理解するための GPS測位計算プログラム入門

（その2）GPS衛星の軌道計算のはなし

独立行政法人　電子航法研究所　福島　荘之介

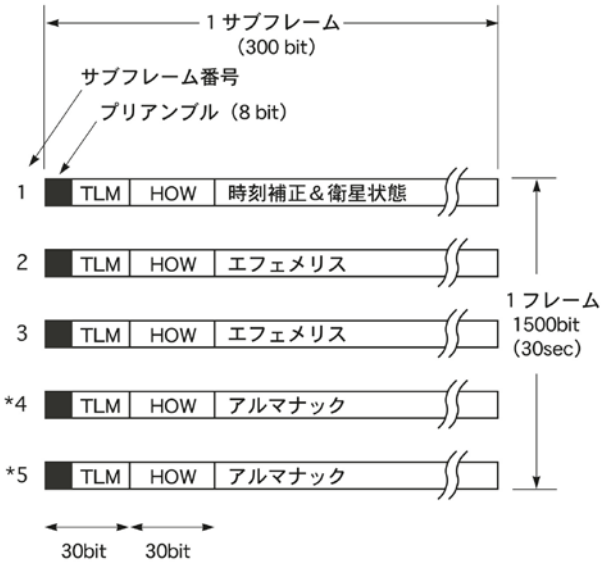
1. はじめに

前回（その1）は，WGS84 座標系を説明し，ECEF（地球中心・地球固定）直交座標と測地座標（経緯度と楕円体高）の相互変換について，また地平座標への変換について説明した。今回は，GPS 衛星から送信される航法メッセージから各衛星の軌道パラメータを抽出し，ある時刻での衛星位置を計算し，ECEF 直交座標で表すことを目的とする。紹介するプログラムではノバテル社の GPS 受信機（RT20）で実際に受信したエフェメリスデータを使用し，受信機が内部で計算する衛星位置とどの程度一致するか比較を試みる。この結果，紹介するプログラムの計算値と受信機が計算する衛星位置は 2cm 以下の差で一致した。

2. 航法メッセージに含まれる軌道パラメータ

GPS 衛星から送信される信号は ICD（Interface Control Document）－GPS[1.7]と呼ばれる文書で規定されており，GPS 受信機はこの文書を基本に設計される。衛星から送信される航法メッセージは，50bps であり1フレームが 1,500bit ある。従って，1 フレームを受信するには 30 秒かかることになる。この1フレームはさらに，5 つのサブフレームから構成されている（1サブフレームは 6 秒，300bit）。このうちサブフレーム1は，各衛星の時計の補正値，健康状態などであり，サブフレーム2，3は，各衛星の軌道パラメータを含むエフェメリス（放送暦）である。サブフレーム4，5は全衛星の概略の軌道情報（アルマナック）や電離層遅延補正係数であり，サブフレーム1～3とは異なって 25 フレーム（12.5 分）受信して初めて全情報が完結する（**図2．1**）。

ノバテル社の GPS 受信機（RT20）には，**GPSolution** というソフトウェア（Windows で動作）



TLM：テリメトリ・ワード（最初の8 bitは同期用プリアンブル）
HOW：ハンドオーバ・ワード（最初の17 bitは週初めからの時刻）
＊：サブフレーム4,5はフレーム毎に内容が変わり，25フレーム（12.5分）で一巡する．これをスーパーフレームと呼ぶ．

図２．１　航法メッセージの構造

が付属しており，受信機をパーソナルコンピュータのシリアルポート（RS232C）に接続し，測位結果の測地座標（緯度，経度，楕円体高），衛星位置などを画面上に表示したり，50 種を超えるフォーマットのデータをファイルに記録することができる。このフォーマットの1つ **REPA**（Raw Ephemeris）は，受信した生のエフェメリスを出力し，サブフレーム1，2，3（バイナリー）を 16 進数表示する。1 サブフレームは 10 ワードで構成され，1 ワードは 6bit のパリティを含む 30bit から成る。**REPA** は，この 1 サブフレーム：10 ワード（300bit）からパリティを除いた 240bit（60 文字）を出力する（**図2．2**）。この内容は，ICDで定義されており，**図2．3～図2．5**による。

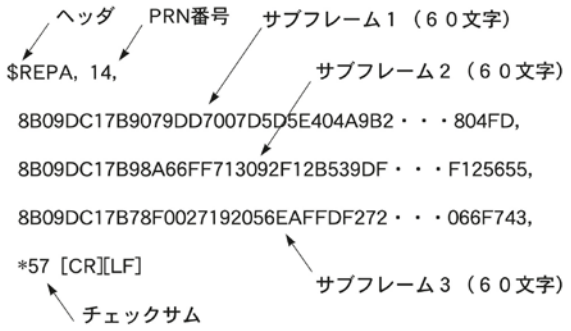


図２．２　ＲＥＰＡのフォーマット

3. 軌道パラメータを抽出するプログラム

REPA から各パラメータを抽出するソースプログラム（get_eph.c）の一部を**リスト２．１**に示す。サンプルデータとしては，電子航法研究所（本所：調布市）で約１日間取得した**表２．１**のデータから，**REPA** だけを抜き出したもの（repa.dat）を使う。main()では，まず実行コマンドの引数で与えるデータファイルをオープン（file_open()）する。次に，decode_repa()で，ファイルの先頭 1 行の REPA データから，衛星番号(PRN)と各サブフレームを読み込んで，デコードする（decode_eph()）。結果は check_print() で表示する。decode_eph() では，get_a_f0()に始まる関数群でサブフレーム内の各パラメータを抽出する。結果は def02.h で定義する ephemeris 型の構造体に入れる。この構造体は，PRN を添字とする配列として定義する。

