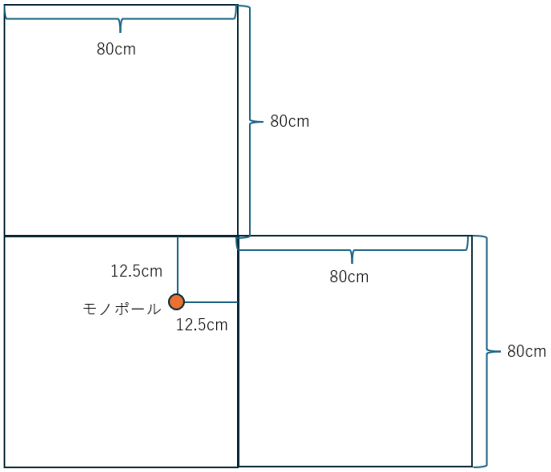

図 3: 3DCR アンテナの展開図

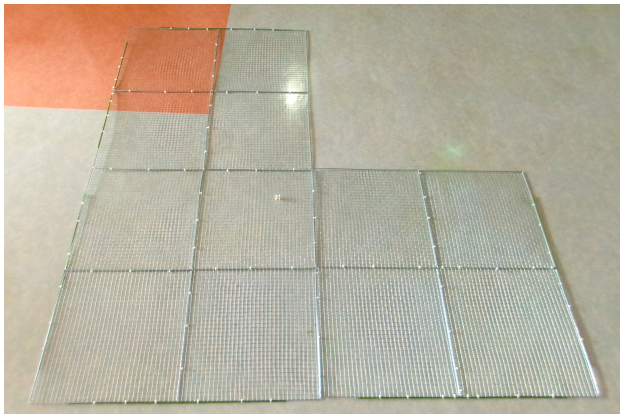

図 4: 作成した 3DCR アンテナを展開した写真

しかし、焼き網だけでは、 3DCR を傾けた際に反射面が自重で大幅に歪んでしまい観測結果に悪影響を与える可能性がある。また、 3D コーナーリフレクターアンテナだけではアンテナの向きを目的の方向に固定することができない。そこでそれらの問題を解決するために枠と台を作成した。

2.1.2 枠

3DCR アンテナだけでは焼き網自身の自重により反射面の形状が大幅に歪んでしまったり、観測対象を狙うためにアンテナを傾けると反射面同士の接続角度が変化してしまったりする。それらの問題を解決するために 3DCR アンテナの形状を固定する枠を作成した。枠は土台が 1 つと柱が 2 つで構成されており、持ち運びが不便にならないようにパチン錠でそれぞれ取り外しが可能な設計にした。必要な長さに切断した木材を釘と直線金具と L 字金具を用いて図 5 のように接続した。組み立てる際は土台と柱に取り付けてあるパチン錠で柱が自立するように固定し、柱同士を縄で結んで固定することで枠が完成する。実際に作成した枠を図 6 に示した。

組み立てた 3DCR アンテナを縄で枠に固定することによって、観測中に反射面の形状や反射面同士の角度が変化するのを防ぐことができる。

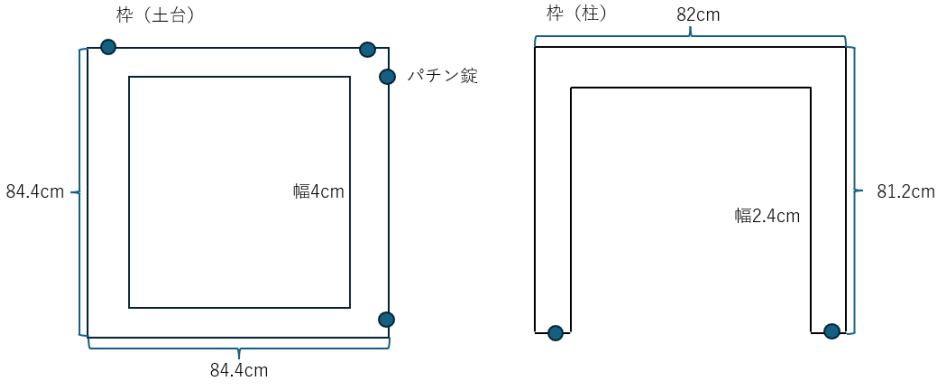

図 5: 枠の設計図

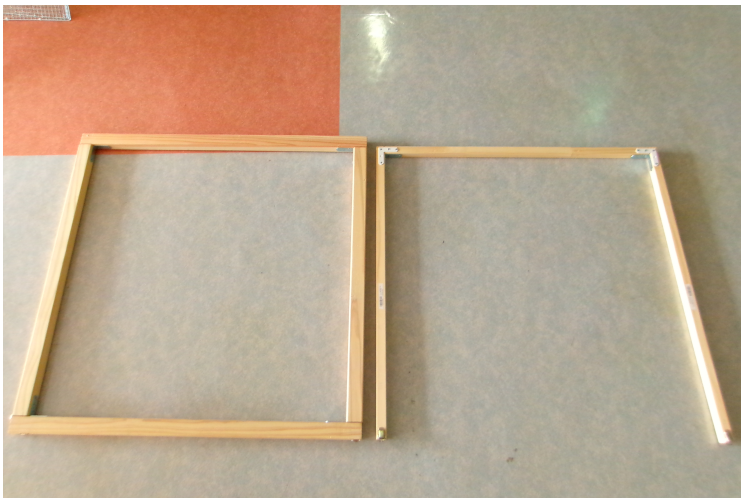

図 6: 作成した枠の写真

2.1.3 台

3DCR アンテナのメインビームの感度が最大になる向きは各反射面に対して 45° 傾いた向きになる。3DCR アンテナを水平な地面に設置した場合、メインビームの中心は仰角が 45° になるような方向を向いていることになる。しかし、アンテナを水平に設置したままでは仰角が 45° 付近の対象しか観測できないため、アンテナを傾けてビーム中心の仰角を自由に調節するための台を作成した。

台は大きさ 120 × 60 cm、厚さ 5 mm の木板を 2 枚蝶番で接続し、上面に L 字金具を 2 つ固定するという設計とした。板と板の間に長さを調節したつっぱり棒を設置し、上面の板の傾きを変えることで、上面に乗っているアンテナの傾きを調整することができる。

仰角が 45° 以上の対象を狙う際は図 7 のように取り付けることで、3DCR アンテナのビーム中心方向の仰角を 45° 以上に調節することができる。また、仰角が 45° 以下の対象を狙う際は図 8 のように取り付けることで、3DCR アンテナのビーム中心方向の仰角を 45° 以下に調節することができる。

図 9 は 3DCR アンテナを台に乗せた状態を上から見た様子である。この台を用いることでアンテナのビーム中心を自由に調節することができ、仰角が 0° − 90° にある対象を狙うことが可能になる。また、実際に 3DCR アンテナを台に乗せて設置した様子を図 10 に示した。

