東北大学アマチュア無線部

Tohoku University Amature Radio Culb

Vol.4

目次

アマチュア無線、東北大学アマチュア無線部とは？

アマチュア無線?

趣味で使うことのできる無線で、日本全国はもちろん世界中と通信することが出来ます。
運用方法などにも様々な方法があり、色々な楽しみ方が出来る無線趣味のことです。

東北大学アマチュア無線部?

アマチュア無線の運用や各種大会出場、電子工作などを行っている学友会の部活動。
川内キャンパスと片平キャンパスの両方の部室を拠点に活動中です。

当部誌について

昨年度から部の活動と部員の研究成果を共有、発表するために刊行を始めました。
皆様のご愛読もあって今号で四号目です。バックナンバーは公式 HP で随時公開中ですので是非そちらもご覧下さい。

アマチュア無線って何だ？（素人目線）

さときょう

無線運用未経験のC0がその魅力について考えてみた。

　タイトルにもあるように僕はまだ部活動を経験したことがありません。2年生になるにもかかわらず無線についてド素人なので 1 年生の皆さんが入部してくれたら一緒に学んでいきたいと思っています笑。またその魅力とは何なのかを知ってから学習することは大切だと思います。そこで自分なりに考えてみました。

１．全く知らない人とつながることができる！

　この点において、現代の SNS を経験している皆さんにはその楽しさをすでに知っているかと思います。携帯一つでさまざまな SNS を使うのも便利で良いですが、無線というアナログな方法を用いるからこその良さもあると思います。最近世間の間でアナログ文化が再燃してきているようなのでアマチュア無線という方法でその波に乗っかってみるのもありではないでしょうか？

２．災害に強い！

地震が発生した時など、一斉に電話回線が集中するとパンクするリスクがありますが、アマチュア無線は基地局や中継アンテナが不要なので、災害時の緊急通信手段としても使うことができます。地震などの緊急事態に備えて、アマチュア無線を運用できるようにするのもオススメです。

こんな感じで簡単にまとめてみましたが、実際に体験してみないとわからない部分が多いかと思います。なので、僕達と一緒に活動してみませんか？

八木・宇田アンテナの製作

部員 K

1.　八木・宇田アンテナとは

八木・宇田アンテナは以下のような複数のエレメントを持った構造をしており、それぞれのエレメントは、導波器（ディレクター）、放射器（ラジエーター）、反射器（リフレクター）と呼ばれている。

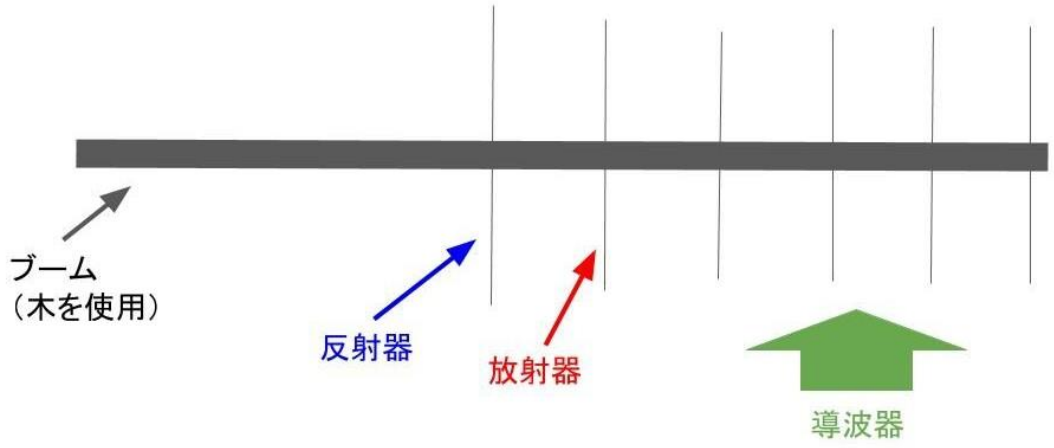


図 1　八木・宇田アンテナの構造

八木･宇田アンテナは、身近なところでは、テレビのアンテナに使われている。特徴としては、１方向へ強い電波を飛ばすことができるという単一指向性アンテナである点である。弱い電波の受信の際には、指向性のある方向の電波を集中的に受信できるため、GP アンテナなど、指向性のないアンテナに比べより強く入感させることができる。

今回は図１にあるような６エレの八木・宇田アンテナの制作を行った。

2.　製作

今回の製作には、「500 円八木アンテナ（https://www.jamsat.or.jp/features/cheapyagi/）」を参考にしながら制作を行った。材料は上記ホームページ

にあるように、ブームには木を、放射器には銅の針金を、それ以外のエレメントにはアルミ棒を使用した。それ以外に、同軸(5D-2V)、M 型コネクタを用意した。また６エレなので、図１にあるように放射器、反射器は１本、導波器は４本のエレメントを使用した。

それぞれのエレメントの長さと、反射器からの距離は上記にある「500 円八木アンテナ」のホームページを参考にし、以下のようにした。

表 1　エレメントの長さと反射器からの長さ

エレメントの長さ(mm)	340	-	315	305	305	279
反射器からの距離(mm)	0	64	140	286	445	610

500 円八木アンテナ（https://www.jamsat.or.jp/features/cheapyagi/）より引用

放射器は少し長めにすることで周波数がより低いところで同調が合うため、エレメントをカットしていくことにより、同調点の周波数を高くすることで調整をした。組み立てには、ペンチ、キリがあれば充分であった。キリにより、木に穴を貫通させ、ペンチにより、アルミ棒や銅棒をカットした。

制作直後に nano VNA で SWR を測定すると以下の画像のようになった。

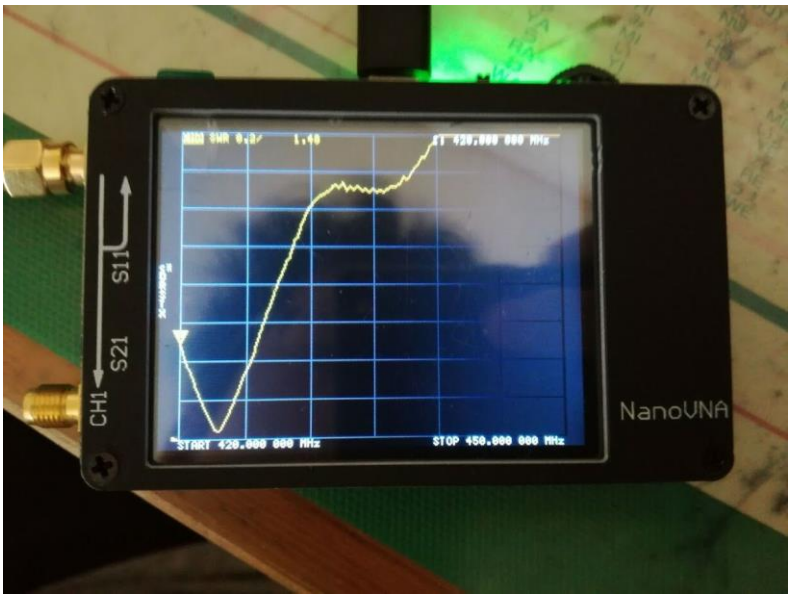


図 2　製作後の SWR

このように 423MHz あたりで同調があっており、はんだ付けも含めしっかりとできていることを確認した。しかしながら、同調点がアマチュアバンド外であるため、エレメントを少し短くすることにより同調点の周波数を高くした。調整後の SWR は以下のようになり、同調点をアマチュアバンド内にもってくることができた。

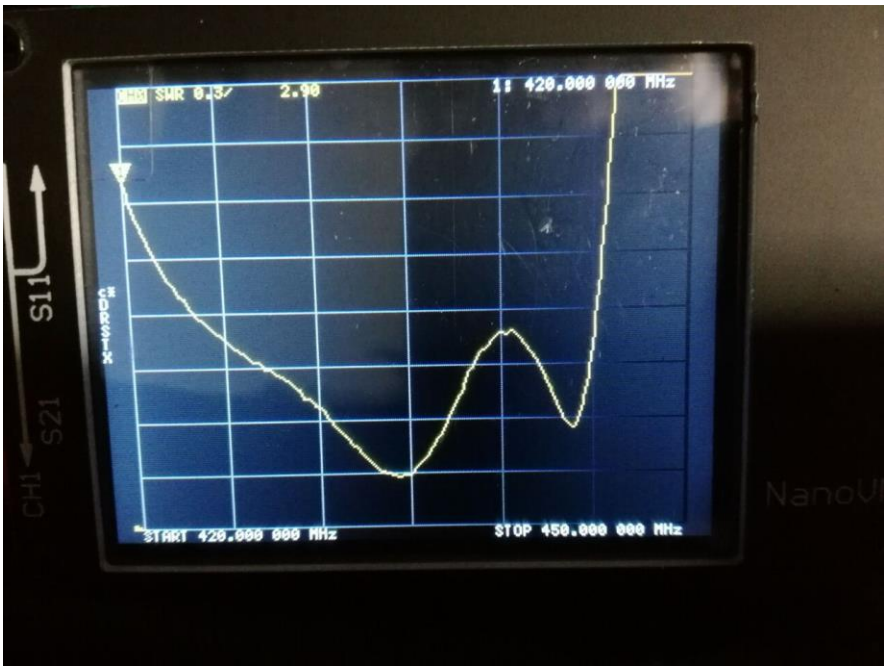


図 3　調整後の SWR

3.　運用

430MHz にて、実際に運用を簡単に行ってみた。QTH は静岡市葵区の賤機山の斜面からである。そこまで、見晴らしの方もいいとは言えない場所であったが、ハンディ機を使って 5W で静岡市駿河区、清水区、富士市の方と QSO することができた。アンテナを向ける方向によってかなり信号強度が変化し、清水区の方との交信においては、ハンディ機の S メーターでアンテナの向きによっては信号が消えてしまうが、最大では S が 5 まで変化し、八木・宇田アンテナの指向性を確認することができた。

図 4 完成後の八木・宇田アンテナ

４．まとめ

ホームセンターと部品屋にあるもので簡単に安価に製作でき、指向性もしっかりあることがわかった。非常にお手軽ですので、皆さんぜひ作ってみてください！

5. 参考文献

[1] “500 円八木アンテナ”,”https://www.jamsat.or.jp/features/cheapyagi/”,2021/04/02 閲覧

アマチュア無線の歴史をざっくりと

富沢いずみ

1.　はじめに

本記事を読んでくださっている読者の方はアマチュア無線を身近なものと感じている方が多いのでしょうが、アマチュア無線はどのように始まったのか、日本ではどのように広がっていったのかなど「アマチュア無線の歴史」についてあまり知る機会が無いように思えます。そこで本記事ではそんな歴史についてわかりやすく時代を追って説明していきたいと思います。

