ソフトウェアGNSS受信機の開発と応用例

東京海洋大学
久保

配布資料

- source_vs2010_32bit
- ifdata_L1_40.96M
- libfftw_for_x64 （x64で動作させる場合に必要なライブラリ）
- presentation_file

全てダウンロードお願いします。Ifdataは1.8GBです。ダウンロード後、可能であればソースを展開して頂き、main文始めのほうのファイル読み込み部分だけ変更し、リビルド及びそのまま実行ができるかご確認願います

G-SAPSEプログラム紹介
（東大、慶応大、海洋大）

セミナーの概要

- 実際のGPS/QZS信号（L1-C/A）を市販フロントエンドよりUSB経由でPCで取得し、そのIF（中間周波数）データをPCで処理し位置計算を実施
- 大部分実際のプログラムの説明にあてており、信号捕捉や信号追尾等を実感して頂くことが目的です
- 上記の信号処理を経て、擬似距離計算、衛星位置計算そして単独測位計算まで実習で行います

本スケジュール

日時	内容
（10：30-12:00）	ソフト受信機概要（GPS/QZS信号処理含む）とAD変換後のデジタルIFデータの取得方法について
（13:00-15:00） 15分休憩	実際にプログラムを利用しながら信号捕捉、信号追尾を説明
（15:15-17:15） 15分質疑	信号追尾後の擬似距離計算、測位計算部分を説明

説明中も適宜質問頂ければありがたいです

ソフト受信機概要

ソフトGNSSの歴史

- コンセプト自体は10年以上前に発表
- GNSSチップメーカーでは、FPGA等で試験されてきた
- 一般の大学研究者に使えるようになってきたのはここ数年。テキストもいくつか出版
- マルチGNSSマルチ周波数の実用的なソフト受信機はドイツのIfen社のみか
- ここ数年、ソフトウエアGNSSを利用した大学研究者の発表論文が増加（ION-GNSS等でも複数のセッション）

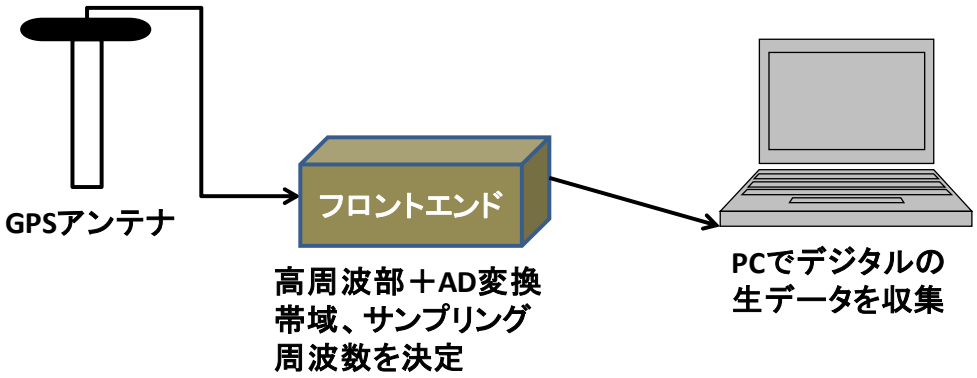
ソフト受信機関連の教科書

- Fundamentals of global positioning system receivers (James bao-yen tsui)
- A software-defined gps and galileo receiver (Kai Borre 他)　本ソースはこれがベース
- GNSS applications and methods (Scott Gleason)
- Navigation Signal Processing for GNSS software receivers (Thomas Pany)
- Digital Satellite Navigation and Geophysics (Ivan, Tsujii)
- Real-time GNSS Software Receiver (Marcel)
- Understanding GPS (Kaplan)→利用者多い
- GPSハンドブックの5章　受信機（ソフトウエア受信機は5.4節）
- RFワールド　2011年2月号の記事　ソフトウエアGPS受信機によるGPS受信機の試み（海老沼拓史）

本スライドで使用する図

- 主に測位航法学会で翻訳した「精説GPS」を利用しています
- それ以外では、ICD、IS-QZS、GNSSのテキスト、過去の精説GPS説明資料、Wikipedia、InsideGNSS等を参考にしています
- 英語と日本語が入り混じります

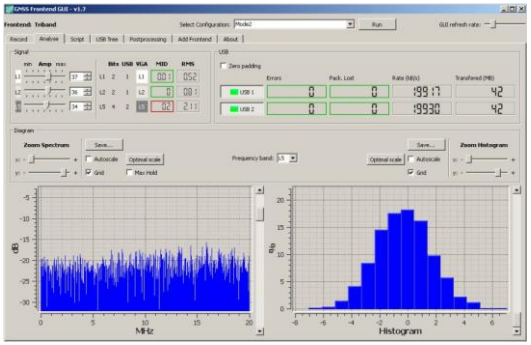
ソフトGNSSのハード部

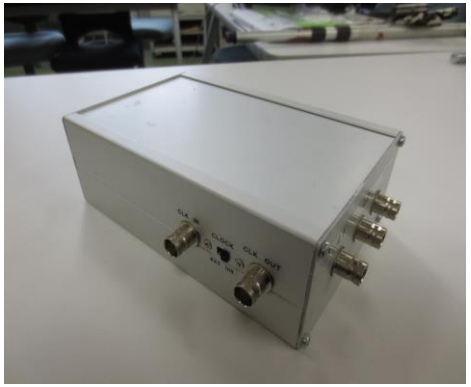


- USB径由で収集するものが多いが、データ転送スピードに限界もある。最近はUSB3.0も利用されている
- 例：40MHz、2bitの場合、1分間で約600MB
- フロントエンドを購入する必要がある。最近はIFデータ収集のみの機能の製品もあり安価になっている
- PC側は、それなりのCPU、メモリが必要
- 用途に応じたフロントエンドが製品化されており、2周波や3周波のものも存在。帯域も2MHzから20MHzまで幅広い

フロントエンドの紹介

- NSL社（マルチGNSSに対応、2波まで）
- IP-solutions社（GPS/GLOのL1）
- Fraunhofer社（L1+L2+L5）



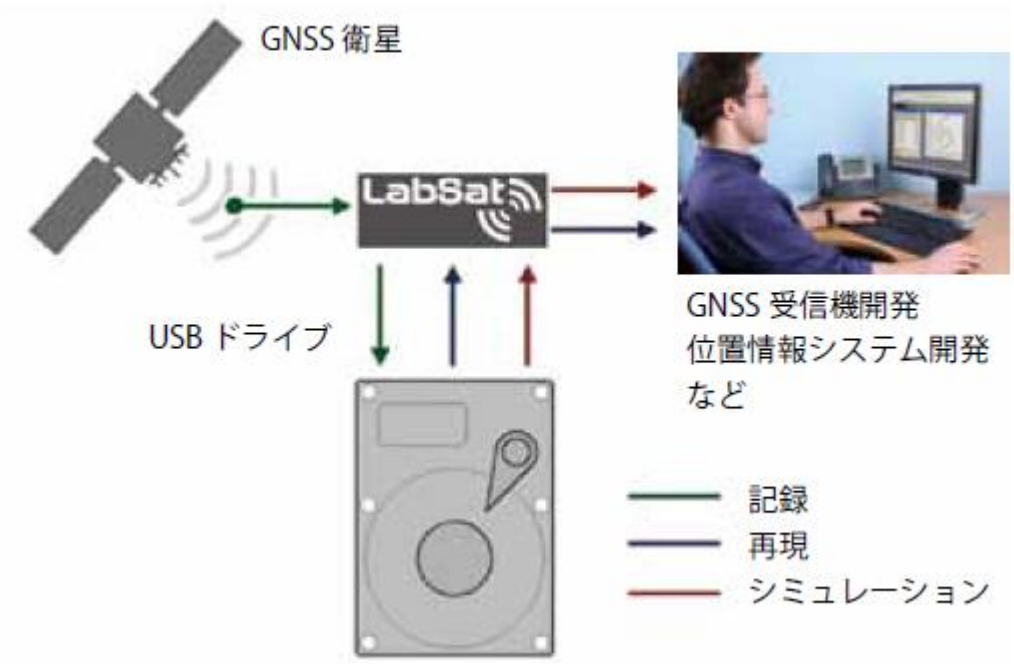


GPS L1/L2/L2C/L5
Galileo L1/E5a/E5b/E6/AltBOC
GLONASS G1/G2/G3
Compass B1/B2/B3

RFのチップセットは
MAX2769やMAX2112が多い

関連商品

システムフロー



LabSat 2 GNSS Simulator

SDR Advantage

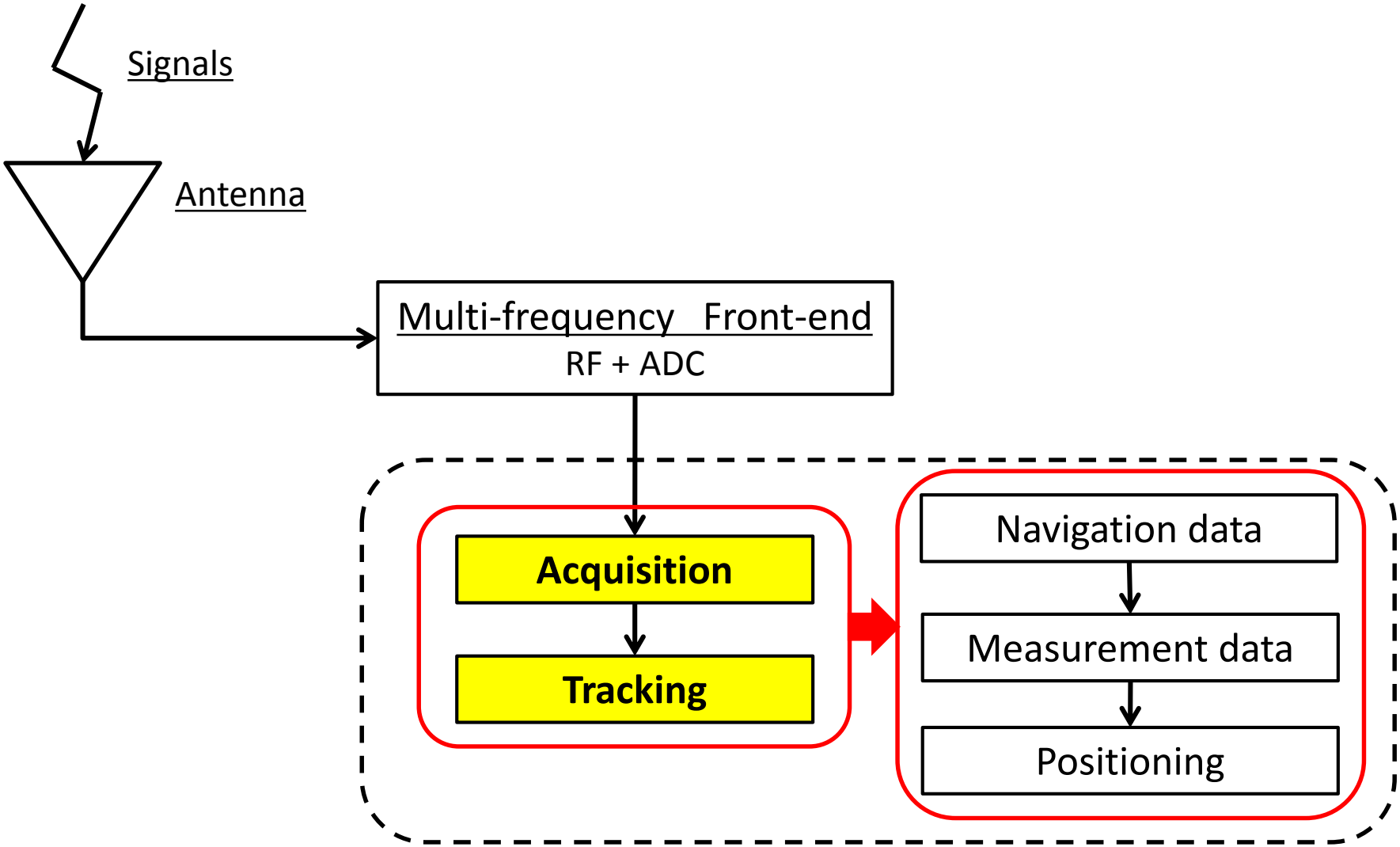
- **Educational tool**
- **Exploration of particular signal characteristics**
- **Signal processing algorithms that are not suitable for analogue implementations**
- **Very good for prototyping**

  -Does not need to wait for new hardware

  -Demonstrate acquisition of new GNSS signals

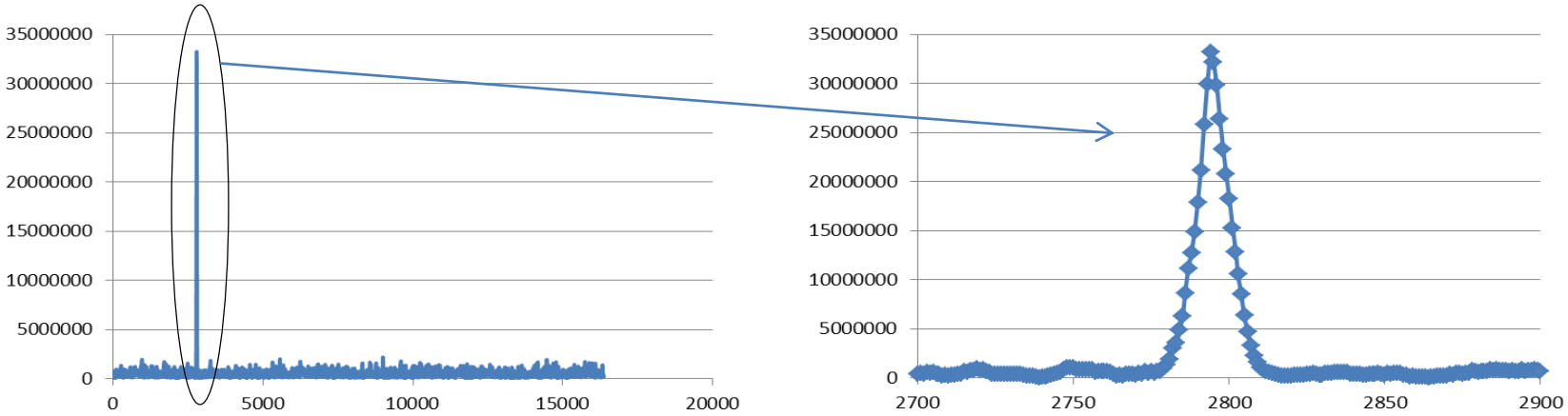
  -Students have converted the GPS SDR to new GNSS SDR within a year

Brief Structure of SDR



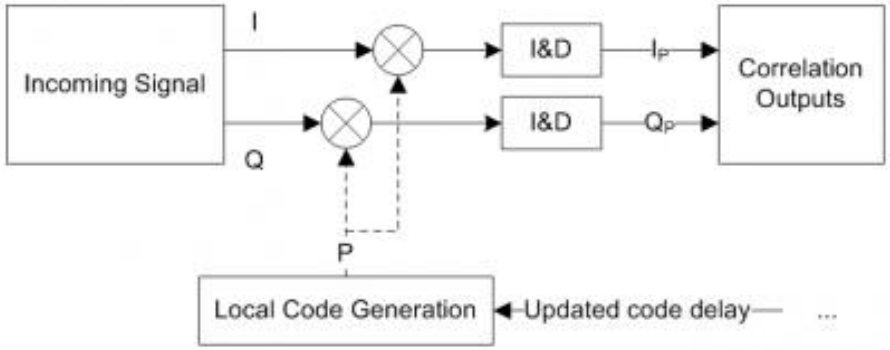
Acquisition (FFT based)

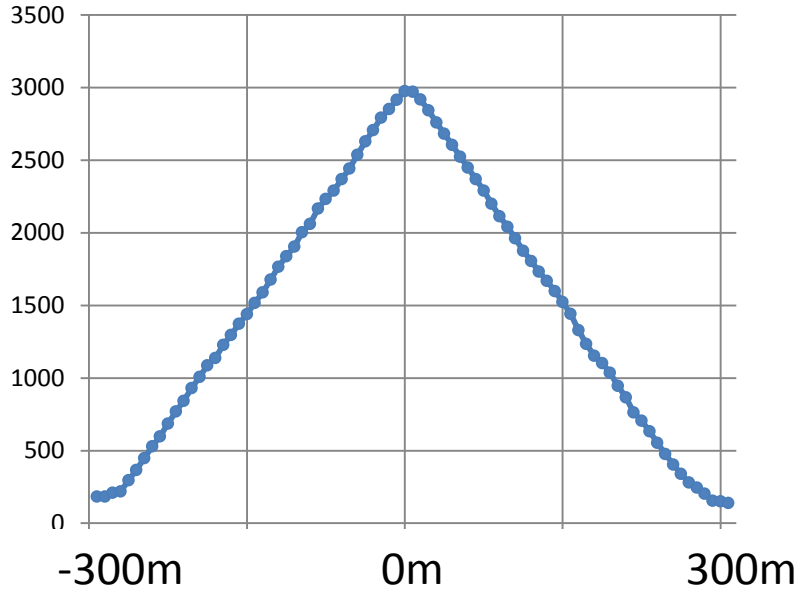
- Acquisition is to acquire the approximate code-phase and Doppler frequency of GNSS signals. Tracking is difficult without acquisition information.

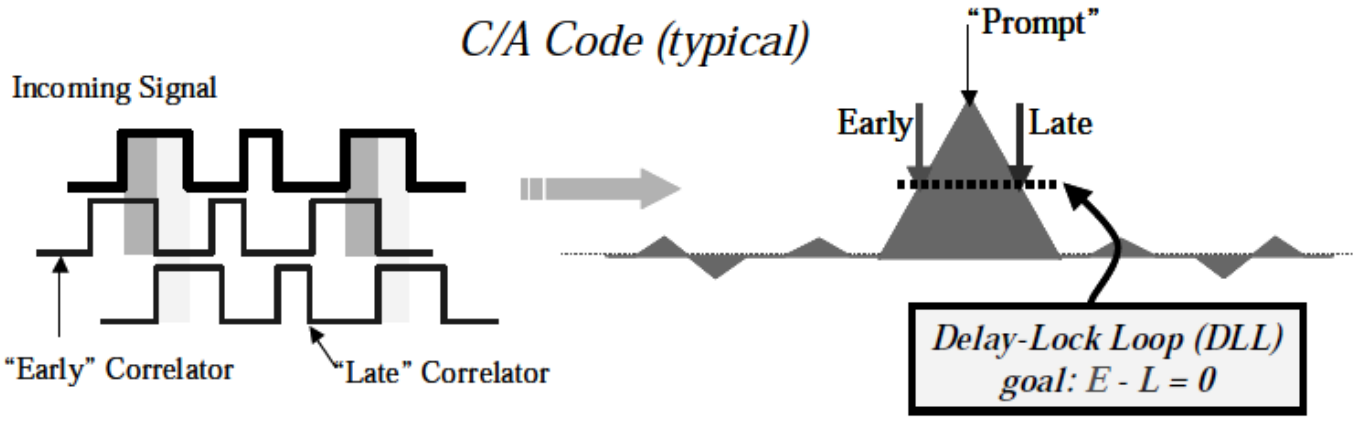


One shot of QZS

Correlation

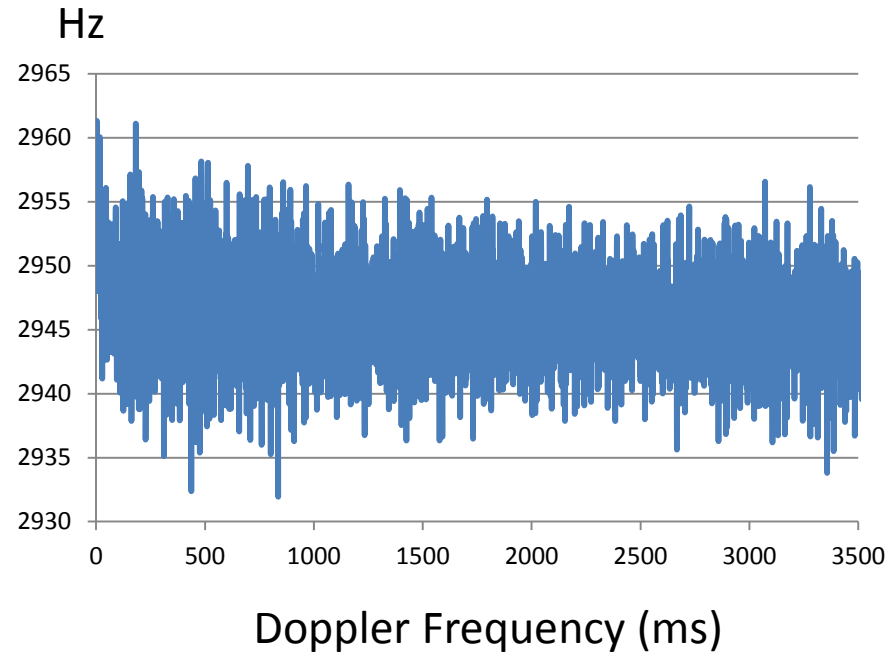


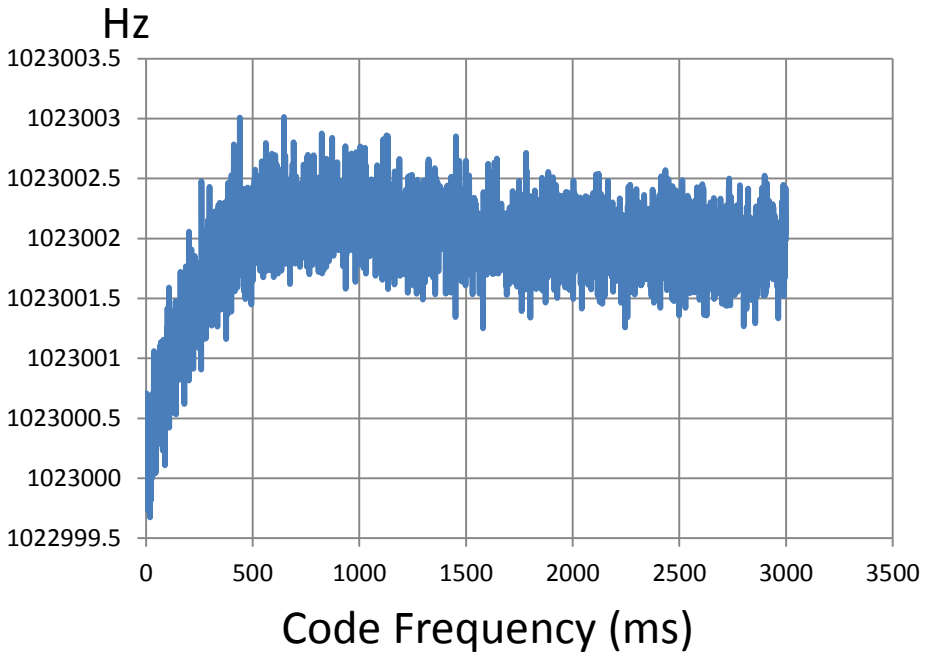




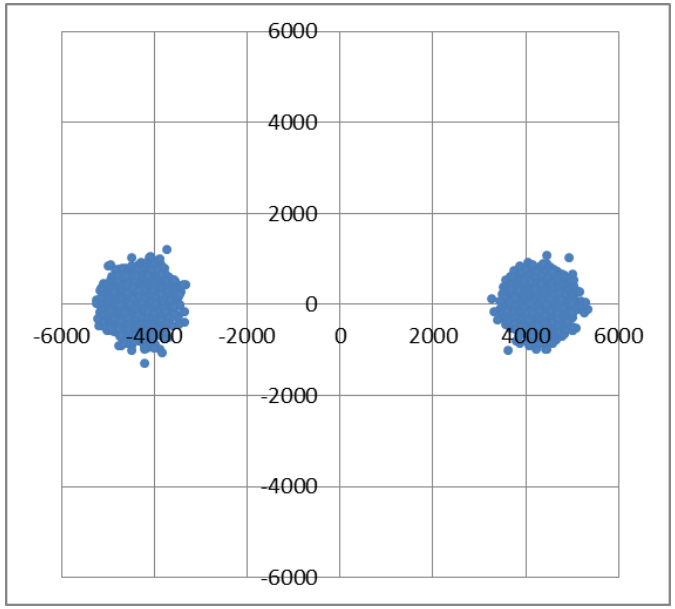
Tracking

- Tracking is to continuously track the code-phase and Doppler frequency of GNSS signals. Loop filter is used in the tracking loop.