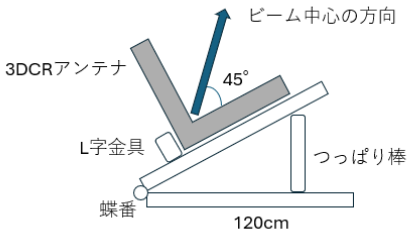

図 7: 仰角 45° 以上の対象を狙う設置方法

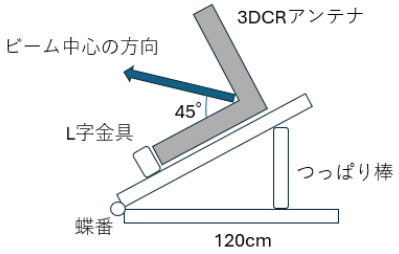

図 8: 仰角 45° 以下の対象を狙う設置方法

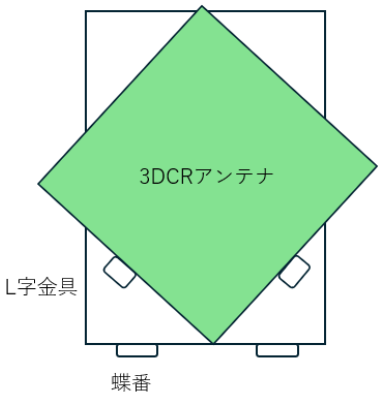

図 9: 台を上から見た様子

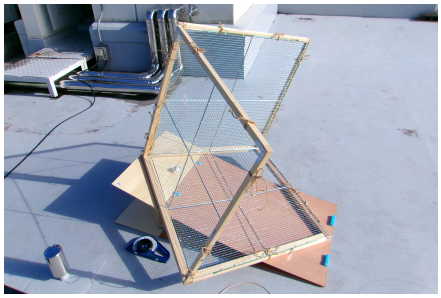

図 10: 実際の 3DCR アンテナの設置

2.2　受信部とデータ取得部

観測システムはアンテナ部、 受信部、データ取得部によって構成されている。

アンテナ部は 3DCR アンテナと枠と台で構成されている。アンテナで受信された信号は始めに受信部に送られる。受信部は SAWbird+H1m と低雑音増幅器（ SPF5189Z ）で構成されている。SAWbird+H1m には低雑音増幅器とバンドパスフィルタ (BPF) が内蔵されており、増幅器の帯域は約 1410 〜 1421 MHz、ゲインは約 40 dB である。また SAWbird+H1m のほかにも低雑音増幅器の SPF5189Z を取り付けた。この増幅器は約 50 MHz 〜 4000 MHz 周波数帯域に対応し、ゲインは約 20 dB である。

次に受信部で増幅された信号はデータ取得部に送られる。データ取得部は SDR とノートパソコンで構成される。送られた信号は SDR(ソフトウェア無線, Software Defined Radio) によってアナログ信号からデジタル信号に変換される。SDR は観測帯域 2.048 MHz でサンプリングし、中心周波数 1420.406 MHz の複素電圧データを 8 ビット符号なし整数で USB を経由し、Windows ノートパソコンのハードディスクに保存される。観測された信号はこのような過程を通してノートパソコンに保存される。観測システムの全体図を図 11 に示した。

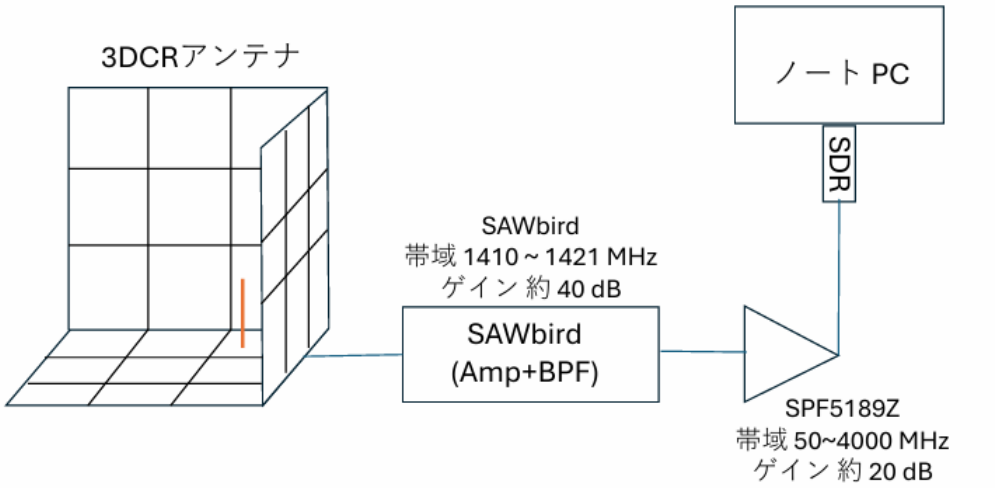

図 11: 観測システムの全体図

3　太陽のドリフト観測

3.1　観測手法

太陽のドリフト観測はアンテナを太陽の経路上に向けて固定したまま、地球自転で太陽がビームを横切る様子を観測し、その時間変化からアンテナの様々な性能（ビームサイズ、ビームパターン、指向性、ゲインなど）を評価する手法である。

今回の観測では 3DCR アンテナで太陽のドリフト観測を行うことによって、観測結果から時間の半値幅を求め、アンテナのビームサイズを評価することを目的としている。従来の太陽のドリフト観測ではアンテナを太陽の経路上に向けて方位角と仰角を固定し、太陽がビームの中心を横切るのを待つという手法を取ってい

る。しかし、3DCR アンテナを赤道儀などの道具を使用して太陽の経路上にアンテナを正確に向けることは困難であると判断した。

したがって、今回の観測では従来の太陽のドリフト観測とは異なり、初めにアンテナを太陽の方向に向けて観測を開始する手法を採用した。この手法を採ることによって赤道儀を使わずに太陽がアンテナのビーム中心にある状態を観測することができる。

太陽がビーム中心を通過する様子を時間と強度のグラフで表すとアンテナビームと太陽円盤の畳み込みの形状になり、通常はほとんどガウシアンになる。従来の手法では得られたデータ点をガウシアンフィットして、そのガウシアンから最大値を求め、ベースラインと最大値の差が半分になる時刻を左右で求める。そして、その差を時間の半値幅とする。そこから太陽の移動角速度を用いて時間の半値幅を角度の半値幅に変換する。一方で本研究の太陽のドリフト観測で用いる手法では、時間と強度の関係を表すグラフの形がガウシアンの右半分であると予想される。したがって、得られたデータを $t = 0$ 付近をピークに持つガウシアンであると仮定してガウシアンフィットをする。そこからベースラインと最大値の差が半分になる時間を求め、その値を 2 倍することで時間の半値幅とした。