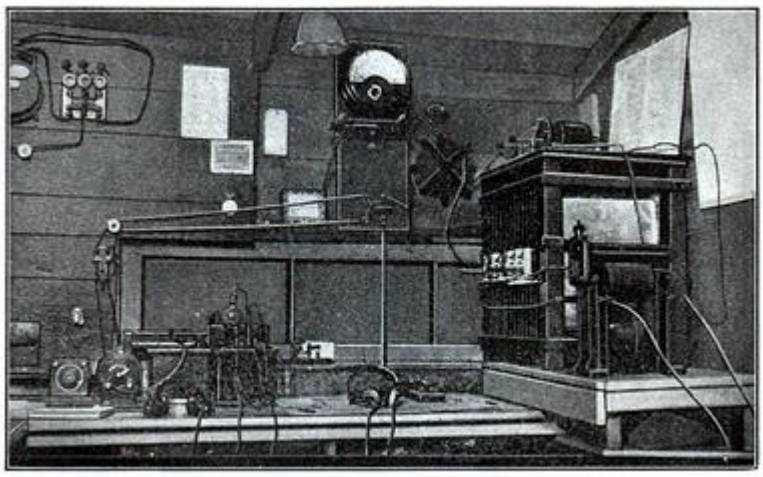
2. 無線通信の拡大とアマチュア無線の広がり

マクスウェルの電磁波の存在の予言、ヘルツの電磁波の発生と検出から始まった電磁波による無線通信技術は多くの科学者の注目の的となり 1895 年、マルコーニが初めて屋外で 1.5Km の無線通信に、1901 年には大西洋横断通信に成功した。これに海軍や企業が注目、マルコーニは無線通信の商用化に成功した。この後も無線通信の普及と技術向上が進む中で生まれてきたのが、金銭目的ではなく純粋に無線技術に興味を持ち無線装置の制作や通信技術の向上を目指す「アマチュア無線」であった。

そんなアマチュア無線が普及する大きなターニングポイントとなったのが

1908 年である。この年、雑誌「電気技師と機械技師」にアマチュア無線に関する記事が載るようになったり、専門誌「近代電気」が相関され無線技術が広められたり、無線部品を販売する店が開店したりした。

これにより 1904 年には 150 局程度だった米国のアマチュア無線家の数は 1909 年には 500-600 局と大きく増加した。ここで問題になるのがアマチュア無線による帯域の専有である。これを解決するため法律規制がなされ、*1912* 年に免許を持ったもののみが無線局を開局できるようなると共に使用できる周波数範囲を制限された。



FRONTISPIECE—An Amateur's Set Heard Over a 500-Mile Range. I. K. W. Station Built by Mr. Ralph Batcher, at Toledo, Iowa, with the Assistance of the 1916 Edition of this Book.

図 1.初期の自作無線送信機

(出典:https://ja.wikipedia.org/wiki/%E3%82%A2%E3%83%9E%E3%83%81%E3%83%A5%E3%82%A2%E7%84%A1%E7%B7%9A%E3%81%AE%E6%AD%B4%E5%8F%B2#/media/%E3%83%95%E3%82%A1%E3%82%A4%E3%83%AB:One_KW_Ham_Station.jpg)

また同年にはアマチュア無線家を組織化した ARRL（American Radio Relay League、米国無線中継連盟）が誕生した。

第一次世界大戦下では禁止されたが終戦後は再開、1925 年には国際的な同盟である IARU（International Amateur Radio Union、国際アマチュア無線連合）が結成、1927 年には CCIR(国際無線通信諮問委員会)が設立された。無線通信の利用が拡大し電波の国家管理が進んだ時代でもアマチュア無線は保護されその裾野は広がった。またアマチュア無線家により長距離無線通信が可能であることや短波帯の有用性が明らかになるなど無線技術全体にも大きく貢献した。

3.　日本国内でのアマチュア無線　前

昭和 25（1950）年の電波法公布までは法制上ではアマチュア無線局の名称はなかったものの、私設無線電信、無線電話実験局として全国に多数の無線局が存在していた。では日本国内でのアマチュア無線はいつから始まったかと言うと、大正 14(1925)頃であると考えられる。大正 13 年に専門雑誌「無線と実験」が創刊され無線への興味が高まる中、大正 14 年には短波帯の私設実験局が関東と関西で実験を開始し双方の交信に成功、次第に交信範囲を国外に広げた。翌年には草間貫吉、梶井健一、笠原功一、仙波猛、磯英治、宮崎清俊等らにより任意団体として JARL（Japan Amateur Radio League，日本アマチュア無線連盟）が設立された。

この年の 6 月 12 日には英文の JARL 設立宣言文が短波送信機を用いて世界中のアマチュア局に向けて打電、海外にも JARL の成立が認識され日本での本格

的なアマチュア無線の歴史が始まった。なおこのとき逓信省（現総務省）は日本国内でのアマチュア無線の存在を知らなかった。この後正式にアマチュア局が認可されたのは昭和 2 年(1927)であり、翌年にはコールサインを国際的標準に変更し国際的なコールサインを用いたアマチュア無線通信が可能になった。

4.　日本国内でのアマチュア無線　後

昭和 10（1935）年 JARL は IARU への加盟が承認され日本のアマチュア無線は国際的に認められた。しかし第二次大戦大戦時において日本国内でのアマチュア無線界隈は苦難を歩むことになる。戦前では憲兵隊からスパイと思われないように通信内容に配慮する必要があり、昭和 16 年には新規の免許申請の停止及び太平洋戦争開戦後は送信受信の停止が命じられた。また関東防空演習において「国防無線隊」として招集されたり無線機器を供出させられたりと日本国内におけるアマチュア無線の暗黒時代であった。

終戦後の GHQ 統治下においてはかつてのアマチュア無線技士らの交渉などにより昭和 20(1945 年)に交信の傍受が可能に、翌年には JARL が再結成された。この新生 JARL は八木秀次氏を会長にし、米国 FCC の規則に沿った規則案からなった。その後も各地で各地での支部創設、署名活動や意見具申などの活動を続けその結果昭和 25（1950）年に電波三法が施行、昭和 27(1952)年に国家試験の結果を通した本免許が下り、日本国内でのアマチュア無線が復活した。ま

た昭和 30（1955）年には IARU への復帰、ARRL の承認を受け正式に国際復帰した。JARL はこのような無線への熱い思いを受けアマチュア無線の発展と振興のために様々な活動を実施した。

図 2.社団法人 日本アマチュア無線連盟 創立総会

(出典: https://www.jarl.org/jarl90th/　閲覧日 2021/03/29)

　その一例が災害時の非常通信網の整備やアマチュア無線人口の増加などを目的とした講習会・講演会・研究会・競技会の実施、機関誌や関連図書の刊行、各種技術的な問題への指導などである。これらの甲斐もあり日本国内のアマチュア無線局数、JARL 会員数は右肩上がりに増加し一大時代を築いた。

現在では携帯電話やインターネットの普及によりアマチュア局数・JARL 会員数は減少している。

5. まとめ

本稿ではアマチュア無線の始まりから現在までの歴史を大まかに説明しました。これまで様々な問題に直面してきたアマチュア無線ですが、無線家達の努力や貢献があって乗り越えてきたことが歴史からも分かると思います。

また、現在の無線従事者の減少という問題についても無線家達がアマチュア無線の重要性や魅力を伝えていくことで必ず前向きな結果が得られると思います。

参考文献

[1] 芳野赳夫,”日本のアマチュア無線の歴史と世界の現状”,“https://www.jstage.jst.go.jp/article/bplus/6/3/6_174/_pdf”,2021/03/28 閲覧

[2] 関根慶太郎,”無線通信の基礎知識”,(2012),CQ 出版社

[3]”JARL 創立 90 周年記念サイト”,”https://www.jarl.org/jarl90th/” ,2021/03/29 閲覧

4G・5G ってなんだ

リー

最近になって始まった 5G ですが、どういった仕組みで早くなっているのかを知らない人も多いかと思います。そのため簡単にですが 1G からの発展と 5G のメリットについて説明していこうと思います。

1．通信方式の違い

実は 5G の通信方式は 4G と全く同じものが使用されていますが,今まではこの通信方式によって劇的に性能が向上していました。まず、1G では FDMA[1]と言われる「周波数分割多元接続方式」を使用していました。これは一端末ごとに異なる周波数を使用して通信をする方式です。イメージとしては無線運用やラジオでしょうか。一番イメージがしやすい形だと思います。ただし非常に近い周波数の情報まで受信してしまうため、ある程度の間隔を開ける必要が出てきます。2G は TDMA(「時分割多元接続方式」)というもので、複数のタイミングのずれた信号を合わせて送受信するものです。このようにする利点としては、時間によって情報を選別しているため、無駄なく周波数全域を使用できる点と

[1] FDMA は Frequency-Division Multiple Access の略であり、FDM とすることもある。同様に T は Time、C は Code、 O は Orthogonal を表す。

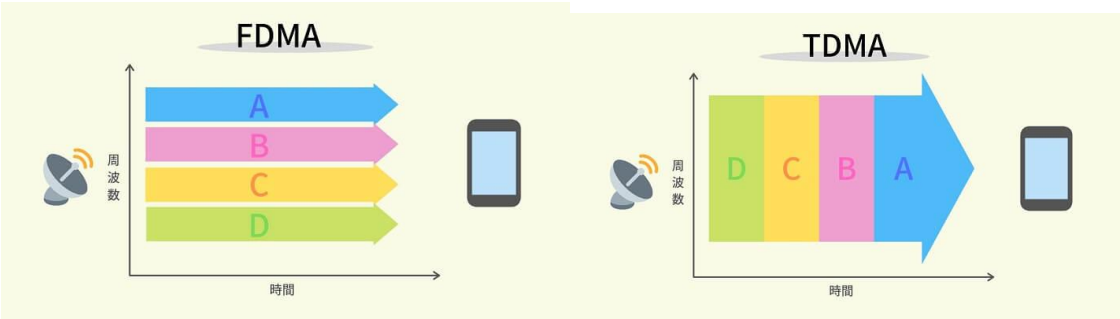


図 5　FDMA(左)と TDMA(右)のイメージ図

（出典:リカテック「5G とは？4G LTE との違いを初心者向けにやさしく解説」https://simpc.jp/rikatech/about-5g/　閲覧日 2021/4/3）

デジタル信号のクロック間の時間を有効活用できる点があります。一方で問題点として距離によって信号が届くまでに時間差が生じるため誤受信しやすいこと、FDMA 同様ノイズに強いわけではないことが挙げられます。

