空間検索機能を目的とした
河川流域データ表示システムの製作

島根大学　総合理工学部　数理・情報システム学科

応用情報学講座　田中研究室

S063047　下田圭亮

平成 22 年 2 月 3 日

目次

第１章　序論

1.1.　研究目的

現在、Google Earth[1]や Google Maps[2]のような地図サーバを用いた Web サイトは地図上に公共機関やレストランなどの店舗情報を表示し、今までのこのような空間検索機能の利用法は行政境界ごとや交通の路線沿いなどの単位が主であり、河川の汚染や洪水といった流域・地域単位での利用は少なく、災害などに対応するのに不十分である。

そこで、河川の汚染や洪水といった流域・地域単位での事象を、一般に公開された地図上に追加して表示するシステムがあれば、災害時の発生場所・規模などの特定や防災に役立ち地域住民や行政に有益であると考えた。

本研究では、斐伊川水系の河川流域データを Google Earth で表示するシステムを製作した。さらに空間検索を目的として流域内で発生した事象と流域外で発生した事象を判断するために、流域の最外周の基本閉路を求めた。

1.2.　システム概要

前述したシステムの製作にあたり、座標データの管理や検索には DBMS を用いた。本システムではその中でも座標データといった文字・数値以外のデータを容易に扱うことが出来、近傍検索といった空間検索を行うことが出来ることからオープンソースの ORDBMS である PostgreSQL[3]とその拡張機能である PostGIS[4]を用いた。

また、Web サーバには動作が軽く、高いシェアを持ち、拡張性が高いオープンソースである Apache を用いた。クライアントーWeb サーバ間をつなぐ言語も必要となってくるのでその言語には効率性、可搬性などの点から Java サーブレットを用い、サーブレットコンテナには安定性が高く、オープンソースである Tomcat を用いた。Web サーバーDB サーバ間の接続にはクライアントーWeb サーバ間に Java を用いたことから JDBC ドライバを用いることにした。

クライアント上への描画には Google Earth 上に追加して描画することの出来る KML を用いた。

なお、河川流域データをデータベースで扱えるようデータの修正を行った。

第２章　空間検索機能

2.1. 空間検索

空間検索とは、図形データと属性データを対応させて空間データとして出力し、情報を得る検索である。

図形データとは、その図形の形状・色・大きさなどによって、情報を表現するデータのことである。地図などがこれに該当する。

属性データとは、非空間・非図形である文字・数字データであり、図形と関連付けられるデータである。人口総数や高齢人口率などがこれに該当する。

これらのデータを組み合わせることにより位置に関する情報を持ったデータを空間データという。例えば、松江市の地図という図形データと松江市の高齢者人口という属性データを組み合わせると、松江市の高齢者人口の分布という空間データを得ることが出来る。

複合データである空間データを表示することで、個々のデータで見るよりも検索結果が理解しやすくなるメリットがある。

2.2. PostGIS

本研究で使用する PostGIS は PostgreSQL に対する空間拡張である。PostGIS は Refraction Research 社によってオープンソースで開発されている拡張機能である。その拡張内容は PostgreSQL に地理オブジェクトのサポートを追加することである。PostGIS を用いることで、PostgreSQL を ESRI の ArcGIS や Oracle Spatial と同様に、地理情報システムのバックエンドで機能する空間データベースとして利用することが出来る。

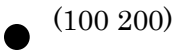
2.3. 空間データ

空間データの型は大まかに Point（点）型、LineString（線）型、Polygon（面）型がある。これらのデータ型を定義することで様々な形を表現できる。例えば、駅の座標は Point、国道は LineString、島根県の行政区画は Polygon を定義することで表現できる。

これらの空間データの表現方法として Well-Known Text（WKT）フォーマット形式がある。これは OGC[5]によって定義されている図形をテキストで表現する方法であり、空間検索システムや SQL などのこれをサポートしているソフトウェア同士でデータの受け渡しを容易に行うことが出来る。本研究で使用する PostgreSQL でも用いている。図 2.3.1 に WKT の例を示す。