get_a_f0()は，衛星時計の補正値である af0 を抽出することを目的とする（**図２．３**右下参照）。関数は，サブフレーム1（s1）の 10 ワード目から 6 文字分（24bit）切り取ってヘキサの文字を整数変換し，上位 22bit マスクして，2bit 右シフトし，**図２．６**にある LSB の値を乗じて単位のある値（物理量という）に変換する。このとき，注意するのは，af0 の値が MSB を符号桁とする2の補数になっていることである。従って，ここでは整数変換した値（alph0_d）が 0x1fffff×2E(-31)より大きいとき，0x400000×2E(-31)から自身を減じることによって，負の数を表現し

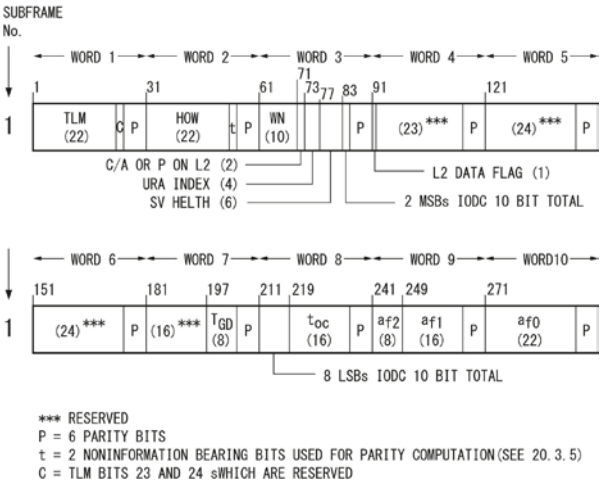


図２．３　航法メッセージ（サブフレーム１）

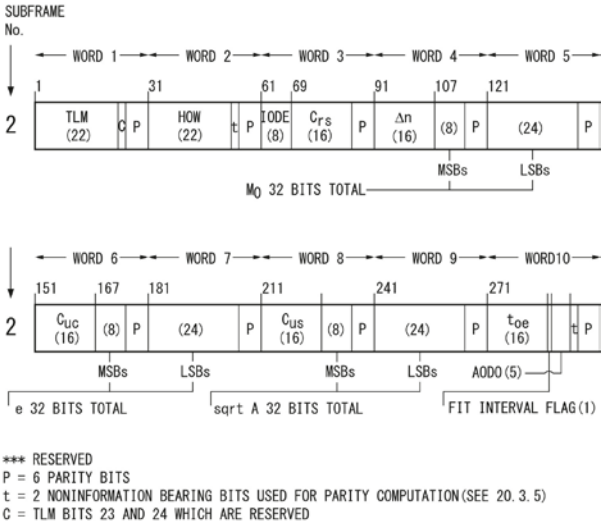


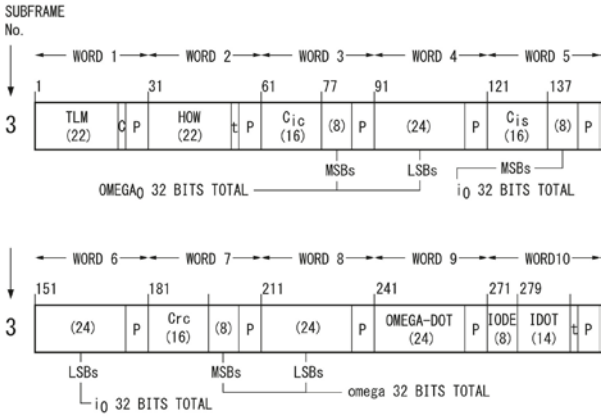
図２．４　航法メッセージ（サブフレーム２）

ている。これは**図２．６**の af0 のビット数（Num. of Bits）22 の右上にある（＊）の注釈に記述があり，他のパラメータも同様である（e，$A^{1/2}$ など一部は異なる）。

プログラムは，入力データ（repa.dat）の1行目だけを読み込み，全軌道パラメータをデコードして，check_print()で表示する。しかし，そのほとんどはコメントアウトし，実際表示するのは

（PRN　i0　$A^{1/2}$　Ω_0）

だけとしている。これだけでもおよその動作チェック



*** RESERVED
P = 6 PARITY BITS
t = 2 NONINFORMATION BEARING BITS USED FOR PARITY COMPUTATION(SEE 20.3.5)
C = TLM BITS 23 AND 24 WHICH ARE RESERVED

図２．５　航法メッセージ（サブフレーム３）

[ICD-GPS200C Table 20-I. Subframe 1 Parameters]