3G になると次は CDMA という「符号分割多次元接続方式」といった符号によって個人個人を判別する方式をとることになります。この方式をとることで多くの情報を合成(ミキサ)して同時に送れるため、TDMA のように同時に送れる情報の数に限りがないことが一番のメリットになります。この頃はスマートフォンも普及した頃で多数の端末からのインターネット接続が必要不可欠であったことも理由の１つです。また、多くの情報をある周波数上で送信するため非常にごちゃごちゃした情報が渡ってくることになります。なので、多少のノイズがあっても正確に復元できること、コードがわからないと復元できないため傍受に強いことも特徴になります。

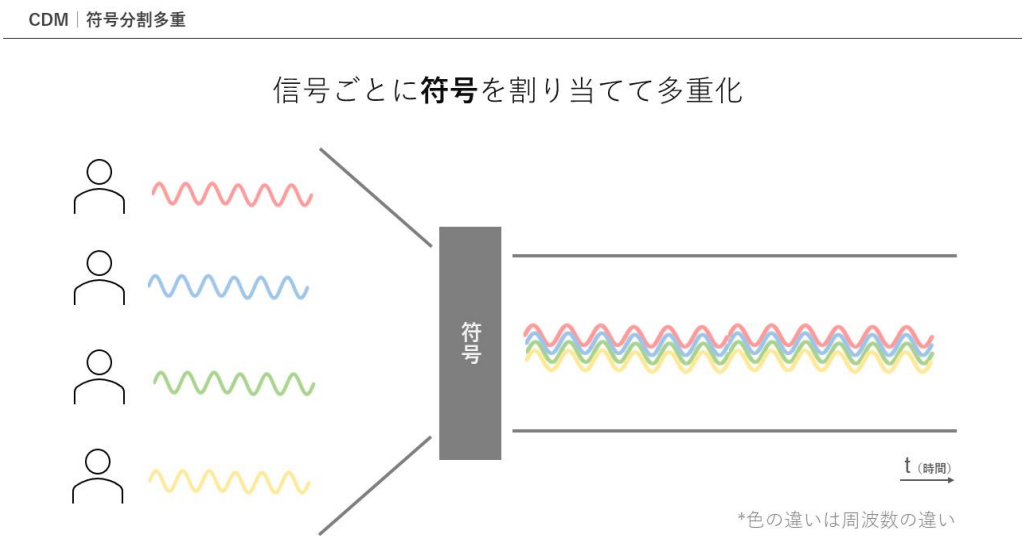


図 6　CDMA のイメージ図

（出典:TechTech ブログ「多重化とは？ 5 つの多重化方式とその原理をわかりやすく解説」https://techtech-bloger.com/multiplexing/　閲覧日 2021/4/3）

4 G では、OFDMA という「直交周波数分割多元接続方式」と SC-FDMA という「シングルキャリア FDMA」の 2 つを通信方式として利用していました。

なぜ 2 つ使う必要があったのかというと基地局が一人一人の携帯に電波を送る場合(下り)と一人一人が基地局に電波を送る場合(上り)のそれぞれに効率的な方法が存在するためです。下りでは OFDMA の方を用います。これは図 3 のようにある信号の振幅が山になる部分で他の信号の振幅を 0 にすることで非常に綺麗な信号をパラレルに送受信できることが特徴です。高速通信・周波数の利用効率を実現するためには信レベルを一定レベルで安定させる必要があります。このため時間経過や端末の移動によって受信レベル(アンテナの数)が変動する

ことであるフェージングの影響の少ないキャリア(周波数帯)を時々刻々とユーザーに割り当てることが行われています。上りでの SC-FDMA は消費電力が少なくなるようにキャリア帯域においてシリアルに送信する方式を採用しています。

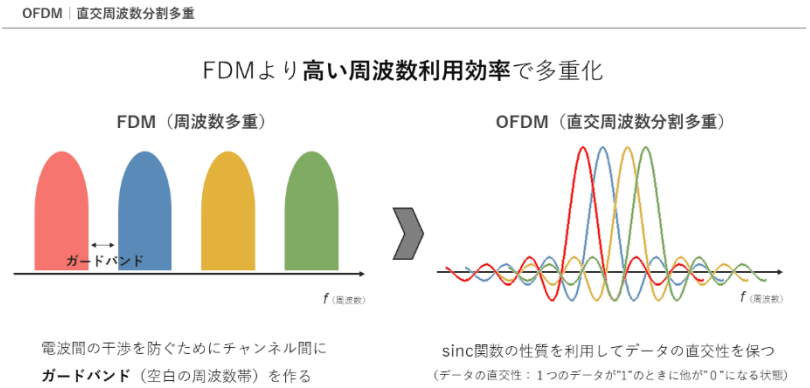


図 7　OFDMA の波長の様子

（出典:TechTech ブログ「多重化とは？５つの多重化方式とその原理をわかりやすく解説」　https://techtech-bloger.com/multiplexing/　閲覧日 2021/4/3）

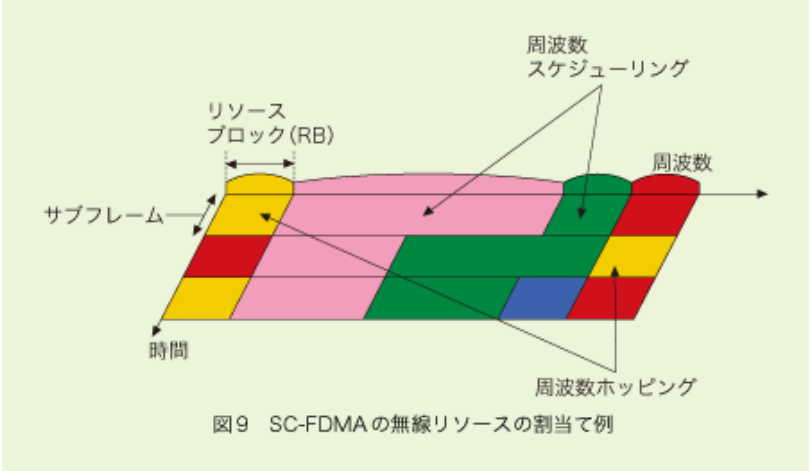


図４　無線リソースの割当の様子

(出典:NTT docomo,“3.2 上りリンク無線アクセス SC-FDMA”,“https://www.nttdocomo.co.jp/corporate/technology/rd/tech/lte/lte01/03/02.html” 閲覧日 2021/4/6)

２．5G のメリットとは

　では、次に本題である 5G について話していきましょう。今までとは異なり通信方式を変えずに大容量・高速通信・低遅延を実用化できたのは主に高周波帯の利用・広い帯域幅・アンテナ技術の向上・アーキテクチャの刷新のおかげです。大容量を実現するために高周波帯と広い帯域幅の２つを利用しています。

　高周波帯の利用で高速になるのは一秒間あたりに振動する数が増える、つまり一秒間あたりの情報量が増えるためです。(1000MHz 程度だったのが 2500MHz 以上)また、帯域幅が広がるとそれだけ情報をパラレルに送信できるようになり、大容量かつ高速に通信できるようになるのです。

　ただし、弱点として高周波帯の電波は直進性が高く、透過しにくい、減衰しやすいという特徴があります。その弱点を解消しつつ高速通信を実現したのがアンテナ技術です。電波が非常に弱いという問題を解決するために狙った方向に強い電波を放出する「ビームフォーミング」により、無駄を省きつつ問題解決に成功しています。また基地局のアンテナの数を 2~8 倍以上に増やし、多数のアンテナを内蔵しているスマートフォンを最大限活用する「Massive MIMO」という技術により、さらにパラレル通信を高速化することも試みられています。

　低遅延実用化のためには、アーキテクチャを新しくしています。遠くと通信するとき発生する伝送時間を短くすることはどうしてもできないため、クラウド

上での遅延時間や伝送時間間隔(約 1/4 倍に短縮)を短くすることで低遅延を実用化しようとしています。

このようにして今までとは異なった形で 5G が誕生したことがわかってもらえたのかなと思います。また、この 5G に関して「なぜ様々な技術を駆使して無理やり達成したのだろう」と思った方もいるかと思います。実際 5G での通信方式として NOMA というものも研究されていたようです。それには現状の 4G だと利用できるスマートフォンの数の限界が見えてきたことや IoT 社会を実現する上で必要不可欠な技術であることなどの理由により”必要に駆られて”達成した技術という印象が強いものになります。こういった社会情勢の観点から物事を見るのも面白いかもしれませんね。

参考文献

[1] 大内孝子, “5G（第 5 世代通信）を基礎から解説、通信の速度や用途は今後どう変わるのか,” “https://www.sbbit.jp/article/cont1/33874,” 2021/4/5 閲覧

[2]石井徹,” 5G ではどうして通信が速くなる？ 仕組みから探る可能性と限界,” ” https://japanese.engadget.com/jp-2020-02-27-5g.html” 2021/4/3 閲覧

[3]渡邊陽介,” 5G とは？4G LTE との違いを初心者向けにやさしく解説,” “https://simpc.jp/rikatech/about-5g/” 2021/4/3 閲覧

[4]藤田宗佑,川崎考蔵,”携帯の行方,” “http://mikilab.doshisha.ac.jp/dia/monthly/monthly08/20080430/sfujita.pdf” 2021/4/5 閲覧

[5]TechTech ブログ,”多重化とは？５つの多重化方式とその原理をわかりやすく解説”,” https://techtech-bloger.com/multiplexing/” 2021/4/3 閲覧

[6]日経ビジネス,”今さら聞けない 5 個のキホン”,” https://special.nikkeibp.co.jp/atclh/ONB/19/5G_IMPACT/article15_2/” 2021/4/5 閲覧

メモリーキーヤーキットを作ってみた

部員 K

1．はじめに

メモリーキーヤーとは、事前に送信するメッセージを記録し記録したメッセージを送信する機能をもっている機器のことである。コンテストの参加の際などには、コールサイン、コンテストナンバーなど同じメッセージを送信するため、一回一回キーで送信するのは大変であり、そのような同じメッセージを繰り返し送信しなければならないときに活躍するのがメモリーキーヤーである。今回はマルツで販売されている「４チャンネル・メモリーキーヤーを作ろう」というキットを購入したのでそれについてまとめたいと思う。

２．このキットについて

このキットはマルツで販売されており、CQ Ham radio 2008 年 6 月号掲載のメモリーキーヤーの制作に関する記事で紹介されている部品をキットとして発売しているものである。

このキーヤーの機能としては、４チャンネルのメッセージメモリによりキーヤーとして使用できるだけでなく、ランダムにコールサインを発信させる機能も持っているため、CW 受信の受信練習機としても使用できる。

3. 制作してみた

制作には２時間程度要した。キットには親切な組み立て説明書が入っており、この内容を参考にしながら制作した。通常のはんだ付けの装備があれば十分であった。制作後の基盤の様子を以下に示す。