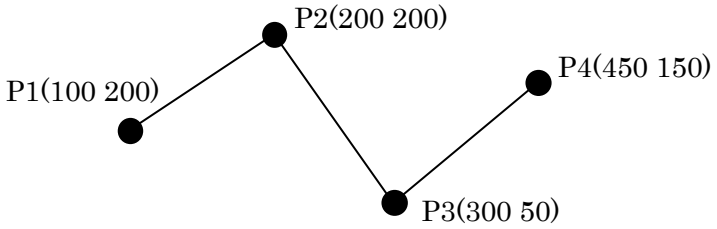
点（Point）：

Point(100 200)



線（LineString）：

LineString(100 100, 200 200, 300 50, 450 150)



面（Polygon）：

Polygon((100 100,200 250, 400 200, 450 50, 250 0, 100 100))

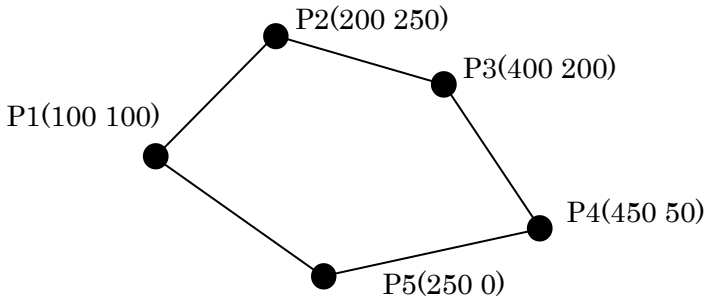


図 2.3.1：WKT の例

Point 型は、x,y 座標を定義することで点を表現出来る。LineString 型は、複数の点を定義し、それを結ぶことで線を表現出来る。Polygon 型は LineString 型と同様に複数の点を定義し、その点同士を結ぶのだが、最初と最後の点を同じにして線を閉じることで面を表現出来る。

第３章　　標準地域メッシュ

3.1.　概要

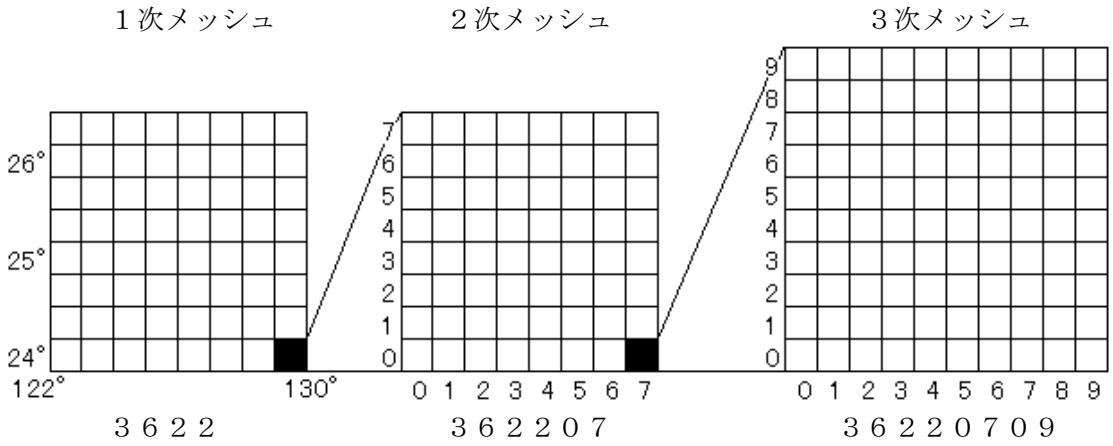
全国的な規模で数値情報を整備する場合、“標準地域メッシュ”方式が広く採用されている。これは、一定間隔の経線緯線によって地域を分割する１つの方法であり、以下に示す階層的な地域区画で全国を分割出来るようになっている（JIS × 0410 地域メッシュコード)。

