緯度・経度・標高からの
ピクセル・ライン計算アルゴリズム説明書

１．与件

B, *L*　緯度・経度、あるいは
X, *Y*　ガウス－クリューゲル投影の座標値
H　楕円体高
　　標定要素（PrismGeo モジュールの関数を通して利用可能）

２．求件

p　ピクセル
l　ライン

３．利用可能な資源

```
Bundle_t PL_to_Bundle(double pixel, double line, Orient_t& Orient, double R_H);
                // ピクセル・ラインから、バンドル方程式を求める。
                // なお、R_H は、大気補正に使用するための概略の標高である。
XandR GetSatPosAlt(double line, Orient_t& Orient);
                // ラインから、衛星の位置と姿勢を求める。
Vector Bundle_on_LSR(Bundle_t Bundle, LSRt& LSR, double H);
                // バンドルが LSR の楕円体高 H の面と交わる点の ECR を求める。
LSRt::LSRt(double B0, double L0, double H0);
                // LSR 座標系の原点を緯度・経度・楕円体高で指定して、LSR を作成する。
Vector ECR_to_LSR(Vector& ECR)     // 地心直交座標 → LSR
Vector LSR_to_ECR(Vector& LSR)     // LSR → 地心直交座標
Vector BLH_to_ECR(double B, double L, double H);
                                   // 緯度・経度・楕円体高を ECR に変換する。
```

４．作成すべき関数（ツール）

4-1. Coordinates.h/ccp

```
class GK_t;
```

（機能） ガウス－クリューゲル投影の情報を格納するクラス。
（参考） 「座標計算アルゴリズム説明書」参照。

```
Vector GK_t::BL_to_GK(double B, double L);
```

（機能） 緯度・経度からガウス－クリューゲル座標を求める。
（手順） 「座標計算アルゴリズム説明書」参照。

```
void GK_t::GK_to_BL_(Vector XY, double& B, double& L);
```

（機能） ガウス－クリューゲル座標から緯度・経度を求める。
（手順） 「座標計算アルゴリズム説明書」参照。

```
void ECR_to_BLH (Vector& ECR, double& B, double& L, double& H);
```

（機能） ECR の座標値を緯度・経度・楕円体高に変換する。
（手順） 「座標計算アルゴリズム説明書」参照。

4-2. PrismGeo.h/ccp

```
void Init_PLH_to_BL(Orient_t& Orient);
```

（機能） PLH_to_BL()を初期値する。
（手順） １．Orient にある画像情報へのポインタを使用して、画像の中間（開始と終了の真中）のラインを決定する。
２．１で決定したラインを GetSatPosAlt()に入力し、衛星の位置の ECR 座標を求める。
３．２で求めた ECR 座標を ECR_to_BLH()に入力し、衛星の位置の緯度・経度を求める。
４．３で求めた緯度・経度を PLH_to_BL()で利用するため、静的に確保した概略の緯度・経度（L_0、B_0）に格納する。
（注意） PLH_to_BL()を呼び出す前にコールすること。異なる画像に対して PLH_to_BL()を呼び出す場合は、本関数を再度コールすること。

```
void PLH_to_BL(double pixel, double line, double H, Orient_t& Orient, double &B,
       double &L);
```

（機能） 楕円体高を仮定して、ピクセル・ラインに対応する緯度・経度を返す。
（手順） １．ピクセル･ラインと標高を PL_to_Bundle()に入力して､バンドル方程式を求める｡標高は、緯度・経度を BL_to_GeoidH()に入力しジオイド高を求め、これと楕円体高（H）を加算して求める。
２．概略の緯度・経度（L_0、B_0）、楕円体高 0 を LSRt::LSRt()に入力して、この点（楕円体上にある）の回りの LSR を作成する。
３．１で得られたバンドル方程式、２で得られた LSR、与えられた楕円体高 H を Bundle_on_LSR()に入力して、バンドルと LSR の交点の ECR 座標を求める。これを ECR_to_BLH()に入力して、交点の緯度・経度・楕円体高を求める。
４．３で求めた緯度・経度を新しい概略の緯度・経度（L_0、B_0）とする。このとき、緯度・経度の変化も求める。
５．与えられた楕円体高 H と３で求めた楕円体高の差、４で求めた緯度・経度の変化が、いずれも一定値（例えば、地上 1mm）以内であれば、３で求めた緯度・経度を出力し、終了する。それ以外の場合は、２に戻る。

```
void Init_BL_to_PLs(Orient_t& Orient, double H);
```