図 8　制作後の基盤の様子

　このキットは、電源(DC7-15V)、パドル、無線機のキー端子に接続することにより使用できる。電源は、無線機用の電源装置に接続することにより、キーヤーを作動させてみた。正常にメッセージが記録され、送信されることを確認しキーヤーとして動作していることを確認できた。

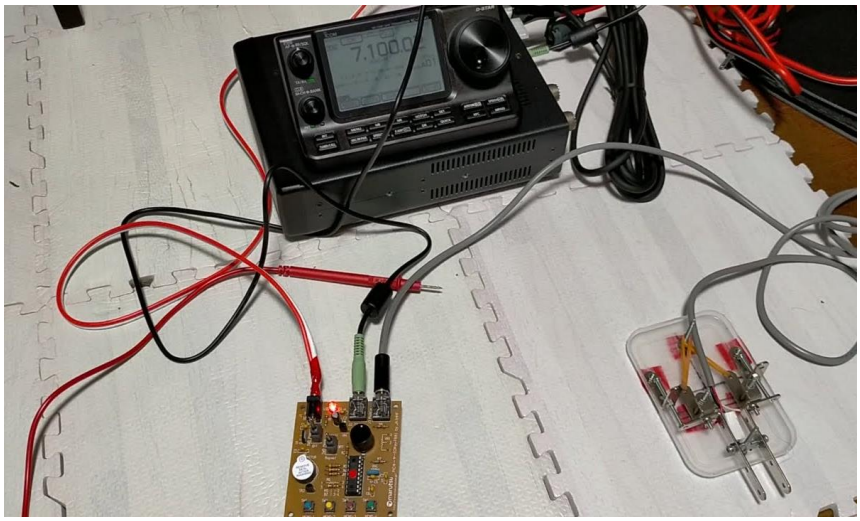


図 9　キーヤーを作動させたときの様子

4. まとめ

　この記事では、メモリーキーヤーの制作の様子をまとめてみた。短時間で簡単にメモリーキーヤーを制作することができた。また、実際に運用で使用してみたいと思った。非常にお手軽ですので皆さん制作してみてください！

FT8(8GFSK)復調・復号プログラムを MATLAB で書いてみた その 1:時間・周波数同期

JP7VTF

1. はじめに

WSJT 系ディジタルモードの一つである FT8 は、手軽に海外交信ができることも相まって、多くのアマチュア無線家が使用するディジタルモードとなりました。手軽に海外交信が可能であるのは、信号対雑音比が低い状況においても復調・復号が可能であるためです。これらは、ディジタル信号処理や強力な前方誤り訂正を使用することで実現しており、その技術に興味を持っているアマチュア無線家も多いのではないでしょうか。

本稿から数回にわたって(もしかしたら途中でやめる可能性もあり)、MATLAB での実装を踏まえながら FT8 の復調・復号プロセスについて解説します。1 回目となる本稿では、復調時の時間・周波数同期について見ていきます。一般に、ディジタル通信においては、送信側・受信側間での各種同期(時間同期、周波数同期、位相同期)が重要であり、その質が信号伝送後の誤り率

に大きく影響します。FT8 で用いられている 8GFSK においてもシンボル判定を行うタイミングや周波数の同期が誤り率に大きく影響します。FT8 ではシンボル系列に埋め込んだ 7×7 のコスタス配列を用いて、5 ms、0.5 Hz の分解能で時間・周波数同期を行った後、シンボル判定を行います。

なお、本稿は参考文献[1]の内容を追試したものになります。また、プログラムの実装については、WSJT-X のソースコード[2]を参考にしました。

2. 原理

FT8 では、送信シンボル列に埋め込んだ 7×7 コスタス配列を用いて時間・周波数ずれを検出・補正することで時間・周波数同期を実現しています。コスタス配列とは、任意の点の間の距離ベクトルがどれをとっても異なるような n×n の配列です(n は自然数)[3,4]。例として、FT8 で用いられている 7×7 のコスタス配列[5]を図 1 に示します。コスタス配列の良いところは、その自己相関特性にあります。図 2 に図 1 のコスタス配列の自己相関係数を示します。自己相関係数は自己相関関数を計算することで得られる係数です。自己相関関数とは、おおざっぱに言うと、ある関数とそれを遅延させた関数との類似性を表すような関数です。図 2 より、x、y 軸のラグ(シフト量)が共に 0 であるとき、自己相関係数が最大となることがわかります。FT8 のような FSK 信号の同期

に応用する場合、図 1 の x 軸を時間軸、y 軸を周波数軸、青色のポイントはシンボルトーンに置き換えて、送信信号中にコスタス配列を挿入します。受信側では、受信信号中のコスタス配列とあらかじめわかっている理想的なコスタス配列との相互相関を調べることで、同期をとることが可能になります。

FT8 では、一つのメッセージには図 1 の 7×7 コスタス配列が 3 組含まれており、これらを用いることで時間・周波数同期を実現しています。

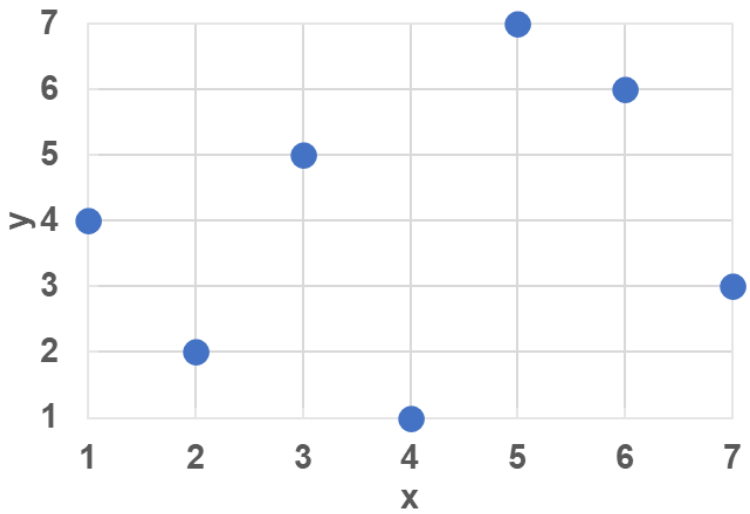


図 1：FT8 で用いられる 7×7 コスタス配列

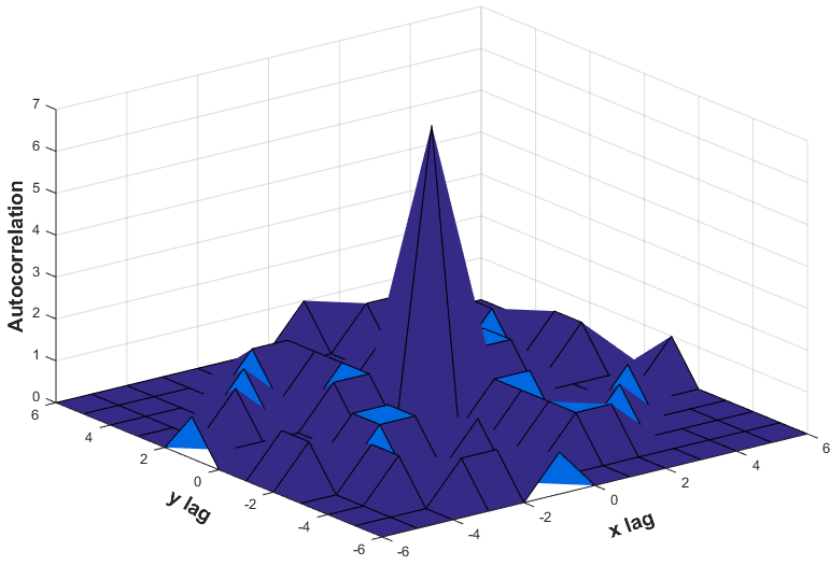


図 2：図 1 のコスタス配列の自己相関係数。x、y 方向のラグが 0 の時に最も大きなピークが現れ、他では 0 または 1 の値しか取らない。

3. 実装

今回、FT8 信号の生成、時間・周波数同期について MATLAB で実装しました。本節では、FT8 の変調方式やプロトコルの概要について述べた後に、同期の手法について述べます。同期処理についてはさまざまな手法で実装することができますが、今回は WSJT-X の手法に沿っています。この手法では、初めに 40 ms、3.125 Hz の分解能で大まかな同期を行った後に、5 ms、0.5 Hz の分解能で細かな同期を行うという 2 段階のプロセスを経ています。

3.1 テスト信号の作成(FT8 の変調方式・プロトコル)

初めにテスト用の信号を作成します。FT8 の変調方式は 8GFSK となっており、1 シンボルあたり $\log_2(8)=3$ bit の情報を送ることができます。変調速度は 6.25 baud に対して、トーン間隔は 6.25 Hz であり、変調指数は 1 となっています。また、BT 積は 2 であるため[2]、ベースバンド信号の帯域制限を行うガウスフィルタの帯域(半値半幅)は 12.5 Hz となります。

図 3 に FT8 のメッセージ構成を示します。FT8 は 1 つのメッセージあたり 79 個のシンボルを持っています。このうちの 58 シンボルはコールサインやグリッドロケータ等の情報が格納されるペイロードに用いられ、残りの 21 シンボル(7 シンボル×3)は 7 ×7 コスタス配列×3 に使用されます。7×7 コスタス配列はメッセージの開始位置、中間位置、終了位置に挿入されています。

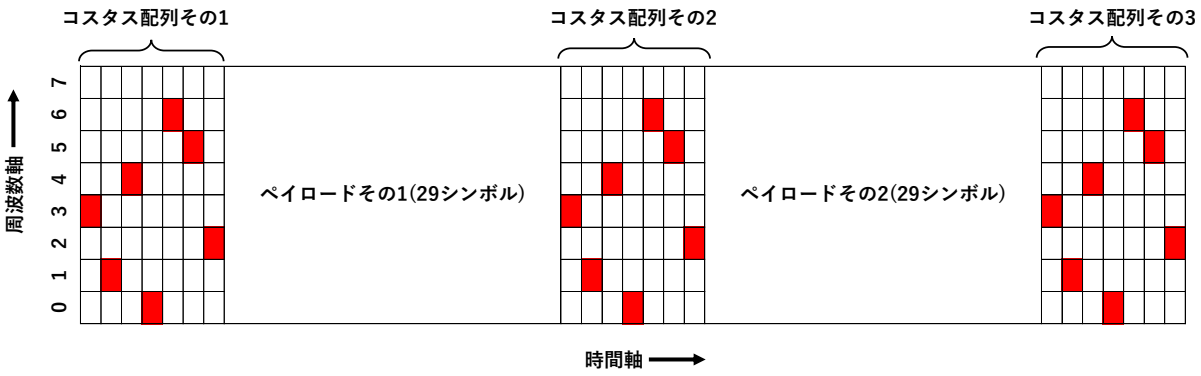


図 3：FT8 のメッセージ構成

[2] Source code¥lib¥ft8¥ft8sim.f90

復調プログラムの作成と検証にあたっては、実際に送信されうる信号を使って確認を行った方が、現実味が増すだろうと考え、「JA7YAA JH7YAA QM65」というメッセージを用いました。詳細については Appendix に譲り、こちらでは結果のみを示します。リスト 1 にメッセージのシンボル列を示します。なおリスト 1 では、8 つあるトーンに対して周波数が低い方から 0, 1, 2, …7 というように数字を割り当てています。また、赤字で示しているのが 7×7 コスタス配列です(図 1 の y 軸の値から-1 した値に一致しています)。