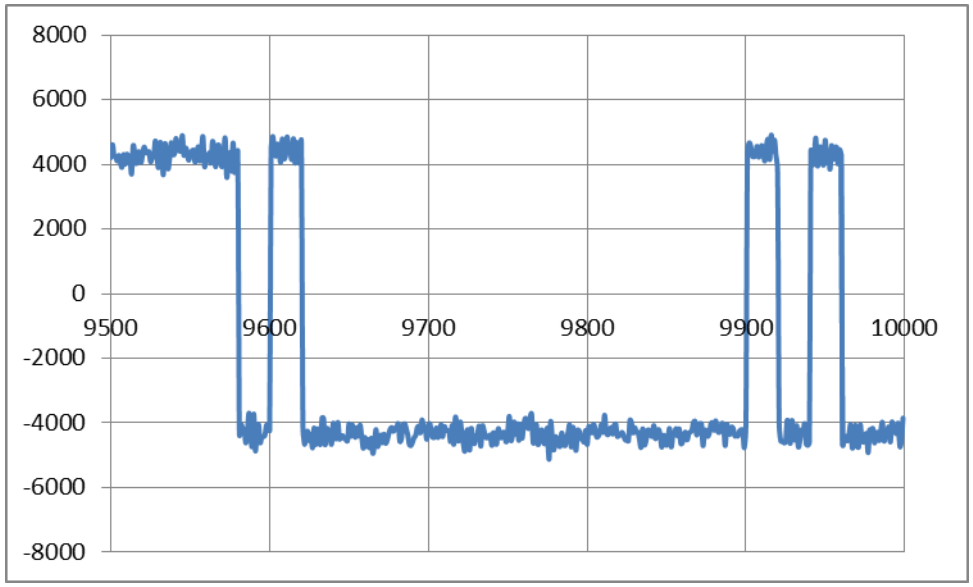




I and Q, Navigation decode (outputs of tracking)



I and Q

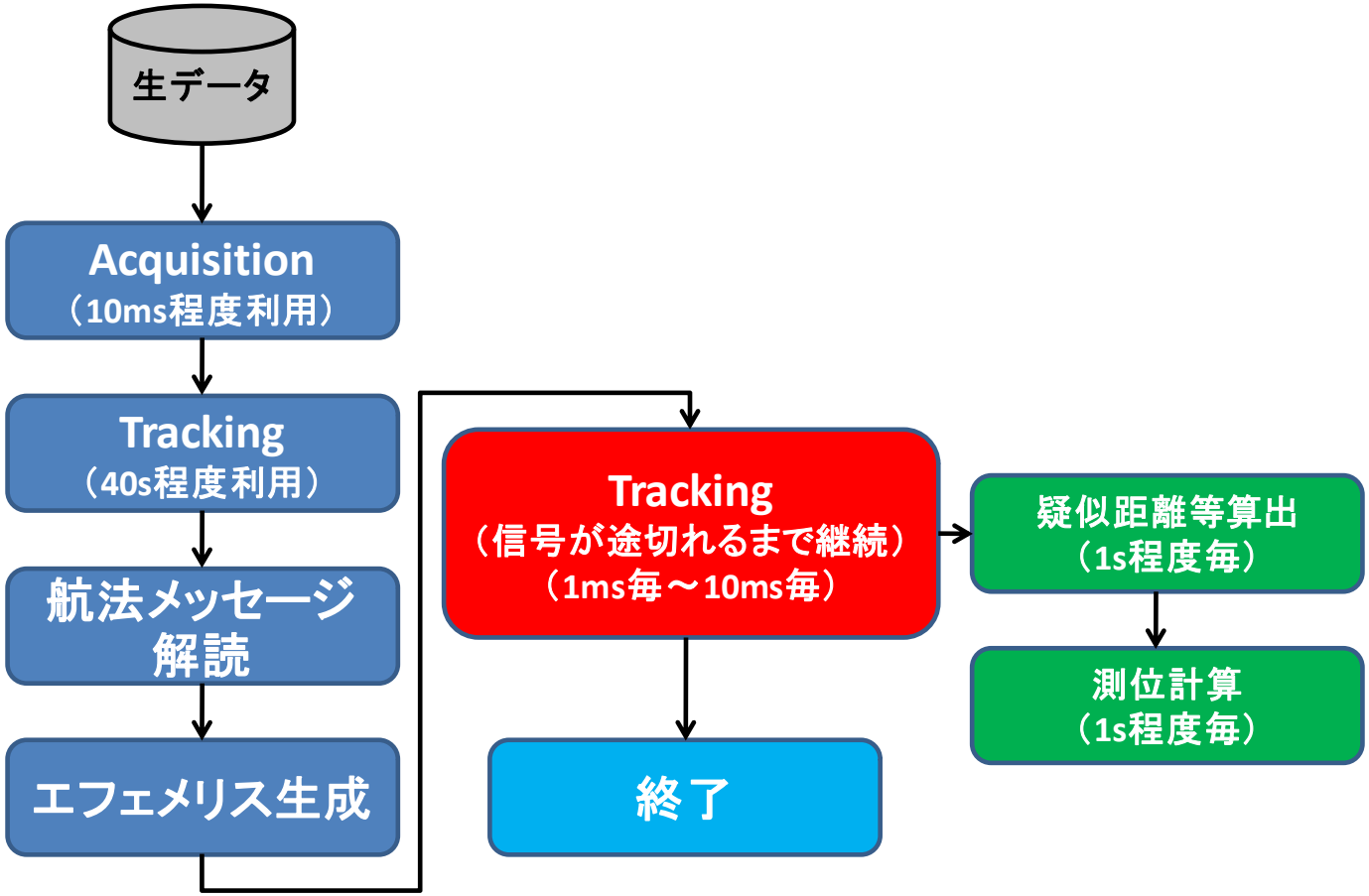


In phase correlation value from 9500ms to 10000ms

Applications of SDR

- Easy to customize source code
- Easy to implement for new GNSS era
- New signal processing development
- Specific receiver development (scintillation monitor etc.)
- Integration with other sensors
- Suitable as an education tool of communication engineering

ソフトGNSSのソフト部

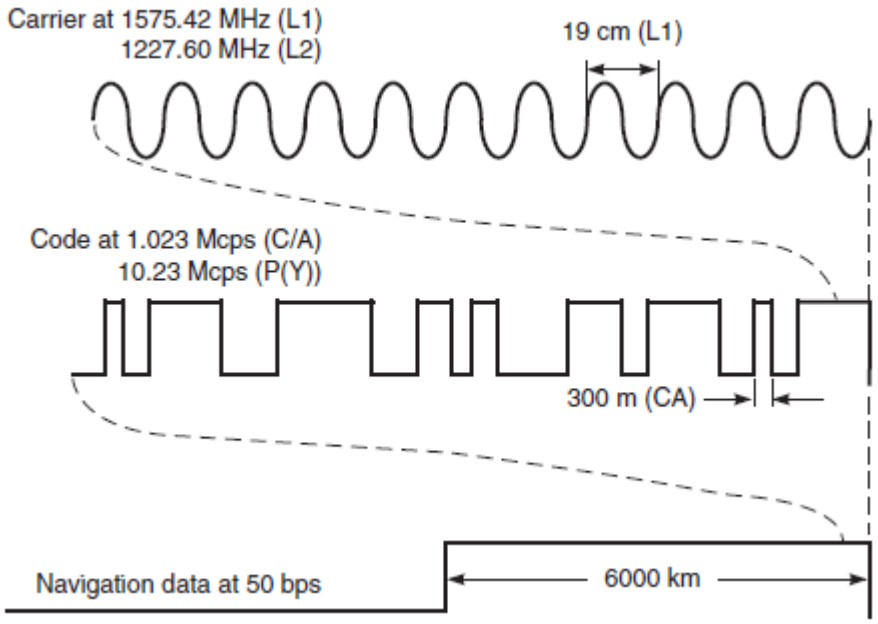


ソフト受信機信号処理部
（GPS/QZS　L1-C/A）

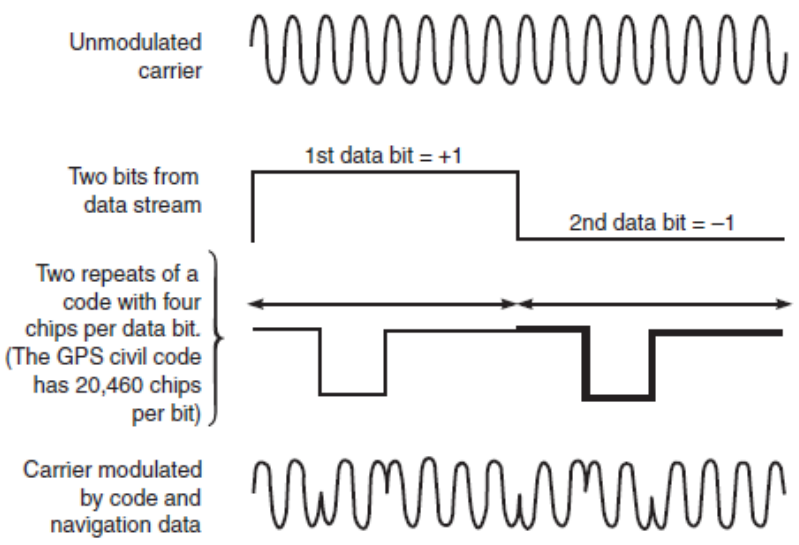
GPSの信号

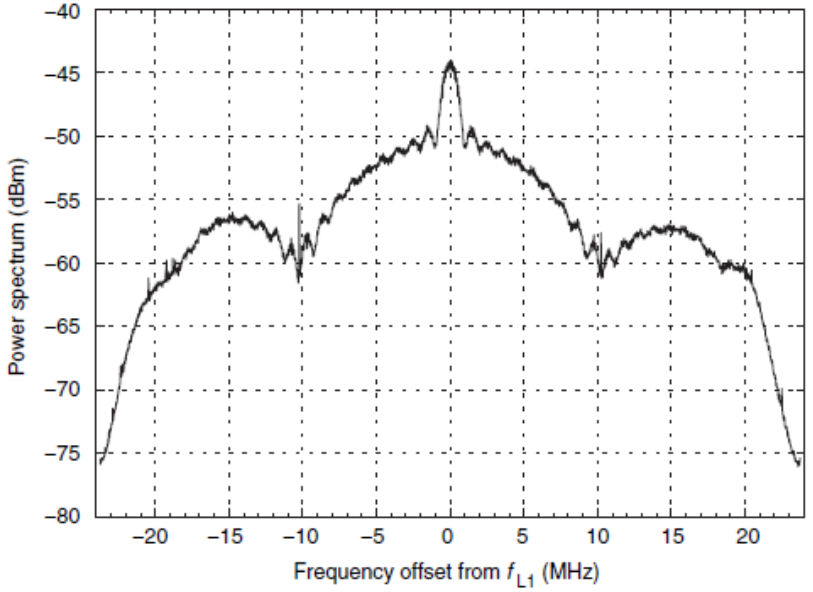
$$S_{L1}(t) = \sqrt{2P_{C1}}D(t)x(t)\cos(2\pi f_{L1}t + \theta_{L1}) + \sqrt{2P_{Y1}}D(t)y(t)\sin(2\pi f_{L1}t + \theta_{L1})$$

$$S_{L2}(t) = \sqrt{2P_{Y2}}D(t)y(t)\sin(2\pi f_{L2}t + \theta_{L2})$$



GPSの信号





2003 年 2 月に計測された PRN31 から送信された C/A コードと P(Y) コードを示す電力スペクトル．スペクトルはスタンフォード大学の直径 46 m のパラボラアンテナを使って測定された．水平軸の目盛りは 5 MHz を示し，最初の P(Y) コードのヌルは ±10 MHz に見られる．大部分の C/A コードのエネルギーは ±1 MHz に集中している．

Spectrum of L1-SAIF signal

RES BW 100 kHz
ATTEN 10 dB
VBW 0.3 kHz
10 dB/
SWP 3.19076 sec
CENTER 1.57542 GHz
SPAN 122.76 MHz
REF 0 dBm

And the spectrum looks very good.

12xfo=12x1.023
=12.276MHz

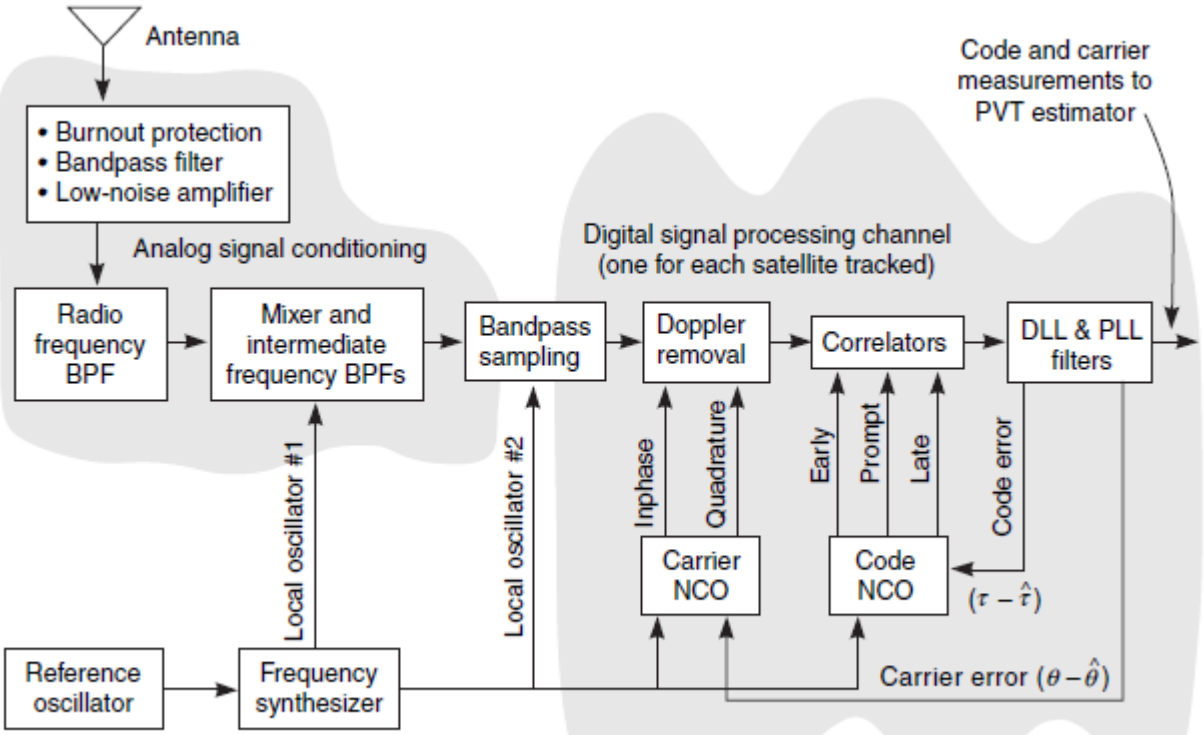
1.57542GHz

−10
−20
−30
−40
−50
−60
−70
−80
−61.38
0
61.38

アンテナ以降の信号の流れ

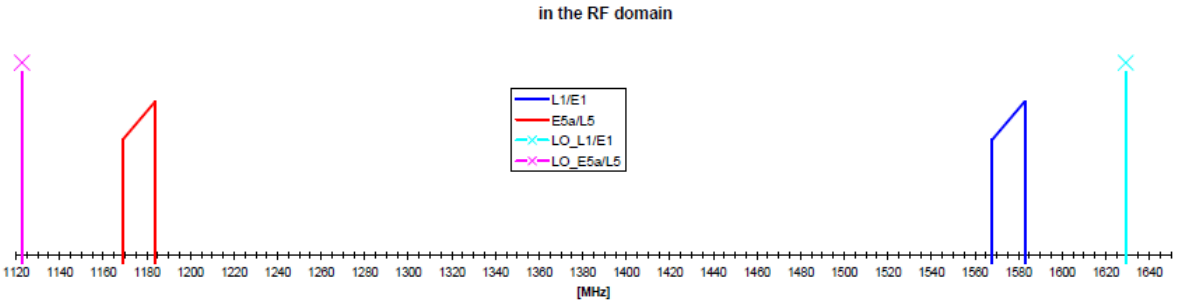
- フロントエンドでの信号の前処理
- 信号捕捉
- 信号追尾

信号処理部のブロック図

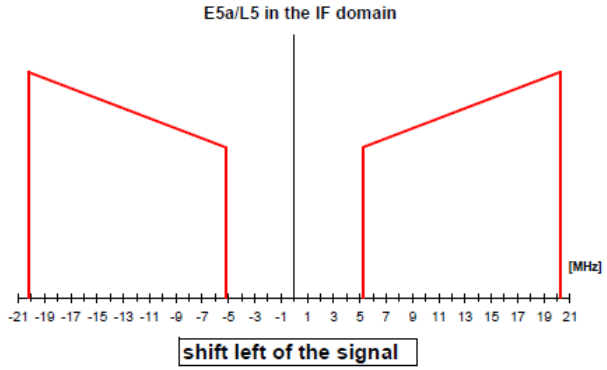


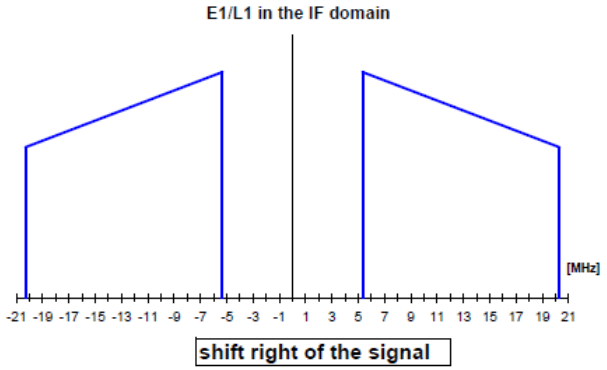
今回利用したFEの参考資料

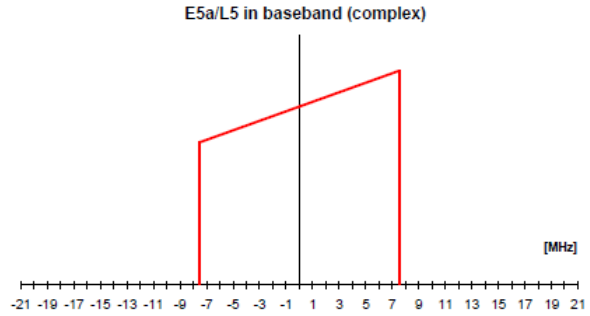
（前処理）

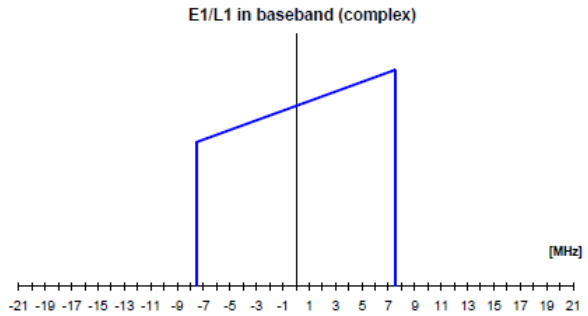


L1,L2は同様

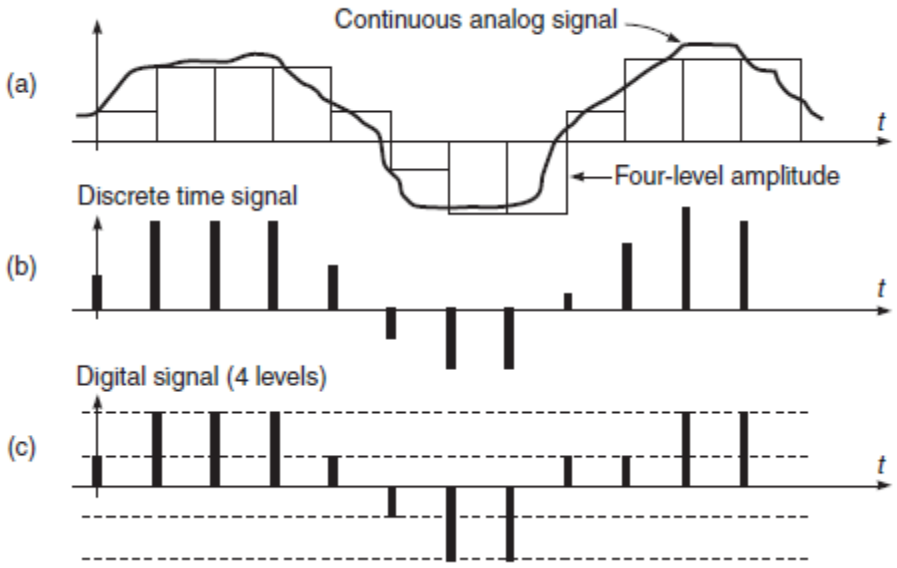








サンプリング



2ビット、3ビット、4ビットのものが多い

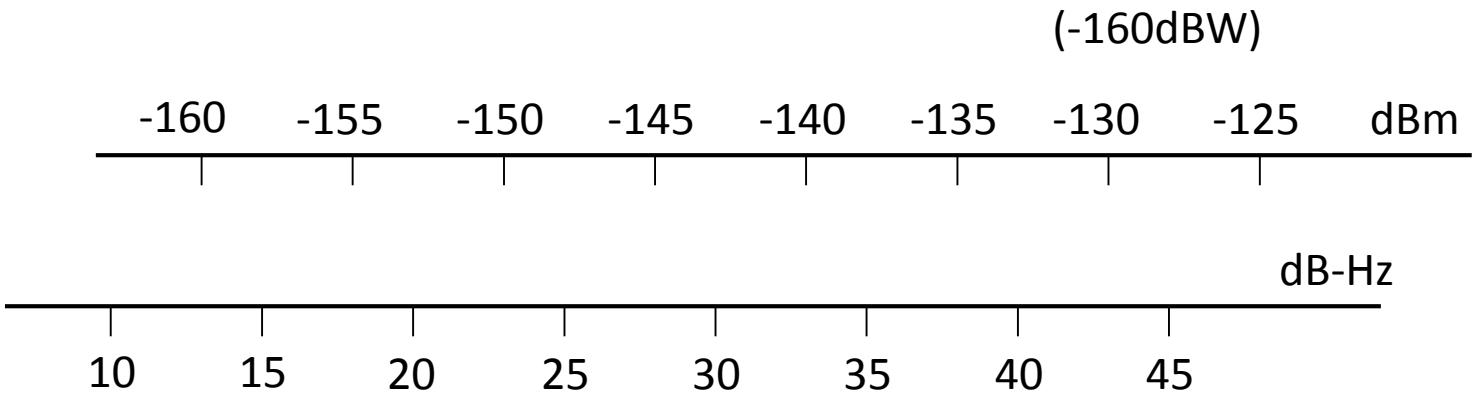
I,Qサンプリング

- 最近のFE（GNSS関連）はほとんどI,Qでサンプリングしたデジタル信号を出力
- FE側でI相と90度位相がずれたQ相のデジタル値を出力。I相のみのときは、例えば信号追尾部分でQ相の値をcosで生成しなければならなかったが、Q値があればそのまま利用できる
- 信号処理の観点よりIサンプリング（20MHz）のみの信号とI,Qサンプリング（20MHz）では、その情報量が異なることは明白