（機能） 楕円体高 H を使用して、緯度・経度等からピクセル・ラインを求める関数 BL_to_PL0()、BLH_to_PL0()、BLH_to_PL()を初期化する。
（手順） １．画像の 4 隅のピクセル・ラインに対して、ピクセル・ラインと与えられた楕円体高を PLH_to_BL()に入力し、4 隅の緯度・経度を求める。
２．１で求めた 4 組のピクセル・ラインと緯度・経度の関係を満足する緯度・経度からピクセル・ラインへの共１次の変換式（下式）の係数を決定する。

$$p = a_{10}B + a_{01}L + a_{11}BL + a_{00}$$
$$l = b_{10}B + b_{01}L + b_{11}BL + b_{00}$$

３．２で求めた係数を、静的領域に記録する。
（注意） BL_to_PL0()、BLH_to_PL0()、BLH_to_PL()を呼び出す前にコールすること。異なる画像に対してこれらの関数を呼び出す場合は、再度本関数をコールすること。

```
void BL_to_PL0(double B, double L, double& pixel, double& line);
```

（機能） 緯度・経度から、共 1 次式を用いて、概略のピクセル・ラインを求める。
（手順） １．Init_BL_to_PL0 ()で計算した係数を使用して、与えられた緯度・経度に対するピクセル・ラインを求める。

```
void BLH_to_PL0(double B, double L, double H, Orient_t& Orient, double& pixel,
       double& line, double pixel0, double line0);
```

（機能） 緯度・経度・楕円体高、概略のピクセル・ラインから、ピクセル・ラインを求める。
（手順） １．概略のピクセル・ラインを、ピクセル・ラインの近似値（p_0，l_0）に代入する。

２．ピクセル・ラインの近似値（p_0，l_0）と与えらた楕円体高 H を PLH_to_BL()に入力し、緯度・経度（B_0，L_0）を計算する。

３．下式に従い、ピクセルライン（p，l）を求める。ただし、a_{00} 等は、Init_BL_to_PLs()で求めた値である。

$$\begin{bmatrix} p \\ l \end{bmatrix} = \begin{bmatrix} p_0 \\ l_0 \end{bmatrix} + \begin{bmatrix} a_{10} + a_{11}L & a_{01} + a_{11}B \\ b_{10} + b_{11}L & a_{01} + a_{11}B \end{bmatrix} \begin{bmatrix} B - B_0 \\ L - L_0 \end{bmatrix}$$

４．２で求めた緯度・経度（B_0，L_0）と、与えられた緯度・経度（B，L）との差が一定値（例えば、地上 1mm）以内であれば，３で求めたピクセル・ライン（p，l）を返し、終了する。それ以外の場合は、３で求めたピクセル・ライン（p，l）を、ピクセル･ラインの近似値（p_0，l_0）として、２に戻る。

（参考） PLH_to_BL()は、楕円体高 H に対応したピクセル・ラインから緯度・経度への真の変換関数 G である。ピクセル・ラインの近似値を（p_0，l_0）、緯度・経度（B，L）に対する真のピクセル・ラインの値を（p_0，l_0）、近似値に対する修正量を（Δp，Δl）とすると、以下のようになる。

$$\begin{bmatrix} B_0 \\ L_0 \end{bmatrix} = G\left(\begin{bmatrix} p_0 \\ l_0 \end{bmatrix} \right)$$

$$\begin{bmatrix} B \\ L \end{bmatrix} = G\left(\begin{bmatrix} p \\ l \end{bmatrix} \right) = G\left(\begin{bmatrix} p_0 \\ l_0 \end{bmatrix} + \begin{bmatrix} \Delta p \\ \Delta l \end{bmatrix} \right) \approx \begin{bmatrix} B_0 \\ L_0 \end{bmatrix} + \begin{bmatrix} \frac{\partial B}{\partial p} & \frac{\partial B}{\partial l} \\ \frac{\partial L}{\partial p} & \frac{\partial L}{\partial l} \end{bmatrix} \begin{bmatrix} \Delta p \\ \Delta l \end{bmatrix}$$

$$\begin{bmatrix} \Delta p \\ \Delta l \end{bmatrix} \approx \begin{bmatrix} \frac{\partial B}{\partial p} & \frac{\partial B}{\partial l} \\ \frac{\partial L}{\partial p} & \frac{\partial L}{\partial l} \end{bmatrix}^{-1} \begin{bmatrix} B - B_0 \\ L - L_0 \end{bmatrix} = \begin{bmatrix} \frac{\partial p}{\partial B} & \frac{\partial p}{\partial L} \\ \frac{\partial l}{\partial B} & \frac{\partial l}{\partial L} \end{bmatrix} \begin{bmatrix} B - B_0 \\ L - L_0 \end{bmatrix}$$

$$\approx \begin{bmatrix} a_{10} + a_{11}L & a_{01} + a_{11}B \\ b_{10} + b_{11}L & a_{01} + a_{11}B \end{bmatrix} \begin{bmatrix} B - B_0 \\ L - L_0 \end{bmatrix}$$

```
void BLH_to_PL(double B, double L, double H, Orient_t& Orient, double& pixel,
       double& line);
```