3.2.　地域メッシュ

地域メッシュには、主に１次メッシュ、２次メッシュ、３次メッシュ（基準メッシュ）の３つのメッシュが存在する（４次メッシュは今回用いないため説明を省略する)。

全国の地域を１度毎の経線と３分の２度（40°）毎の緯線によって縦横に分割したものが１次メッシュ（国土地理院発行の縮尺 20 万分の１地勢図の通常の区画に相当する範囲)。１次メッシュの縦横をそれぞれ８等分したものが２次メッシュ（国土地理院発行の縮尺２万５千分の１地形図の通常の区画に相当する範囲)。２次メッシュの縦横をそれぞれ 10 等分したものが３次メッシュ（ほぼ１平方キロメートル）である。

例：メッシュコードが３６２２０７０９のとき



１次メッシュは４桁のメッシュコードで表す。
上２桁：南端緯度　×　１．５　（ただし、分の単位も含む）
下２桁：西端経度の下２桁
【例】　南端緯度２４度００分　西端経度　１２２度００分の場合
上２桁：２４×１．５＝３６　下２桁：２２　　→　３６２２

２次メッシュは６桁のメッシュコードで表す。
上４桁：１次メッシュコード
５桁目：１次メッシュを縦に８等分し、南から０～７の番号を付け、これをそれぞれのメッシュを示す数字とする。
６桁目：１次メッシュを横に８等分し、西から０～７の番号を付け、これをそれぞれのメッシュを示す数字とする。
【例】　３６２２０７

３次メッシュは８桁のメッシュコードで表す。
上６桁：２次メッシュコード
７桁目：２次メッシュを縦に 10 等分し、南から０～９の番号を付け、これをそれぞれのメッシュを示す数字とする。
８桁目：２時メッシュを横に 10 等分し、西から０～９の番号を付け、これをそれぞれのメッシュを示す数字とする。
【例】３６２２０７０９

第４章　　流域境界データの作成

4.1.　概要

河川流域境界データとしてまず、流域境界データを作成した。今回使用したデータは旧建設省国土地理院が作成した流域界･非集水界線位置データを島村[6]が単位流域ごとの１枚のポリゴンに直したものである（図 4.1.1）。

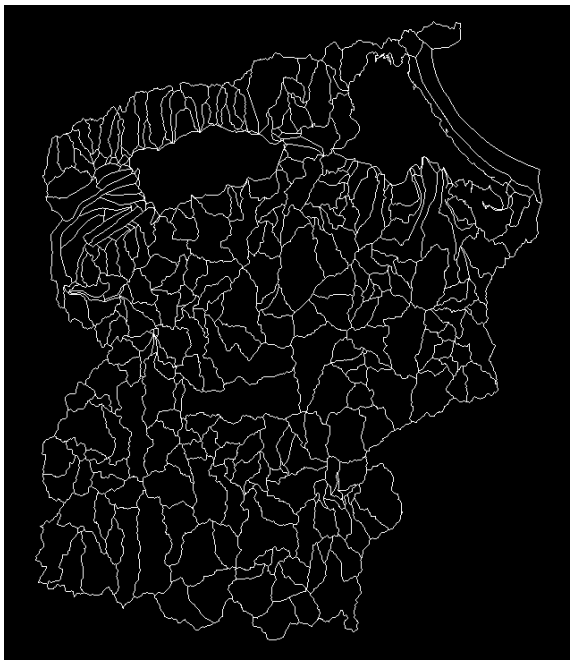


図 4.1.1：流域境界データの元データ

元データの説明を行う。今回使用したデータは建設省国土地理院が作成した国土数値情報細密数値情報の CD-R メディア内の KS-273（流域界・非集水界線位置）である。KS-273 には雨が降った際、ある特定の河川に集中すると考える地域を分合流点で分けた単位流域毎に分けられた座標値が入っている。

KS-273 は「３．２地域メッシュ」で説明した２次メッシュのベクトル形式の地図データである。データの項目は図葉番号、流域界、非集水界線区分[左側（水系域コード、単位流域コード）右側（水系域コード、単位流域コード）]、始終点タグ、非集水界線[空欄、始終点タグ]、２次メッシュ正規化位置である[7]。