Parameter	No. of Bits**	Scale Factor (LSB)	Effective Range***	Units
Code on L2	2	1		discretes
Week No.	10	1		week
L2 P data flag	1	1		discretes
SV accuracy	4			(see text)
SV health	6	1		discretes
T_{GD}	8*	2^{-31}		seconds
IODC	10			(see text)
t_{oc}	16	2^{4}	604,784	seconds
a_{f2}	8*	2^{-55}		sec/sec^2
a_{f1}	16*	2^{-43}		sec/sec
a_{f0}	22*	2^{-31}		seconds

* Parameters so indicated shall be two's complement, with the sign bit(+ or -) occupying the MSB;

** See Figure 20-1 for complete bit allocation in subframe;

*** Unless otherwise indicated in this column, effective range is the maximum range attainable with indicated bit allocation and scale factor.

図２．６　サブフレーム１の緒元

[ICD-GPS200C Table 20-III. Ephemeris Parameters]

Parameter	No. of Bits**	Scale Factor (LSB)	Effective Range***	Units
IODE	8			(see text)
Crs	16*	2^{-5}		meters
Δn	16*	2^{-43}		semi-circles/sec
M0	32*	2^{-31}		semi-circles
Cuc	16*	2^{-29}		radians
e	32	2^{-33}	0.03	dimensionless
Cus	16*	2^{-29}		radians
$(A)^{1/2}$	32	2^{-19}		meters$^{1/2}$
toe	16	2^{4}	604,784	seconds
Cic	16*	2^{-29}		radians
$(OMEGA)_0$	32*	2^{-31}		semi-circles
Cis	16*	2^{-29}		radians
i_0	32*	2^{-31}		semi-circles
Crc	16*	2^{-5}		meters
omega	32*	2^{-31}		semi-circles
OMEGADOT	24*	2^{-43}		semi-circles/sec
IDOT	14*	2^{-43}		semi-circles/sec

* Parameters so indicated shall be two's complement, with the sign bit(+ or -) occupying the MSB;

** See Figure 20-1 for complete bit allocation in subframe;

*** Unless otherwise indicated in this column, effective range is the maximum range attainable with indicated bit allocation and scale factor.

図２．７　エフェメリスの緒元

はできる。このプログラムの計算結果が正しいかどうかは，受信機に付属する **Convert** というソフトの出力と比較することによって確かめられた。このソフトは，ノバテル社のフォーマットのデータを RINEX ファイルに変換する。RINEX（Receiver Independent Exchange Format）とは，受信機固有のフォーマットに依存しない共通データ形式のことであり，観測（observation）ファイル（o-file），航法（navigation）ファイル（n-file）などテキスト形式のファイルからなる。このうち，航法ファイル（ファイルの

拡張子が 02n となっている。02 は西暦の下2桁）には各衛星の軌道パラメータが格納される。フォーマット内容を記述した文書[2.1]はインターネットで配布されている。注意としては，例えば Ω_0 の単位は，**semi-circles**（**図２．７**）であるが，RINEX では **radian** を使っている。**semi-circles** から **radian** への変換は π を乗じるだけであり，紹介したプログラム（get_eph.c）でも後で扱いやすいように単位を **radian** に統一した。

リスト2．1は少し改良すると全てのデータを読み込める。これは，decode_repa()の中で入力データを1行に制限している個所，

```
if (i==1) break;
```

を，例えば `i==50` として 50 行まで読み込むことである。各衛星の軌道パラメータは，2時間に1回更新されるので，ephemeris 型の構造体に入る結果は最新のものとなる（PRN が添字のため）。

4. 軌道パラメータの意味とは

さて，**リスト2．1**のプログラムで得られた軌道パラメータはどういう意味を持っているのだろうか。ケプラーの軌道要素など既に詳しくご存知の方も多いと思うが，ここでは少し噛み砕いた説明をしてみる。まず，**図2．8**のような軌道面を考え，前回説明した ECEF 直交座標にどう置くか（原点を通る）を考える。（これは例えば「下じき」を机の角にくっつけて，どう固定するかと同じことになる）軌道面を固定するには，赤道面（X-Y 平面）からの角度 **i（軌道傾斜角）**と軌道面と赤道面の交線の X 軸からの角度 **Ω（昇交点赤経）**の2つを決めればよい。次に，（別の紙に）楕円を描く。楕円の定義とは「2定点から距離の和が一定となるような点の軌跡」であり，この形状は **a（長半径）**と **e（離心率）**で決める。2定点は楕円の焦点と呼ばれ，**e** は楕円の中心から焦点のずれを示す。この（別の紙に書いた）楕円を軌道面（下じき）の上に重ねるとき，1つの焦点を直交座標の原点（机の角）に置き，楕円上の焦点から一番近い点（近地点）の方向を軌道面と赤道面の交線から角度 **ω（近地点引数）**にする。これだけで衛星の軌道は決定できる。衛星はこの楕円軌道上を運動する。