地球は恒星日に 360° 回転するため、天体は天球上を一定の角速度で移動しているように見える。恒星日の長さは 86164 s であるため、赤道上（赤緯 0°）での見かけの角速度は 360°/86164 s となる。すなわち、赤道上では 1 時間で 15.041° 移動しているように見える。しかし、実際の角速度は天体の赤緯に依存し赤道上の角速度より小さくなる。赤道上の角速度を ω_0 と置いたとき、赤緯 δ 上の角速度 ω は $\omega = \omega_0 \cos\delta$ となる。

観測時刻に太陽が赤緯 δ にあるとき、時間の半値幅を ΔT とおくと、半値幅で表したビームサイズを求める式は

$$\theta = \omega_0 \Delta T \cos\delta \cdots (5)$$

のようになる。測定に使用する観測時刻の太陽の赤緯は Python の astropy を用いて求めた。

また、3DCR アンテナを太陽に向けて設置する際は、3DCR アンテナのメインビームの中心方向が各反射面に対して 45° 傾いた向きになることを考慮する必要がある。例えば、観測開始時点での太陽の仰角が 30° であった場合、図 8 のように台と 3DCR アンテナを設置し、台の傾きが 15° となるようにつっぱり棒で固定する。すると、3DCR アンテナは水平な地面に設置した場合と比べて、ビーム中心方向の仰角が 15° 小さい方向を見ている状態になる。すなわち、このとき 3DCR アンテナのビーム中心方向の仰角は 30° となる。このように 3DCR アンテナのビーム中心を太陽に向けて設置する際は、ビーム中心方向の仰角を台で調節しながら設置することが必要になる。

観測は太陽から放出される 1420 MHz の連続波の強度の計測を計 4 回行った。観測結果はそれぞれ図 12 - 15 に示した。観測日時と天気、観測当時の太陽の赤緯、時間の半値幅、ビームサイズを表 1 に示す。

3.2 結果

表 1: 太陽のドリフト観測の結果

図	観測日時	観測時間	天気	太陽の赤緯 δ	時間の半値幅 ΔT	ビームサイズ （半値幅） θ
9	2026/1/20/13:20	4000s	晴れ	$-20.202°$	1713.1 s	$6.72°$
10	2026/2/10/10:10	4000s	曇り	$-14.518°$	2452 s	$9.92°$
11	2026/2/10/12:20	5400s	曇り	$-14.489°$	452.6 s	$1.83°$
12	2026/2/10/14:30	5400s	曇り	$-14.460°$	1000.7 s	$4.05°$

観測結果は図 12 - 15 のようになった。図 12 - 15 の青い点は観測されたデータ点であり、赤い線はガウシアンフィットの結果である。どの観測結果も図から時間経過とともに強度が減少している様子が確認できる。測定結果をガウシアンフィットしたのち、ビームサイズを求めると、表 1 のようになった。

しかし、今回作成した 3DCR アンテナのビームサイズは平均 $25°$ であるため、計測値のビームサイズが予想値と異なる結果となってしまった。

また、観測開始とともに強度が急激に減少したり、観測途中で強度が増加したりするケースも見られる。そのため、測定結果が整ったガウシアンの形をしていないものがほとんどとなっている。そのため、これらの観測結果から今回作成した 3DCR アンテナのビームサイズを評価するのは困難であると考えられる。

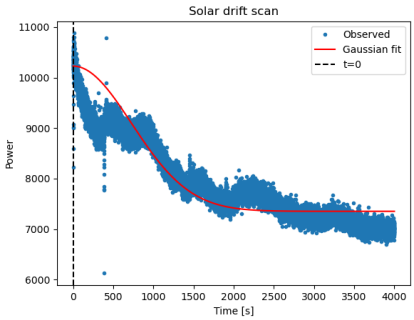

図 12: 1 回目のドリフト観測

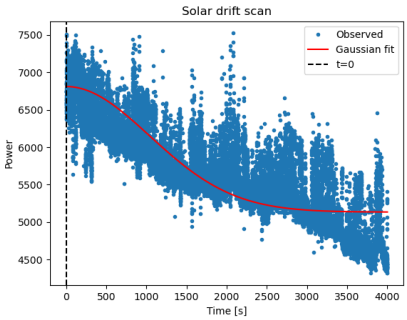

図 13: 2 回目のドリフト観測

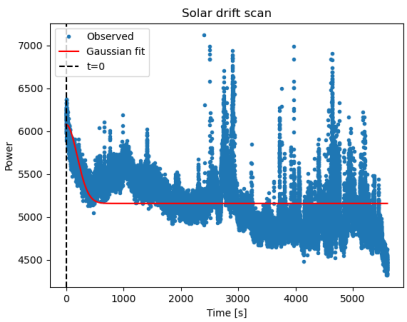

図 14: 3 回目のドリフト観測

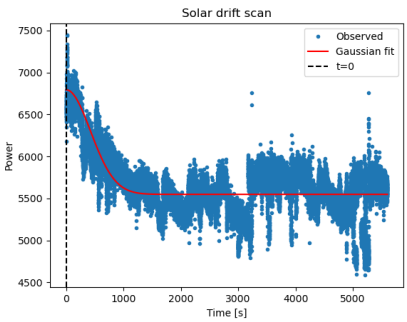

図 15: 4 回目のドリフト観測

3.3 考察

太陽のドリフト観測では、観測結果から求めたビームサイズの計測値が予想値と大きく異なる結果となってしまった。このような結果になってしまった大きな原因は、アンテナのビーム中心を初めから正確に太陽に向けることができなかったことが原因と考えられる。また、観測ごとに異なるビームサイズが計測されたことから、観測ごとに太陽に対してビーム中心を向ける方向が異なっていたと考えられる。すなわち、現段階では 3DCR アンテナを狙い通りに対象に向けることが困難であると思われる。

正確な太陽のドリフト観測を行うためには、ビーム中心を対象に高い精度で向ける必要がある。それを実現させるためには、アンテナのビーム中心を狙った方向に正確に向けるための改良やアンテナを特定の方位角や仰角に向けて固定できるよう改良することが必須であると考えられる。

4 21cm 線の観測

4.1 観測手法

1 章のタンジェントポイント法の説明で示した通り、タンジェントポイント法とは内銀河の 21 cm 線スペクトルの最大視線速度 $v_{\max}$ を用いて、内銀河の回転速度を測定する方法である。この観測では 21 cm 線の観測の結果にタンジェントポイント法を適用して天の川銀河の回転曲線を導出することを目的としている。また、本研究では水素原子の速度分散補正とランダム運動（熱運動、乱流、非円運動など）を無視する。