（機能） 緯度・経度・楕円体高から、ピクセル・ラインを求める。

（手順） １．与えられた緯度･経度（B, L）を BL_to_PL0()に入力して、概略のピクセル･ライン（p_0, l_0）を求める。

２．１で求めた概略のピクセル・ライン（p_0，l_0）をピクセル・ラインの初期値として、BLH_to_PL0()をコールし、ピクセル・ラインを求める。

4-3. DEM.h/ccp

```
void Init_DEM(char *DemFileName);
```

（機能） DEM を読み込み、モジュールを初期化する。

```
double BL_to_DEMH(double B, double L);
```

（機能） 緯度、経度から標高を求める。

4-4. Geoid.h/ccp（ジオイド補正は行わないが、ジオイド補正を行う場合に必要な関数をコールすること）

```
void Init_Geoid(char *GeodiFileName);
```

（機能） ジオイド高ファイルを読み込み、モジュールを初期化する。

（処置） 何もしない。

```
double BL_to_GeoidH(double B, double L);
```
（機能） 緯度、経度からジオイド高を求める。
（処置） 常に 0 を返す。

５．正射画像の作成方法の概要

１． Init_PLH_to_BL()をコールし、PLH_to_BL()を初期値する。
２． その地域における典型的な楕円体高を Init_BL_to_PLs()に入力し、BL_to_PLs()を初期化する。
３． 出力範囲が与えられていない場合は、未補正画像の 4 隅の画素について、PLH_to_GK()をコールして、それらのガウスークリューゲル座標の最小値・最大値をもって、出力範囲とする。
４． 出力画像の全てのラインについて、５~８以下の操作を行う。
５． 出力画像の当該ラインの全ての画素について、以下の方法で緯度・経度・楕円体高を計算する。
5-1. ガウスークリューゲル座標の座標値（*X*, *Y*）を計算する。
5-2. 5-1 で求めた座標値（*X*, *Y*）を GK_to_DEMH()に入力して標高を求め、また、GK_to_GeoidH()に入力してジオイド高を求める。両者を足して、楕円体高 *H* を求める。
６． 出力画像のピクセル番号の昇順に、BLH_to_PL()を使用して、近似値なしで入力画像のピクセル・ラインを求める。最初に有効なピクセル・ラインを得られたならば、この画素のデータを出力画像の当該画素にコピーして、次に進む（この時の出力画像のピクセル番号を *po* とする）。有効性については、「６．画像の範囲外データへの対応」参照。
７． *po* から降順に BLH_to_PL0()を使用して、入力画像のピクセル・ラインを求め、出力画像の対応する画素にコピーする。概略のピクセル・ラインは、*po* +1 のピクセル・ラインとする。有効なピクセル・ラインが得られなければ、次に進む。
８． *po* から昇順に BLH_to_PL0()を使用して、入力画像のピクセル・ラインを求め、出力画像の対応する画素にコピーする。概略のピクセル・ラインは、*po* -1 のピクセル・ラインとする。有効なピクセル・ラインが得られなければ、次に進む。

なお、有効なピクセル・ラインが得られなかった場合は、0 を埋める。６で、有効なピクセル・ラインを得られなかった場合は、そのラインの全ての画素に 0 を埋める。

６．画像の範囲外データへの対応

PrismGeo.cpp の Pixel_to_CCD()の 421 行目（`if (i > MaxCCD) {`）では、画像の範囲外のピクセルを与えられた場合、異常終了するようになっている。同様に、GetSatPosAlt()の 491 行目（`if (l1<1 || l2>Orient.ImInf->lines) {`）では、画像の範囲外のラインを与えられた場合は、異常終了するようになっている。正射画像作成においては、範囲外のピクセル、ラインを与えられる場合があるので、これらの関数及びそれを引用する関数（今回作成する関数を含む）を、以下のように修正する。

・正常なピクセル、ラインが与えられたかどうかを示すグローバル変数 Invarid_Pixel_Line を用意する。
・PrismGeoInit()では、Invarid_Pixel_Line を false にする。
・Pixel_to_CCD()、GetSatPosAlt()では、正常なピクセル・ラインが与えられた場合には、Invarid_Pixel_Line を false にし、処理を継続する。それ以外の場合には、Invarid_Pixel_Line を true にして、処理を中断する。
・Pixel_to_CCD()、GetSatPosAlt()を呼び出した場合（間接的に呼び出した場合も含む）は、Invarid_Pixel_Lin をチェックし、true であれば処理を中断する。