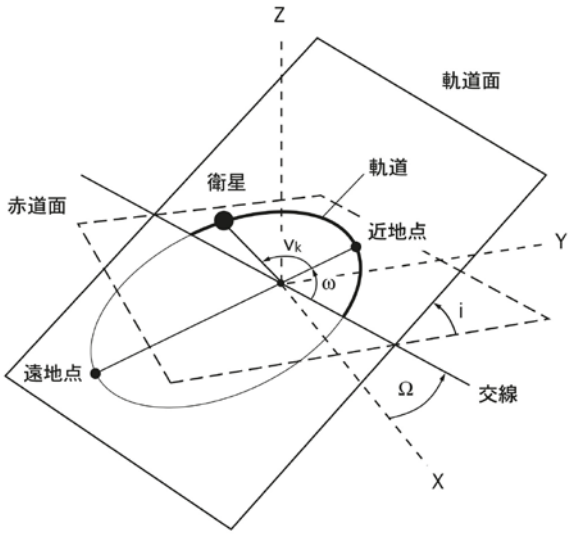


図２．８　ＧＰＳ衛星の軌道要素（Ω，ｉ，ω）

5. 衛星位置を求める

次に，「ある時刻に衛星は楕円軌道上のどこにいるのか？」を求める。このためには，ICDに書いてある関係式（**図2．9～図2．11**）を使って**図2．8**の **v_k（真近点離角）**を求める必要がある。軌道パラメータにはある基準の時刻（元期という）があり，衛星が楕円上のどこにいるかを示す量として **M_0（元期の平均近点角）**と軌道の**元期 t_{oe}** が含まれる。ここで，楕円軌道の中心位置（2つの焦点の中点）を中心とする楕円の長半径に等しい半径の円軌道を一定の角速度 **n（平均運動）**で運動している仮想の衛星を考えると，エフェメリスの元期からの時刻 t での平均近点角 M_k は，

$$M_k = n(t - t_{oe}) + M_0$$

となる。M_k をケプラー方程式

$$M_k = E_k - e\sin E_k$$

に代入すれば E_k（離真近点角）が求まり，幾何関係により v_k が求められる。

v_k が求まれば，後は前回説明した軸の回転で

ECEF 直交座標の値に変換することができる。ただし，その前に，摂動力（地球の重力乱れ，月・太陽の引力，潮汐など）を考慮した補正（エフェメリスに補正項が含まれる）を行う必要がある（**図2．9～図2．10**）。この補正は，**4．**で述べた **i** と**Ω**にもおこなわれており，実際，**3．**でデコードした i0（元期の軌道傾斜角）とΩ_0（週初め（tow=0）の昇交点赤経）を補正して計算される。軌道面は天球に対して固定で，ECEF に対して回転しているため，この計算には地球の自転速度Ω_e-DOT とその補正値Ω-DOT が関係する。

6. 軌道パラメータから衛星位置を求めるプログラム

リスト2．1で航法メッセージから軌道パラメータを抽出した。今度は，この軌道パラメータから衛星位置を求めるプログラムを説明する。プログラムは少し長くなり，内容も複雑になるが，基本的には**4．，5．**で述べた ICD の計算をしているだけである。メインとするプログラム名は eph2ecef.c（**リスト2．2**）とする。main()の中では，まず前回同様入力データファイルをオープン（file_open1()）する。ただし今度は，抜き出した **REPA** だけ（repa.dat）ではなく，**表1**の全データフォーマットを含むファイルを扱う（kmusen020425.gps）[1]。main()では次に read_file()関数を実行している。

この関数は，データファイルのヘッダを判定（just_look_head()）し，**REPA** だったら前回と同様に decode_eph()を実行して ephemeris 型の構造体に最新のエフェメリスを読み込む。また後で結果が正しいかどうか確かめるために，受信機内で計算している ECEF 直交座標の衛星位置を含む **SVDA**（SV Position in ECEF XYZ Coordinates with Corrections）も読み込んでおく。さらに，後で説明するが衛星位置を正確に求めるためには擬似距離が必要となるので，これを含む **RGEA**（Channel Range Measurements）も読み込んでおく

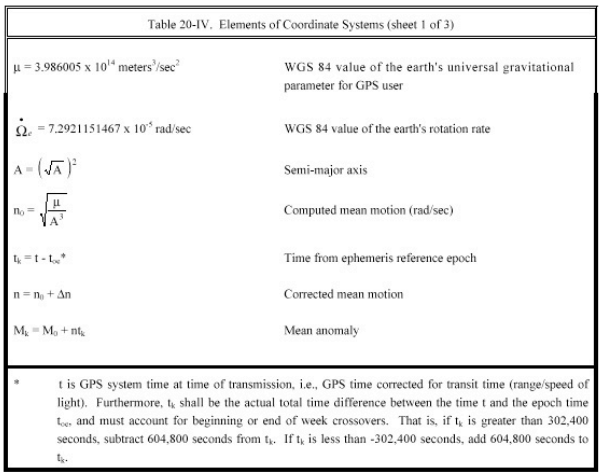


Table 20-IV. Elements of Coordinate Systems (sheet 1 of 3)