Honma et al. (2012) [4] より、 v_0 と R_0 はそれぞれ $v_0 = 238$ km/s, $R_0 = 8.05$ kpc とする。観測の結果から $\sin l$ と $v_{\max}$ の値が測定できれば式 (1) と式 (2) を用いて v と R の関係、すなわち回転曲線を求めることができる。この観測では初めに 21 cm 線の取得データを周波数スペクトルとして表し、最小二乗法を用いて 1 次ベースラインフィットをする。次にドップラーシフトの式 (3) によって周波数スペクトルを視線速度スペクトルに変換して最大視線速度 $v_{\max}$ を求める。今回の観測では銀経 $30° - 90°$ を観測したので接点上に位置する中性水素原子ガスの回転方向は観測者から遠ざかる向きである。よって、最大視線速度は正の値であると考えられる。本研究では最大視線速度の定義を強度がベースラインノイズの標準偏差 σ_n の 3 倍以上の値を持つ速度の中で、最も速度の値が大きいものとした。

また 21 cm 線を地球上から観測した際、地球の自転と公転、太陽の固有運動を考慮した補正が必要になる。すなわち、局所静止基準に対する観測者の速度を考慮する必要がある。v_{corr} を局所静止基準に対する観測者の速度を考慮した際の補正値としたとき式 (2) は正確には

$$v = v_{\max} + v_0 \sin l + v_{\mathrm{corr}} \cdots (4)$$

と書き換えられる。補正値の計算には Python の astropy を用いた。

そこで求まった v と R の関係をグラフで表すと回転曲線を導出することができる。

本研究の観測では銀河面上の銀経 $l = 30°, 45°, 60°, 75°, 90°$ の 5 点を観測した。1 回の観測はおよそ 30 秒間行った。観測日時は表 2 に記載した。観測当日の天気は 1/20, 1/21 のどちらも晴れであった。

観測の結果、得られた周波数スペクトルは図 16 - 20 のようになった。

4.2 結果

表 2: 21 cm 線観測の結果

l	$\sin l$	R [kpc]	$v_{\max}$ [km/s]	$v_0 \sin l$ [km/s]	v_{corr} [km/s]	v [km/s]	観測時刻
30°	0.5	4.03	79.73	119	24.44	223.17	2026/1/21/11:30
45°	0.71	5.69	71.29	168.98	20.26	260.53	2026/1/21/11:41
60°	0.87	6.97	13.88	207.06	14.12	234.06	2026/1/20/16:22
75°	0.97	7.78	12.99	230.86	7.66	251.51	2026/1/20/15:57
90°	1.0	8.05	8.81	238	0.67	247.48	2026/1/20/16:05

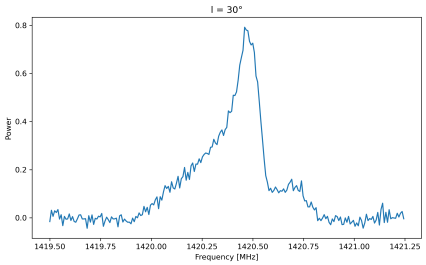

図 16: $l = 30°$ の周波数スペクトル

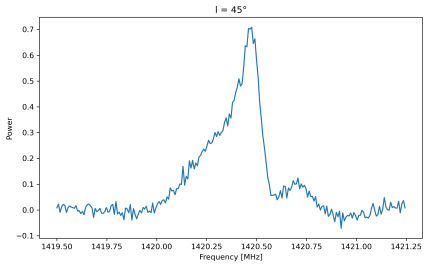

図 17: $l = 45°$ の周波数スペクトル

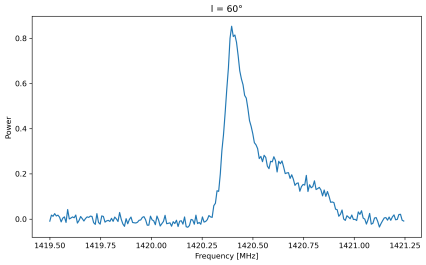

図 18: $l = 60°$ の周波数スペクトル

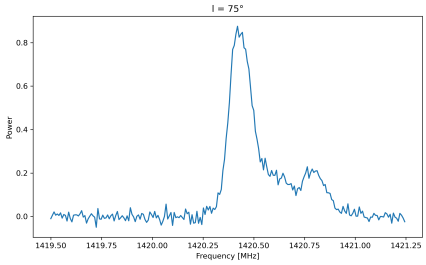

図 19: $l = 75°$ の周波数スペクトル

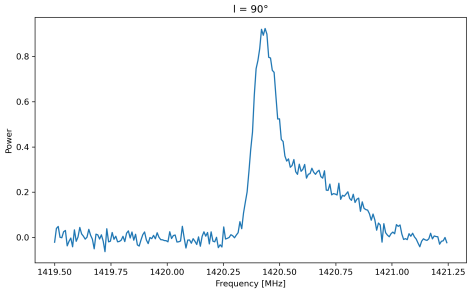

図 20: $l = 90°$ の周波数スペクトル

図 16 - 20 の周波数スペクトルは既に 1 次ベースラインフィットを行いベースラインを 0 に揃える補正をしている。ここで得られた周波数スペクトルをドップラーシフトの式 (3) を用いて視線速度スペクトルに変換すると図 21 - 25 のようになった。