Acquisition（信号捕捉）

- GPS信号はスペクトラム拡散符号で変調されており、かつ地上で受信可能な信号レベルは-125dBmから-130dBmと非常に低い
- 認識できる信号に戻すために、コード位相やドップラーを含んだ搬送波の正確な遅延量を知る必要がある。そこで、レプリカ信号と入力信号の相関をとる
- 相関器はコード位相の遅延と周波数のレプリカが入力信号を一致したときに最も高い値を取得する。この過程はすべての衛星に対して同様に行われる

dBmとdB-Hz



いくつかの場所での典型的な信号レベル

環境	信号レベルdBm	信号レベルdBHz
屋外	-123 to -130	51-44
1-2階建の建物	-130 to -150	44-24
窓のオフィス	-135 to -160	39-14
駐車場	-135 to -150	39-24

GPSの送信信号と受信信号

$$s(t) = \sqrt{2P_{\text{tmt}}}D(t)x(t)\cos(2\pi f_L t + \theta_{\text{tmt}})$$

$$r(t) = \sqrt{2P_{\text{rcv}}}D(t-\tau)x(t-\tau)\cos\left(2\pi(f_L + f_D)t + \theta_{\text{rcv}}\right) + n(t)$$

ここで，$P_{\text{tmt}} >> P_{\text{rcv}}$ である．下付き文字の“tmt”と“rcv”はそれぞれ“送信時”と“受信時”を表している．残りの記号はすでにお馴染みであろう．しかし，雑音 $n(t)$ だけが障害となるわけではない．他の GPS 衛星からの信号も同じ周波数に同時に重なっている．

このような状況においても，受信機はこの微弱な信号を捕らえ，私たちが必要とする位置・速度・時刻の情報を算出する．具体的には，受信機は位置と時刻を得るために τ を，速度を得るために f_D を推定し，さらに高精度な位置を得るために θ を推定する．

受信信号を眺める

$$s(t) = \sqrt{C}D(t-\tau)x(t-\tau)\cos\left(2\pi(f_{IF}+f_D)t+\delta\theta\right)+n(t)$$

信号はディジタル処理に適した形になっているが，まだまだたくさんの処理を行わなければならない．まず，到着時刻 τ を推定しなければならない．その理由は，τ には基本距離（basic range）およびユーザ位置およびクロックオフセットを計算に必要な時刻情報が含まれているためである．また，ドップラシフト（f_D）も推定しなければならない．これは，ユーザ速度とクロック周波数の計算に使用する擬似距離変化率が含まれているからである．究極の測位精度を望む場合には，搬送波位相オフセット（$\delta\theta$）の推定および追尾が必要となる．

鍵となる三つの要素 $(\tau, f_D, \delta\theta)$ は，二つの段階に分けて推定される．最初の段階は，(τ, f_D) の概算値を広域探索（global search）する動作であり，信号捕捉（signal acquisition）として良く知られている．第二段階は，(τ, f_D) の正確な推定を行う狭域探索（local search）であり，これには搬送波位相の推定が含まれることもある．狭域探索が搬送波位相を推定する場合，コヒーレント信号追尾と呼ばれる．搬送波位相を無視する検索は，非コヒーレント信号追尾と呼ばれる．本節では，信号捕捉について述べる．信号追尾については，第 12 章で解説する．

Acquisitionの種類

信号捕捉は時間を消費する．受信機は，信号捕捉の時間を極力少なくするため，利用可能な手がかりを巧みに利用する．受信機の電源が始めて入れられ，事前情報がない状態での信号捕捉は**コールドスタート**（cold start）となる．何らかの事前情報がある状態では，**ウォームスタート**（warm start）が可能となる．受信機が信号捕捉を終え，信号を追尾している場合，たったひとつの衛星が一時的に遮られた状態から回復することを，信号の**再捕捉**（reacquisition）と呼ぶ．

コールドスタートは事前情報がない状態から開始され，捕捉まで数分を要する．現在の位置および時刻の推定値がない場合，受信機はどの衛星が上空にいるかを把握できない．この場合，受信機は衛星をランダムに探し始める．ウォームスタートでは，受信機位置を 200~300 マイルの精度で，時刻を 10 分以内の精度で把握していれば，アルマナックを用いて処理を開始できる．これにより，受信機は探索すべき衛星がわかり，一般に仰角の高い衛星から探索を始める．仰角の高い衛星が選ばれるのは，信号が遮られる可能性が低く，C/N_0 も一般に高いためである．

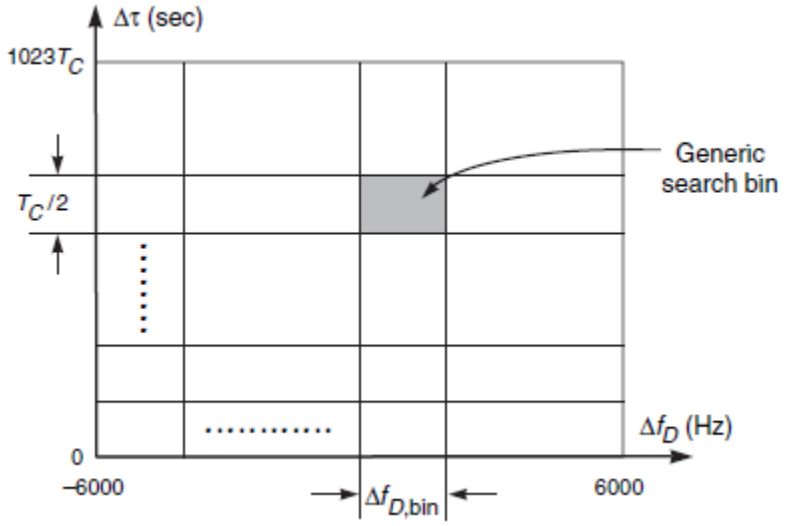
コールドスタートまたはウォームスタートにより信号を捕捉すると，信号追尾が開始される．しかし，すべての航法データを獲得するまで時間を要する．第 4 章で述べたように，航法データの 1 フレームの送信には 30 秒を要する．位置の算出にサブフレーム 1, 2, 3 が必要であり，18 秒間継続する．

再捕捉のポイント

たったひとつの衛星が一時的に遮られた状態から回復することを，信号の再捕捉と呼ぶ．その他の衛星は連続的に捕捉されており，受信機は位置およびクロックオフセットの非常によい推定値を既に得ているため，再捕捉には通常数秒しか要しない．さらに，失われた信号の遅延ロックループは，その衛星に対する適切なコード遅延 τ の推定値を保持している場合もある．受信機は，信号が失われた時点でのコード遅延とドップラ周波数の近傍を探索するだけでよい．

本節では，信号捕捉を補助する事前情報を活用する多くの手法について触れていない．また，シリアルサーチ（serial search）についてのみ説明し，パラレルサーチ（parallel search）やフーリエ変換による信号捕捉については割愛する．

探索領域



信号の探索領域は，ドップラ周波数（Δf_D）とコード位相（$\Delta\tau$）をカバーする．探索領域のセルの総数 M は，コード位相のビン数とドップラのビン数の積である．

もしサンプル周波数が16MHzの場合、1msの1コード長で16000のチップのサンプリングとなる。同様に500Hz間隔で10kHzのドップラサーチには41の組み合わせが必要。
合計、16000×32衛星×41
＝20992000シフトが必要！

実際には、縦のコード位相ビンはサンプリング周波数に
横の周波数ビンは積分時間に応じて決定する

ソフト受信機での信号捕捉

- ベースは全く同じであるが、処理時間を考えてパラレルサーチ（フーリエ変換利用）を行う
- その中でもサーキュラコリレーションの手法（今回のソースもこの手法）がとられる。この方法ではコード位相のサーチを並列処理できる。よってサーチが必要なのはドップラ周波数のみ→詳細は教科書
- C/Aコードレプリカの FFT等は前もって準備

CoherentとNon-Coherent

- Coherent積分は、1ms以上のデータを捕捉する場合のこと。1msデータはコード長も1msのためバンド幅は1kHz。10msの場合100Hzとなる
- Non-Coherent積分とは個々のデータが別々に生成されたものの集まり。Coherentと比較すると積分による効果がやや悪いが、航法データビットの位相の変化に対する影響が少ない

ドップラ周波数について

- 衛星とユーザ受信機との相対速度で考えると±5kHzで十分だが、受信機の時計誤差やジッタを考慮して±7kHz程度サーチする
- 搬送波に与えるドップラ周波数の影響はそのままオフセットとしてあらわれる
- コードに与えるドップラ周波数の影響は1/1540（1.023MHzと1575.42MHz）である。静止時最大の5kHzで影響は3.2Hzである
- OCXOやルビジウムを基準発信器とすると、受信機側の周波数オフセットはほぼ0になる→出力されるドップラ周波数はそのままオフセットを含まない値

ソースコードでの実際

- Acquisitionでは、10ms程度のデータを利用。屋上で仰角20～30度以上であれば1～2msで十分。10度くらいの場合2～4ms程度
- L1CやL2Cはコード長が長いため、探索する周波数の幅を狭くする必要がある
- 最初の捕捉がうまくいった後、例えばL1-C/A（1ms分）を10ms分にしてFFTをかけて（コード位相が判明しているため）細かいドップラ周波数を算出する。この手法は、20ms間ビット反転のないようなコードには利用できるが、ビット反転があると厳しい

信号追尾

本章では遅延ロックループ（DLL：Delay Lock Loop）と位相ロックループ（PLL：Phase Lock Loop）を紹介する．付録では周波数ロックループ（FLL：Frequency Lock Loop）を議論する．第 11 章で議論したように，信号捕捉はコード位相 τ とドップラ周波数 f_D の概略の推定値を与える．DLL は，このコード位相の初期推定値の精度を高め，その後の変化に追従する．このため，DLL は衛星が送信している PRN コードのレプリカ（模造）を発生し，受信コードとの同期を維持する．擬似距離全体を得るため，コード位相の推定値は 5.1 節で説明した他の項と一緒に用いられる．FLL はドップラ周波数の初期推定値の精度を高め，その後の変化に追従する．このため，FLL は正弦波の搬送波を発生し，受信搬送波の周波数との同調を維持する．FLL と同様，PLL は搬送波追尾ループであるが，搬送波位相 θ の推定値も与える．PLL は入力搬送波の位相に同期させるため，レプリカである正弦波の周波数と位相を調整する必要がある．コードと搬送波のこれら操作の全体を**信号追尾**と呼ぶ．

信号追尾器の概要

これらのことから，追尾器は自動車の自動速度制御器に似ている．速度制御においては自動車が制御対象であり，道路の状態が変化しても運転者が指定した速度となるように，自動車の速度を制御することが望まれる．このため，自動車の実際の速度を測定し，制御器にフィードバックする．制御器は，測定速度と指定速度の差である誤差信号を計算する．この誤差信号は制御器に供給され，制御器は誤差を小さくしようと動作する．自動車が指定速度より速い場合，制御器はエンジンへの燃料の供給を減らす．遅い場合には燃料の供給を増やす．自動速度制御器と GPS 受信機で用いられているループは驚くほどよく似ている．

DLL ではコード生成器が制御対象である．DLL は，レプリカ（模造）コードが受信コードに時間軸上で一致するようにコード生成器を制御する．レプリカコード生成器の速度を制御するクロックに対し，加速あるいは減速命令を出すことで，これを実現する．制御器はレプリカコードの時間軸上にて，急激なステップ変化を伴う命令は行わない．生成器を制御するクロック速度を単に加速したり，減速したりするだけである．レプリカコードが進んでいる場合，DLL はコード生成器に減速するよう命令する．レプリカコードが遅れている場合，DLL はコード生成器に加速するよう命令する．これらの命令により，レプリカコードは受信コードに次第に一致する．

PLL では，数値制御発振器（NCO：Numerically Controlled Oscillator）が制御対象である．NCO は，レプリカコードではなくレプリカ搬送波を生成する点を除き，コード生成器に似ている．PLL は，レプリカの周波数と位相を受信搬送波の周波数と位相に同期させようとする．このため，PLL は位相差を検出し，この差を減らすよう NCO に命令する．ここでも，NCO の出力を急激に変化させるような命令は行わない．命令は，同期を維持するよう NCO を単に加速したり減速したりするだけである．

遅延ロックループ(DLL:Delay Lock Loop)

1. 目的

   ユーザー位置計算のための擬似距離の測定

2. 動作

   衛星が送信しているPRNコードのレプリカ(模造)を発生し，受信コードとの同期を維持

位相ロックループ(PLL:Phase Lock Loop)

1. 目的

   航法データの復調

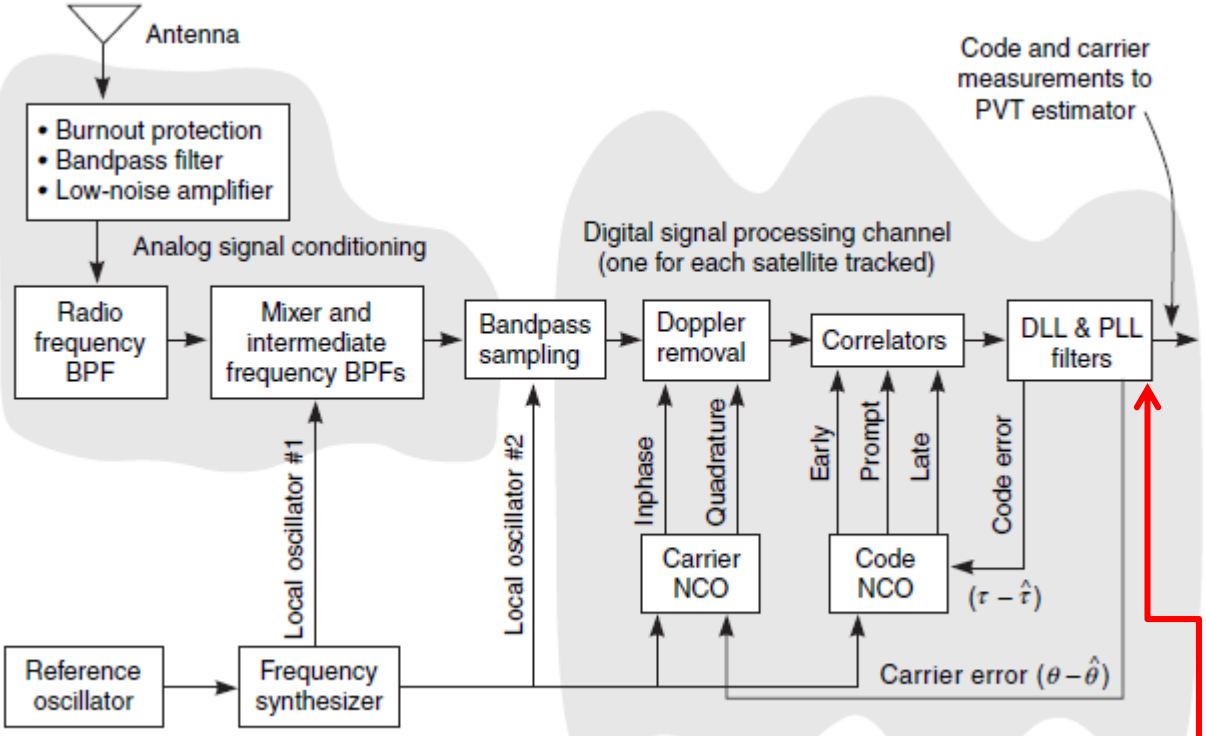
   ユーザー速度計算のための擬似距離変化率の測定

   RTK等のための搬送波位相算出

2. 動作

   入力搬送波の位相に同期させるため，レプリカである正弦波の周波数と位相を調整

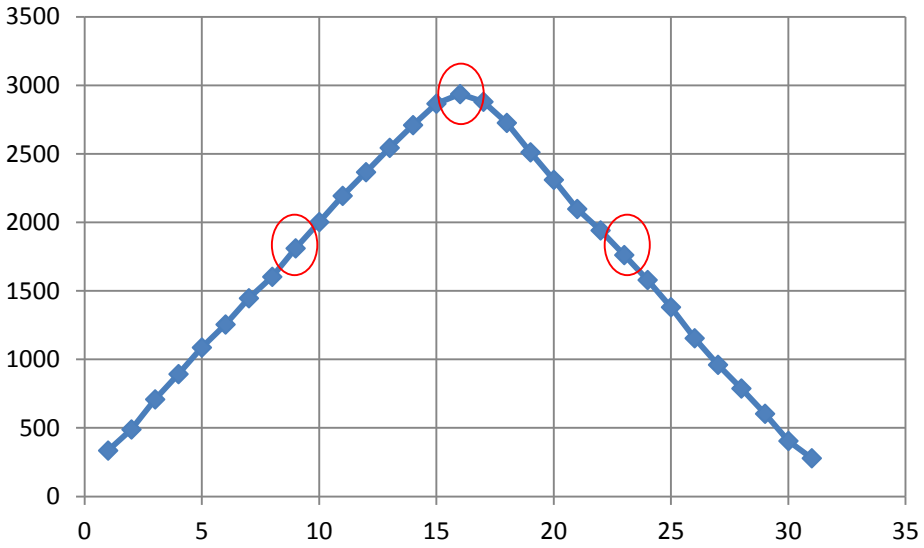
再度　信号処理のブロック図（DLLとPLLのケース）

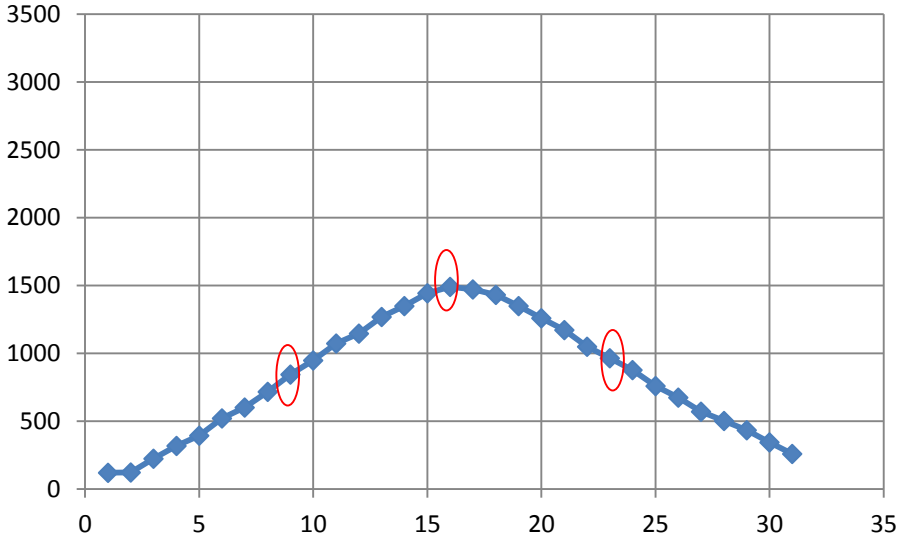


ここに弁別器（discriminator）とループフィルターがある

DLLの相関器による1ショット

横軸1目盛りが約18m（16MHzサンプリング） 縦軸は相関値



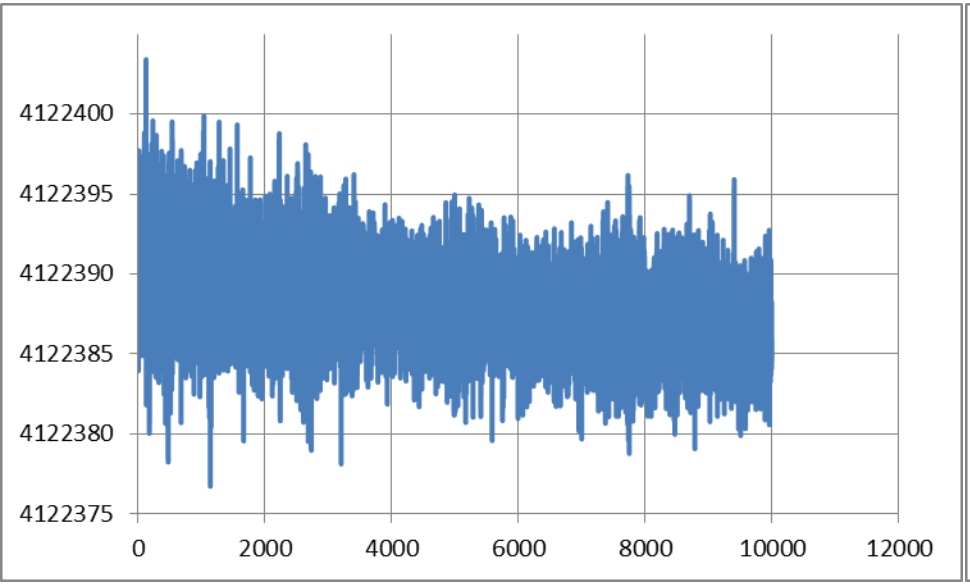


実習で解析するデータより　193番（準天頂）
左からEarly,Prompt,Late

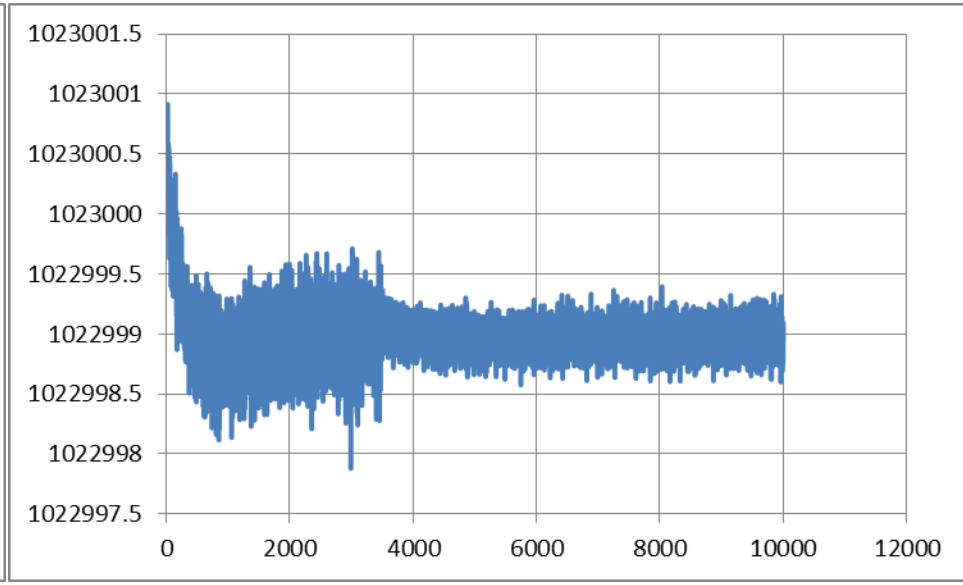
同じくPRN32

PLLによる周波数追尾とコード位相

横軸はms　縦軸は左がキャリアで右がコードの周波数



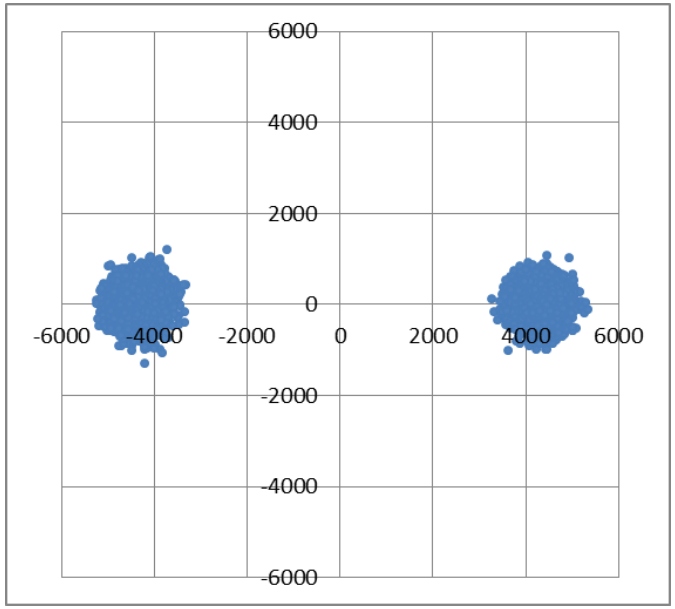
PLLによる周波数の追尾



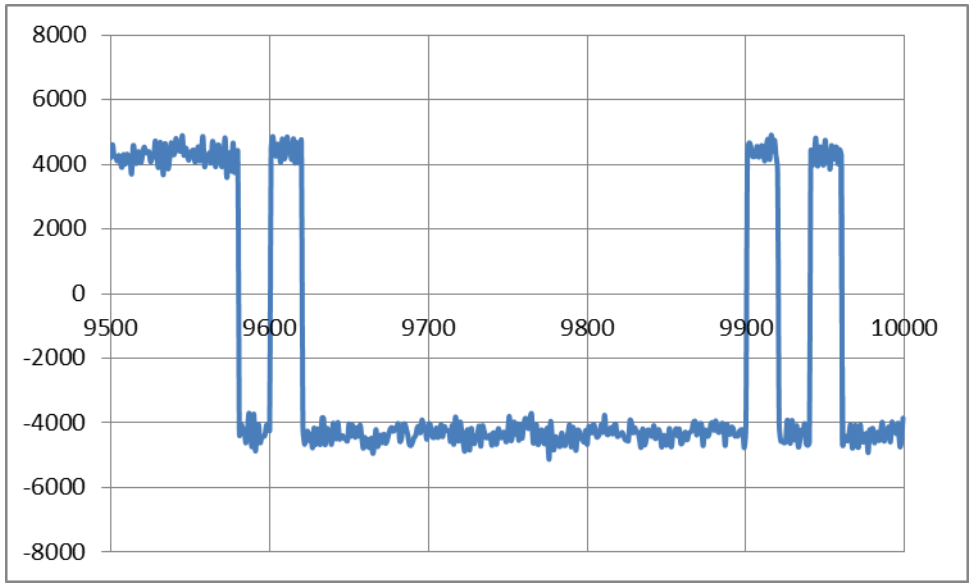
コード位相の追尾

フロントエンドのIF周波数が4.12MHz付近

I相とQ相のPromptの推移



横軸がI相　縦軸がQ相の値



9500msから10000msまでのI相の推移

利用しているDiscriminator

- DLL

codeError = (sqrt(I_E*I_E + Q_E*Q_E) - sqrt(I_L*I_L + Q_L*Q_L))/(sqrt(I_E*I_E + Q_E*Q_E) + sqrt(I_L*I_L + Q_L*Q_L));