$\mu = 3.986005 \times 10^{14}$ meters^3/sec^2	WGS 84 value of the earth's universal gravitational parameter for GPS user
$\dot{\Omega}_e = 7.2921151467 \times 10^{-5}$ rad/sec	WGS 84 value of the earth's rotation rate
$A = (\sqrt{A})^2$	Semi-major axis
$n_0 = \sqrt{\mu / A^3}$	Computed mean motion (rad/sec)
$t_k = t - t_{oe}$*	Time from ephemeris reference epoch
$n = n_0 + \Delta n$	Corrected mean motion
$M_k = M_0 + nt_k$	Mean anomaly

* t is GPS system time at time of transmission, i.e., GPS time corrected for transit time (range/speed of light). Furthermore, t_k shall be the actual total time difference between the time t and the epoch time t_{oe}, and must account for beginning or end of week crossovers. That is, if t_k is greater than 302,400 seconds, subtract 604,800 seconds from t_k. If t_k is less than -302,400 seconds, add 604,800 seconds to t_k.

図２．９　軌道要素の関係式（１／３）

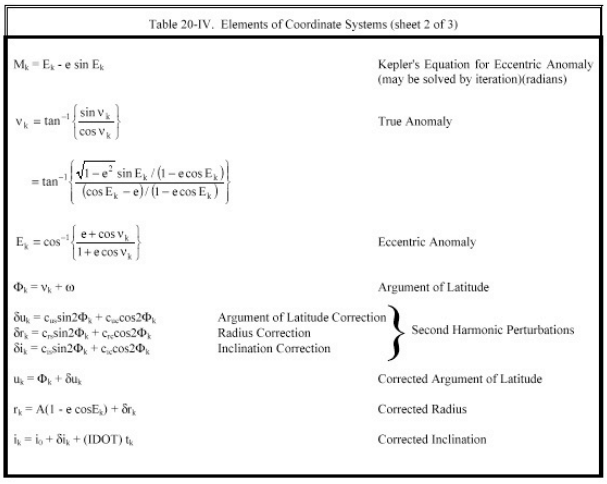


Table 20-IV. Elements of Coordinate Systems (sheet 2 of 3)

$M_k = E_k - e \sin E_k$		Kepler's Equation for Eccentric Anomaly (may be solved by iteration)(radians)
$\nu_k = \tan^{-1}\left\{\frac{\sin \nu_k}{\cos \nu_k}\right\}$		True Anomaly
$= \tan^{-1}\left\{\frac{\sqrt{1-e^2}\sin E_k/(1-e\cos E_k)}{(\cos E_k - e)/(1-e\cos E_k)}\right\}$		
$E_k = \cos^{-1}\left\{\frac{e+\cos \nu_k}{1+e\cos \nu_k}\right\}$		Eccentric Anomaly
$\Phi_k = \nu_k + \omega$		Argument of Latitude
$\delta u_k = c_{us}\sin 2\Phi_k + c_{uc}\cos 2\Phi_k$	Argument of Latitude Correction	Second Harmonic Perturbations
$\delta r_k = c_{rs}\sin 2\Phi_k + c_{rc}\cos 2\Phi_k$	Radius Correction	
$\delta i_k = c_{is}\sin 2\Phi_k + c_{ic}\cos 2\Phi_k$	Inclination Correction	
$u_k = \Phi_k + \delta u_k$		Corrected Argument of Latitude
$r_k = A(1 - e\cos E_k) + \delta r_k$		Corrected Radius
$i_k = i_0 + \delta i_k + (\text{IDOT})\, t_k$		Corrected Inclination

図２．１０　軌道要素の関係式（２／３）

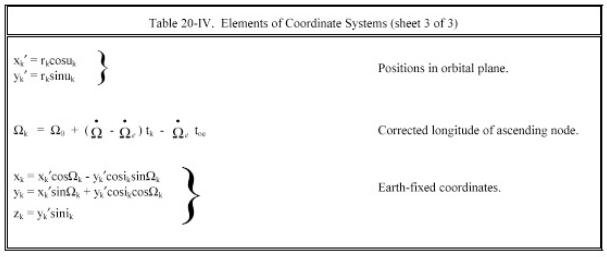


Table 20-IV. Elements of Coordinate Systems (sheet 3 of 3)

$x_k' = r_k \cos u_k$ $y_k' = r_k \sin u_k$	Positions in orbital plane.
$\Omega_k = \Omega_0 + (\dot{\Omega} - \dot{\Omega}_e)\, t_k - \dot{\Omega}_e\, t_{oe}$	Corrected longitude of ascending node.
$x_k = x_k' \cos\Omega_k - y_k' \cos i_k \sin\Omega_k$ $y_k = x_k' \sin\Omega_k + y_k' \cos i_k \cos\Omega_k$ $z_k = y_k' \sin i_k$	Earth-fixed coordinates.