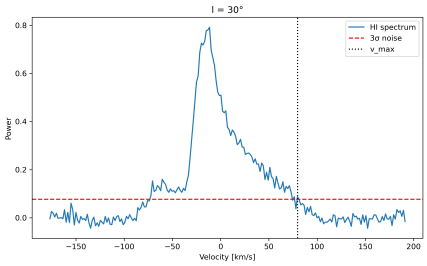

図 21: $l = 30°$ の視線速度スペクトル

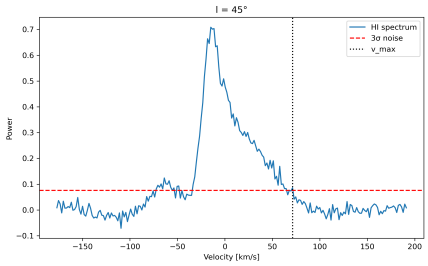

図 22: $l = 45°$ の視線速度スペクトル

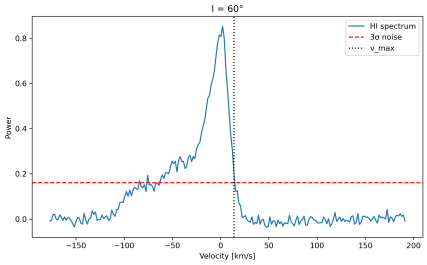

図 23: $l = 60°$ の視線速度スペクトル

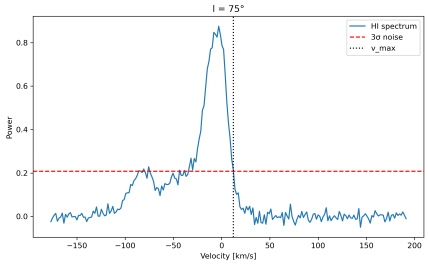

図 24: $l = 75°$ の視線速度スペクトル

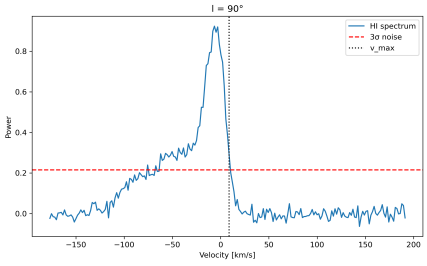

図 25: $l = 90°$ の視線速度スペクトル

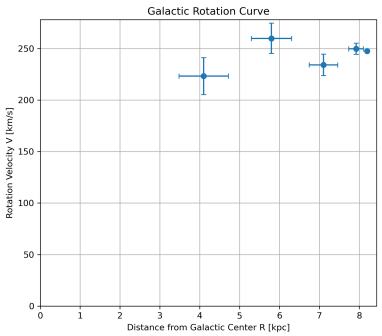

図 26: 天の川銀河の回転曲線

ここで最大視線速度の定義を、強度がのベースラインノイズの標準偏差 $\sigma_{\rm n}$ の 3 倍以上の値を持つ速度の中で最も値が大きいものとしたとき、 $\sin l$ と $v_{\rm max}$ の関係は表 2 のようになる。また、局所静止基準に対する観測者の速度を考慮したとき、$v_{\rm corr}$ の値は表 2 のようになった。

これらの結果を式 (1) と式 (4) に代入すると v と R の関係を求めることができる。その関係を図 26 に示した。

図 26 のエラーバーは銀経 5° のずれを表している。図 26 の結果から 4 - 8 kpc の間で回転速度はおよそ 200 - 260 km/s となり、あまり大きな変化をしないことが分かる。この結果は先行研究 [2] [3] とも矛盾せず、銀河の回転曲線が平坦であることが確認できた。

4.3 考察

3DCR アンテナを用いた 21 cm 線の観測では整った形状の周波数スペクトルを得られ、また計算結果から先行研究と矛盾しない回転曲線を導出できた。このことから、3DCR アンテナはアンテナとしての性能を十分に有していることが確認できる。

しかし、本研究では中性水素原子ガスの速度分散による補正を無視している。さらに、太陽のドリフト観測の考察でも述べたように、3DCR アンテナはビーム中心を狙った方向へ正確に向けることが困難であるため、本研究で導出した回転曲線は実際の回転曲線と比較して正確ではない。より正確な回転曲線を導出するためには、本研究で行えなかった補正を実行し、3DCR アンテナのビーム中心を狙った方向へ正確に向けるための改良が求められる。

5 今後の展望

本研究では太陽のドリフト観測から 3DCR アンテナのビームサイズを評価すること、そして 21 cm 線の観測から天の川銀河の回転曲線を目標にして観測を実行した。それらの結果を踏まえると、21 cm 線の観測から 3DCR アンテナはアンテナとしての性能を有することが確認できたが、太陽のドリフト観測の結果からはアンテナのビーム中心を正確に対象へ向けることが困難であることが確認できた。

3DCR アンテナをより正確なアンテナ装置として活用していくためには、その問題を解決する必要がある。そのためにはビーム中心を狙った方向に設置できるような仕組みを作ったり、反射面の歪みを抑える工夫をしたりするなどの取り組みが必要になると考えられる。

また、今回の観測で 3DCR アンテナがアンテナとしての性能を有することが確認できたため、今後は他の周波数での観測を行い、様々な天体や星間物質の電波観測が可能であるかを検討していく。

6 謝辞

本研究を実施するにあたり、多くの方々のご指導とご支援をいただきました。この場をお借りして心より感謝申し上げます。

はじめに、指導教員である百瀬宗武教授には観測機器の扱い方、データ解析、論文作成に至るまで多くのご指導とご助言を賜りました。そのおかげで、本研究を完成させることができましたことを心より感謝申し上げます。また、観測手法のご意見やご指摘を賜り、観測の実施にもご協力いいただいた米倉覚則教授にも、厚く御礼申し上げます。さらに、電波天文学観測研究室の皆様には研究活動を進める中で多くのご助力やご指摘な