- PLL

carrError = atan(Q_P / I_P) / (2.0 * pi);

このあたりの詳細は参照教科書のUnderstanding GPSが良いです。
Discriminatorの基本的な役割は相関値のEarlyとLate値を利用して常に信号ピークをとらえ続けることです。
3つ前のスライドの相関波形を見るとイメージしやすいです

利用しているループフィルタ（2次）

- DLL

codeNco = oldCodeNco[prn] + (tau2code/tau1code) * (codeError - oldCodeError[prn]) + codeError * (PDIcode/tau1code);

- PLL

carrNco = oldCarrNco[prn] + (tau2carr/tau1carr) * (carrError - oldCarrError[prn]) + carrError * (PDIcarr/tau1carr);

ループフィルタの基本的な役割は、雑音の多いDLLやPLLの
出力値を真値を損なわないようにスムーズにすることです。
出力値とは、コード位相をどれだけ進ませるかのCodeNco
等になります。NcoとはNumerical Control Oscillatorの意味です。
このフィルタで利用される係数設定は重要です

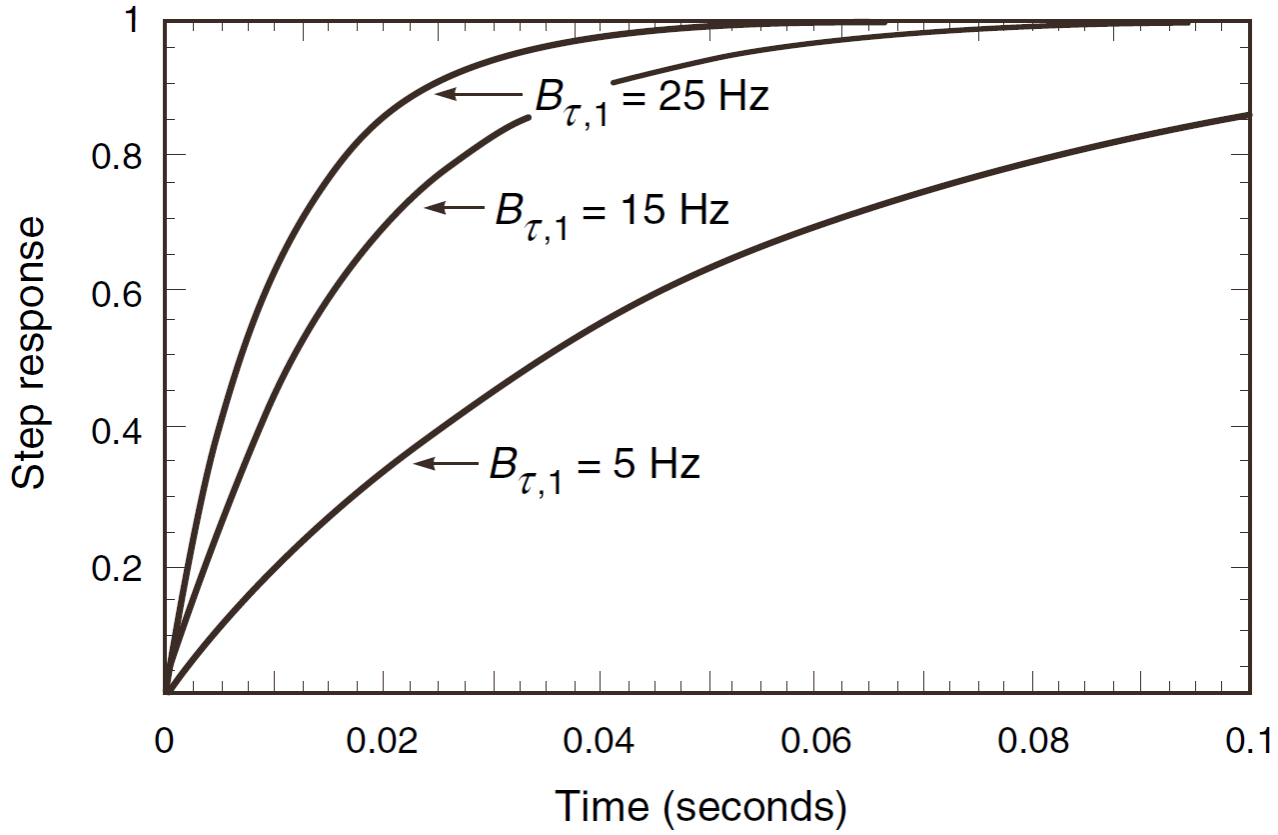


図12.7　非支援一次遅延ロックループのステップ応答

- $B_{\tau,1}$が大(小)→素早く(緩やかに)追随
- 帯域幅が広い(狭い) →雑音減衰が少(多)
- 雑音特性と動的追従特性はトレードオフの関係

レート支援遅延ロックループ

コード遅延推定値　＝

前回コード遅延推定値
＋搬送波追尾ループの変化率推定値
＋(DLL測定値ー前回コード遅延推定値)比例項

1)搬送波追尾ループの変化率推定値は十分な精度がある
2)DLL測定値は，搬送波追尾ループの変化率推定値の
　長時間バイアス除去が目的
3)レート支援により雑音帯域幅を十分に小さくし
　(0.005Hz程度)，雑音を減らす

白色雑音下における性能

雑音に対するDLL応答は

$$\mathrm{var}\{\Delta\hat{\tau}\} = \frac{B_{\tau,1} d T_C^2}{2C/N_0}\left(1 + \frac{2}{T_{CO}C/N_0}\right) \quad \mathrm{sec}^2$$

二乗損失の項

非コヒーレント弁別器は，搬送波位相と
航法データビットを取り除くことが出来るが，
その代り大きな二乗損失を被る

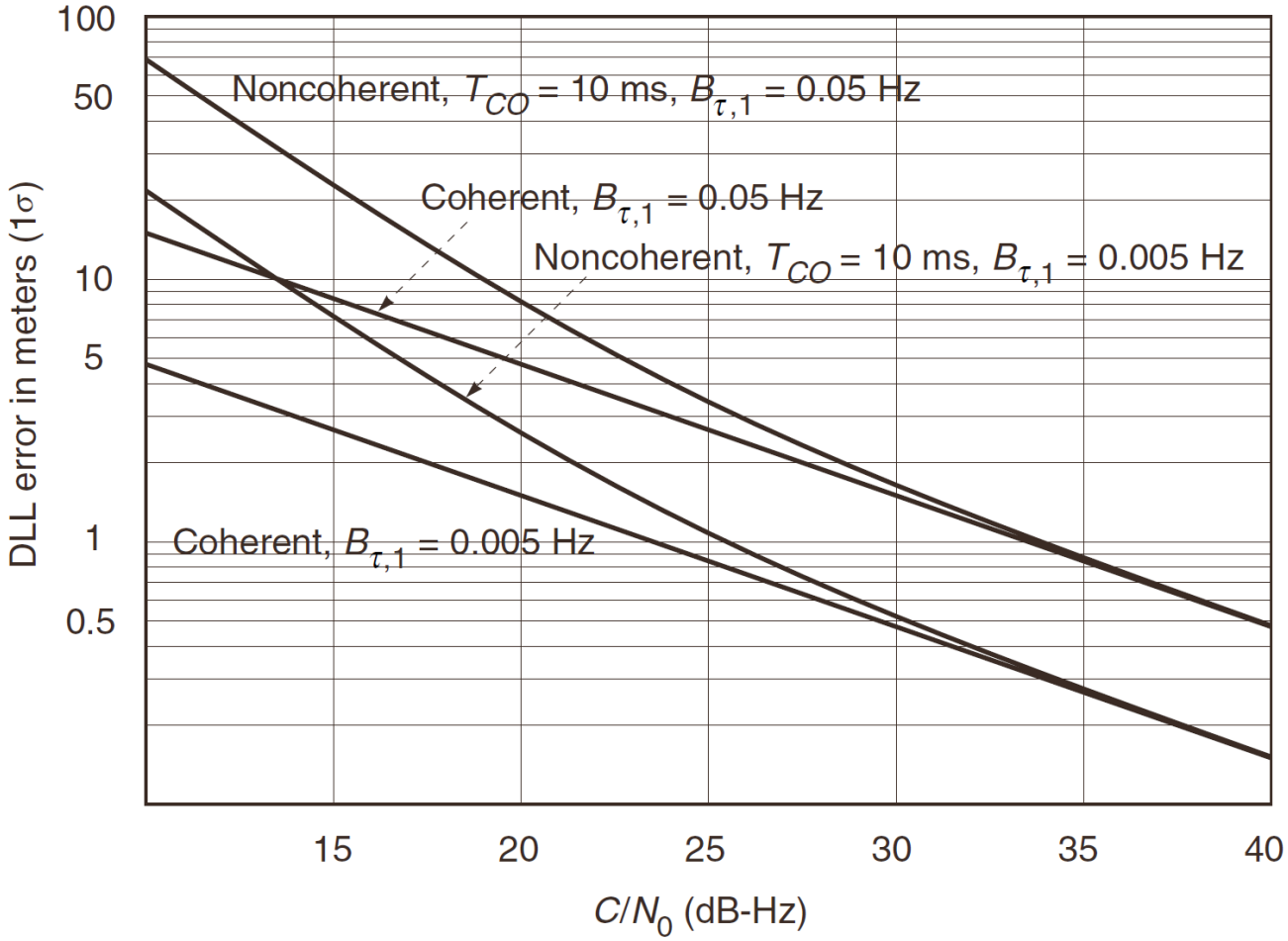
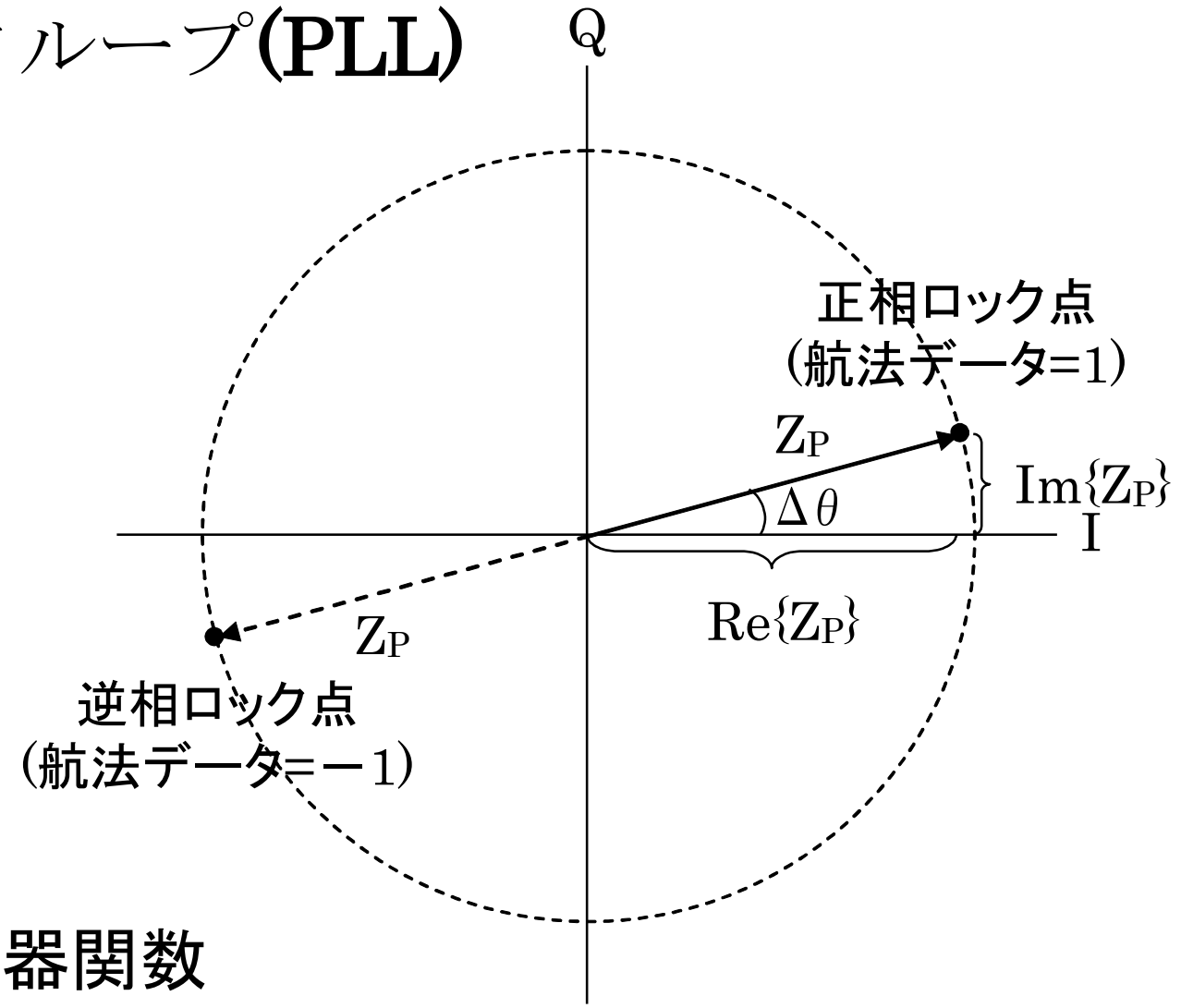


図12.8　白色雑音下におけるDLLの性能

信号対雑音比25dB-Hz以上 →二乗損失影響少
GPS信号は30dB-Hz以上で運用→
二乗損失は通常考慮する必要なし

位相ロックループ(PLL)



PLLの弁別器関数

コヒーレントPLL： $L_\theta = \mathrm{Im}\left\{\tilde{Z}_P\right\} D$

コスタスPLL: $L_\theta = \mathrm{Re}\{Z_P\}\,\mathrm{Im}\{Z_P\}$

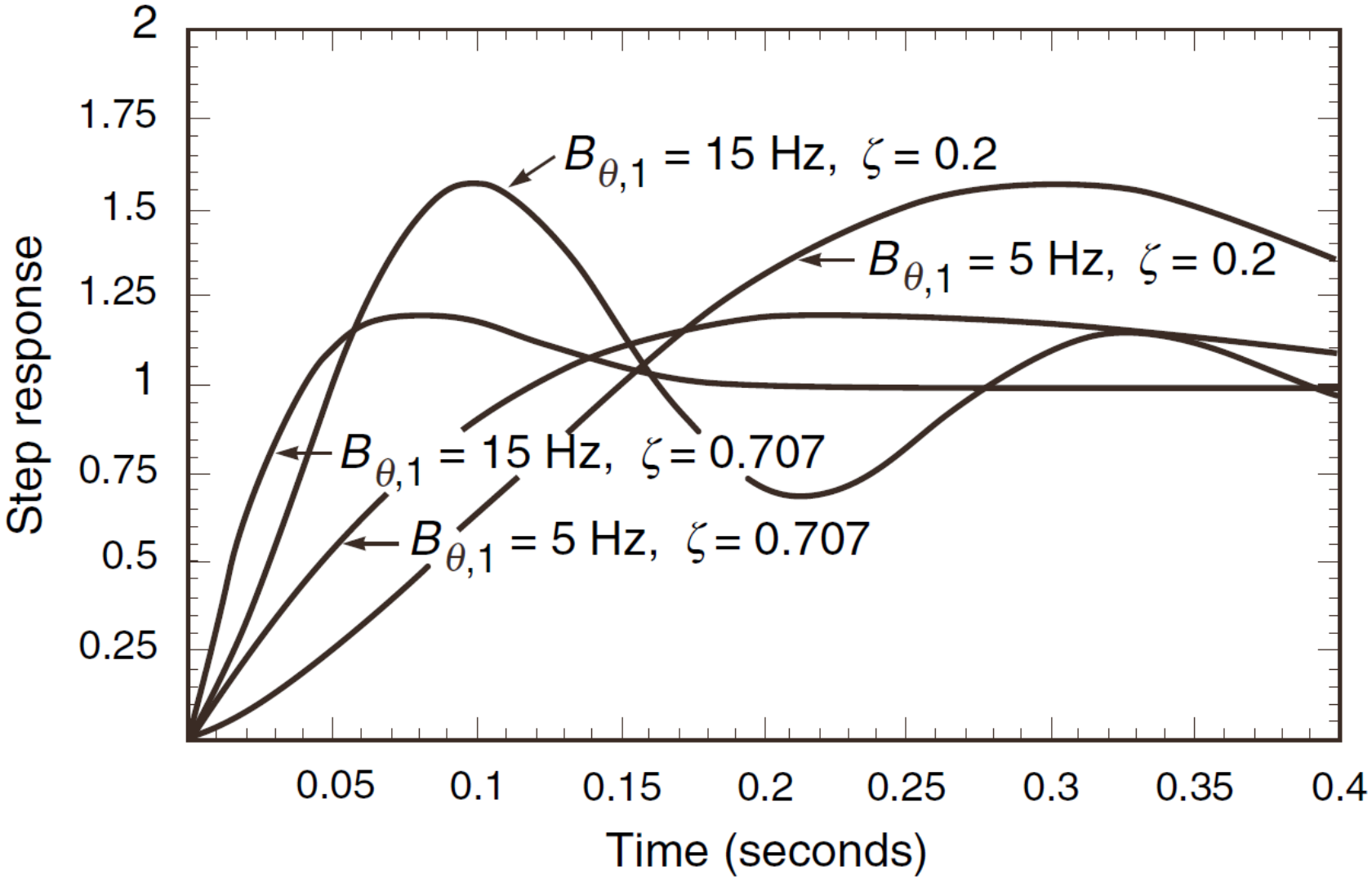


図12.11　二次ステップ応答

PLLの外乱要因と追尾特性

外乱要因	ループ帯域 (等価雑音帯域幅)
	狭い ⟷ 広い
ユーザーの運動	× ⟷ ○
白色雑音	○ ⟷ ×
クロック変動	× ⟷ ○

各外乱に対する追尾特性のバランスを取るため
最適なループ帯域の選定が必要

二次ループの定常状態誤差

ユーザーの運動に対して

ステップ，ランプ(等速)：　追従できる

パラボラ(等加速度)：　　定常状態誤差は以下となる

$$\lim_{t\to\infty}\Delta(t)=\frac{k_G\,g\cdot 360}{4\lambda B_{\theta,1}^2}\qquad \text{degrees}$$

PLLは位相誤差が15度以上になるとロックを外しやすい

従って

追従限界 [g]	帯域幅 [Hz]
1	17.6
0.25	8.8

白色雑音下におけるコスタスループの性能

位相誤差の分散は以下となる

$$\mathrm{var}(\Delta\hat{\theta}) = \frac{B_{\theta,1}}{C/N_0}\left(1 + \frac{1}{2T_{CO}C/N_0}\right) \quad \mathrm{rad}^2$$