データは、全国が１つのファイルにまとめられており、島村は島根県の宍道湖に流れ込む斐伊川水系の単位流域データを抜き出し、並び替え、グラフィックツールで表示できるようにした。

4.2.　データの修正

「２．２．空間データ」で述べたように、Polygon 型のデータは最初と最後の点を同じにしなければ DB に格納することが出来ない。しかし、元のデータはグラフィックツールで表示する目的で作成されているため polygon 型データになっておらず、最初と最後の点が同じでない部分がある。そこで最初と最後の点を同じにするべく最初の点を最後の点の次の点

として付け加えた（図 4.2.1）。

906681　1006526	**906681　1006526**
905231　1004865	905231　1004865
906437　1003329	906437　1003329
906100　1004500	906100　1004500
907400　1004865	907400　1004865
906700　1006300	906700　1006300
	906681　1006526

図 4.2.1：データの修正

4.3.　法線向きの統一

流域境界データをクライアントのブラウザに表示させると、法線が逆向きになって表示される箇所が多く存在する。法線向きが逆になっている場合、3D 表示する際、地表からのプラスマイナスの概念があるグラフィックツールを使用すると、ポリゴンデータが地下に潜り込んでしまう。そこで、法線が下向きに表示されている部分を修正し、全ての法線が上を向くように統一した。この方法は文献[8]によるものである。

4.3.1.　法線不統一の原因

法線の向きはデータの並びに依存する。ちょうど右ねじの法則のように、ある点から次の点へ線を引く回転方向が右回りであれば法線は下向きになる（図 4.3.1）。逆に、左回りであれば法線は上を向く。

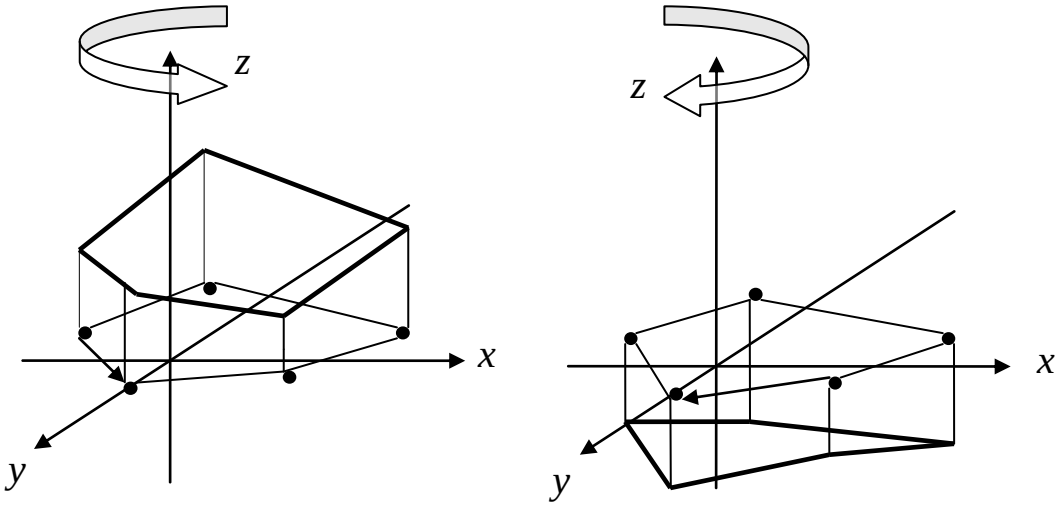


図 4.3.1：データの並びと法線の向き

4.3.2.　修正方法・修正結果

前節で、法線の向きは、ポリゴンデータの流れの左右を判定することで分かることを述べた。その判定方法には、外積ベクトルの公式を用いる。

外積の公式では、ベクトル a,b を辺とする平行四辺形の面積と、面積を求める際のベクトルの回転方向を知ることができる。

$$a \times b = |a||b|\sin\theta$$

公式より、外積 $a \times b$ の符号が分かれば、$\sin\theta$ の符号が分かる。つまり、$a \times b$ と θ の符号は等しいため、外積を計算した結果が正であるか負であるかで θ の回転方向が判定できる。以上より、