図２．１１　軌道要素の関係式（３／３）

（SVDA, RGEA のデータ型は str_rt20.h の中で定義する）。ただし，ここで読み込む **SVDA** と **RGEA**

[1] ファイルサイズが 125MB と大きいため，インターネット上には圧縮した 41MB のファイル（拡張子が.gz），１時間データ（2MB）も置く．

はファイル中の全データではなく，同じ時刻の1組のデータだけとし，1時刻毎に全計算を終えて，次の1組のデータを読み込む処理に戻る。このやり方だと全データを読んでメモリが溢れるのを防ぐことができる（フィルター型プログラムと呼ばれる）。データが読み込まれたら，衛星位置の変換（関数 calc_sat_position()）を実行する。この関数は，calc_sat_position.c（リスト2．3）の中にあり，**SVDA** にある受信機内部の時刻 tow（time of week：週はじめからの通算秒，リセットされるのは日曜日の早朝，UTC 0 時 00 分 0 秒）と衛星の PRN 番号を関数 eph2xyz()に与え，ECEF 座標の衛星位置 pos（ベクトル）を計算する。

関数 eph2xyz()（**リスト2．3**）の中身は，**図2．9～図2．11**に示された ICD の計算そのものである。ケプラー方程式は，ニュートン法（newton()）を使って解を求めており，適当な初期値（ここでは 1.0）を与えて収束するまで，繰り返し計算している。この関数は一般的なもの（math_util.c に含む）で，アルゴリズムは数値計算の教科書で詳しく説明されている。

実行するには，eph2ecef.c，calc_sat_position.c の他，データ読み込みのための関数 read_data.c 以前作った get_eph.c, math_util.c を同時に make する。

7．GPS受信機の計算する衛星位置と一致しますか？

以上である時刻における GPS 衛星の ECEF 直交座標位置を計算できたはずである。しかし実際にノバテル受信機が計算する **SVDA** の位置とは数 km 以上も隔たりがある。この原因の1つは，時刻の扱いにある。**6．**の計算で使った **SVDA** の時刻は，GPS 衛星からの信号を受信した時刻であった。しかし，ICD を詳しく読むと**図2．9**の時刻 t は，衛星から信号が送信された時刻である（脚注*に説明がある）。そこで，calc_sat_position()の中で，eph2xyz()に与える時刻を，送信時刻に変更するため **RGEA** から読み込んだ擬似距離 pr を使い，送信時刻 time を

```
time = tow – pr /C
```

と求める。ここで C は光速であり，受信時刻から擬似距離の伝搬時間を差し引いている。これで ICD に書いてあることは全て実施した。しかしノバテル受信機の衛星位置とはまだ差がある。ただし，Z 方向の差だけは数メートルにまで縮まった。

そこで試行錯誤の結果，次の方法が有効であることがわかった。その1つは，擬似距離 pr に衛星の時計の誤差を補正することである。擬似距離は衛星と受信機間の真距離とは異なり，これ以外に受信機の時計オフセットと衛星の時計オフセットが含まれる。受信機のオフセットは測位してみないとわからないが，衛星のオフセットは，エフェメリスで放送される af0, af1, af2 から計算できる。これを使って擬似距離を補正し送信時刻をより正確に求める。すると Z 軸の衛星位置はノバテル受信機の値と 1mm 以下の差で一致した。

残るは X,Y 軸の差であるため，地球の回転に起因すると予想できる。そこでよくよく考えてみると，衛星位置というのは，信号の送信時刻で計算され，かつ信号が受信される時刻の ECEF 座標での位置である。つまり，今まで計算していたのは送信時刻の ECEF 座標での衛星位置であり，地球が回転する分だけ衛星位置が異なっていた。そこで，eph2xyz()の中で Ω_0 の補正をするときに，地球の回転速度 Ω_e(dot)を補正する部分で，t_{oe} を `pr/C` だけ進めて地球の回転を戻す。すると X,Y 軸の計算値とノバテル受信機の計算値は 2cm 以下で一致する。このプログラムの出力である計算結果とノバテル受信機の差は，**リスト2．3**の最終行で出力しており，この結果をグラフに出力すれば，差を確認できる。

8．おわりに

これで本稿の目的は達せられた。紹介したプログラムの計算結果とノバテル受信機の差は，Z 軸で 1mm 以下，X,Y 軸で 2cm 以下となった。これで実用上はまったく問題ないが，X,Y 軸に関してはまだ少し改良の余地があるかもしれない（もしこれに関して情報をいただけたら参考になります）。

私が知る限り，高精度の GPS 受信機で，衛星の ECEF 位置を出力するのはノバテル社の受信機だけで，好んで使っている理由の1つです。次回は，いよいよ測位計算プログラムを説明をします。「これもノバテル受信機の計算結果と一致するかどうか？」ご期待ください。

参考文献

[2.1] W. Gurtner, “RINEX: The Receiver Independent Exchange Format version 2.10,” インタネット文書（http://www.ngs.noaa.gov/CORS/rinex210.txt）

表2．1：取得データ履歴

取得場所： 電子航法研究所アンテナ試験塔上
（http://www.enri.go.jp/research/sisetu/2401j.htm）
日時： 2002.4.25 17:55～4.26 17:12(JST)
アンテナ位置：
35.68006587E, 139.56102778N, 110.656
データ項目(周期)：
ALMA(onchange), RALA(onchange),
REPA(onchange), SVDA(ontime 1.0),
RGEA(ontime 1.0), POSA(ontime 1.0)
マスク角： 5度