リスト 1 「JA7YAA JH7YAA QM65」を表すメッセージのシンボル列。

同期用のコスタス配列を赤字で示している。

3 1 4 0 6 5 2 5 2 4 3 3 6 4 7 2 1 1 0 1 4 6 4 4 1 0 1 4 6 2 5 3 3 7 0 5 3 1 4 0 6 5 2 4 3 7 1 3 6 5 7 1 3 6 2 0 2 1 2 3 0 7 1 0 4 5 5 1 4 1 5 7 3 1 4 0 6 5 2

上記のメッセージを用いて、周波数が最も低いトーンの周波数(以後、f_cと定義)が 1500 Hz となるような 8GFSK 信号を生成し、サンプリング周波数 12 kHz の wav ファイルとして保存しました[3]。また、図 4 のように WSJT-X で実際にデコードさせ、正しい信号が生成できているか確認しました。

[3] WSJT-X ではサンプリング周波数を 12 kHz としているため。wav 形式にしたのも、WSJT-X に読み込ませるためです。また、WSJT-X 上ではオーディオ出力の周波数の範囲で設定できますが、この設定値は最も周波数の低いトーン信号の周波数を表しています。

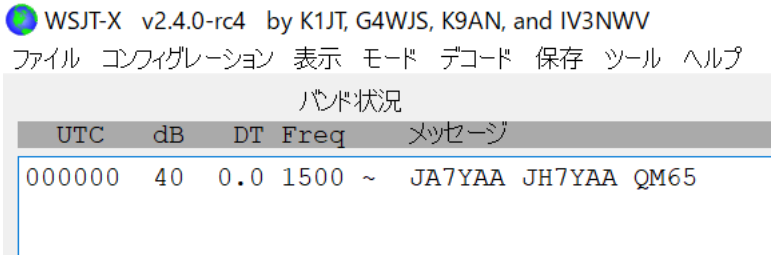


図 4：生成した信号を WSJT-X で復調した結果

FT8 では、15 秒の T/R シーケンスで交信が行われます。受信側では、相手から信号が送られてくるタイミングで、受信信号を 15 秒間録音します。この時のサンプリング周波数は 12 kHz であるため、録音データのサンプル数は 15 s×12 kSample/s = 18 kSample となります。15 s の信号を録音し終えたところから、下記に示す周波数・時間同期に関する処理を始めます。

3.2 大まかな周波数・時間同期

まずは、40 ms、3.125 Hz の分解能で大まかな同期を行います。録音データの端から端まですべてを細かな分解能で掃引して周波数・時間同期しようとすると、処理に時間がかかってしまうため、大まかに当たりを付けるといったところでしょうか。

受信信号に対して、時間窓 160 ms、時間掃引幅 40 ms で短時間フーリエ変換(STFT：Short Time Fourier Transform)します。リスト 2 に MATLAB

で実装したプログラムを示します。

160 ms というのは、ボーレート 6.25 Hz の逆数であり、シンボル周期を表しています。時間掃引幅が 40 ms であるため、1/4 シンボルずつ時間窓をずらしながら離散フーリエ変換(DFT：Discrete Fourier Transform)します[4]。160 ms の窓で切り出すサンプル数は 12 kSample/s × 160 ms = 1920 Sample となります。WSJT-X の実装では、この 1920 Sample の後ろに 1920 Sample(160 ms 分)の 0 を追加した合計 3840 Sample で DFT を実行します(ゼロパディング[5])。DFT では、周波数分解能(周波数ビン)=サンプリング周波数/時間サンプル数となるため、3840 Sample では、12 kHz/3840 = 3.125 Hz となります。DFT の結果は、-6 kHz から 6 kHz の範囲の振幅スペクトルとして出力されますが、実数信号をフーリエ変換においては、正の周波数と負の周波数とでは複素共役の関係にあり、振幅の大きさについては同じであるため、以後の処理は正の周波数の成分のみを用います。

---

[4] 高速フーリエ変換(FFT：Fast Fourier Transform)は DFT を高速化できるアルゴリズムのこと。今回作ったプログラムでも FFT を使用しています。サンプリング数が 2 のべき乗ではないので、中でどうなってるか知りませんが、今回は使えればおっけー。

[5] サンプリング周波数を固定としたとき、フーリエ変換の不確定性原理を考えると、時間幅・周波数幅≧const. の関係があり、両者を共に小さくすることはできません。短時間フーリエ変換においては、窓の時間幅を短くすると時間分解能は向上するが周波数分解能は劣化し、窓の時間幅を長くすると周波数分解能は向上するが時間分解能は劣化するというトレードオフがあります。また、窓の時間幅を長くすると複数のシンボルを取り込んでしまい、スペクトル上に複数のトーンが表れてしまう可能性があります。

WSJT-X の実装では、窓の幅を FT8 のシンボル周期と同じ 160 ms としています。また、窓で切り出した時間信号にゼロパディングしたうえで DFT することで、周波数分解能の向上を図っているようです。

この正の周波数成分について、電力スペクトルを計算します。図 5 にリスト 2 の計算結果として得られるスペクトログラムを示します[6](プログラム中の変数名は powerSpectrum)。15 s の録音データであるため、窓で切り出せる時間方向のサンプル数は、15 s/40 ms−3=372 個となります。このため、変数 powerSpectrum は行が周波数、列が時間を表しており、サイズは 1920×372 となります。160 ms の時間窓で 40 ms ごと時間掃引しているため、powerSpectrum(:,1)は 0~160 ms、powerSpectrum(:,2)は 40 ms~200 ms、powerSpectrum(:,3)は 80 ms~240 ms…という対応関係になります。

リスト 2：大まかな周波数・時間同期のために STFT を実行するプログラム

```
% FT8 synchronization & demodulation
%% ファイルの読み出し
disp('ファイルの読み出し');
GFSK = audioread('FT8_signal.wav');

%% 160 msの時間窓を40 msごと時間掃引してFFT(15 s / 40 ms - 3 = 372点)
disp('160 msの時間窓を40 msごと時間掃引してFFT');
NMAX = 12000 .* 15;      % 全サンプル数(=180000 sample)
NSPS = 12000 .* 0.16;    % 1シンボルあたりのサンプリング数(=1920 sample)
NSTEP = NSPS./4;         % 窓を掃引時のタイムステップサイズ(=480 sample)
NSYM = NMAX./NSTEP - 3; % スペクトル数(=372こ)

for I=0:(NSYM-1)
```

[6] 皆様おなじみのウオーターフォール画面では縦軸が時間、横軸が周波数ですが、図 5 では逆になっています。

```
    temp = fft([GFSK((NSTEP.*l+1):(NSTEP.*l+NSPS),1); zeros(NSPS,1)]); % 160 msの
ゼロパッディングをしてfft
    ampSpectrum(:,l+1) = temp(1:NSPS,1); % 正の周波数成分のみを使用
    powerSpectrum(:,l+1) = abs(temp(1:NSPS,1));
end

% グラフを書くための設定。本題とは関係なし
X = 0:0.04:(371.*0.04);
Y = 0:3.125:(6000-3.125);
imagesc(X,Y,powerSpectrum);
axis xy; ylim([-62.5 62.5]./2+1500+6.25.*3.5);
xlabel('Time [s]'); ylabel('Frequency [Hz]');
```

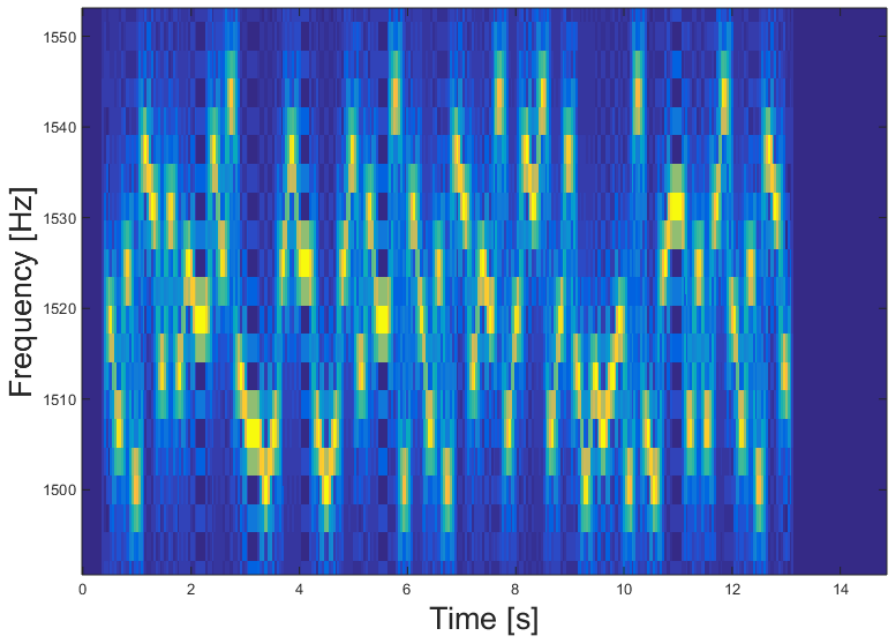


図 5：リスト 2 の処理で得られたスペクトログラム。

次に、FT8 の信号が 15 秒間のスペクトログラム中のどこにあるのか探しま

す。スペクトログラム中での位置は、時間t_0(録音データの始まりを$\tau = 0$ と定義、 $-2 \leqq \tau \leqq 3$)[7]、周波数f_c(3.1 節での定義と同じ)で表すことができます。信号の探索は時間遅延が$-2 \leqq \tau \leqq 3$(powerSpectrum は 40 ms ステップで時間掃引しているため、$-2 \leqq \tau \leqq 3$は 5/0.04=125 段階に相当)、周波数は WSJT-X 実行時のウオーターフォールに表示されての範囲(例えば、200~2500 Hz)で行われます。

信号の位置検出は、信号中のコスタス配列の位置検出によって実現されます。これは、ある時間τ、周波数f_cにおいて、コスタス配列の信号があると予想される位置の電力スペクトル密度の総和と、信号あるべきではない位置の電力スペクトル密度の総和の比を計算し、その比が最大となる位置を探し出すことで実装できます。リスト 3 にこの電力比を求めるプログラムを示します。