二乗損失の項

二乗損失はC/N_0が25dB-Hz以上では問題にならない
GPS信号は通常30dB-Hz以上である

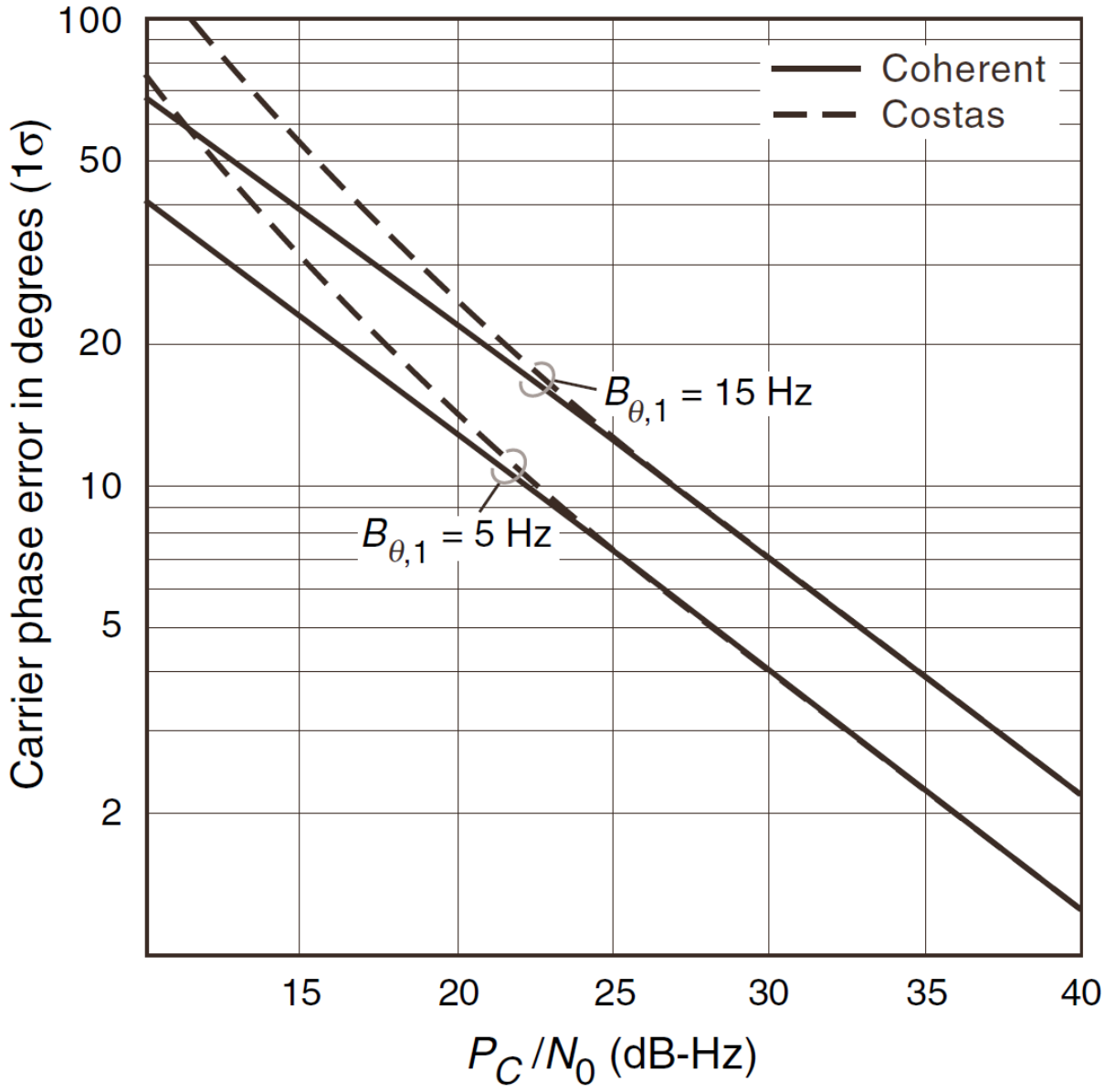
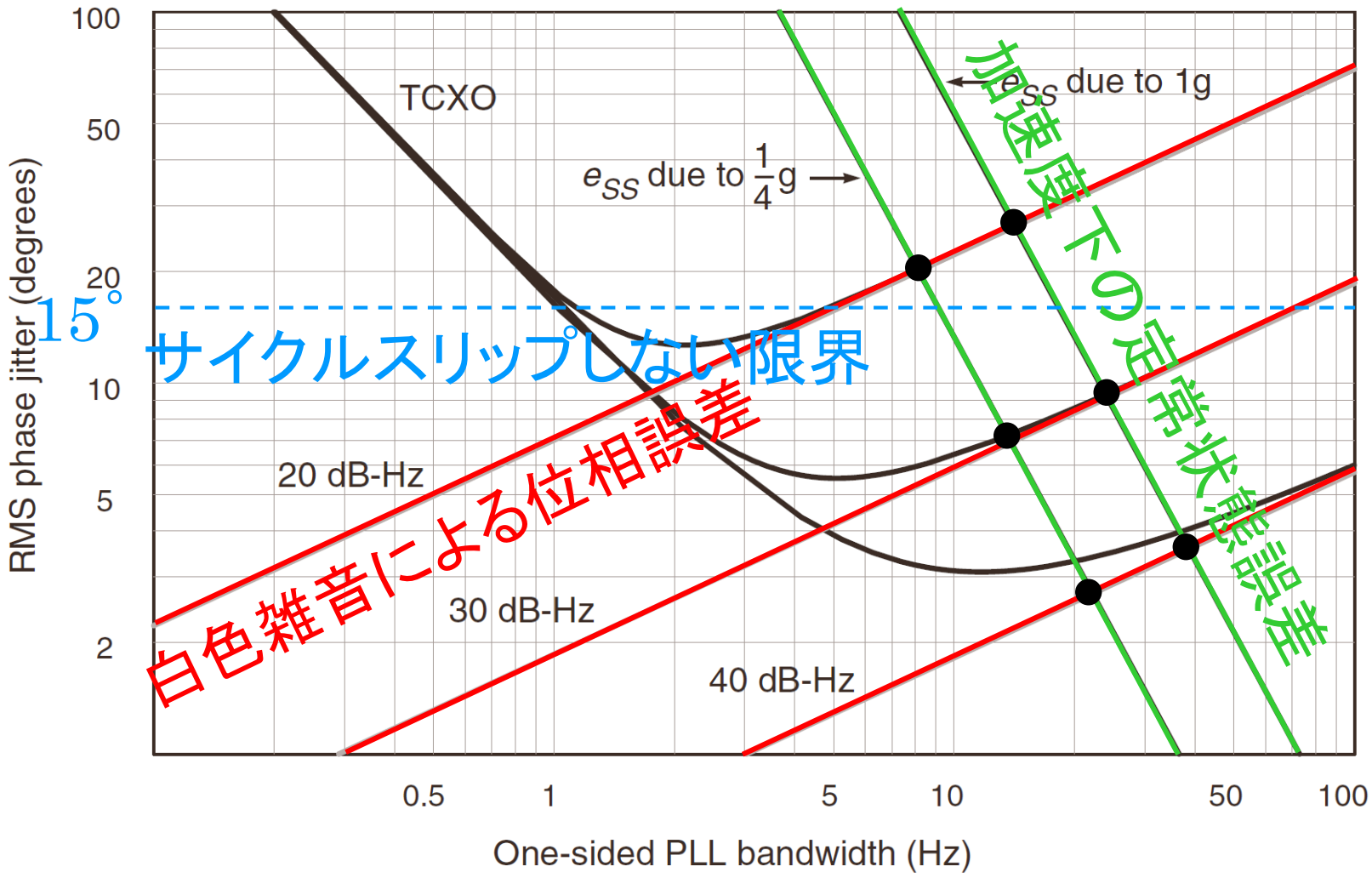


図12.12　白色雑音下におけるPLLの性能

ループ帯域幅の選定

● 動作点



位相誤差の小さい(なるべく下寄りの)動作点を選ぶ

付録12.A　周波数ロックループ(FLL)

FLLの役割

　レプリカ信号と受信信号の周波数差を弁別器で検出し，周波数差をゼロにするようにレプリカ信号発振器を制御する。位相のロックは考慮しない

FLLの目的

1) PLLに同じ

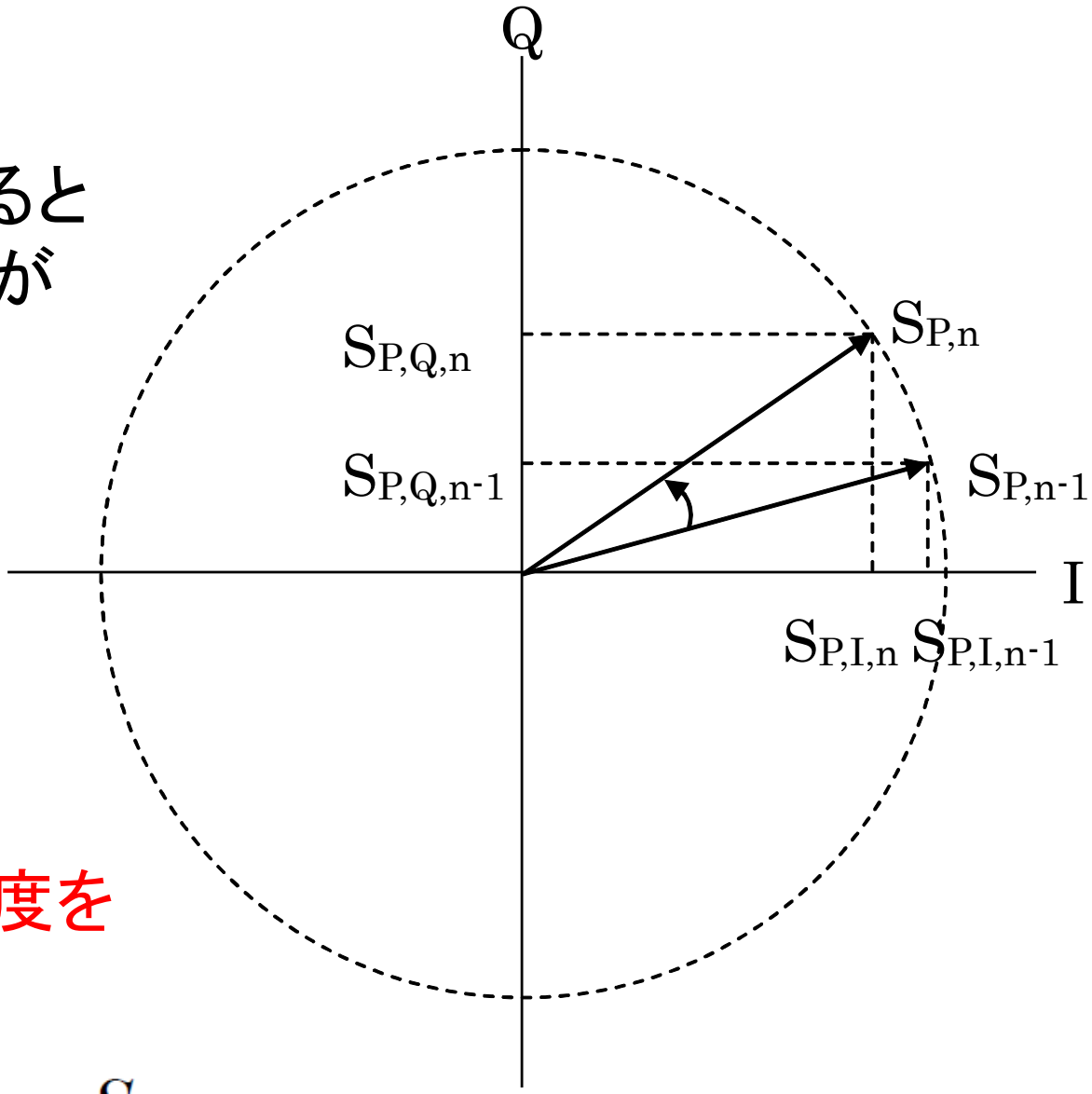
　　加えて．．．

2) 信号捕捉からPLL開始までの間の周波数引き込み

3) 劣悪な受信状態での概略追尾

FLLの弁別器関数

周波数がずれていると
相関値ベクトル$S_{P,n}$が
回転する

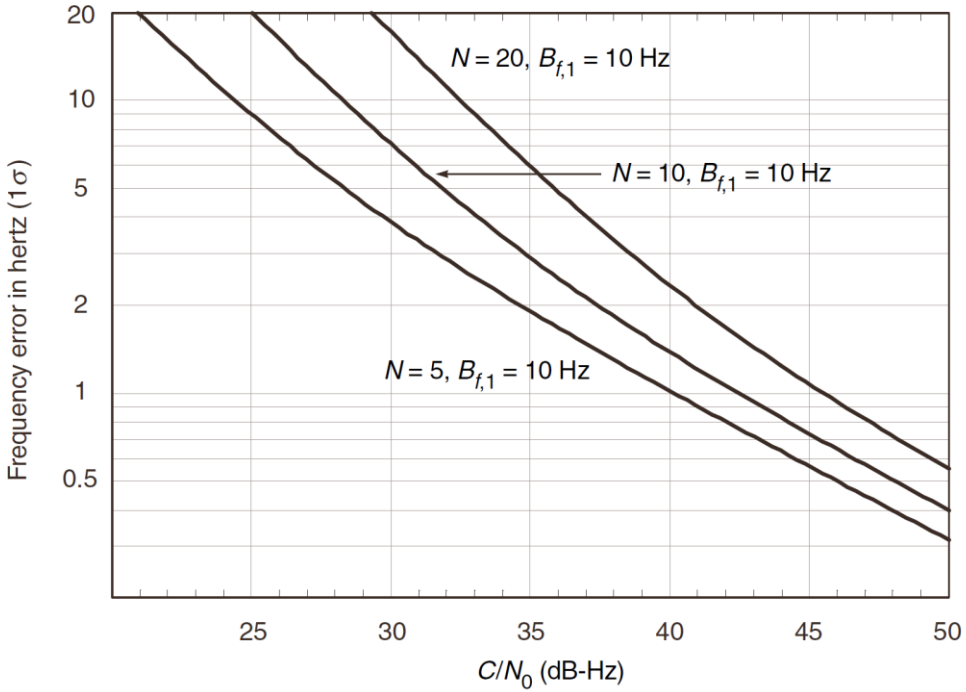


周波数ずれ＝角速度を
外積で近似する

$$S_{P,I,n-1}S_{P,Q,n} - S_{P,I,n}S_{P,Q,n-1}$$

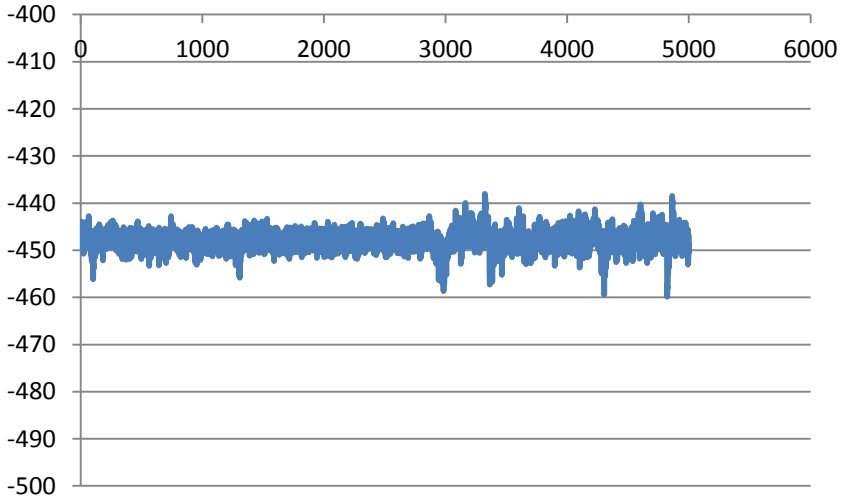
白色雑音下におけるFLL特性

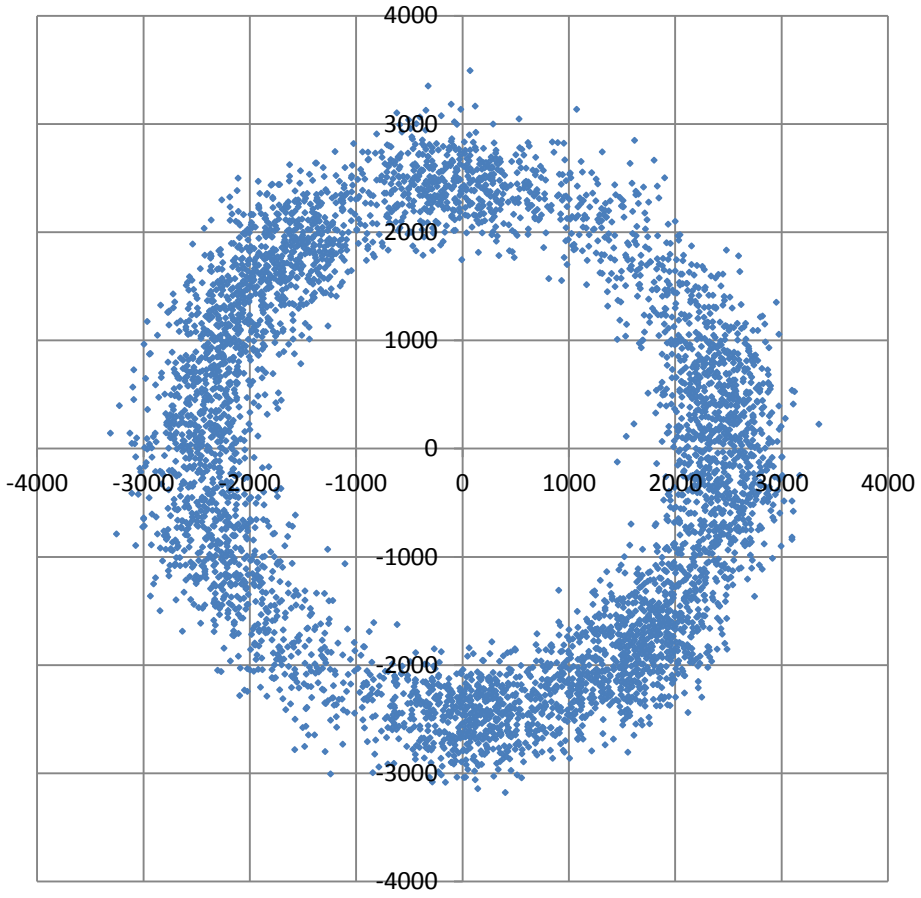
$$\mathrm{var}\{\Delta f\} = \frac{B_{f,1}N}{2\pi^2(N-1)^2 T_{CO}^2 C/N_0}\left(1+\frac{N-1}{2(C/N_0)T_{CO}}\right)$$



	加算時間T_{CO}	加算回数N
信号捕捉時	$0.1T_B = 2ms$	10
追尾中	$0.5T_B = 10ms$	2

FLLとDLLで追尾時のIとQ値（実データ：QZS）





ここから実際のソースを利用
（信号捕捉、信号追尾）

利用するIFデータ

- 2013年4月5日の研究室屋上データ（約45秒 1.8GB）。10時12分（L1）
- Fraunhofer社製（ドイツ）
- Sampling rate : 40.96MHz.
- Bit resolution: 2
- Bandwidth: 13MHz.

Front-end Configurations

b) Selection of the data output configuration

1 : L1-band via *USB1* with 4bit resolution
L2-band via *USB2* with 4bit resolution

2 : L1- and L2-band via *USB1* with 2bit resolution each
L5-band via *USB2* with 4bit resolution

For further information about the composition see capture 6.2.

Band	Center Frequency	Analog IF [MHz]	Digital IF [MHz]	IF spectrum orientation	Bandwidth [MHz]
L1	1575.42MHz	53.780000	12.820000	Flipped	13
L2	1227.60MHz	53.800000	12.840000	Flipped	13
L5	1176.45MHz	53.700000	12.740000	Non flipped	13

Table3: Intermediate frequencies

The sampling frequency f_s is 40.960000MHz for all bands.

32ビットと64ビット

- ソースファイルは32ビット版です。64ビットで計算する場合は、x64でコンパイルが必要
- 違いはAcquisitionで用いている「fftw（フーリエ変換）」を32ビットまたは64ビットで生成したかのみです→よってライブラリのみアップしています
- 64ビットのほうが若干速いです（詳細はわかりません）
- VS2010のExpress版（フリー）では64ビット版はないようです

ソフトの設定

- sdr_test.slnがプロジェクト
- プロジェクトのsdr_testプロパティでC/C++のリンカのシステムのスタックサイズを1GB程度に設定（64ビット版は2GB程度）
- 同じくリンカの入力でfftのライブラリを3つ記入。libfftw3-3.lib libfftw3f-3.lib libfftw3l-3.lib
- Release、Debugモードどちらでも動作します
- ただし、Debugモードで途中ブレークポイントを設けるとバグが発生することに気付き、現在対応中です。アップしたソースはバグがでないことは確認していますが、確認をとったPCが少ないためセミナー中に発生することも十分考えられます

入力ファイルと出力ファイル

- C言語のソースファイルは43個
- main.cppがメイン
- 入力ファイルはフロントエンドでADされたIFデータのみ
- 出力ファイルはいくつかあり、自由に設定してください。デフォルトでは、適当な途中経過を出力しています

計算時間

- 高速化をほとんどしていないため、計算時間を要します
- SF=40.96MHz：Corei7（3770）の3.40GHz、メモリ8GB、64ビット計算機で約40秒のIFデータから、Acquision及び40秒のトラッキング（航法メッセージ復調）、さらに40秒のトラッキング＋測位演算で320秒要した（7チャネル）。1分のトラッキング＋測位演算のみで160秒
- 多大な計算を要しているのは、信号補足や信号追尾部の相関処理部分
- 単純にマルチスレッドにして8チャネル40秒データ（追尾＋測位演算のみ）が30秒程度。16チャネルにするとその倍程度→デモ

IFデータの読み込みについて

40秒分のIFデータ

まず40秒分のデータで信号捕捉（捕捉自体は10ms程度のデータ）や航法メッセージ取得のための信号追尾を行う

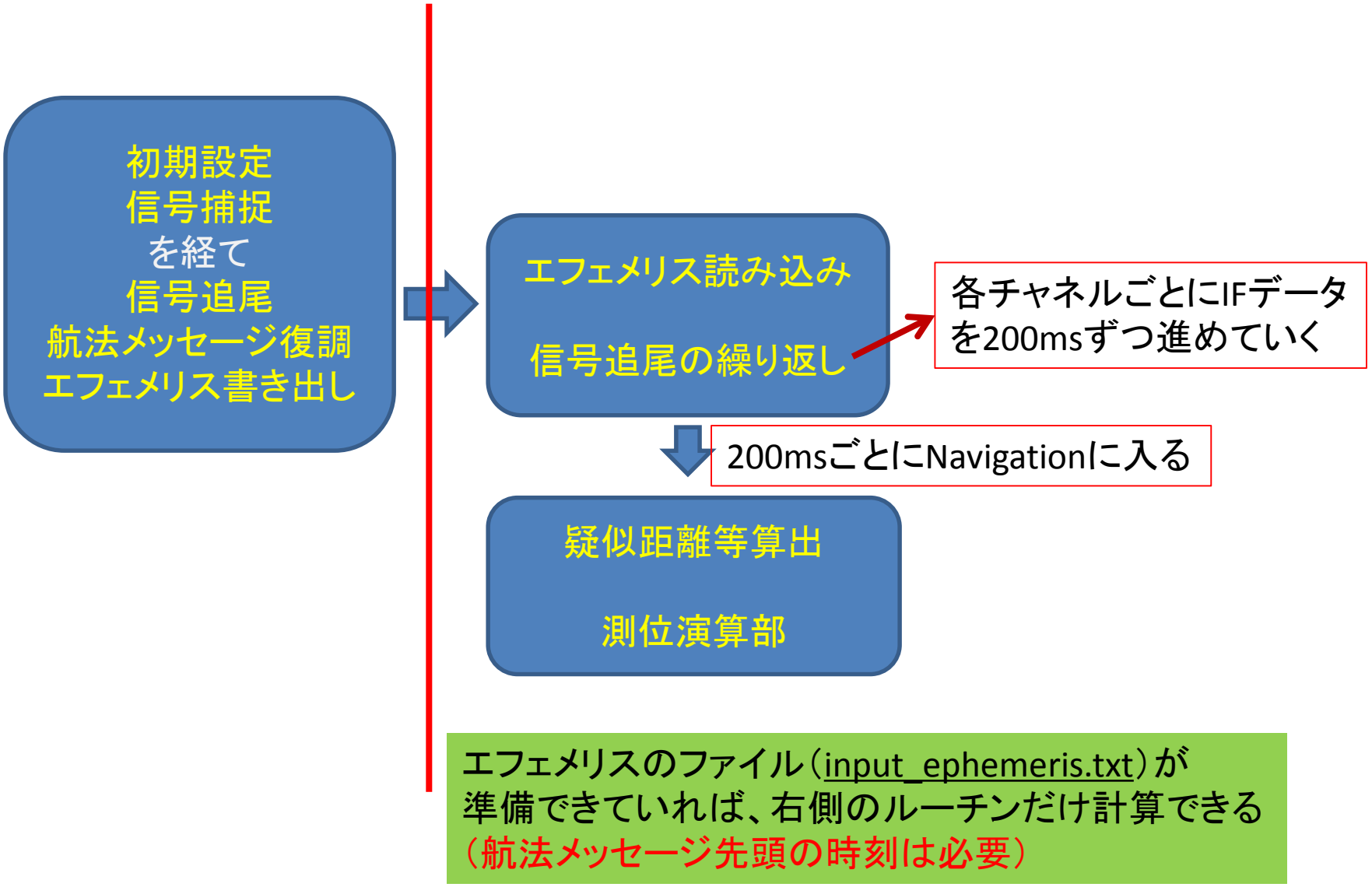
上記で航法メッセージの先頭の時刻も判明し、エフェメリスも取得できたので、通常のトラッキングかつ疑似距離算出、測位演算を行う

注意すべき点は、
＊航法メッセージを取得するために読み込んだデータを
また最初から読み込んで信号追尾し測位演算を行う点
＊基本的に信号捕捉は1回なので、そのときに捕捉した衛星しか利用しない。新しい衛星が入る部分は対応していない
＊最初に捕捉した衛星でよければ、1時間分でも計算は可能。ただしエフェメリスは2時間ごとに更新される

外部変数

- 関数の外で利用するような変数は、すべて外部変数として定義しています
- 外部変数だけでなく、変数定義、メモリ管理等、プログラムが大変利用しづらく申し訳ありません

プログラムの大枠


初期設定
信号捕捉
を経て
信号追尾
航法メッセージ復調
エフェメリス書き出し
エフェメリス読み込み
信号追尾の繰り返し
各チャネルごとにIFデータを200msずつ進めていく
200msごとにNavigationに入る
疑似距離等算出
測位演算部
エフェメリスのファイル（input_ephemeris.txt）が
準備できていれば、右側のルーチンだけ計算できる
（航法メッセージ先頭の時刻は必要）