表2．2：プログラムの説明

1．プログラム名：eph2ecef.c
目的：航法データから取り出したエフェメリスパラメータから各衛星の ECEF 座標位置を計算し，その結果を SVDA にある ECEF 位置と比較。
動作：
main()：メイン
file_open1()：入力データファイルのオープン
read_file()：REPA, SVDA, REGEA の読み込み
・REPA を1行読み込む
decode_eph()：get_eph.c の関数
get_svd_one()：read_data.c の関数
get_rge_one()：read_data.c の関数
・同時刻であれば，
calc_sat_position()：calc_sat_position.c の関数
・EOF になるまで繰り返す

2．プログラム名：get_eph.c:
関数：バイナリーの航法データ（**REPA**）からエフェメリスパラメータを抽出して物理量に変換する．
decode_eph()：ビット列から物理量への変換
get_a_f0()：af0 の抽出（構造体 eph へ代入）
get_a_f1()：af1 の抽出（〃）
（以下，各パラメータに抽出の関数を準備）
check_print()：確認のための表示

3．プログラム名：read_data.c
目的：**SVDA**，**RGEA** データの読み込みのための関数群
関数：
get_word()：バッファ中の次のコンマまでの文字列を取り出し詰める
get_word_ast()：上記と同様だが，コンマまたは＊
get_svd_one()：**SVDA** データを構造体へ読み込む
get_rge_one()：**RGEA** データを構造体へ読み込む

4．プログラム名：calc_sat_position.c
関数：
eph2xyz()：エフェリスパラメータから ICD の計算（**図 2.9～図 2.11**）を行い，ECEF 直交座標位置を求める。ただし，伝搬時間分の地球の回転を考慮
calc_sat_position()：
・af0，af1，af2 で擬似距離を補正
・衛星から電波を発射した時刻を求める（補正擬似距離から求めた伝搬時間と受信時刻から）
eph2xyz()：上記関数
求めた衛星位置と SVDA 位置の差を求める

リスト2．1 **get_eph.c**（一部のみ）

```
/*---
  $REPA(NovAtel OEM3: Raw Ephemiris)からエフェメリスを取り出す
  2002.8.21: S.Fukushima(ENRI)
---*/
（中略）
FILE *infp;
char buf[FRM_MAX+1];
ephemeris    eph[33];
```

```
static char retword[3];
char *stop;

void file_open(int argc, char **argv)
{
  if(argc != 2){
  puts("usage: get_eph.out (in-file)\n");
  exit(0);
  }

  if((infp = fopen(argv[1], "r")) == NULL){
  puts("Error: NOT OPEN input file\n");
  exit(0);
  }
}

void get_a_f0(int prn, char *s1, char *s2, char *s3)
{
  char alph0[7];
  unsigned int alph0_d;

  strncpy(alph0, &s1[(10-1)*6], 6); /*--- s1, 10word, 6chara*/
  alph[6] = '\0';
  alph0_d = strtoul(alph0,&stop,16);
  eph[prn].clock_bias = ((alph0_d & 0xfffffc) >> 2) * pow(2.,-31.);
  if(eph[prn].clock_bias > (0x1fffff * pow(2.,-31))) /*-- check MSB */
  eph[prn].clock_bias -= 0x400000 * pow(2.,-31);
}
（中略）

void decode_eph(int prn,char *s1,char *s2,char *s3)
{
  int i, j;

  eph[prn].prn = prn;                  /*--- prn    */
  get_a_f0(prn, s1, s2, s3);           /*--- a_f0 */
  get_a_f1(prn, s1, s2, s3);           /*--- a_f1 */
  （中略）
  get_trans_msg(prn, s1, s2, s3);    /*--- Transmission time of message */
}

void decode_repa()
{
  int i = 0, j;
  char *ret;
  char *dumy;
  int prn;
  char sub1[61], sub2[61], sub3[61];

  while( feof(infp) == 0 ){
  ret = fgets(buf, FRM_MAX, infp);
  if(ret == 0) break;
  if(i == 1) break;      /* <--- 1 行だけ読む */
  sscanf(buf,"$REPA,%2d,%60c,%60c,%60c",&prn,&sub1,&sub2,&sub3);
  i++;

  decode_eph(prn, sub1, sub2, sub3);
  }
}

/*---
int main(int argc, char *argv[])
{
  file_open(argc, argv);
  decode_repa();
  fclose(infp);
}
 ---*/
/*---
26  9.712323734411E-01  5.153597038269E+03 -2.195012619292E+00
---*/
```