どをいただきました。誠にありがとうございました。

最後に、本研究を実施するにあたり、ご支援をいただいた多くの皆様に心より感謝申し上げます。

7 付録

観測や解析に用いた Python コードを以下に記載する。

7.1 太陽のドリフト観測の観測コード

```
import tkinter as tk
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
import numpy as np
import datetime
import glob
import os
from rtlsdr import RtlSdr
import matplotlib.pyplot as plt

plt.rcParams['font.family'] = 'Yu␣Gothic'

# ユーザー設定

FREQ_CENTER = 1420.4e6
SAMPLE_RATE = int(2.048e6)
CHUNK = 131072
GAIN_LIST = [2,3,6,9,11,14,16,17,19,21,22,25,27,29,32,34,36,37,38,40,42,43,44,45,47,50]
OBS_TIME = 10.0       # 観測時間秒 []


class Application(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.master.title("Solar␣Drift␣Scan")

        frame = tk.Frame(self.master)
        fig = Figure(figsize=(10, 7.5))
        self.ax = fig.add_subplot(1, 1, 1)

        self.fig_canvas = FigureCanvasTkAgg(fig, frame)
        self.toolbar = NavigationToolbar2Tk(self.fig_canvas, frame)
        self.fig_canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

        self.ax.text(0.05, 0.5, "電波終端を Sawbird␣の入力に取り付けて HI␣\「準備」ボタンを押してくださ
            い。n",
                     fontsize=26)
        self.ax.axis("off")
```

```
        self.fig_canvas.draw()

        frame.pack()

        fonts = ("", 13)
        tk.Button(self.master, text="準備\n 一度だけ
            ()", command=self.prep, font=fonts).place(x=0, y=0)
        tk.Button(self.master, text="太陽の連続観測\n ドリフトスキャン ()",
                  command=self.sun_obs, font=fonts).place(x=200, y=0)
        tk.Button(self.master, text="クリア
            ", command=self.fig_clear, font=fonts).place(x=450, y=0)
        tk.Button(self.master, text="終了
            ", command=self.master.destroy, font=fonts).place(x=650, y=0)


    # グラフ消去
    def fig_clear(self):
        self.ax.clear()
        self.fig_canvas.draw()

        # ゲイン最適化 SDR
    def prep(self):
        self.ax.clear()
        self.fig_canvas.draw()

        sdr = RtlSdr()
        sdr.sample_rate = SAMPLE_RATE
        sdr.center_freq = FREQ_CENTER

        gain = None
        for g in GAIN_LIST:
            sdr.gain = g
            dat = sdr.read_samples(CHUNK)

            r_hist, _ = np.histogram(dat.real, range=(-1, 1), bins=256)
            i_hist, _ = np.histogram(dat.imag, range=(-1, 1), bins=256)
            N0 = (r_hist/np.sum(r_hist) + i_hist/np.sum(i_hist)) * 256

            print(f"gain={g},␣peak={np.max(N0):.2f}")
            if np.max(N0) < 7:
                gain = g
                break

        sdr.close()


        if gain is None:
            self.ax.text(0.1, 0.5, "入力レベルが低すぎます。\アンテナ・ゲインを確認してください。n",
                         fontsize=20)
            self.fig_canvas.draw()
```

```
        return

    for f in glob.glob("load_*.csv"):
        os.remove(f)

    np.savetxt(f"load_{gain}.csv", N0, delimiter=",")

    self.ax.text(0.1, 0.7, f"の最適ゲイン値は SDR␣{gain}␣です。", fontsize=26)
    self.ax.text(0.1, 0.55, "終端を外しアンテナを接続してください。", fontsize=22)
    self.ax.text(0.1, 0.4, "太陽方向にアンテナを固定したら\「太陽の連続観測」を押してください。n",
                fontsize=22)
    self.ax.axis("off")
    self.fig_canvas.draw()


# 太陽ドリフトスキャン

def sun_obs(self):

    load = glob.glob("load_*.csv")
    if not load:
        self.ax.text(0.1, 0.5, "先に準備を行ってください。", fontsize=20)
        self.fig_canvas.draw()
        return

    self.gain = int(load[0].split("_")[1][:-4])
    self.pws_bl = np.genfromtxt(load[0], delimiter=",")

    self.sdr = RtlSdr()
    self.sdr.sample_rate = SAMPLE_RATE
    self.sdr.center_freq = FREQ_CENTER
    self.sdr.gain = self.gain

    self.t_list = []
    self.p_list = []

    self.ax.clear()
    self.ax.set_xlabel("Time␣[s]")
    self.ax.set_ylabel("Normalized␣Power")
    self.ax.set_title("Solar␣Drift␣Scan␣(Real-time)")
    self.line, = self.ax.plot([], [], color="blue")

    self.start_time = datetime.datetime.now()
    print("---␣太陽ドリフトスキャン開始␣---")

    self.observe_step()

def observe_step(self):
    try:
```

```
            dat = self.sdr.read_samples(CHUNK)
        except Exception as e:
            print("SDR␣error:", e)
            self.finish_obs()
            return

        spec = np.fft.fftshift(np.fft.fft(dat))
        freq = np.fft.fftshift(np.fft.fftfreq(len(dat), d=1/SAMPLE_RATE))
        mask = np.abs(freq) < 50e3

        pwr = np.mean(np.abs(spec[mask])**2)
        p_norm = pwr / np.mean(self.pws_bl)


        t = (datetime.datetime.now() - self.start_time).total_seconds()
        self.t_list.append(t)
        self.p_list.append(p_norm)

        self.line.set_data(self.t_list, self.p_list)
        self.ax.relim()
        self.ax.autoscale_view()
        self.fig_canvas.draw_idle()

        if t >= OBS_TIME:
            self.finish_obs()
            return

        self.master.after(1, self.observe_step)

    def finish_obs(self):
        self.sdr.close()

        now = datetime.datetime.now().strftime("%Y%m%dT%H%M%S")
        out = np.vstack([self.t_list, self.p_list]).T
        np.savetxt(f"sun_drift_{now}.csv", out, delimiter=",")

        plt.figure(figsize=(10,4))
        plt.plot(self.t_list, self.p_list)
        plt.xlabel("Time␣[s]")
        plt.ylabel("Normalized␣Power")
        plt.title("Solar␣Drift␣Scan␣Result")
        plt.grid(True)
        plt.savefig(f"sun_drift_{now}.png", dpi=200)

        print("観測完了・保存しました")



# メイン
```

```
root = tk.Tk()
app = Application(master=root)
app.mainloop()
```

7.2 ドリフト観測の結果をガウシアンフィットするコード

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# データ読み込み（time[s], ）power
t, p = np.loadtxt("sun_drift_20260120T142857.csv", delimiter=",", unpack=True)



dt_peak = 30.0
mask_peak = np.abs(t) < dt_peak
P_peak = np.mean(p[mask_peak])

mask_base = (t < -300) | (t > 300)
B0 = np.mean(p[mask_base])

def gaussian_peak_fixed(t, sigma, B):
    return B + (P_peak - B) * np.exp(-(t**2) / (2 * sigma**2))

from scipy.optimize import curve_fit

sigma0 = 100.0 # 初期値[s]
p0 = [sigma0, B0]

popt, pcov = curve_fit(gaussian_peak_fixed, t, p, p0=p0)

sigma_fit, B_fit = popt

FWHM_time = 2.355 * sigma_fit
print(f"FWHM␣=␣{FWHM_time:.1f}␣s")



t_fit = np.linspace(t.min(), t.max(), 1000)
p_fit = gaussian_peak_fixed(t_fit, sigma_fit, B_fit)

plt.figure()
plt.plot(t, p, ".", label="Observed")
plt.plot(t_fit, p_fit, "r-", label="Gaussian␣fit")

plt.axvline(0, color="k", ls="--", label="t=0")
```

```
plt.xlabel("Time␣[s]")
plt.ylabel("Power")
plt.title(f"Solar␣drift␣scan")

plt.legend()
plt.savefig("Solar␣drift␣scan_20260120T142857.png")
plt.show()
```

7.3 銀河座標を地表座標に変換するコード

```
#銀河座標を地表座標に変換する
from astropy.coordinates import SkyCoord, AltAz, EarthLocation
from astropy.time import Time
import astropy.units as u

# 入力
# 日本時間 () JST
jst_time = "2026-01-20␣13:57:00" # JST

# 観測地点
lat = 36.6977  # deg
lon = 140.695 # deg




# 銀河座標
l = 15.0 # deg
b = 0.0  # deg

# JST → UTC
t_jst = Time(jst_time, scale="utc") - 9 * u.hour


# 座標定義
location = EarthLocation(
    lat=lat * u.deg,
    lon=lon * u.deg
)

gal = SkyCoord(
    l=l * u.deg,
    b=b * u.deg,
    frame="galactic"
)
```

```
# 地平座標へ変換
altaz = gal.transform_to(
    AltAz(obstime=t_jst, location=location)
)


# 出力
print(f"入力 () JST:␣{jst_time}")
print(f"UTC␣␣␣␣␣␣␣␣:␣{t_jst.iso}")
print(f"Azimuth␣␣␣␣:␣{altaz.az.deg:.2f}␣deg")
print(f"Altitude␣␣␣:␣{altaz.alt.deg:.2f}␣deg")
```

7.4 太陽の位置を調べるコード

```
#太陽の位置

from astropy.time import Time
from astropy.coordinates import EarthLocation, AltAz, get_sun
import astropy.units as u