$a \times b > 0$ であれば、 $\theta > 0$ で、右回り
$a \times b < 0$ であれば、 $\theta < 0$ で、左回り

となる。これを図で示すと図 4.3.2 のようになる。

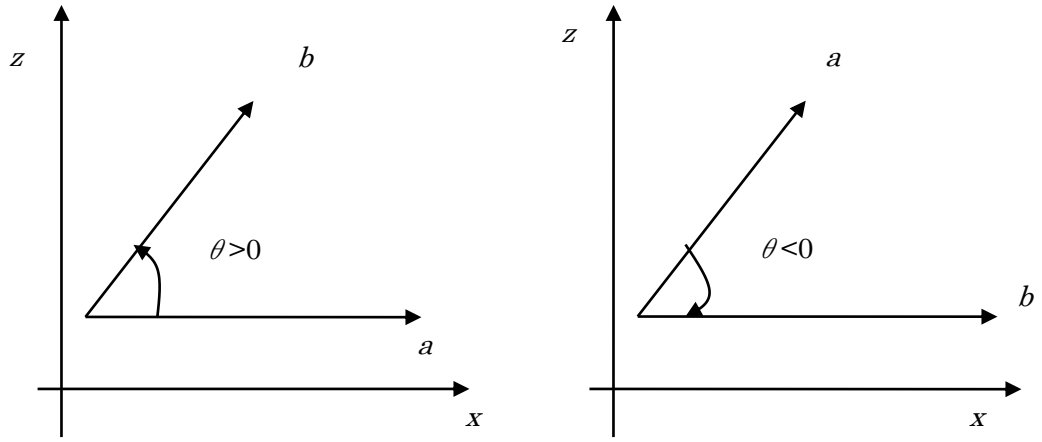
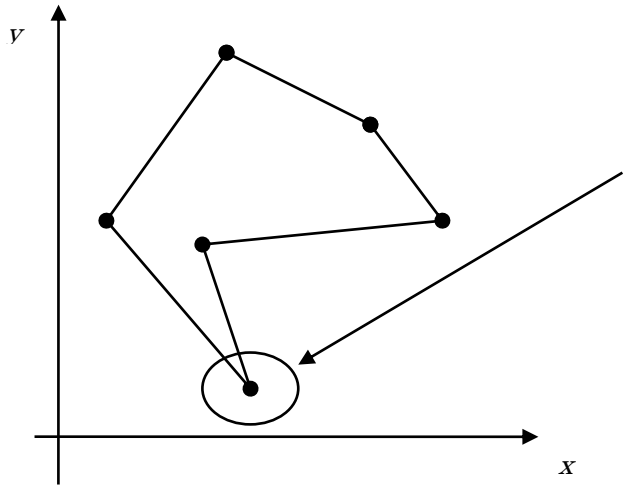


図 4.3.2：ベクトルの回転方向と θ の符号の関係

次に、この外積の考え方を実際にポリゴンデータに用いる方法について説明していく。

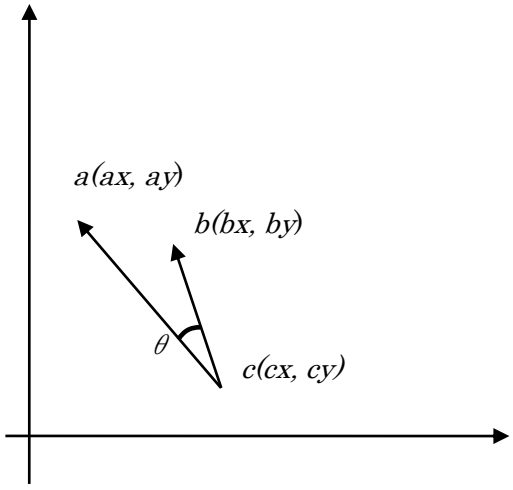
1．ポリゴンデータの中で y 軸の座標値が最小の点を見つける



x	*y*
906681	1006526
905231	1004865
906437	**1003329**
906100	1004500
907400	1004865
906700	1006300
906681	1006526