ファイルの場所

- sdr_GPSQZS_win32
- そのフォルダの中にsdr_test.slnがあります。そのディレクトリに全てのCのソースがある
- 最初の入力ファイルであるIFデータは適当な場所にコピーしてください。IFデータ読み込み部分はmain.cppの64行目付近からです。L1+L2+L5で3つのファイルがあります。注意点は、L5は同時に取得していないことです
- 出力結果ファイル（主にエクセルファイル）はsdr_testディレクトリ下にあります。同時間帯に取得したNetR9の観測データもコピーしています

プログラムの順番に概要説明

- main.cppがメイン部分
- global.hは外部変数定義部
- global_main.hは外部変数を他の関数で利用するための定義部

set_initial_value.cpp

- プログラムに必要となる初期設定を読み込み変数に格納。重要なパラメータがあります
- トラッキングループで使用する搬送波のSINとCOSのテーブルを準備
- 相関波形用の変数を準備
- レファレンスの位置を緯度、経度、高度から地球中心のX,Y,Zに変換

makeCaTable.cpp

- GPSと準天頂衛星のC/Aコードのテーブルを準備。合わせて0.25チップ早いものと0.25チップ遅いC/Aコードを準備（40.96MHzなので、1チップで約40個の相関値をとれる。±10としています）
- なお、現在の準天頂衛星（PRN193）は本ソースでは33番としている（PRN33番は実際はGPSに割り当てられている）

その他の信号

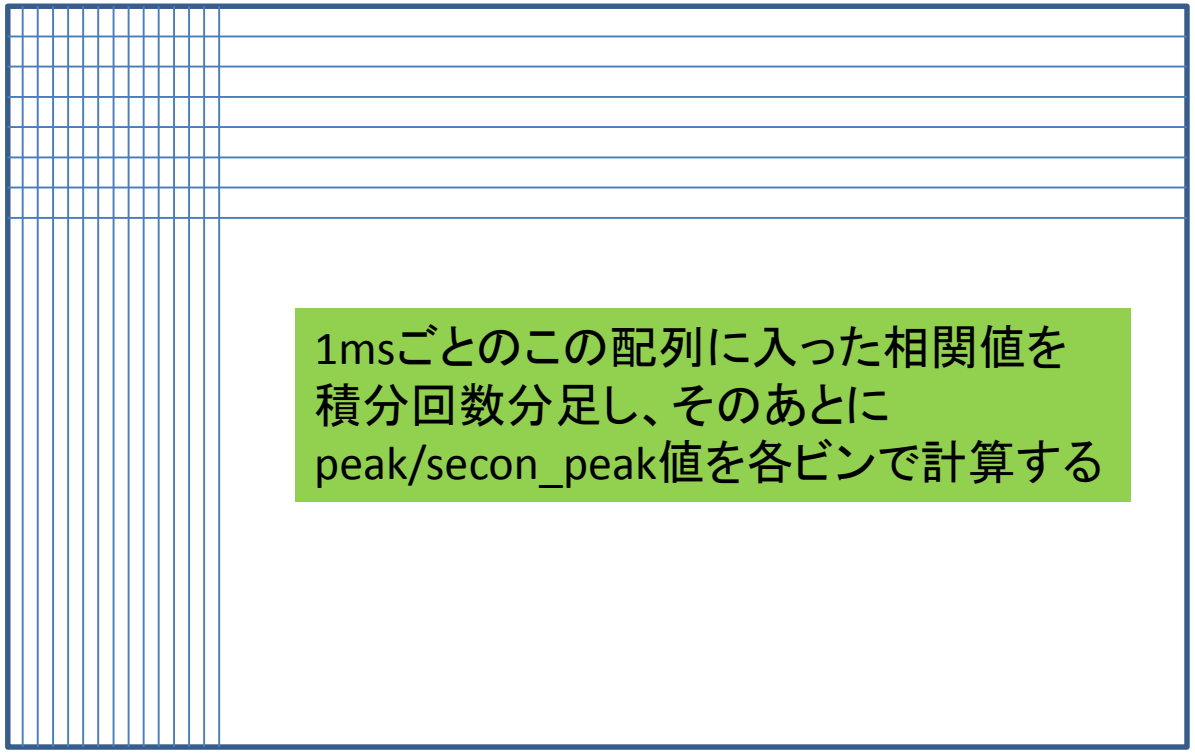
- makeL1CTable→BOC信号のため±6（early+late）としています
- makeL2CTable→L2CMのみ生成。CLの部分は0としています。±20（early+late）としています
- makeL1CTable→ L5Iのみ生成。10.23Mcpsのため、±2（early+late）としています

acquisition.cpp

- 信号捕捉を実施。L1-C/AとL1Cは500Hz毎、L2Cは10Hz毎、L5は250Hz毎
- 信号捕捉が成功できたと判断した信号について10msのデータでさらに細かいドップラー周波数を算出（L1-C/AとL1Cのみ）。
- ノンコヒーレント→：L1-C/A（4ms）、L1C（コード長分）、L2C（コード長分）、L5（4ms）
- 相関をとるベース技術はFFTを利用したサーキュラーコリレーション
- L2CはL1-C/Aのドップラ推定値を利用

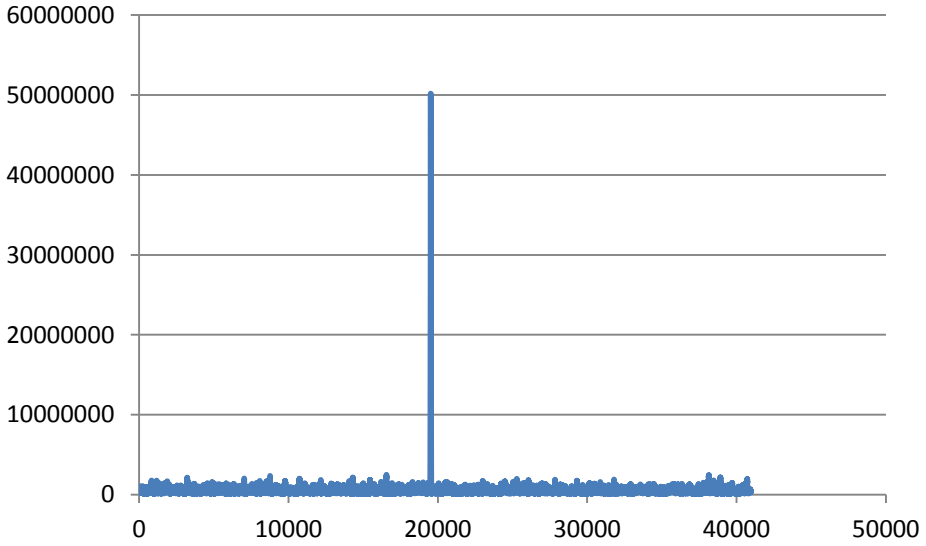
ノンコヒーレント

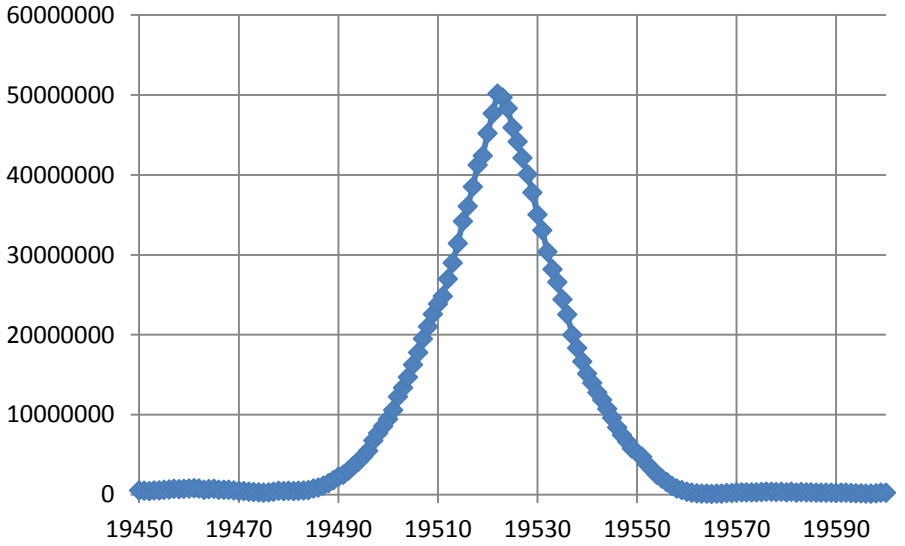
周波数ビン
28または56個程度



コード位相ビン
（40960個程度）

PRN5の信号捕捉（L1-C/A）（右が拡大図、4ms分のFFT）





C/Aコード1023チップを40960等分したときに、
ちょうど19522番目付近にC/Aコードの先頭が存在する
peak/second_peakを見つけるときに、peak周辺1チップは無視。ドップラ周波数は1748Hz

set_channel.cpp

- Acquisitionで捕捉された衛星の情報について整理し、以降の処理でその変数を利用
- 以降のTracking等で衛星番号で管理はしているが、信号の大きいと思われる順に0からの番号を付加

tracking_pre.cpp（L1-C/A）

- この前段階のTrackingは航法メッセージを取り出すためのもの
- 各衛星を500msごと進ませているため、トータルで80回繰り返して40秒の航法メッセージを取り出す
- 40秒分の航法データがあると、エフェメリスを解読できる（追尾に失敗しなければ）
- 後で出てくるtracking.cpp関数と追尾ループの内容は同じ

その他の信号

- L1C、L2C、L5は基本的にこの信号追尾が確認できる部分まで紹介しています。この後FEC等の復調とNHコードが存在します
- それぞれの関数名で信号追尾を確認
- なお、追尾ループのパラメータはそれぞれの信号で異なります。set_initial_value.cppのファイルを参照してください

tracking_test.cpp（L1-C/A）

- この関数では、DLL+PLL以外にFLLをテストしています。時間があれば紹介します

tracking_pre.cppでの初期値

- すべての信号において、Acquisitionで得た各衛星のコード位相の先頭位置とドップラー周波数を利用
- 下（102行目）が最初にファイルポインタを移動させる部分（コード位相の先頭）と蓄積部分（200msごとに繰り返されるため）

```
pos = 0 + channel.codePhase[channelNr]-1 +
blksize_accumulation[prn];
fsetpos(fp, &pos);
```

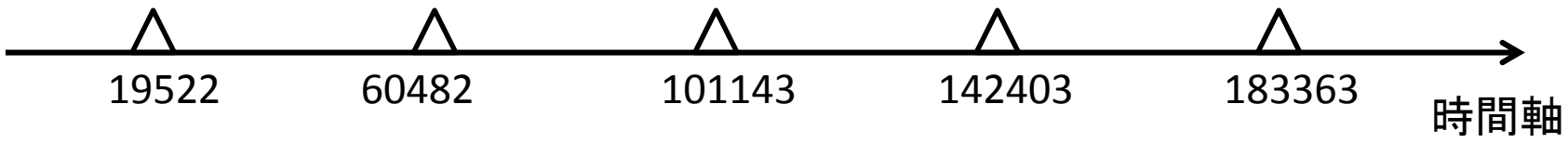
相関計算部

```
for(i=blksize;i>=0;i--){
    //1つの例　sin(2*pi*f*t)を求めるとき、tは時間なので、「i/settings.samplingFreq」となる
    trigarg[i] = (carrFreq[prn] * 2.0 * pi)*double(i)/settings.samplingFreq+remCarrPhase[prn];
    //ここでは実際にsin,cosを求めるために位相を出している
    residual = trigarg[i] - pi*2.0*int(trigarg[i]/(pi*2.0));
    index = int(residual*100);
    j = int(rawSignal[i]);

    switch(j){//準備済みのテーブルを利用してcos,sin値を計算
    case  3:qBaseband=Table_cos3[index]; iBaseband=Table_sin3[index]; break;
    case  1:qBaseband=Table_cos[index]; iBaseband=Table_sin[index]; break;
    case -1:qBaseband=Table_cos2[index]; iBaseband=Table_sin2[index]; break;
    case -3:qBaseband=Table_cos4[index]; iBaseband=Table_sin4[index]; break;
    case  0:qBaseband=0; iBaseband=0; break;
    }

    //6つの相関値を以下で計算
    I_E = I_E + earlyCode[i] * iBaseband;
    I_P = I_P + promptCode[i] * iBaseband;
    I_L = I_L + lateCode[i] * iBaseband;
    Q_P = Q_P + promptCode[i] * qBaseband;
    Q_E = Q_E + earlyCode[i] * qBaseband;
    Q_L = Q_L + lateCode[i] * qBaseband;
}
```

搬送波とコードの位相を正確に追尾できないと、
上記の相関値に影響が出る

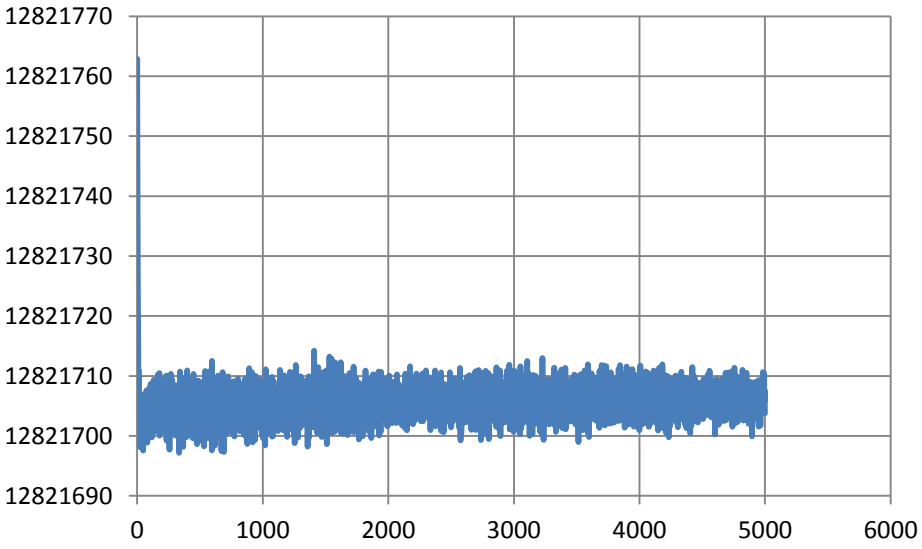
相関のイメージ

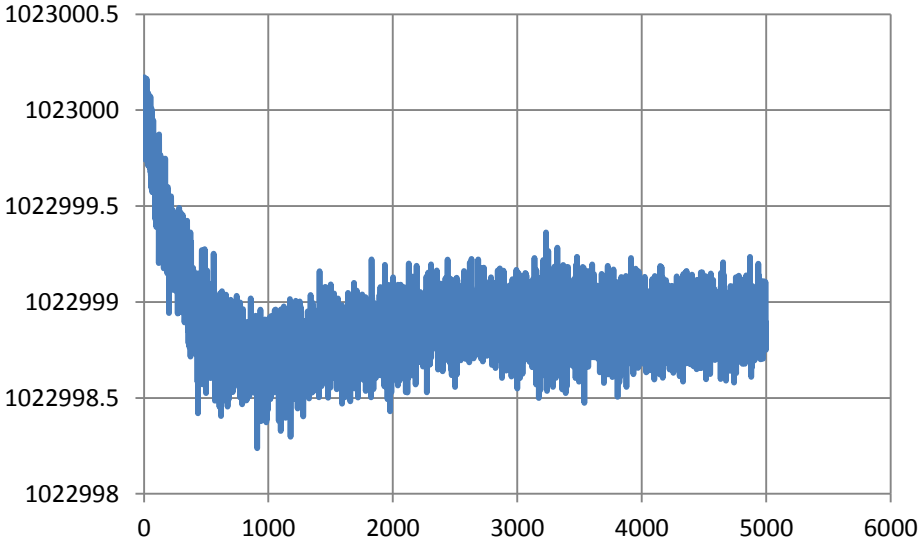
最初にAcquisitionで得たコード位相の先頭がほぼ正しいため、
それ以降はIFデータを1ms分のサンプリング値進ませることになる。
ただしドップラー周波数等で微妙に変化していく



上記は実際のPRN5番の数値（最初から）
IFデータをほぼ40960進ませることになる。
これはIFデータの中にあるC/Aコードの先頭がそこに存在することを意味する。前スライドのレプリカコードは、ちょうどこのIFデータのC/Aコードの先頭から1ms分掛け算して足されている。
この相関値のピークをI_Pとしている。
もし搬送波位相の位相が受信信号とずれてくると、
コリレータのピーク値が振幅する

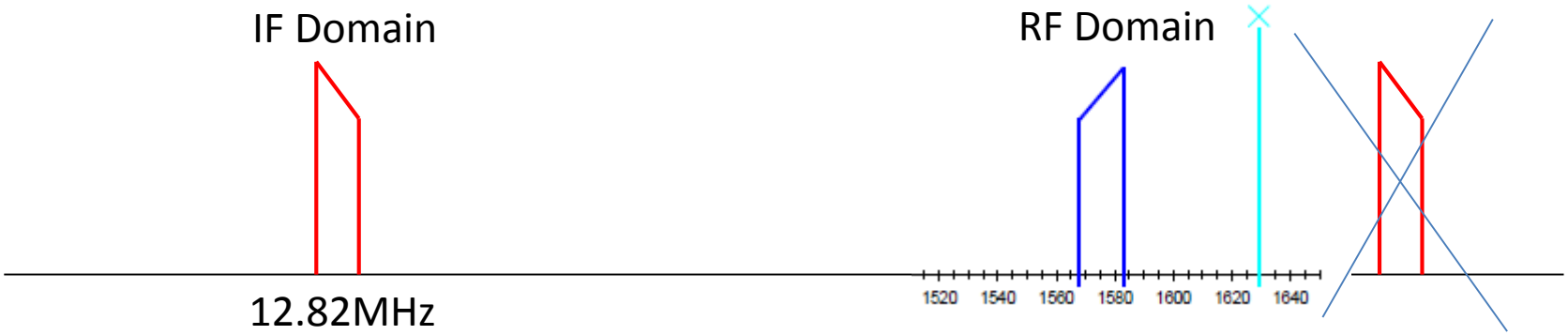
キャリアとコードの周波数
（PRN5　最初の5秒）





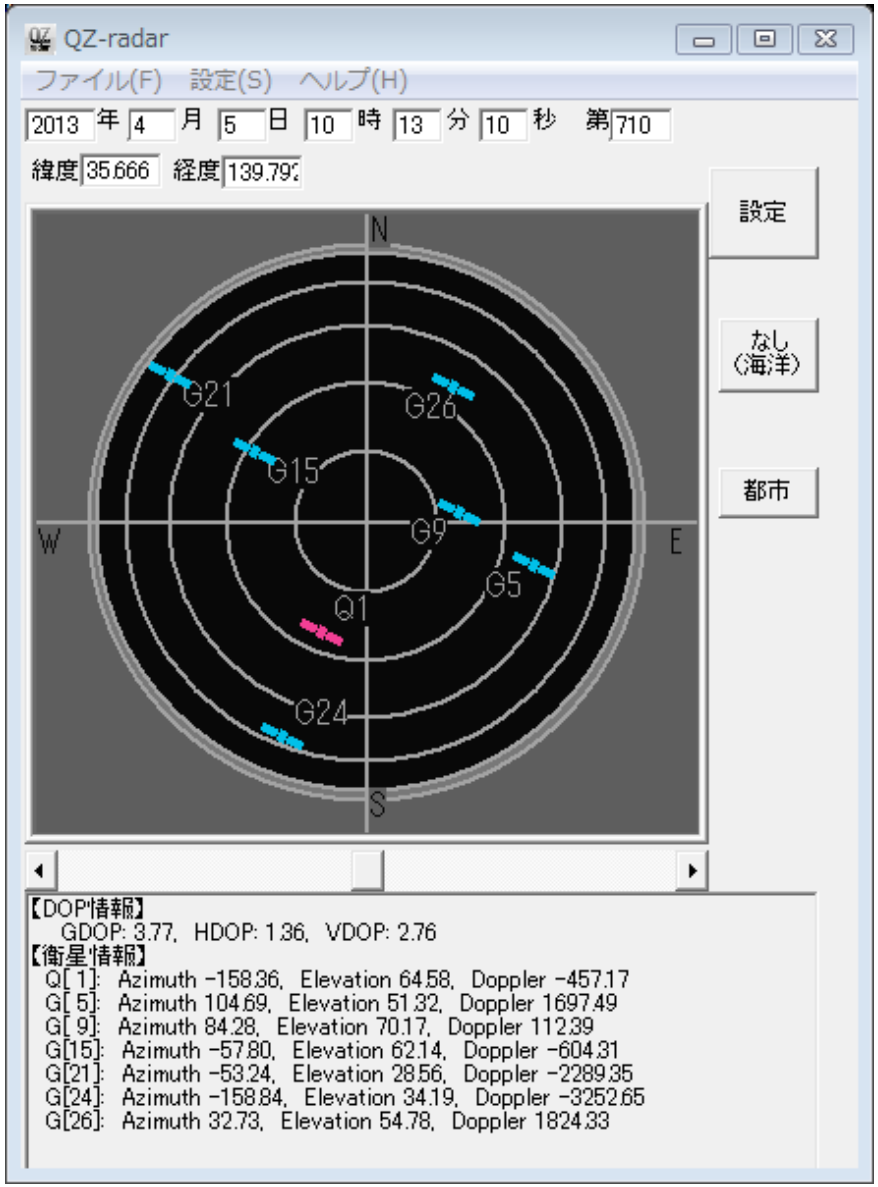
L1帯のIF周波数は12.82MHz（FEによる）のため、
ドップラ周波数は上記から12.82MHzを
引いた値となる（受信機クロックの周波数誤差は含む）　　？
Acquisitionでの推定値は1748Hz
Trackingでは1705Hz程度（実際は逆？）

補足説明(私の理解)



The sign in the IF domain is opposite to the sign in the RF domain by mixing.