# 観測地点
location = EarthLocation(
    lat=36.6977 * u.deg, # 緯度
    lon=140.695 * u.deg, # 経度
    height=0 * u.m
)


# 日本時間 () を入力 JST
t_jst = Time("2026-02-10␣14:25:00", scale="utc") - 9*u.hour


# 太陽位置の計算
sun = get_sun(t_jst)
sun_altaz = sun.transform_to(
    AltAz(obstime=t_jst, location=location)
)


# 出力
print(f"高度␣Alt␣=␣{sun_altaz.alt.deg:.2f}␣deg")
print(f"方位角␣Az␣=␣{sun_altaz.az.deg:.2f}␣deg")
```

7.5 21 cm 線の観測コード

```
import tkinter as tk
from matplotlib.figure import Figure
from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg, NavigationToolbar2Tk
import numpy as np
import datetime
from scipy import fftpack
from rtlsdr import *
from matplotlib import pyplot
import glob
import os
import argparse

class Application(tk.Frame):
    def __init__(self, master=None):
        super().__init__(master)
        self.master = master
        self.master.title('matplotlib␣graph')

        #------------------------------------------------

        # 配置用フレーム matplotlib
        frame = tk.Frame(self.master)

        # の描画領域の作成 matplotlib
        fig = Figure(figsize=(10.0, 7.5))
        # 座標軸の作成
        self.ax = fig.add_subplot(1, 1, 1)
        # の描画領域とウィジェット matplotlib(Frame) の関連付け
        self.fig_canvas = FigureCanvasTkAgg(fig, frame)
        # のツールバーを作成 matplotlib
        self.toolbar = NavigationToolbar2Tk(self.fig_canvas, frame)
        # のグラフをフレームに配置 matplotlib
        self.fig_canvas.get_tk_widget().pack(fill=tk.BOTH, expand=True)

        self.ax.text(0.0, 0.5, "電波終端を Sawbird␣の入力に取り付けて、HI\「準備」のボタンを押してくださ
            い。n", fontname="MS␣Gothic", fontsize=28)

        self.ax.axis('off')
        self.fig_canvas.draw()

        # フレームをウィンドウに配置
        frame.pack()


        # ボタンの作成
        fonts = ("", 13)
```

```
    button0 = tk.Button(self.master, text = "準備\n 観測前に一度だけ実行
        ()", command = self.prep, font=fonts)
    button1 = tk.Button(self.master, text = "天の川の観測\n 観測前に「クリア」を押す
        ()", command = self.sky_obs, font=fonts)
    button2 = tk.Button(self.master, text = "クリア", command = self.fig_clear, font=fonts)
    button3 = tk.Button(self.master, text = "プログラムの\終了
        n", command=self.master.destroy, font=fonts)
    button0.place(x=0, y=0)
    button1.place(x=375, y=0)
    button2.place(x=670, y=0)
    button3.place(x=850, y=0)

    #-----------------------------------------------



def fig_clear(self):
    self.ax.clear()
    self.fig_canvas.draw()


def close_window(self):
   self.master.quit()


def prep(self):
    self.ax.clear()
    self.fig_canvas.draw()

    # configure device
    sdr = RtlSdr()
    sdr.sample_rate = 2.048e6 # Hz
    sdr.center_freq = 1420.4e6  # Hz

    Glist = [2,3,6,9,11,14,16,17,19,21,22,25,27,29,32,34,36,37,38,40,42,43,44,45,47,50]

    for k in range(len(Glist)):
        sdr.gain = Glist[k]
        NSAMP = int(2.048e6)
        dat = sdr.read_samples(NSAMP)
        r_hist, r_bins = np.histogram(dat.real, range=(-1, 1),bins=256)
        i_hist, i_bins = np.histogram(dat.imag, range=(-1, 1),bins=256)
        N0 = (r_hist/np.sum(r_hist)+i_hist/np.sum(i_hist))*256
        print("gain=%d,␣%.1f,␣%.3f" % (Glist[k],sdr.get_gain(), np.max(N0)))
        if np.max(N0) < 7:
            gain = Glist[k]
            break
        elif k == len(Glist)-1:
            print("\n")
            print("電波の入力レベルが低すぎます")
```

```
            print("ケーブルの接続や装置の不具合を確認してください")

            print("\n")
            exit()
    sdr.gain = gain
    pws_bl = np.zeros(256)
    for i in range(Nint):
        NSAMP = int(2.048e6)
        dat = sdr.read_samples(NSAMP)
        nData = len(dat)
        signal_spectrum = np.fft.fftshift(np.fft.fft(dat))
        pwrN = np.abs(signal_spectrum/nData*2)**2
        pws = np.sum(pwrN.reshape(256,int(nData/256)),axis=1)
        pws_bl += pws

    sdr.close()

    try:
        for f in glob.glob("load_*.csv"):
            os.remove(f)

    except OSError as e:
        pass

    np.savetxt("load_%d.csv" % (gain,), pws_bl, delimiter=",", fmt="%s")
    ########################################
    self.ax.text(0.2, 0.8, "の最適ゲイン値は SDR% でした。
        d" % (gain,), fontname="MS␣Gothic", fontsize=24)
    self.ax.text(0.0, 0.7, "電波終端を Sawbird␣の入力コネクタから外し、
        HI" , fontname="MS␣Gothic", fontsize=24)
    self.ax.text(0.0, 0.6, "アンテナと Sawbird␣を同軸ケーブルで繋いでください。
        HI", fontname="MS␣Gothic", fontsize=24)
    self.ax.text(0.0, 0.4, "アンテナを天の川に向けたら「クリア」ボタンを押し、
        ", fontname="MS␣Gothic", fontsize=24)
    self.ax.text(0.0, 0.3, "「天の川の観測」のボタンを押して観測してください。
        ", fontname="MS␣Gothic", fontsize=24)
    self.ax.axis('off')
    self.fig_canvas.draw()



def sky_obs(self):
    self.ax.clear()
    load = glob.glob("load_*.csv")

    if len(load) == 0:
        self.ax.text(
            0.1, 0.5,
            "先に「準備」を実行してください。",
            fontname="MS␣Gothic",
            fontsize=24
```

```
            )
            self.ax.axis("off")
            self.fig_canvas.draw()
            return

        pws_bl = np.genfromtxt (load[0], delimiter=",")
        gain = int(load[0].split('_')[1][:-4])

        sdr = RtlSdr()
        # configure device
        sdr.sample_rate = 2.048e6 # Hz
        sdr.center_freq = 1420.4e6  # Hz
        sdr.gain = gain

        pws_sky = np.zeros(256)
        for i in range(Nint):
            NSAMP = int(2.048e6)
            dat = sdr.read_samples(NSAMP)
            nData = len(dat)
            signal_spectrum = np.fft.fftshift(np.fft.fft(dat))
            pwrN = np.abs(signal_spectrum/nData*2)**2
            pws = np.sum(pwrN.reshape(256,int(nData/256)),axis=1)
            pws_sky += pws

        sdr.close()

        now = datetime.datetime.now()
        skyname = now.strftime("%Y%m%dT%H%M%S")