図 4.3.3：ポリゴンデータの y 軸の座標値が最小の点

２．Y 座標の最小の点と、その前後２点の３点で外積の計算をする。



x	*y*	
906681	1006526	
905231	1004865	*a*
906437	**1003329**	***c***
906100	1004500	*b*
907400	1004865	
906700	1006300	
906681	1006526	

図 4.3.4：y 軸の座標値が最小の点と前後の２点

ここで、y 座標値最小の点が始点であれば、始点より前の点は存在しないため、始点、始点+1、終点-1 の 3 点での外積を考える。y 座標値最小の点が終点であった場合も同様に、終点、終点-1、始点+1 の 3 点で外積を考える。

3 点 *a,b,c* の外積の計算式は

$$a \times b = (ax - cx) * (by - cy) - (ay - cy) * (bx - cx)$$

となる。計算結果が負であれば θ も負であり、データの流れは左回りとなるので、法線向きを上向きと判断する。上図の例で外積を計算すると

$$a \times b = (905231 - 906437) * (1004500 - 1003329) - (1004865 - 1003329) * (906100 - 906437) = -894594 < 0$$

となり、負となるのでデータに修正は加えない。この計算を全てのポリゴンデータに行い、法線が逆向きになっているポリゴンには修正を加えていく。

4.4. ポリゴンデータの併合

流域内外の事象を判断するための機能として SQL/MM に ST_Within 関数がある。SQL/MM とはマルチメディア・データベースに必要なデータ型を標準ライブラリとして整備を行っている。本研究で利用する PostGIS は SQL/MM で定義している関数をサポートしている。ST_Within 関数を用いることで「２．２．空間データ」で説明したような Point 型、LineString 型、Polygon 型で格納されているデータと検索範囲の図形との空間関係をチェックして、前者が後者に「完全に含まれているかどうか」を判定できる。これを図示したものが図 4.4.1 である。

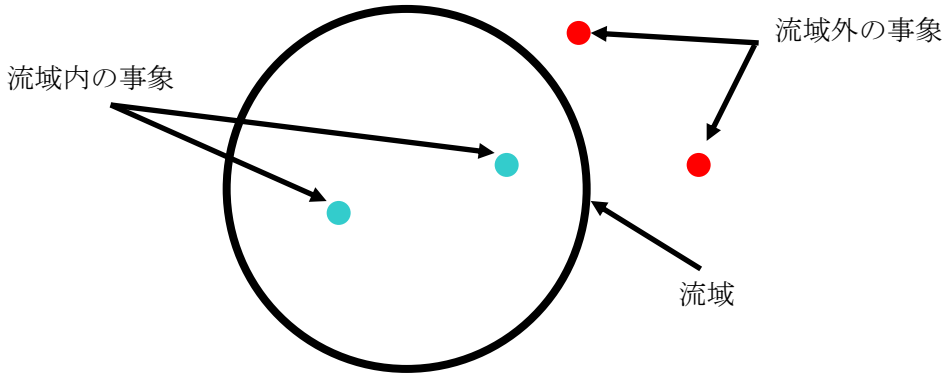


図 4.4.1：ST_Within 関数による流域内外の事象の判定

図 4.4.1 では、「完全に含まれている」点を青、「含まれていない」点を赤で示している。空間検索範囲の図形が「流域」の円で示されている。ST_Within 関数を用いることで流域の円に「完全に含まれる」点とそれ以外の点を判定している。

ST_Within 関数を用いるためには流域境界データの最外周の対応する PostGIS の関数は基本閉路の座標点を抽出するのである。そのためには、流域境界データの複数のポリゴンデータを併合して１つのポリゴンデータにすることで実現する。