リスト 3：スペクトログラム中からコスタス配列を探索するプログラム

```
%% マスクをかけてパワーを算出
disp('コスタス配列のマスクをかけてパワー比を算出');
NSSY = 4;    % 1シンボルあたりのスペクトル数
NFOS = 2;    % 周波数ビンのオーバーサンプリングファクター
costasArrayTable = [3 1 4 0 6 5 2]; % 7x7コスタスアレー

nfstart=1; nfstop = 1920./2;
```

[7] WSJT-X ではτ上のどこを DT=0 と定義しているのか定かではありませんが、図 4 の結果を得るために時間遅延量を調整した結果から考えると、録音データの最初から 0.5 秒のようです。

```
ntstart = -2./0.04; ntstop = 3./0.04;

for nf=nfstart:NFOS:nfstop
    for nt0=ntstart:ntstop
        ta=0; tb=0; tc=0;
        t0a=0; t0b=0; t0c=0;
        for n=0:6
            m = n.*NSSY + nt0;

            if(m > 1)
                ta = ta + powerSpectrum(NFOS.*costasArrayTable(1,n+1)+nf,m);
                t0a = t0a + sum(powerSpectrum(NFOS.*[0:6] + nf,m));
            end

            tb = tb + powerSpectrum(NFOS.*costasArrayTable(1,n+1)+nf,m + 144);
            t0b = t0b + sum(powerSpectrum(NFOS.*[0:6] + nf,m + 144));

            if((m+288) < NSYM)
                tc = tc + powerSpectrum(NFOS.*costasArrayTable(1,n+1)+nf,m + 288);
                t0c = t0c + sum(powerSpectrum(NFOS.*[0:6] + nf,m + 288));
            end
        end

        t=ta+tb+tc;
        t0=(t0a+t0b+t0c-t)./6;
        sync_abc = t./t0;

        t=tb+tc;
        t0=(t0b+t0c-t)./6;
        sync_bc = t./t0;

        sync2d(nf,nt0-ntstart+1) = max([sync_abc sync_bc]); % sync2d(nf-
nfstart+1,nt0-ntstart+1) = max([sync_abc sync_bc]);に等価
    end
end
```

```
%% 候補を選別する
disp('候補の選別');
% まずは表にする
[L,N] = size(sync2d);
clear candidate;
cc = 1;
for l=1:L % freq
    for n=1:N % time
        if (sync2d(l,n) > 1.5)
            candidate(cc,:) = [l n sync2d(l,n)]; % 変数candidate [周波数 時間 パワ
一比]
            cc = cc + 1;
        end
    end
end
candidate = flipud(sortrows(candidate,3));


% ±4 Hz以内の候補は削除

cc = 1;
while (cc < length(candidate))
    candidate((abs(candidate(cc,1) - candidate(:,1)) < 2) & (candidate(:,2) ~=
candidate(cc,2)),:) = [];
    cc = cc + 1;
end

if (length(candidate) > 200)
    candidate(201:end,:) = []; % 上位200こに絞る
end

% [s], [Hz]に単位変換
candidate(:,1) = (candidate(:,1)-1).*6000./1920; % 1? 0?
candidate(:,2) = (candidate(:,2)-1).*0.04-2;
```

例えば$\tau = 0$を仮定したときの 3 つあるコスタス配列の内の 1 つ目については、powerSpectrum(:,1)の$f_c + 18.75$ Hz、powerSpectrum(:,5)の$f_c + 6.25$ Hz、powerSpectrum(:,9)の$f_c + 25$ Hz、powerSpectrum(:,13)の$f_c + 0$ Hz、powerSpectrum(:,17)の$f_c + 37.5$ Hz、powerSpectrum(:,21)の$f_c + 31.25$ Hz、powerSpectrum(:,25)の$f_c + 12.5$ Hzの周波数ビンの値を足し合わせ、コスタス配列があると予想される位置の電力スペクトル密度の和を求めます。リスト 3 でのプログラム中では、この和を変数 ta に格納しています。信号の中間、最後に挿入されているコスタス配列についても同様に計算した後に変数 tb、tc に格納しており、さらに、ta、tb、tc の和を t に格納しています。トーン信号があるべきではない位置の電力スペクトル密度の和の計算については、powerSpectrum 中のトーンがあると思われる周波数ビンの値($f_c + 0$、$f_c + 6.25$、$f_c + 12.5$、$f_c + 18.75$、$f_c + 25$、$f_c + 31.25$、$f_c + 37.5$ Hz に対応)を足し合わせることを、powerSpectrum(:,1)、powerSpectrum(:,5)、powerSpectrum(:,9)…のそれぞれに対して行い、さらにその結果をすべて足し合わせた後に ta、tb、tc を引き、変数 t0 に格納します。これらの結果を元に、両者の比である t/t0 を計算します。

上記の処理について、τ、f_cを掃引しながら行います。これらの結果を図 6 に

示します。図 6 では、$\tau = 0.56$ s、$f_c = 1500$ Hzにおいて、最大値を取っている

ことがわかります。

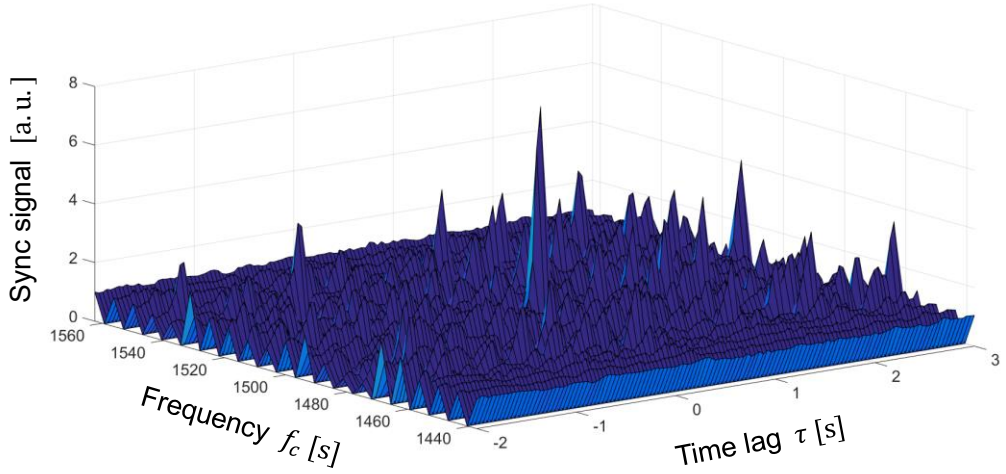


図 6 リスト 3 のプログラムで得られた電力比(変数 sync2d)。

この値が最大値を取る点から実際の時間遅延・周波数を推定する。

上図では、$\tau = 0.56$ s、$f_c = 1500$ Hzで最大値を取っている。

リスト 3 のプログラムでは、sync2d から、遅延時間・周波数の候補をリス

ト化し、最大値を取る時間遅延量・周波数を探索します。リスト化する際に

は、電力比の値が 1.5 以上となるものについてリスト(変数 candidate)に追加

しています[8]。ソートを行った後に、上位の候補から±3.125 Hz 以内にある

[8] 同期信号が 1.5 以上となる候補がない場合には、デコードを終了すると書いてあるので、本来は終了する際の処理についても書いておくべきですが、めんどくさいのでここでは書いていません。実際に使用するソフトを作る場合には、そういったところも十分考慮

下位の候補を削除します。さらに、そこから上位 200 位の候補を抜き出します。WSJT-X では、上位の候補から以後の処理を行い、デコードできなければ下位の候補のデコードを試みるという処理を繰り返します(本稿のプログラムでは省略)。

3.3 細かな周波数・時間同期

前節で述べた処理では、40 ms、3.125 Hz の分解能で信号の同期が可能ですが、逆に言うと、それだけのあいまいさが残っています。本節の処理では、5 ms、0.5 Hz というさらに高い分解能で同期を行います。

本節で行う同期には、相互相関係数を利用します。相互相関関数とは、大雑把言うと、２つの関数の類似性を示すような関数です。自己相関関数はある関数とそれを遅延させた関数を用いて計算するのに対して、相互相関関数ではある関数と遅延させた別の関数を用いて計算します。WSJT-X での実装では、1 つ目として関数は受信信号、2 つ目の信号として基準となる理想的な信号を用いて相互相関係数を計算し、相互相関係数が最大を取る時間遅延量・周波数を算出することで、同期を実現します。

具体的なプログラムをリスト４に示します。まず、相互相関係数を計算する

して作る必要あります。

際に使用する 7×7 コスタス配列の基準信号を生成します。サンプリング間隔は 5 ms(FT8 のシンボル間隔は 160 ms であるため、1 シンボルあたり 32 sample)とし、最も低周波なトーンの周波数が 0 Hz となるような複素信号(以下、ベースライン信号と呼称)を生成します。

受信信号側についてもいくつか処理を施します。まず、受信信号に対してゼロパディングをしたうえで DFT を実行します。受信信号は 12 kHz × 15 s = 180000 Sample の時系列データですが、これを 12 kHz × 16 s = 192000 Sample の時系列データとなるように 12000 Sample 分のゼロパッディングを施します。周波数分解能は 12 kHz / 192000 Sample = 0.0625 Hz となります。その後、関数 fft を用いて周波数領域の写像に変換し、変数 ampSpectrum2 に格納します。

次に、ダウンコンバージョンによるベースライン信号への変換と、必要な帯域の信号のみを取り出すための帯域制限を行います。まず、前節で計算した f_c(今回使用したテスト信号では 1500 Hz)が 0 Hz となるようにダウンコンバージョンします。ampSpectrum2 に格納された振幅スペクトル中の$f_c - 9.375$ [Hz]から$f_c - 53.125$ [Hz]の成分のみを抜き出し、Raised-cosine filter(シンボル周期 1/(900×0.0625)=17.8 ms、ロールオフ率α=1/9≅0.11 相当[9])を

[9] Raised-cosine filter は PSK や QAM のベースバンド信号に対して符号間干渉フリーで帯域制限するために使用されます。FSK 信号に対して今回のような使い方をするのは理

かけます。なお、前節で計算したf_cには 3.125 Hz のあいまいさがあるため、そこで、本節ではダウンコンバージョンする際の周波数変化量について、f_cを中心に 0.5 Hz 刻みで±2.5 Hz の範囲で掃引し、それぞれの場合について相互相関係数の評価を行います。次にこの結果を逆 FFT して、時間領域の写像に変換します。逆 FFT を行うに当たっては、f_cが 0 Hz となるように変数 ampSpectrum3 にスペクトルの数値を格納しています。周波数分解能 0.0625 Hz のスペクトルをサンプリング周期 5 ms(サンプリング周波数 200 Hz)の時間信号に変換するためには、200 Hz÷0.0625 Hz=3200 個の bin が必要となります。このうちの 1000 個については Raised-cosine filter をかけた後に得られた 62.5 Hz 分のデータを用い、他についてはゼロで埋めています。