2013/4/5 10:13:10のデータ

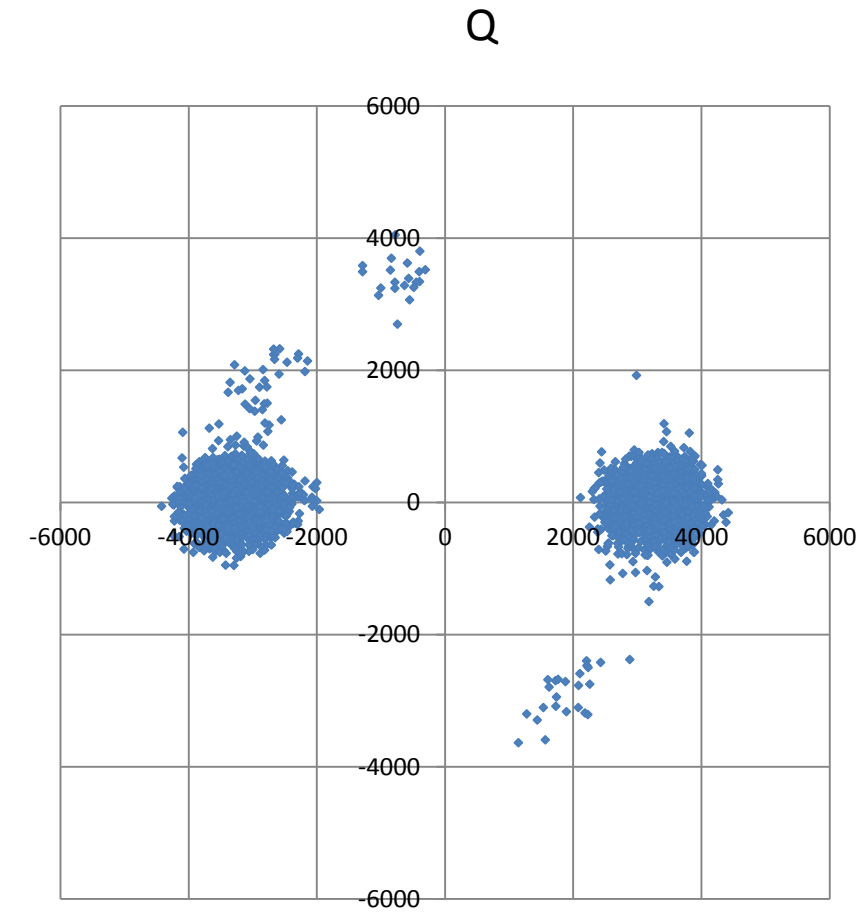


GPSTIME	PRN	Pseudo-range	Doppler	QZ-Vision Diff		C/N_0	Ele	Azi
436390	9	20099347.99	121.927	112.41	9.517	49	70.2	84.3
436390	5	21456107.13	1708.16	1697.49	10.67	48.9	51.3	104.7
436390	15	20707739.16	-593.82	-604.15	10.33	48.2	62.1	302.2
436390	26	20582048.72	1834.52	1824.52	10	49.2	54.8	32.7
436390	24	22419295.21	-3242	-3252.67	10.71	44.7	34.2	201.2
436390	33	37281649.71	-446.74	-457.29	10.55	45.9	64.6	201.7
436390	21	23324356.71	-2279.7	-2289.57	9.918	42.3	28.6	306.8

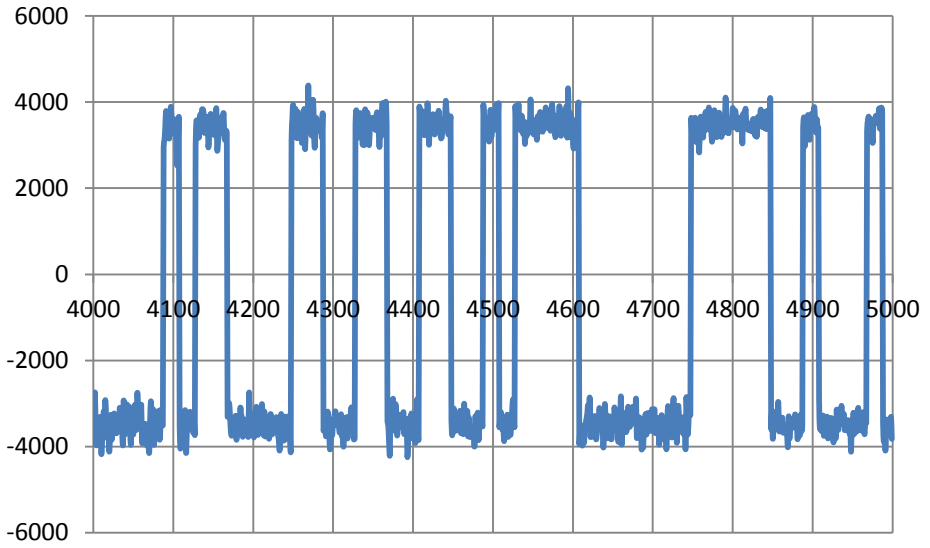
ソフト受信機の出力結果にQZ-raderを付加

＊周波数オフセットは10Hz程度
＊QZSはこの位置で、実際には東京に対して距離が近づくことに注意

信号追尾OK


Q
6000
4000
2000
0
-2000
-4000
-6000
-6000
-4000
-2000
0
2000
4000
6000

I


6000
4000
2000
0
-2000
-4000
-6000
4000
4100
4200
4300
4400
4500
4600
4700
4800
4900
5000

その他の信号の解析

- settings.skipAcquisition = 1;//1:Acq　0:NO Acq
- settings.skipDecode =　0;//1:Decode　0:NO Decode (only L1-C/A --> 1)
- settings.Navigation =　0;//1:NAV　0:NO NAV (only L1-C/A --> 1)
- settings.frequency =　5;//0:L1-C/A 1:L1C 2:L2C 5:L5

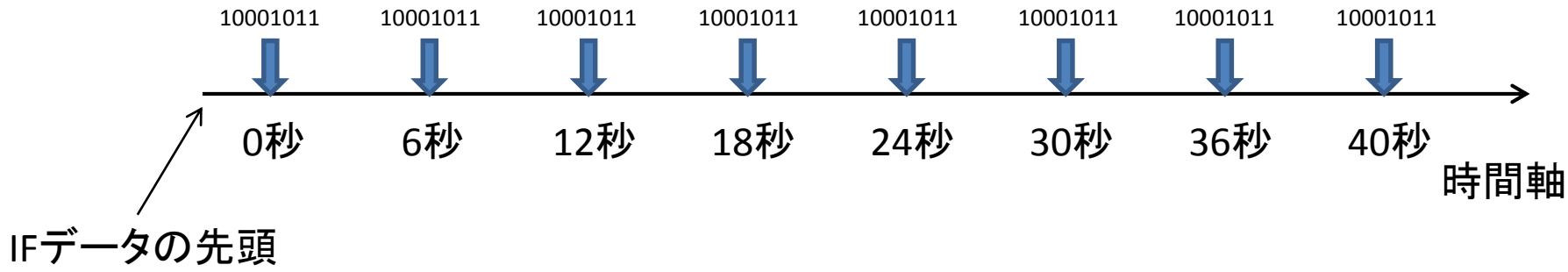
- 確認用設定値

① 1,0,0,0
② 1,0,0,1
③ 1,0,0,2
④ 1,0,0,5
⑤ 1,1,1,0（次ページ以降のスライド） エフェメリス生成＋時刻＋NAV
⑥ 0,0,1,0（次ページ以降のスライド） エフェメリスと時刻があればNAVのみ

ここから擬似距離計算や測位演算

findPreambles.cpp（L1-C/A）

- 航法メッセージの先頭のプリアンブルを見つける
- GPSのC/Aコードでは、8ビットの10001011を探す（6秒ごとに入っているはず）
- 前のtracking_preで取り出した20msごとの（1、-1）を、20ms分で1ビットとして作成する。そのビットより上記のプリアンブルを探す
- 下図より、6秒ごとに繰り返すことを利用する。よって信号が復調できていれば、必ず最初から6秒以内にプリアンブルが見つかることになる



さらにTLMとHOWでのチェック

プリアンブル　　Parity

TOWのカウントメッセージ　　Parity
（GPS時刻に変換：**TOWの時刻は次のサブフレームの時刻！**）

TLMとHOWは各サブフレームの先頭から順番に30ビットずつあり、
この30ビットの最後の6ビットのparityでさらにデータチェックする。
1つのサブフレームは300ビット（6秒＝300×0.02秒）である。サブフレームは
1から5まで順番に流れているので、どのサブフレームからきたとしても、
30秒（6秒×5）で1周することになる。よって40秒分程度読み込んでおけば、
エフェメリスの存在するサブフレーム1から3を確実に読み込むことができる

サブフレーム先頭の積算時刻をストック

これがIFデータの最初から6000msまでとすると
PRN9は、4883ms目にプリアンブルの先頭が存在

衛星	PRN9	PRN5	PRN15	PRN26	PRN24	PRN193	PRN21
先頭のms	4883	4888	4885	4885	4891	4941	4894

準天頂衛星（PRN193）だけ距離が長い分、遅れる（1msで約300kmの差）
各衛星の1msオーダの距離差はここで把握しておく

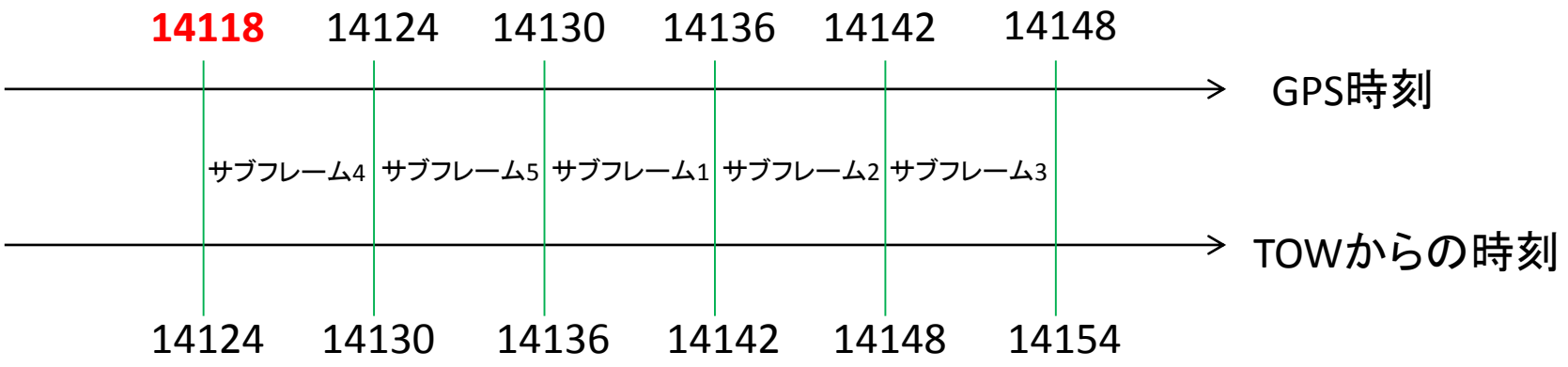
decode_navigation.cpp

- プリアンブルを見つけたことにより、サブフレームの先頭がわかったので、その先頭から順番に1500ビット（5つのサブフレーム分）分読み込む
- それらのビットデータよりエフェメリスを復調。フォーマットについては、ICD-GPS-200を参照
- 一応サブフレーム2のIODEとサブフレーム3のIODEが一致しているかチェック

ephemeris.cpp

- GPS-ICD-200に基づいて復調する部分
- この関数では、エフェメリス復調だけでなくこの後の計算に重要なGPS時刻（正確には衛星がそのフレームを発射した時刻）も計算

以下は実例（添付データではない）



本プログラムではサブフレームを5つ読み終えた時点（サブフレーム3の先頭）でTOW（GPS時刻）を求めているため、30秒引いている！（14148-30=14118）

write_ephemeris.cpp

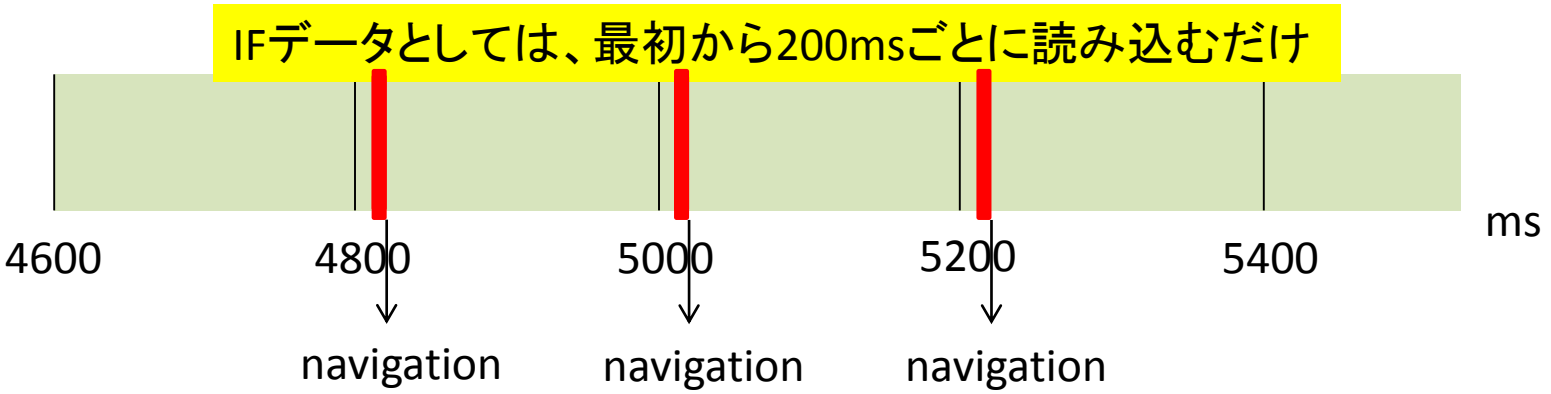
- デコードしたエフェメリスを書き出す部分
- input_ephemeris.txtファイルを開いて書き込む
- このファイルは、次のread_ephemeris.cppで読み込まれる
- よって、最初から通して計算することもできるが、input_ephemeris.txtファイルが準備できれば（あとTOWの時刻も）このread_ephemerisから計算することが可能

read_ephemeris.cpp

- 前のスライド参照
- エフェメリス＋（衛星数、TOW）を読み込む

tracking.cpp

- trackingの200msごとのバッチ処理とnavigationとの関係を以下に示した。先頭衛星が4883ms（TOW=436386秒）のため、この時刻が衛星で刻印されて、おおよそ70ms前後で受信機側に到着したと考える。よって、この4883msの位置は、おおよそGPS時刻の整数秒+70ms前後である。ここでは、GPS時刻の整数に近づけるため、70ms前後を引いたタイミングでNavigationを行っている



実際には83-68=15のため、4815、5015、5215のタイミングで観測値を算出
「70ms前後の値」は、測位計算に一度入ると自信のクロック誤差も判明し、それを考慮した一番短い擬似距離を基準としている

tracking.cpp

- 追尾部分やフィルターはtracking_preと同じ
- 異なる部分は、これ以降の擬似距離算出とドップラ周波数算出のために、新しい変数を追加している個所

擬似距離算出用

```
if(code_phase[prn]>=0)
    PseudoRem[prn] = Diff_ms[prn] + double(code_phase[prn]/blksize) +
    (1023 - tcode[int(blksize)-1])*293.07/(settings.c / 1000);
else
    PseudoRem[prn] = Diff_ms[prn] + double(code_phase[prn]/blksize) +
    (1023 - tcode[int(blksize)-1])*293.07/(settings.c / 1000);
```

上記のPseudoRem[prn]は、擬似距離そのもので、これに適当な伝搬時間を足して光速をかけることにより、擬似距離を実際に算出している（calculatePseudoranges_testで）

Diff_ms[prn]：衛星ごとの1ms単位の差
double(code_phase[prn]/blksize)：受信機クロックに応じて決定した時刻での各衛星の位相差（±40960サンプルの範囲内）
1023 - tcode[int(blksize)-1])：1msごとにコードをすすませているが、衛星毎に余りが発生。その余り分を考慮

ドップラー周波数算出用

//for Doppler

- if(step-(iter-1)*settings.navSolPeriod>=Subframe_top-(Transit_Time+1)-10 && step-(iter-1)*settings.navSolPeriod<=Subframe_top-(Transit_Time+1)+9){

  Acq_carrier_frequency[prn] = Acq_carrier_frequency[prn] + carrFreq[prn]-settings.IF;//DopplerにE必K要v
- }

Subframe_topはサブフレーム先頭msの余り
衛星から電波が発射された時のGPS整数時刻付近での
観測値を求めるために、このサブフレーム先頭から伝搬時間を引き
前後10ms分で平均したドップラー周波数を利用する

navigation.cpp

- この関数は擬似距離計算以外は、単独測位を行っている。よって詳細は「GPSのための実用プログラミング」等を参照してください。中身は異なりますが、おおよそ同じような感じです。結果もおおよそ同じになると思います
- ここでは擬似距離計算とドップラー周波数からの速度計算を説明します
- なお、航法メッセージ4と5は解読していないため、電離層のモデル係数だけはここで定義する必要があります

trans_llh_xyz.cpp

- 緯度、経度、高度から地球中心のx,y,z座標系に変換する関数

satpos.cpp

- 衛星位置をエフェメリスより計算する関数
- 衛星の発射時刻が必要になります

calc_angle.cpp

- 衛星位置計算で求めた衛星位置と自身のアンテナ位置より、各衛星の仰角と方位角を求める関数

calc_iono_krb.cpp

- クロバッチャモデルより、各衛星の電離層遅延量を推定する関数

calc_tropo.cpp

- Saastamoinen modelより、対流圏遅延量を推定する関数

calc_pos.cpp

- 単独測位を行う関数
- 実際には、least_square.cpp関数で最小二乗法により測位演算が行われる
- この関数で、測位計算で求まった受信機の時計誤差を反映していることに注意

calc_posに続く関数

- satpos.cpp
- satpos_prev.cpp
- satpos_next.cpp
- re_calc_pos.cpp

上記の2つの赤色の関数は、1回目の測位計算で受信機時計誤差を求め、それによって疑似距離等を補正した後に再度衛星位置から求めている部分。さらにprev,nextの衛星位置計算は、速度計算用に前の時刻（transmitTime - settings.navSolPeriod/1000.0）と後の時刻（transmitTime + settings.navSolPeriod/1000.0）の衛星位置を求めている

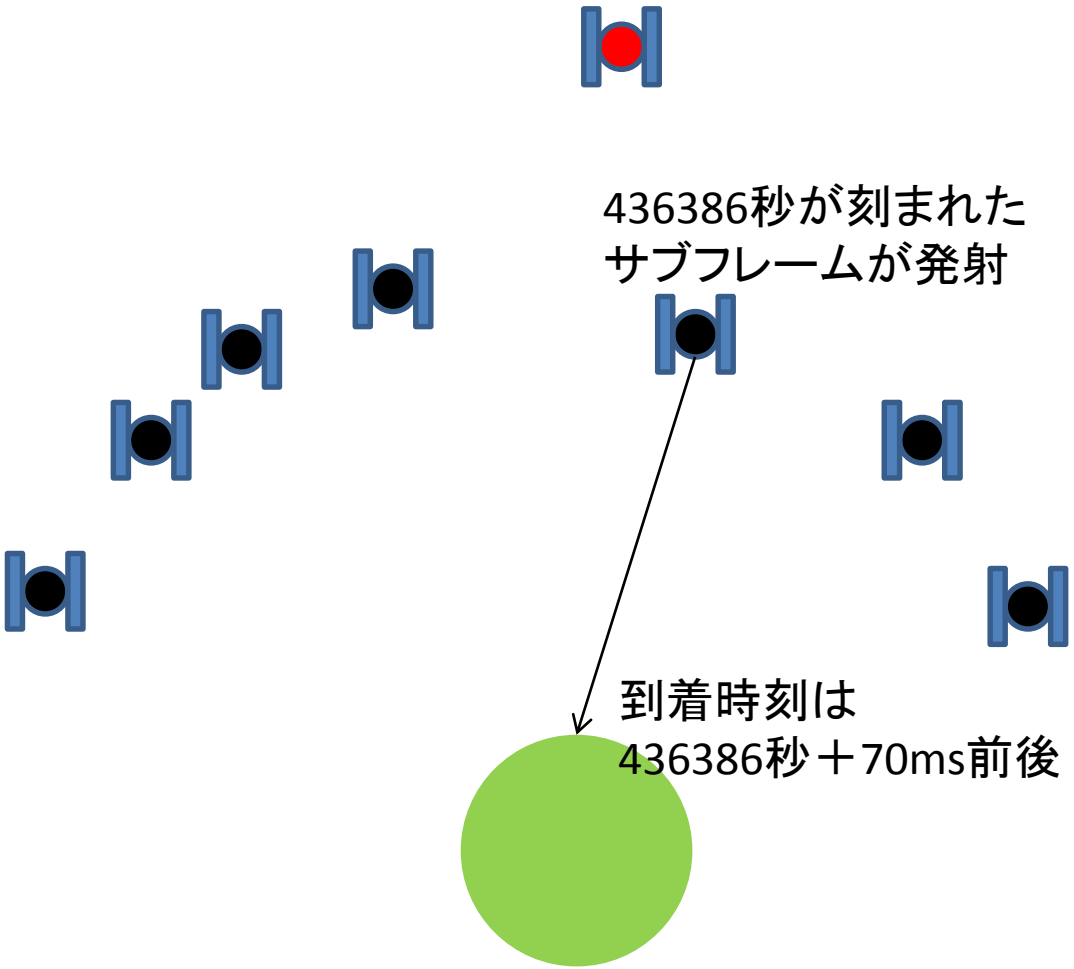
least_square.cpp

- 最小二乗法により測位演算を行う関数
- さらに速度計算用のルーチンも存在

- 測位演算部のコアの部分は、以下のr2[i]とr3[i]変数の計算。Init[0,1,2]はx,y,zに対応しており、「初期値＋毎回の最小二乗法で求めたずらす量」が入る。r3[i]は「自身で求めた疑似距離や衛星時計誤差、電離層、対流圏遅延推定量」から上記のr2[i]を引いている。よって、上の赤線部分が正しければ正しいほど良い精度で位置が求まることになる

```
r2[i] = sqrt((SVx[prn]-init[0])*(SVx[prn]-init[0])
                                        +(SVy[prn]-init[1])*(SVy[prn]-init[1])
                                        +(SVz[prn]-init[2])*(SVz[prn]-init[2]));

r3[i] = Pseudoranges[prn] + SV_corrtime[prn] -
                                         Iono[prn] - Tropo[prn] - r2[i];
```

calculatePseudoranges.cpp（イメージ図）



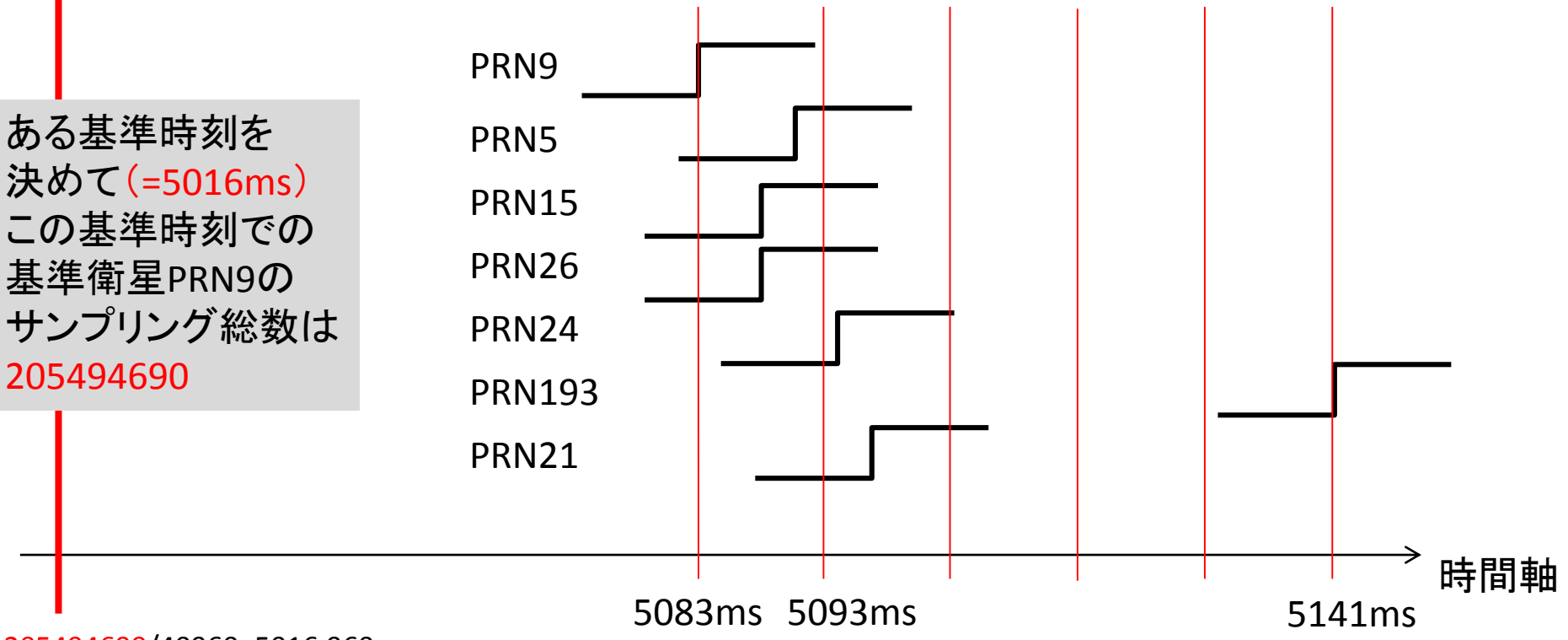
5番衛星の436386秒の整数時での疑似距離を求め、その時の測位計算を行う

436386.07秒前後に、サブフレームの先頭が受信されているので、0.07秒前の時刻を設定。
さらに、1回目の測位計算で受信機側の時計誤差もわかるので、そこから計算した疑似距離を利用してもう少し正確な整数時での疑似距離を求める。よって、衛星位置計算は436386秒よりも前の衛星の位置を求めることになる