        HIspectrum = (pws_sky - pws_bl) / pws_bl
        HIspectrum = HIspectrum - np.min(HIspectrum)
        HIspectrum=HIspectrum/np.max(HIspectrum)

        self.ax.plot(freq, HIspectrum)
        self.ax.set_xlabel('Frequency␣[MHz]')
        self.ax.set_ylabel('Normalized␣intensity␣[a.u.]')
        self.ax.set_title("Hydrogen␣line␣signal␣of␣our␣milkyway", fontsize = 16)
        self.ax.set_ylim(np.min(HIspectrum),1.01)
        Outdata = np.zeros((2,256))
        Outdata[0] = freq
        Outdata[1] = HIspectrum
        self.fig_canvas.draw()
        np.savetxt(skyname+".csv", Outdata.T, delimiter=",", fmt="%s")
        self.fig_canvas.print_png(skyname+".png")



parser = argparse.ArgumentParser(description='Filterbank␣format(.fil)␣file')
```

```
parser.add_argument('-d', '--duration', type=int, help='Observing␣duration␣(seconds)',
    default=25)

# Notebook では unknown arguments (-f kernel...) を無視する
args, unknown = parser.parse_known_args()


Nint = args.duration
print("Observing␣Duration:␣" + str(Nint) + "sec")
freq = [1420.4 -2.048/2 + 0.004 + 0.008*i for i in range(256)]

root = tk.Tk()
sdr = RtlSdr()
sdr.close()

app = Application(master=root)
app.mainloop()
```

7.6 21 cm 線の周波数スペクトルをベースラインフィットするコード

```
#ベースラインの補正
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# CSV 読み込み
df = pd.read_csv("20260120T160513.csv", header=None)
df.columns = ["frequency_MHz", "power"]

freq = df["frequency_MHz"].values
power = df["power"].values


# 周波数範囲マスク
# 解析対象（1419.5 〜1421.25 ） MHz
mask_use = (freq >= 1419.5) & (freq <= 1421.25)

freq_use = freq[mask_use]
power_use = power[mask_use]

# ベースラインに使う領域
mask_base = (
    ((freq_use >= 1419.5) & (freq_use <= 1420.0)) |
    ((freq_use >= 1421.0) & (freq_use <= 1421.25))
)
```

```
x_base = freq_use[mask_base]
y_base = power_use[mask_base]


# ベースラインフィット（次）1
coef = np.polyfit(x_base, y_base, 1)
baseline = np.polyval(coef, freq_use)

print("Baseline␣coef␣[slope,␣intercept]␣=", coef)


# ベースライン補正
power_bs = power_use - baseline



# プロット

plt.figure(figsize=(8,5))

plt.plot(freq_use, power_bs)

plt.xlabel("Frequency␣[MHz]")
plt.ylabel("Power")
plt.title("l␣=␣° 90")

plt.tight_layout()
plt.savefig("20260120T160513_baseline_corrected.png", dpi=300)
plt.show()
```

7.7 周波数スペクトルを視線速度スペクトルに変換するコード

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt


# 定数
F0 = 1420.405751 # MHz
C = 299792.458 # km/s


# CSV 読み込み
df = pd.read_csv("20260120T160513_baseline_corrected.csv")

freq = df["frequency_MHz"].values
power = df["power_baseline_corrected"].values
```

```
# 速度軸変換

velocity = C * (F0 - freq) / F0

# 並び替え
idx = np.argsort(velocity)
velocity = velocity[idx]
power = power[idx]


# ノイズσの推定（ベースライン領域）

mask_noise = (
    ((freq >= 1419.5) & (freq <= 1420.0)) |
    ((freq >= 1421.0) & (freq <= 1421.25))
)

noise_power = power[mask_noise]
sigma = np.std(noise_power)

print(f"Noise␣sigma␣=␣{sigma:.3e}")


# 最大強度

Imax = np.max(power)


# 条件マスク

mask_detect = (
    power >= 3.0 * sigma
)


# 最大視線速度（正側）

v_max = velocity[mask_detect].max()

print(f"Maximum␣LOS␣velocity␣=␣{v_max:.2f}␣km/s")


# 可視化

plt.figure(figsize=(8,5))
plt.plot(velocity, power, label="HI␣spectrum")
```

```
plt.axhline(3.0 * sigma, color="red", ls="--", label="σ 3␣noise")
plt.axvline(v_max, color="black", ls=":", label=f"v_max")
plt.xlabel("Velocity␣[km/s]")
plt.ylabel("Power")
plt.title("l␣=␣° 90")
plt.legend()

plt.tight_layout()

plt.savefig("20260120T160513_HI_V.png", dpi=300)
plt.show()
```

7.8 局所静止基準に対する観測者の速度補正のコード

```
import astropy.units as u
from astropy.time import Time
from astropy.coordinates import SkyCoord, EarthLocation
import numpy as np
def lsr_correction(
    obstime_utc,
    l_deg,
    b_deg,
    lat_deg,
    lon_deg,
    height_m=0
):



    # 観測時刻
    t = Time(obstime_utc, scale="utc")

    # 観測地点
    location = EarthLocation(
        lat=lat_deg * u.deg,
        lon=lon_deg * u.deg,
        height=height_m * u.m
    )

    # 銀河座標
    coord = SkyCoord(
        l=l_deg * u.deg,
        b=b_deg * u.deg,
        frame="galactic"
    )

    # 地球→太陽
```

```
    v_helio = coord.radial_velocity_correction(
        kind="heliocentric",
        obstime=t,
        location=location
    ).to(u.km/u.s).value

    # 太陽の固有運動
    U = 11.1  # km/s 銀河中心方向 ()
    V = 12.2  # km/s 銀河回転方向 ()
    W = 7.3   # km/s 北銀極方向 ()

    l = np.deg2rad(l_deg)
    b = np.deg2rad(b_deg)

    v_sun_lsr = (
        U * np.cos(l) * np.cos(b) +
        V * np.sin(l) * np.cos(b) +
        W * np.sin(b)
    )

    # 合計補正量
    v_lsr_corr = v_helio + v_sun_lsr

    return v_lsr_corr

obstime = "2026-01-20␣07:05:13" # UTC
l = 90.0
b = 0.0

lat = 36.6977
lon = 140.695
height = 0

v_corr = lsr_correction(
    obstime, l, b, lat, lon, height
)

print(f"補正量 LSR␣=␣{v_corr:.2f}␣km/s")
```

8 参考文献

[1] 谷敷怜空, 2024, 天文月報, 117, 315
[2] 谷敷怜空, 2025, 天文月報, 118, 53
[3] 本間希樹, 1999, 天文月報, 92, 617
[4] Honma, M., et al., 2012, PASJ, 64, 1
[5] 根本靖彦 ほか, 2026, 天文月報, 119, 160