　次に、逆 FFT で得られた時間信号から 7×7 コスタス配列があると予想される時間の信号を読み出し、理想的な 7×7 コスタス配列との相互相関係数を計算します。さらに、自身の複素共役との積を計算しておきます。元の相互相関係数は複素数ですが、複素共役との積(元の複素数の大きさの自乗に等しい)は必ず実数になります。FT8 の信号には 3 つのコスタス配列が含まれています

由があるのかないのかよくわかりません。ロールオフ率 α=1/9 は WSJT-X の実装に倣ったものです。シンボル周期 1/(900×0.0625)=17.8 ms、ロールオフ率 α=1/9≅0.11 相当の Raised-cosine filter は、透過率 1 の帯域が 50 Hz(=6.25 Hz×8)、透過率 0 となる帯域が 62.5 Hz(=6.25×10)となります。変数 ampSpectrum2 に格納された振幅スペクトラムの分解能は 0.0625 Hz であるため、周波数ビンの数は 50 Hz は 50 Hz÷0.0625 Hz=800 個、62.5 Hz÷0.0625 Hz=1000 個とキリのいい数字になります。

が、それぞれにおける相互相関係数の大きさの自乗を計算し、足し合わせたものを変数 S に格納します。相互相関係数は、ラグが-8 sample から 8 sample の範囲で計算します。これはすなわち、-40 ms から 40 ms の範囲で時間掃引していることに相当します。周波数・時間掃引をして計算した相互相関係数のグラフを図 7 に示します。図 7 より、オフセット時間・オフセット周波数がそれぞれ 0 ms、0 Hz において相互相関係数が最大値を取っています。ここから受信信号が始まっていると推定でき、これによって周波数・時間同期が達成されます。最後に大まかな同期と細かな同期での結果を組み合わせて、最終的な結果を計算します。今回の用いた信号の場合は、周波数が 1500 Hz、時間は 0.56 s となりました。以後は、ここで得られた周波数・時間遅延量を元に、ペイロード中のシンボルの読み出しを行います。

リスト 4 相互相関係数の計算・比較による細かな時間・周波数同期の実装

```
%% より細かい同期において基準とするコスタス配列のベースライン信号を生成
phase = 0;
clear Ref;
for l=0:6
    for m=1:32
        Ref(32.*l+m) = exp(1i.*(phase+pi./2));
        phase = phase - 2.*pi.*6.25.*costasArrayTable(1,l+1).*0.005; % 位相の回転方向がとっても大事
        phase = mod(phase,2.*pi);
    end
```

```
end
Ref=Ref';

%% 候補の読み出し
fc = candidate(1,1);
dt = candidate(1,2);

%% もっと周波数分解能を細かくしてFFT
NMAX2 = 12000 .* 15;      % 全サンプル数(=180000 sample)
NSPS2 = 12000 .* 16;    % 1シンボルあたりのサンプリング数(=192000 sample)
NSTEP2 = 12000 .* 0.05;          % 窓を掃引時のタイムステップサイズ(=600 sample)
NSYM2 = NMAX2./NSTEP2; % スペクトル数(=300こ)

ampSpectrum2 = fft([GFSK; zeros(NSPS2-NMAX2,1)]);

clear S;

% raised cosine filterの用意
RCF = ones(1000,1);
RCF(1:100,1) = 0.5 .* (1-cos((0:99).*pi./100));
RCF(901:1000,1) = 0.5 .* (1+cos((0:99).*pi./100));


for l=-5:5 % 周波数 0.5Hz x 11 step = ±2.5 Hz

    ampSpectrum3 = ampSpectrum2((((fc-0)-6.25.*1.5+0.5.*l).*...
        192000./12000+1):(((fc-0)+6.25.*8.5+0.5.*l).*192000./12000),1);
    ampSpectrum3 = ampSpectrum3 .* RCF; % フィルタをかける

    %% 周波数ダウン
    ampSpectrum3 = [ampSpectrum3(151:1000,1); zeros(2200,1); ampSpectrum3(1:150)];

    %% 逆FFT
    baseline = ifft(ampSpectrum3);
    % baselineとGFSKの間には0.06 s程のズレがあったので補正
    baseline = circshift(baseline,[round(0.06.*200) 0]);
```

```
    %% 相互相関関数の計算
    ndt = round(dt./0.005);

     xc1 = xcorr([zeros(8,1); Ref; zeros(8,1)],[zeros(8,1);
baseline((ndt+1):(ndt+224),1); zeros(8,1)]);
     xc2 = xcorr([zeros(8,1); Ref; zeros(8,1)],[zeros(8,1);
baseline((ndt+1152+1):(ndt+1152+224),1); zeros(8,1)]);
     xc3 = xcorr([zeros(8,1); Ref; zeros(8,1)],[zeros(8,1);
baseline((ndt+2304+1):(ndt+2304+224),1); zeros(8,1)]);


    % 5 ms x 17 step = ±40 ms

    S(l+6,:) = abs(xc1(232:248)).^2 + abs(xc2(232:248)).^2 + abs(xc3(232:248)).^2;
end

[N,L] = size(S);
X = (0:(L-1)).*5-40;
Y = (0:(N-1)).*0.5-2.5;
figure;
surface(X,Y,S); view(3); grid;
%xlabel('Offset time [ms]');
%ylabel('Offset frequency [Hz]');
xlim([-40 40]);
ylim([-2.5 2.5]);

[C,I] = max(S(:));
[N,L] = ind2sub(size(S),I);

fc_offset = (N-6) .* 0.5;   % Hz
tstart_offset = (L-9) .* 0.005; % second

% 最終的に得られた、信号が始まる時間・周波数の計算
fc_total = fc + fc_offset
tstart_toral = dt + tstart_offset
```

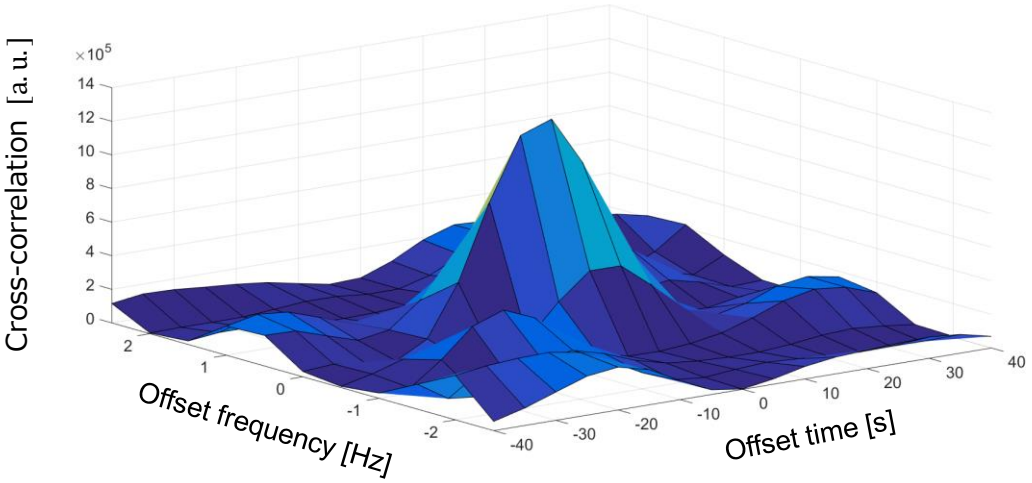


図 7 相互相関係数の計算結果。オフセット時間 0 ms、

オフセット周波数 0 Hz において相互相関係数が最大となる。

4. まとめ

本稿では、FT8 の復調における、時間・周波数同期について、MATLAB での実装を交えながら説明してきました。今回の実装例は WSJT-X の実装を参考にしたもので、時間軸・周波数軸を 5 ms、0.5Hz という細かい分解能で同期することが可能です。次回(ないかもしれませんが)は、誤り訂正符号の復号などについて書きたいと思います。

Appendix テスト信号の生成

FT8 の信号の生成の流れは、同じ WSJT-X のディジタルモードである FST4W と同様ですので、この Appendix については前号の部誌の記事[6]を参考にしながら読んでください。初めにリスト 5 にプログラムを示します。

リスト 5：テスト信号を生成するプログラム。

```
% メッセージをを生成(i3 = 1(standart message))
payload = [c28('JA7YAA') 0 c28('JH7YAA') 0 0 g15('QM65') 0 0 1];

% CRCの付与
data = [payload zeros(1,14)];
data_crc = crcRemainder([data zeros(1,5)],...
    logical(unicode2native(dec2bin(hex2dec('6757')),'Shift_JIS')-48));
data_crc = data_crc(1,6:96) + data;

% FECの付与
data_ldpc = logical(mod(data_crc * generatorMatrixFT8,2));

% 7x7のコスタス配列を準備
S = [3 1 4 0 6 5 2];

% チャネルシンボルに変換
M_A = baseband2FT8symbol(data_ldpc(1,1:87));
M_B = baseband2FT8symbol(data_ldpc(1,88:174));
chSym = [S M_A S M_B S];

% オーバーサンプリングしてBT=2のガウスフィルタにかける
%OSRchSym = 64;
OSRchSym = 128.*15;
chSymOS = repelem(chSym,OSRchSym);
```

```
%chSymGauss = conv(chSymOS,gaussianImpulse(2./0.16,201,OSRchSym./0.16));
chSymGauss = conv(chSymOS,gaussianImpulse(2./0.16,1001,OSRchSym./0.16));
freqz(gaussianImpulse(2./0.16,1001,OSRchSym./0.16),1,10001,OSRchSym./0.16);

Amp = 0.1;
dftone = 1./0.16;
fc = 1500;
Fs = 12000;
GFSK = zeros(1,15.*Fs);
f = zeros(1,15.*Fs);

phase = 0;
for l = 1:length(chSymGauss)
    for Csample = 1:(Fs.*0.16./OSRchSym)
        phase = mod(phase + 2.*pi.*(fc + (dftone .* chSymGauss(1,l)))./Fs,2.*pi);
        f(1,l.*(Fs.*0.16./OSRchSym)+Csample) = (fc + (dftone .* chSymGauss(1,l)));
        GFSK(1,l.*(Fs.*0.16./OSRchSym)+Csample) = Amp .* sin(phase);
    end
end

DT = 0.46; %0.25
t = (0:(15.*Fs-1))./Fs;

% 立ち上がりと立下りの処理
intd = 0.05;
t = (0:(15.*Fs-1))./Fs;

GFSK = circshift(timeWindowFT8(),[0 intd.*Fs]) .* GFSK;
GFSK = circshift(GFSK,[0 DT.*Fs]);

figure;plot(t,GFSK); % 表示が微妙
hold on; plot(t,circshift(f-fc,[0 DT.*Fs])); ylim([0 50]); % 表示が微妙
% 出力
audiowrite(['FT8_signal','.wav'],GFSK,Fs);
```