4.4.1. 併合方法

図 4.4.2 のようにポリゴン A を小流域と定義し、ポリゴン B を小流域 A に隣接する小流域と定義する。また、この時データの並びは法線向きを統一しているため反時計回りとする。

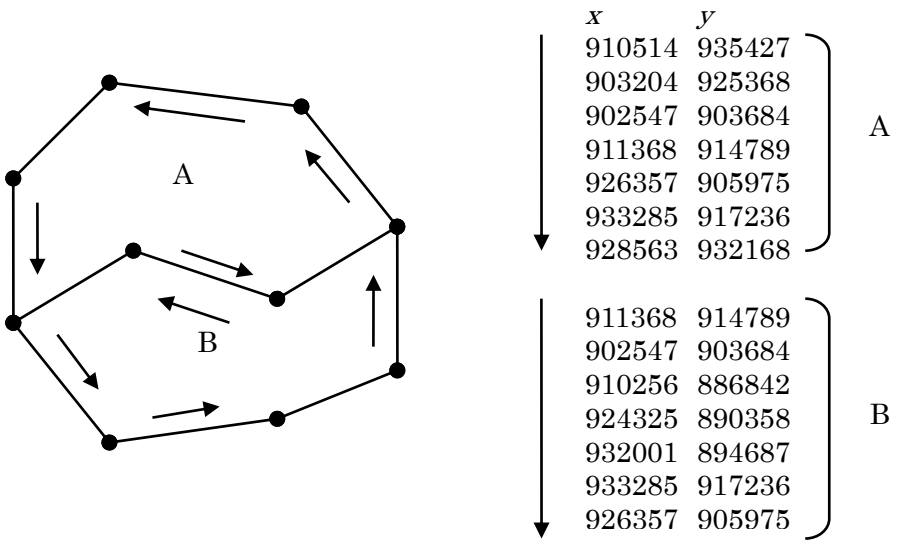


図 4.4.2：ポリゴンデータの併合

1．全データを１つずつ比較し、重複している点があるかどうかを調べる。
重複している点が存在する場合、その点に重複している点の数だけデータに数字をつける。（重複している点が２つの場合は２）
重複している点が存在しない場合、その点に数字の０をつける。

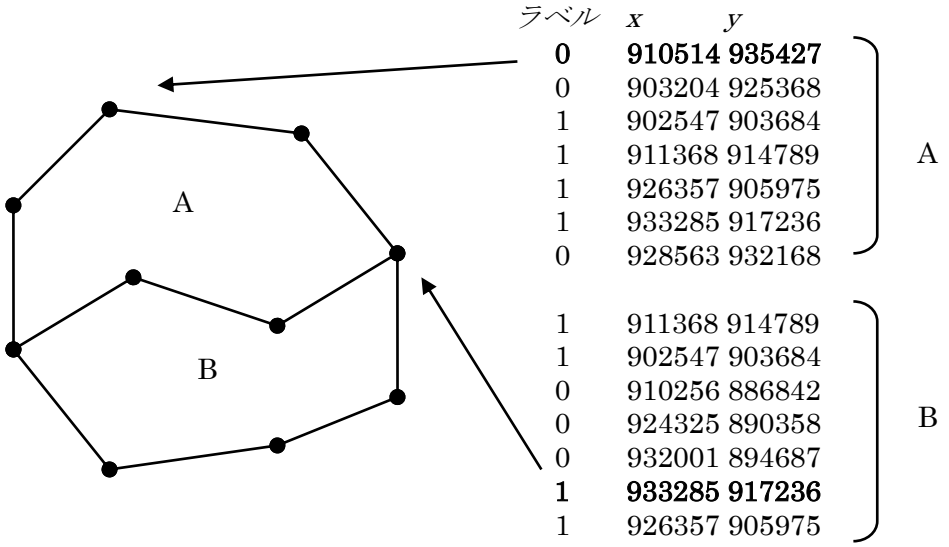
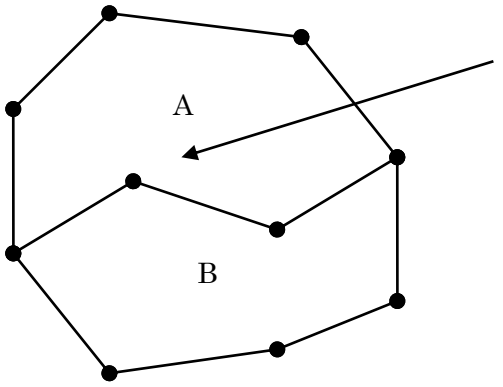