各衛星の擬似距離を求める

- サブフレームの先頭は以下の通り

ある基準時刻を
決めて(=5016ms)
この基準時刻での
基準衛星PRN9の
サンプリング総数は
205494690

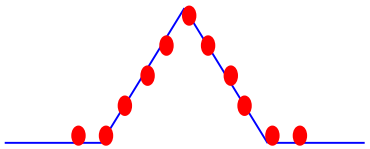


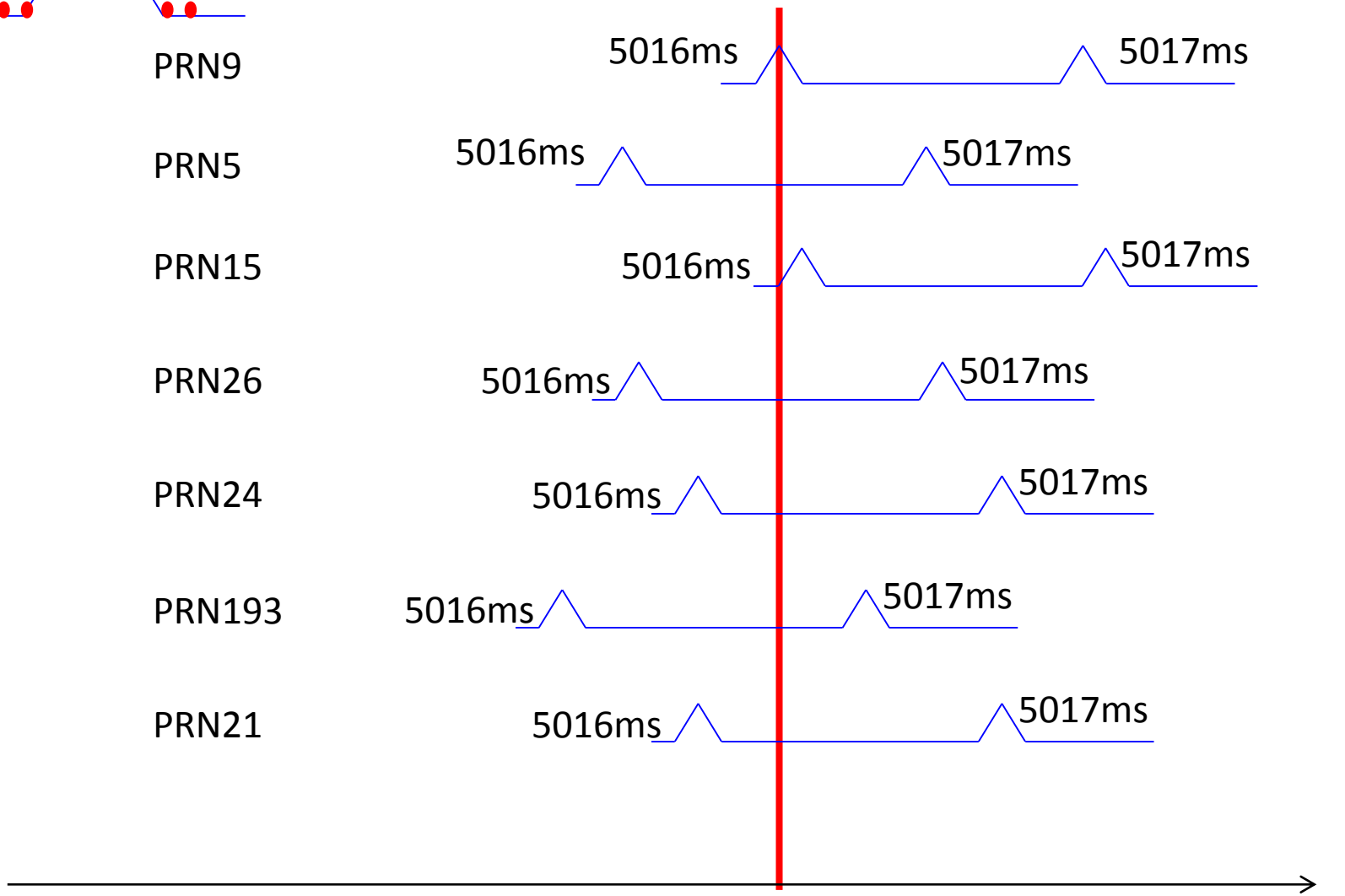
205494690/40960=5016.960…

まだ1ms以内の正確な到達時刻の差はわからない
(.960……はPRN9のコード位相が39317のため)

正確な時刻差を計算

205494690のときに各衛星のずれはいくらか？





擬似距離（m）＝（67ms+diff[prn]）×光速×0.001

受信GPS時刻＝TOW+…

tcode変数について

```
while(value < (blksize-1)*codePhaseStep+remCodePhase[prn]){
        tcode[i++] = (remCodePhase[prn])+codePhaseStep*i;
        value = tcode[i-1];
}
remCodePhase[prn] = (tcode[int(blksize)-1] + codePhaseStep) -
    1023.0;
```

- 上記のtcode値をチェックするとすぐにわかるが、この値はコード周波数に応じた1サンプリングで進ませるコード量と前回の1023に達しなかった余り（remCodePhase）に関連している。tcode値は各衛星が1サンプリングで進ませる量を決定しており、常に連続した流れになっている点でも重要な量である（衛星側の時計を監視している）

L1-C/Aの場合1023チップ

- 例えば、さきほどの5016ms時点での各衛星のtcode値の1ms間の最後の値は以下の通り

衛星	PRN9	PRN5	PRN15	PRN26	PRN24	PRN193	PRN21
最後の値	1022.994	1022.986	1022.993	1022.984	1022.988	1022.989	1022.983

- 1サンプリングでのSTEPが1023/40960=0.024975..で、ドップラ周波数があるため必ず端数が最後に残る。よって、観測値を出す時点での各衛星のバイアスを考慮している。tcodeそのものはバイアスを考慮した結果になっているため、この数値も利用できるはずだが、まだ実装していない

受信機のクロックに基づいて観測データ算出タイミングを決定

- 今回実装している観測値を出すタイミングは、受信機のサンプリング周波数に依存している。つまり、受信機クロックに依存している
- 200ms毎に基準時刻を出す際、200×40960分進ませていることからもわかる（衛星の1ms毎の進ませるサンプルは40960とは限らない）
- 最終的に測位演算や二重位相差処理で時計誤差はなくなるため、大事なことは、このタイミングに忠実に全ての衛星の観測値を同時に出力することにあるように思う

calculatePseudoranges_test.cpp

- ここでは、前述で求めた擬似距離の元となる量とドップラの元となる量より、擬似距離とドップラ周波数を算出している
- 必要に応じて、搬送波位相もtracking関数内で求める。搬送波位相はドップラ周波数の積分値になる

速度計算(教科書抜粋)

6.2 擬似距離変化率から求める位置と速度

6.2.1 速度の推定

衛星とユーザの相対運動は，衛星信号の観測周波数の変化となる．ドップラシフトは，GPS 受信機の搬送波追尾ループ（CTL）で常に測定される［12 章］．衛星の速度が得られた場合，ドップラシフトはユーザ速度の推定に使われる．ドップラシフトまたはこれと同等な距離変化率［1.3.2 節］は，衛星の視線ベクトルの相対速度ベクトルの射影として記述できる．しかし，測定値は受信機のクロックバイアス変化率（すなわち，周波数オフセット）によってバイアスを持ち，実際に測定されるのは擬似距離変化率である．搬送波位相の測定値から得られるデルタ擬似距離変化率は，平均擬似距離変化率，または時間領域で衛星に対するユーザの視線ベクトルに比例する．擬似距離変化率のモデルは式（6.1）の微分から得られる．

$$\begin{aligned}\dot{\rho}^{(k)} &= \dot{r}^{(k)} + (\dot{b} - \dot{b}^{(k)}) + \dot{I}^{(k)} + \dot{T}^{(k)} \\ &= (\mathbf{v}^{(k)} - \mathbf{v}) \cdot \mathbf{l}^{(k)} + \dot{b} + \varepsilon_{\dot{\rho}}^{(k)},\end{aligned} \tag{6.33}$$

ここで，$\mathbf{v}^{(k)}$ は衛星の放送メッセージからわかる衛星の速度ベクトルであり［練習問題 4-12］，$\mathbf{v}$ は推定されるユーザの速度ベクトルである．$\mathbf{v}^{(k)}$ と $\mathbf{v}$ は，ECEF 座標で表現される．ユーザからみた衛星の視線単位ベクトルの $\mathbf{l}^{(k)}$ は，ユーザ位置の推定値から決定される．$\dot{b}$ と $\dot{b}^{(k)}$ は受信機と衛星のクロック変化率（m/s）であり，$\varepsilon_{\dot{\rho}}^{(k)}$ は衛星のクロックと電離層と対流圏の測定値の変換に起因する結合された誤差を意味する．留意すべきは，地球上の物体の速度は ECEF 直交座標でゼロであるということである．

教科書の続き

1990 年代に式（6.33）の主な誤差源は SA による，衛星クロック周波数のゆれ（dithering）$\dot{b}^{(k)}$ であった．現在 SA はなくなり，この項は無視できる．電離層遅延，対流圏遅延，マルチパスの測定区間での変化による誤差は，ほとんどの場合小さい．中程度のユーザ速度で，速度推定の主誤差源は，予測衛星位置と速度 $\mathbf{v}^{(k)}$ の誤差である．予測衛星位置（特に移動方向に沿った成分と直交する成分）は視線方向誤差に現れ，衛星速度ベクトルの内積を拡大する．

問題はユーザのダイナミクスが過大な場合に生じる．デルタレンジは，時間間隔の間の平均速度のみを与える．高い加速度やジャークは，明らかに問題となる．そのようなダイナミクスは，受信機クロックの位相雑音に起因してさらなる誤差源となる［4.2.3 節］．これらの理由により DoD は速度推定性能の仕様を避けている（古い文書［JPO（1991）］によれば，一定速度シナリオに基づく PPS 性能仕様は，0.1 m/s(rms，全方向)，0.2 m/s（2 drms）である．これらは次に示すように過度に保守的である．）

式（6.33）はユーザの速度成分に関して線形であり，

$$(\dot{\rho}^{(k)} - \mathbf{v}^{(k)} \cdot \mathbf{l}^{(k)}) = -\mathbf{l}^{(k)} \cdot \mathbf{v} + \dot{b} + \varepsilon_{\dot{\rho}}^{(k)}$$

と再記述できる．$(\dot{\rho}^{(k)} - \mathbf{v}^{(k)} \cdot \mathbf{l}^{(k)})$ を $\tilde{\dot{\rho}}^{(k)}$ と表記すれば，K 機の衛星からの測定値の結合集合は，行列表記で一組の方程式として，

$$\tilde{\dot{\boldsymbol{\rho}}} = \mathbf{G}\begin{bmatrix}\mathbf{v}\\ \dot{b}\end{bmatrix} + \tilde{\boldsymbol{\varepsilon}}_{\dot{\rho}}, \tag{6.34}$$

と簡潔に表現できる．ここで，行列 **G** はユーザと衛星の幾何を特徴づけており，式（6.10）ですでに定義されている．擬似距離変化率に基づくユーザ速度の推定問題は，式（6.9）の擬似距離からのユーザ位置の推定問題と同一であることである．また，前述した様に最小二乗解と DOP パラメータを定義でき，推定値の rms 誤差と関連づけることもできる．

擬似距離変化率から求めた速度推定値を図 6.8 に示す．25 km 離れた固定アンテナの受信機ペアで 1 分間隔で 1 日擬似距離変化率を記録した．図 6.8（a）は，一つの受信機の水平速度推定値の散布図を示す．水平速度誤差は，50%で 0.03 m/s，95%で 0.08 m/s である．前に議論した予測衛星位置と速度誤差の効果は，相対速度計算を低減し劇的な改善をもたらす．二つの受信機間の相対速度の推定値を図 6.8（b）に示す．

速度の垂直成分の推定値は水平成分ほど良くない．この理由は，本節の最初に述べた従来処理による水平・垂直位置推定の精度の議論と同様である．

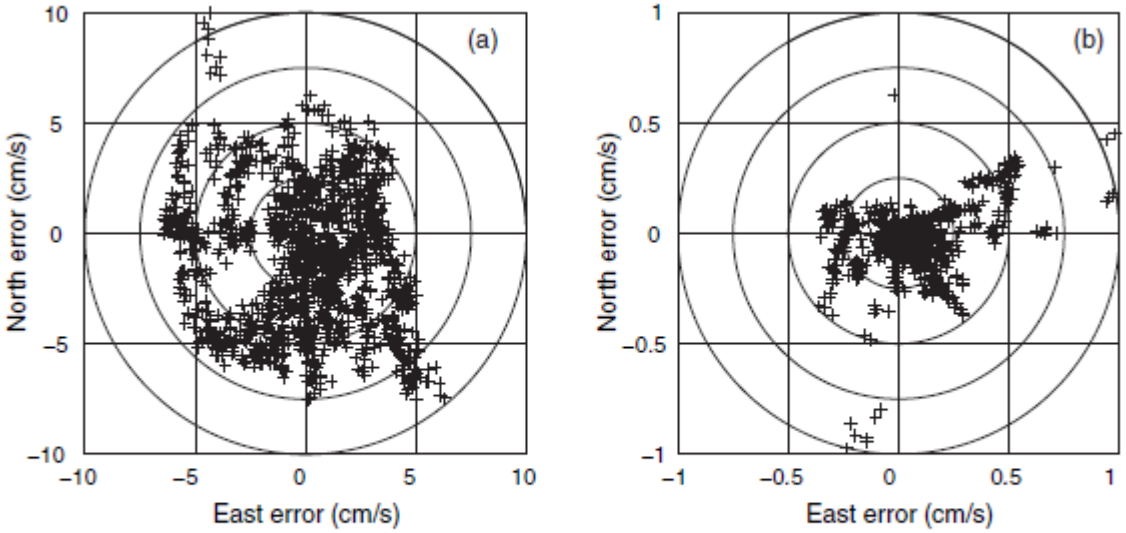


図 6.8　25 km 程離れた固定アンテナ間で 1 分間隔で 24 時間測定した擬似距離レート（ドップラ周波数）から計算した GPS 速度推定値の水平誤差の散布図．（a）速度推定値の誤差，（b）相対速度推定値の誤差．（1 m/s ≈ 2 knots）

速度計算を行っている関数

- satpos_prev.cpp
- satpos_next.cpp
- least_square.cpp

6.34式の行列Gは基本的に単独測位で利用しているユーザ衛星間の単位ベクトルに他ならない。よってそのまま利用できる。あとは、ユーザのベクトルvを、擬似距離変化率（または搬送波位相変化率、ドップラ周波数）より求める。

実際に擬似距離変化率を用いた場合と搬送波位相の変化率では精度に大きな差がある。搬送波位相はドップラ周波数の積分値であるともいえる。本ソースではドップラ周波数を用いて速度ベクトルを計算した

補足事項

I,Qサンプリング

- residual = trigarg[i] - pi2*int(trigarg[i]/(pi2));
- index = int(residual*100);
- j = int(rawSignal[i]);

- //depends on receiver (which number is used ?)
- switch(j){
- case 3:qBaseband=Table_cos3[index]; iBaseband=Table_sin3[index]; break;
- case 1:qBaseband=Table_cos[index]; iBaseband=Table_sin[index]; break;
- case -1:qBaseband=Table_cos2[index]; iBaseband=Table_sin2[index]; break;
- case -3:qBaseband=Table_cos4[index]; iBaseband=Table_sin4[index]; break;
- case 0:qBaseband=0; iBaseband=0; break;
- }

上記の相関処理部のようにI相のみのIFデータでは、I相からQ相の値をcosをかけて人工的に求めている。もし、Q相のデータがI,Qの順番で存在する場合、上記のqBasebandのところにそのままほうりこむ

信号強度算出方法

$$S/N = 20\log_{10}(IP/T/FEのノイズレベル)$$

$$C/N_0 = S/N + 27.0$$

今回IPの値は20msで平均化。よってT=20
FEのノイズレベルは無相関での相関値の1シグマとした

- Fraunhofer　278

PLI（Phase Locked Indicator）の算出方法　2通り

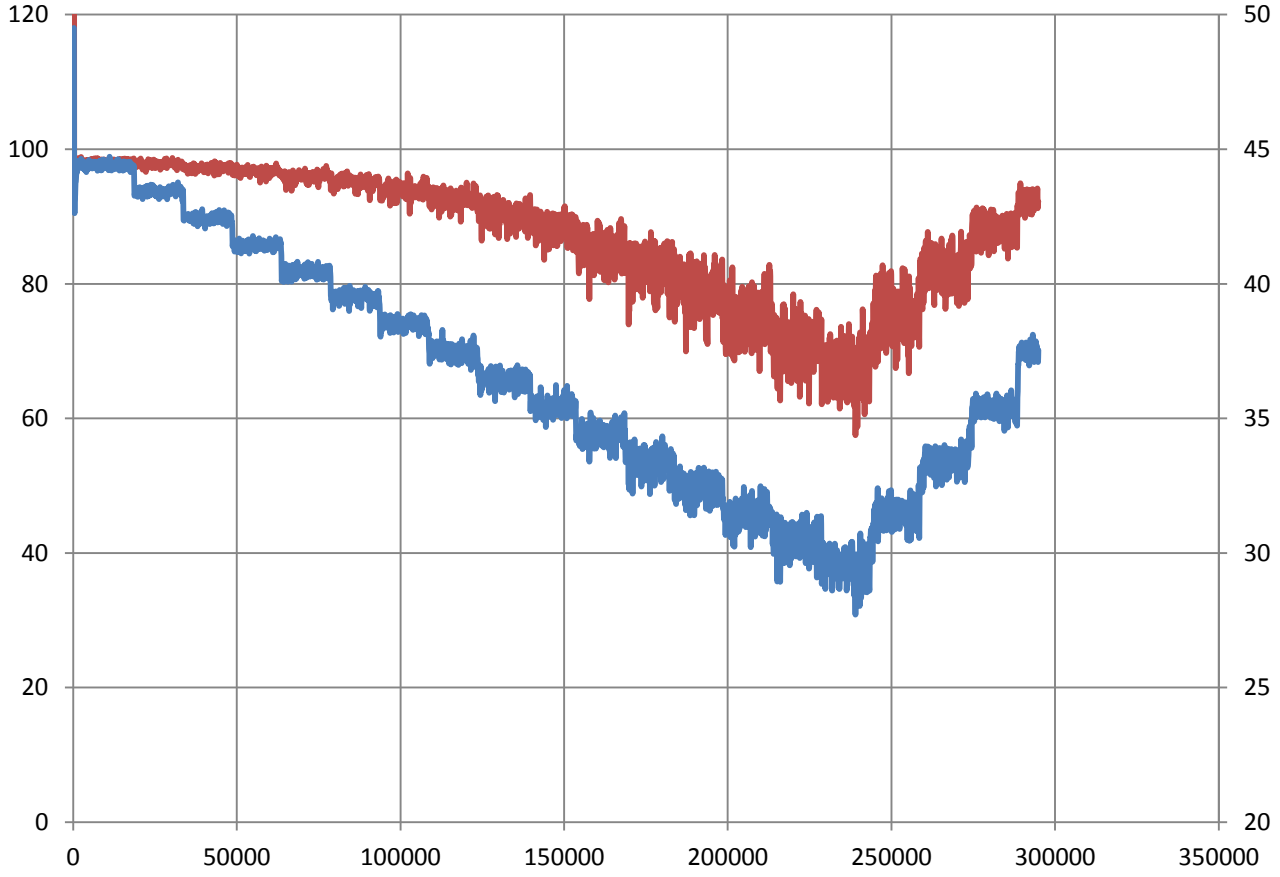
$$PLI_1 = \frac{IP^2 - QP^2}{IP^2 + QP^2}$$

$$PLI_2 = \frac{IP^2}{IP^2 + QP^2}$$

PLLの位相ロックはずれを約15度と設定すると、上記で0.85程度の値

DLL:1Hz / PLL:20Hz / 1msの追尾結果

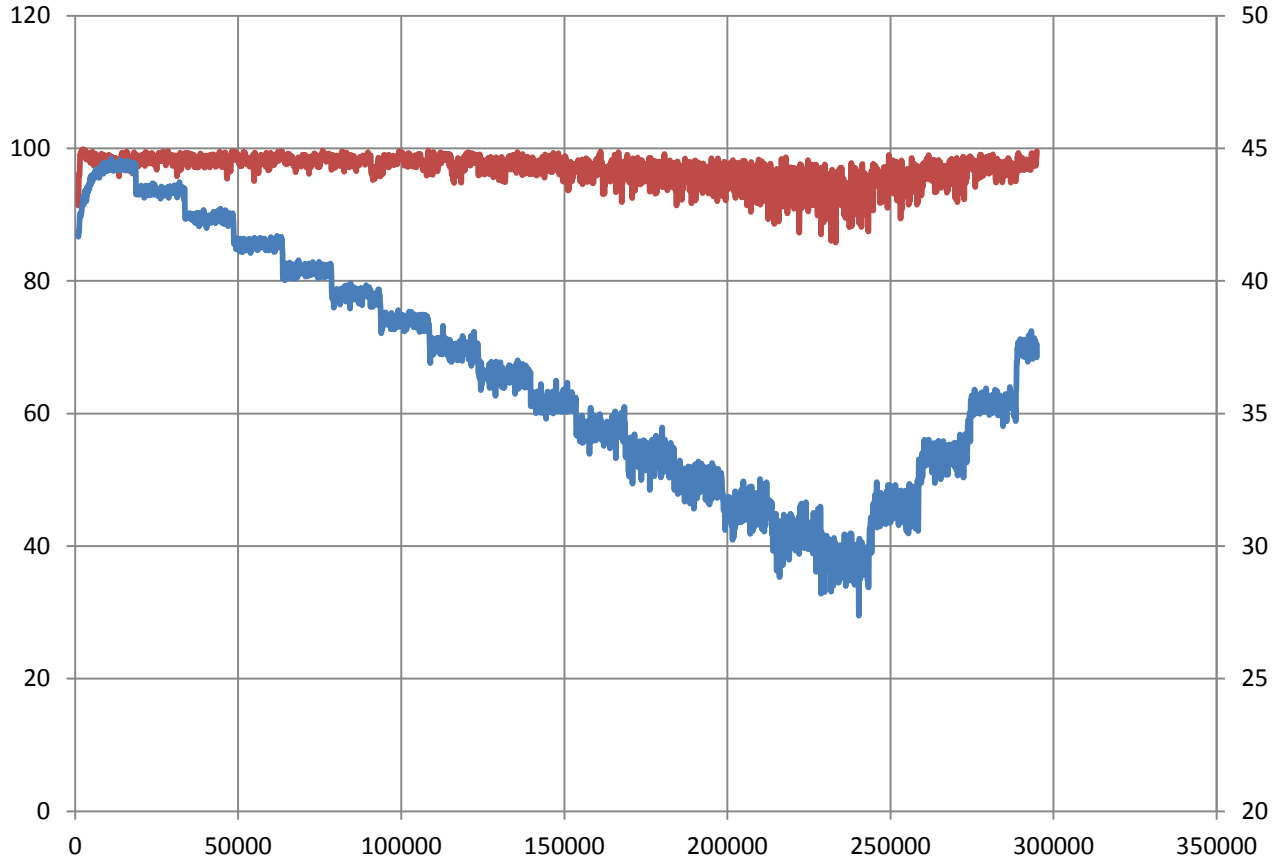
赤がPLI



青が信号強度

DLL:1Hz / PLL:5Hz / 10msの追尾結果

赤がPLI



青が信号強度

ご静聴ありがとうございました