このプログラムでは、「JA7YAA JH7YAA QM65」というメッセージを生成します。FT8 をはじめとした WSJT-X 系のモードでは、メッセージを可逆圧縮する Source Encoding の形式が統一されています(ただし、前号で掲載した FST4W は例外)。今回のプログラムでは、参考文献[5]の Table 1 中の i3=1 Std Msg 形式を用いています。この形式を用いたときの 77 bit ペイロードへのビット割り当てを図 8 に示します。「c28」にはコールサイン、「r1」、には「/R」を付けるか否かのフラグ[10]、「R1」には了解を示す「R」を付けるかのフラグ、「g15」にはグリッドロケータの情報が入ります。コールサインから「c28」形式、グリッドロケータから「g15」形式への変換を関数化したものをリスト 6、7 に示します。これらの処理は前号の部誌の記事[6]で紹介した FST4W のものと全く同じです。

MSB LSB

コールサイン 28 bit	"/r" 1 bit	コールサイン 28 bit	"/r" 1 bit	"R" 1 bit	グリッドロケータ 15 bit	形式指定 3bit (001)

図 8：std Msg におけるペイロードのビット割り当て

[10] 北米の VHF コンテストにおいて「rover」を表すためのサフィックスらしい。

リスト６ コールサインを符号化する関数 c28()

```
function y = c28(callSign)
% matlabはunicodeが使われているのでASCIIに変換
callSign_int = uint64(unicode2native(callSign,'Shift_JIS'));

% 1文字目
if callSign_int(1,1) == 32
    callSign_int(1,1) = 0; % スペースを固有の整数値に変換
elseif ((48 <= callSign_int(1,1)) & (callSign_int(1,1) <= 57))
    callSign_int(1,1) = callSign_int(1,1) - 47; % 数字を固有の整数値に変換
elseif ((65 <= callSign_int(1,1)) & (callSign_int(1,1) <= 90))
    callSign_int(1,1) = callSign_int(1,1) - 54; % 英字を固有の整数値に変換
end

% 2文字目
if ((48 <= callSign_int(1,2)) & (callSign_int(1,2) <= 57))
    callSign_int(1,2) = callSign_int(1,2) - 48;
elseif ((65 <= callSign_int(1,2)) & (callSign_int(1,2) <= 90))
    callSign_int(1,2) = callSign_int(1,2) - 55;
end

% 3文字目
callSign_int(1,3) = callSign_int(1,3) - 48;

% 4～6文字目
for m=4:6
    if callSign_int(1,m) == 32
        callSign_int(1,m) = 0; % スペースを固有の整数値に変換
    elseif ((65 <= callSign_int(1,m)) & (callSign_int(1,m) <= 90))
        callSign_int(1,m) = callSign_int(1,m) - 64; % 英字を固有の整数値に変換
    end
end

N = callSign_int(1,1).*36 + callSign_int(1,2);  % 1-2文字目
N = N.*10 + callSign_int(1,3);  % 3文字目
```

```
N = N.*27 + callSign_int(1,4);  % 4文字目
N = N.*27 + callSign_int(1,5);  % 5文字目
N = N.*27 + callSign_int(1,6);  % 6文字目

% 謎の処理
NTOKENS = 2063592;  % 謎の数字 = 2^3 * 3 * 85983
MAX22 = 4194304;    % =2^22
N = N + NTOKENS +MAX22; % 謎の足し算
N_lgc = logical(unicode2native(dec2bin(N),'Shift_JIS')-48);
bitmask28 = logical(unicode2native(dec2bin(2.^28-1),'Shift_JIS')-48);
while length(N_lgc) < 28
    N_lgc = logical([0 N_lgc]);
end
while length(bitmask28) < length(N_lgc)
    bitmask28 = logical([0 bitmask28]);
end
y = and(N_lgc,bitmask28);
end
```

リスト７ グリッドロケータを符号化する関数 g15()

```
function y = g15(loc)
%% グリッドのコーディング
loc = upper(loc);
loc_int = uint64(unicode2native(loc,'Shift_JIS'));

loc_int(1,1:2) = loc_int(1,1:2) - 65;
loc_int(1,3:4) = loc_int(1,3:4) - 48;
M = ((loc_int(1,1) .* 18 + loc_int(1,2)).* 10 + loc_int(1,3)).* 10 + loc_int(1,4);

M_lgc = logical(unicode2native(dec2bin(M),'Shift_JIS')-48);
while length(M_lgc) < 15
    M_lgc = logical([0 M_lgc]);
end
```

```
y = M_lgc;

end
```

次に、77 bit ペイロードの情報に対して、CRC を付与し、91 bit のメッセージに変換にします。CRC の生成多項式としては 0x6757 を使用します。少しややこしいですが、77 bit ペイロードの LSB 側に 5 ビット分の 0 を追加した 82 ビットの系列に対して剰余を計算します。このとき得られた 14 bit の剰余を、元の 77 bit の系列の LSB 側に続けることで CRC の付与を行います。これにより 91 bit の系列が得られます。

続いて、LDPC を用いた誤り訂正符号の生成を行います。これは、生成行列との乗算により求めることができます。リスト 8 に FT8 の生成行列を出力する関数を示します。FT8 の生成行列については、ソースコード[2]の lib¥ft8¥ldpc_174_91_c_generator に 16 進数表示で記述されています。

リスト 8 生成行列を出力する関数 generatorMatrixFT8()

```
function y = generatorMatrixFT8
%% LDPC生成行列の準備
% ソースコードのlib¥ft8¥ldpc_174_91_c_generator.f90よりコピペした行列
generatorMatrixChara = [...
'8329ce11bf31eaf509f27fc';
-中略-
'608cc857594bfbb55d69600'];
```

```
for m = 1:(174-91)
    for n=0:4
        temp = logical(unicode2native(dec2bin(hex2dec(...
            generatorMatrixChara(m,(n.*4+1):(n.*4+4)))),'Shift_JIS')-48);
        while length(temp) < 16
            temp = [false temp];
        end
        generatorMatrix(m,(n.*16+1):(n.*16+16)) = temp;
    end

    temp = logical(unicode2native(dec2bin(hex2dec(...
        generatorMatrixChara(m,21:23))),'Shift_JIS')-48);
    while length(temp) < 12
        temp = [false temp];
    end
    temp(:,12) = [];
    generatorMatrix(m,81:91) = temp;
end

y = [eye(91,91) generatorMatrix'];

% いちいちこれを計算するのは非効率なので、一度計算したらどっかに保存しておけばよい
end
```

誤り訂正符号を付与した 174 ビットの信号列を 3 ビットごとに区切り、それぞれを 8FSK のシンボルに変換します。この際にグレーコードへの変換も行います。グレーコードへの変換はリスト 9 に示す関数を用いて行っています。

リスト 9 8FSK への変換、グレイコードへの変換を行う関数

FT82basebandSymbol()

```
function y = baseband2FT8symbol(baseband)
    GrayMapTable = [0 1 3 2 5 6 4 7];
    for l=0:(fix(length(baseband)/3)-1)
        y(1,l+1) = GrayMapTable(1,4.*baseband(1,3.*l+1) +...
            2.*baseband(1,3.*l+2) + baseband(1,3.*l+3) + 1);
    end
end
```

シンボル列に変換する際には、メッセージは最初、中間、最後に 7×7 のコスタス配列の挿入も行います。MATLAB での実装では、行列の間に行列を挿入するだけなので非常にシンプルに書けます。

コスタス配列を含めた全 79 シンボルについて、BT 積が 2 となるガウスフィルタ(半値半幅 12.5 Hz)で帯域制限を行った後に変調指数が 1 となるように 8FSK で変調します。

最後に、信号の開始と終了にテーパーを付けて、信号帯域幅の狭窄化を図ります。このときの包絡線形状については参考文献[5]を参照してください。最後に audiowrite 関数で wav ファイルとして保存します。

参考文献

[1] M.P.Hasselbeck(WB2FKO), “Synchronization in FT8,”

http://laarc.weebly.com/uploads/7/3/2/9/73292865/ft8syncv8.pdf,

Feb. 2019, 2021 年 4 月 10 日閲覧.

[2] WSJT Developers,WSJT-X Source code,

https://physics.princeton.edu/pulsar/k1jt/wsjtx-2.4.0-rc4.tgz.

[3] J. P. Costas, “A study of a class of detection waveforms having

nearly ideal range-Doppler ambiguity properties,” Proc. IEEE, vol.72,

no.8, pp 996-1009, Aug. 1984.

[4] @tommyecguitar, “コスタス配列はどんなものか”,

https://qiita.com/tommyecguitar/items/79de16cd3474bb699235, 2021

年 4 月 10 日閲覧.

[5] Stave Franke, K9AN, Bill Somerville, G4WJS and Joe Taylor, K1JT,

“The FT4 and FT8 communication,” QEX July/August, May. 2020.

[6] JP7VTF, “WSJT 系新モード「FST4/FST4W」のプロトコルを読み解く”,

東北大学アマチュア無線部部誌 Vol.3 新年号, pp.18-47, 2021.

編集後記

部員 K です。今回は製作してみたという記事を書きました。アンテナの自作や電子工作はまだまだ初心者でして、簡単なものからチャレンジしており今回は八木・宇田アンテナと、メモリーキーヤーを制作しました。メモリーキーヤーについては動作のテストしかしておらず、実際に運用で使っていないため、また実際に使ってみたいと思います。また、新たな製作にも挑戦したいと思います。

リーです。初めて時事ネタに手を出してみたのですが、詳しい資料が少ない上に、下手なことは書きにくいときたもので非常に面倒臭かったです。もう２度と時事ネタには手を出さないかも知れません w。それはそうと、今回の記事はどうだったでしょうか。私としては書きたかった通信方式の部分と 5G の発展の経緯の話を掛けたので大満足です。これからもどうにかこうにか部誌を更新していこうと思うので、よかったら今後も読んでみてください!

JP7VTF です。投稿がおそくなってすいませんでした。また、わかりにくい文章ですいません。詳しくは参考文献[1]をご覧ください。

富沢いずみです。仙台でも桜が満開になったかと思えばもう散るころのようで時の速さを感じます。時の速さといえばもう三度目の新歓の時期ですし、この部誌も四号です。

毎度のことながら編集長をまたやらせて頂きましたが部誌の内容や記載についてはこちらとしてもしっかりとした認識の上、編集・発行して行きたいと思います。今後ともよろしくお願いします。

とうほくだいがく
あまちゅあむせんぶ

発行者：東北大学アマチュア無線部

URL：http://www.ja7yaa.org.tohoku.ac.jp/index.php

Twitter：@JA7YAA