リスト2．2 eph2ecef.c（一部のみ）

```
/*---
 $REPA(NovAtel OEM3: Raw Ephemiris)から切り出した
 エフェメリス・パラメターを ECEF(XYZ)に変換する。をよぶ，メイン。
 2002.8.28: S.Fukushima(ENRI)
---*/
（中略）
char *just_look_head()
/*--- バッファに入っているデータのヘッダをみる */
{
 int i;

 for(i=0;i<FRM_MAX;i++){
 if(tbuf[i] == ','){
                buf_h[i] = '\0';
                return buf_h;
 }
 buf_h[i] = tbuf[i];
 }
}

void read_file()
{
 int i = 0, j;
 char *ret;
 char *dumy;
 int prn;
 char sub1[61], sub2[61], sub3[61];

 while( feof(infp) == 0 ){
 ret = fgets(tbuf, FRM_MAX, infp);
 if(ret == 0) break;
 /* if(i == 10000) break;   / <--- 1 行だけ読む */
 if(strcmp(just_look_head(),repat) == 0){
                sscanf(tbuf,"$REPA,%2d,%60c,%60c,%60c",&prn,&sub1,&sub2,&sub3);
                decode_eph(prn, sub1, sub2, sub3);
                continue;
  }else if(strcmp(just_look_head(),svdat) == 0){
                get_svd_one();
  }else if(strcmp(just_look_head(),rgeat) == 0){
                get_rge_one();
 }
 if(rge.seconds == svd.seconds){
                calc_sat_position();
 }
 i++;
 }
}

int main(int argc, char *argv[])
{
 file_open1(argc, argv);
 rge.seconds = 0.0;
 svd.seconds = 0.1;
 read_file();
 fclose(infp);
}
```

リスト2．3 calc_sat_position.c（一部のみ）

```
/*---
 $REPA(NovAtel OEM3: Raw Ephemiris)から切り出した
 エフェメリス・パラメターを ECEF(XYZ)に変換する。
 2002.8.28: S.Fukushima(ENRI)
---*/
（中略）
vector eph2xyz(ephemeris eph, double t, double p_range)
```

```
{
 （中略）
 sq_a = eph.square_root_of_semimajor_axis;
 a = sq_a * sq_a;
 n0 = sqrt(MU/(a*a*a));
 toe = eph.time_of_ephemerides;
 prn = eph.prn;
 tk = t - toe;    /* Time from ephemeris reference epoch */
 if (tk >  302400.0) tk -= 604800.0;/* Correct for possible end-of */
 if (tk < -302400.0) tk += 604800.0;/* GPS-week crossovers */

 dn = eph.mean_motion_difference;
 n   = n0 + dn;
 m0 = eph.mean_anomaly;
 mk = m0 + n * tk;

 /* Solve Kepler's equation for ek */
 e = eph.eccentricity;
 ek = newton(1.0, mk, e);

 /* True anomaly */
 cosvk = cos(ek) - e;
 sinvk = sqrt(1 - e * e) * sin(ek);
 vk = atan2(sinvk, cosvk);
 /* Argument of latitude */
 w = eph.argument_of_perigee;
 pk = vk + w;

 /* Argument of latitude correction  */
 cus = eph.sine_term_for_u_correction;
 cuc = eph.cosine_term_for_u_correction;
 duk = cus * sin(2 * pk) + cuc * cos(2 * pk);
 uk = pk + duk;
 /* Radius correction  */
 crs = eph.sine_term_for_r_correction;
 crc = eph.cosine_term_for_r_correction;
 drk = crc * cos(2 * pk) + crs * sin(2 * pk);
 rk = a * (1 - e * cos(ek)) + drk;

 /* Correction to inclination  */
 cis = eph.sine_term_for_i_correction;
 cic = eph.cosine_term_for_i_correction;
 dik = cic * cos(2 * pk) + cis * sin(2 * pk);
 i0 =          eph.inclination;
 idot =        eph.rate_of_inclination;
 ik = i0 + dik + idot * tk;

 /* Positions in orbital plane  */
 xkd = rk * cos(uk);
 ykd = rk * sin(uk);

 /* Corrected longitude of ascending node */
 omega0 = eph.nodes_longitude;
 omegadot = eph.rate_of_nodes_longitude;
 omegak = omega0 + (omegadot - OMEGADOTE) * tk -
 OMEGADOTE * (toe + p_range/(C));

 /* Earth fixed co-ordinates */
 xk = xkd * cos(omegak) - ykd * cos(ik) * sin(omegak);
 yk = xkd * sin(omegak) + ykd * cos(ik) * cos(omegak);
 zk = ykd * sin(ik);
 solv.n = 3;
 solv.a[0] = xk;
 solv.a[1] = yk;
 solv.a[2] = zk;
 solv.err = 0;

 return solv;
}

void calc_sat_position()
{
 int i, j, prn;
 double toe, af0, af1, af2, tow, time, pr;
 vector pos;

 tow = svd.seconds;
 for(j=0;j<svd.obs;j++){
  prn = svd.xyzc[j][0];
  af0 = eph[prn].clock_bias;
  af1 = eph[prn].drift;
  af2 = eph[prn].drift_rate;
  toe = eph[prn].time_of_ephemerides;

  for(i=0;i<rge.obs;i++){
    if(svd.xyzc[j][0] == rge.sat[i].prn){
      pr = rge.sat[i].psr + (af0 + (tow - toe) * af1
           + SQR(tow - toe) * af2) * C;
      time = tow - pr / C;
      pos = eph2xyz(eph[prn], time, pr);

      printf("%10.1f %3d %8.3f %8.3f %8.3f\n",
          svd.seconds, prn,  svd.xyzc[j][1]-pos.a[0],
          svd.xyzc[j][2]-pos.a[1], svd.xyzc[j][3]-pos.a[2]);
     }
    }
  }
}
```