図 4.4.3：データの比較

2．１以上の数字が付いている点について調べる（今回はポリゴン A の４番目のデータとする）。１以上の数字が付いている点の前後を調べ、前後の点にも１以上の数字が付いている場合、その点を削除する。



ラベル	*x*	*y*	
0	910514	935427	
0	903204	925368	
1	902547	903684	
1	**911368**	**914789**	A
1	926357	905975	
1	933285	917236	
0	928563	932168	
1	911368	914789	
1	902547	903684	
0	910256	886842	
0	924325	890358	B
0	932001	894687	
1	933285	917236	
1	926357	905975	

図 4.4.4：データの削除

３．削除後はデータの並びがバラバラになっているので１つのポリゴンとなるように並び替えを行う。

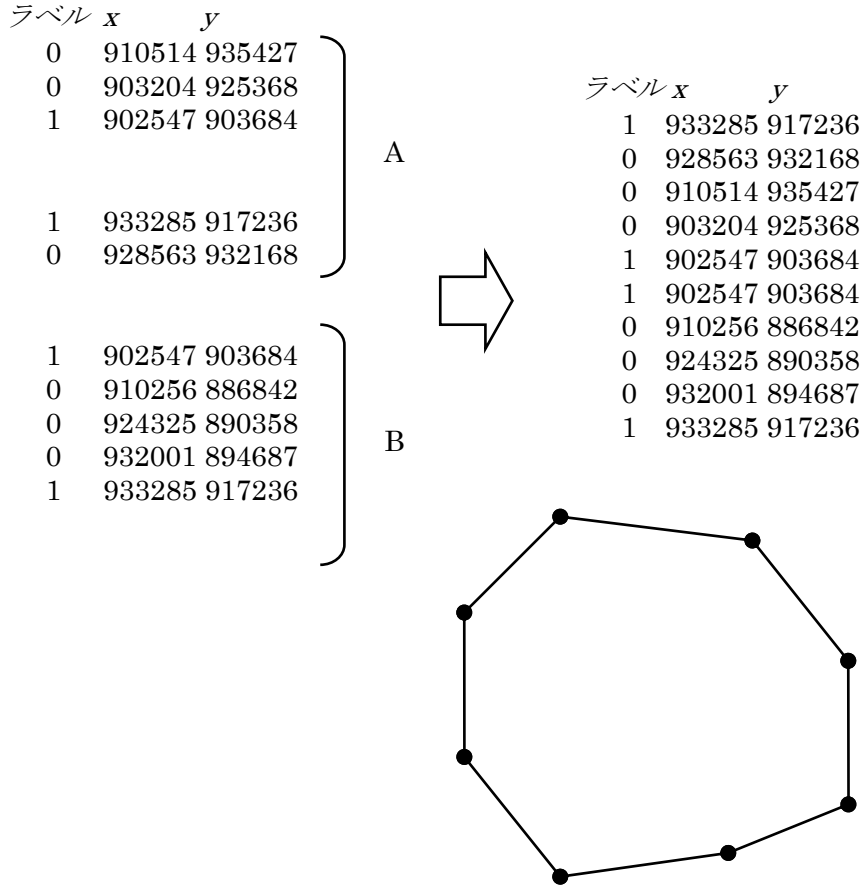


図 4.4.5：データの並び替え

なお、今回のアルゴリズムを用いてポリゴンデータを併合する場合、ポリゴン型のデータのままだとアルゴリズムを適用することが出来ない。その理由を説明する。

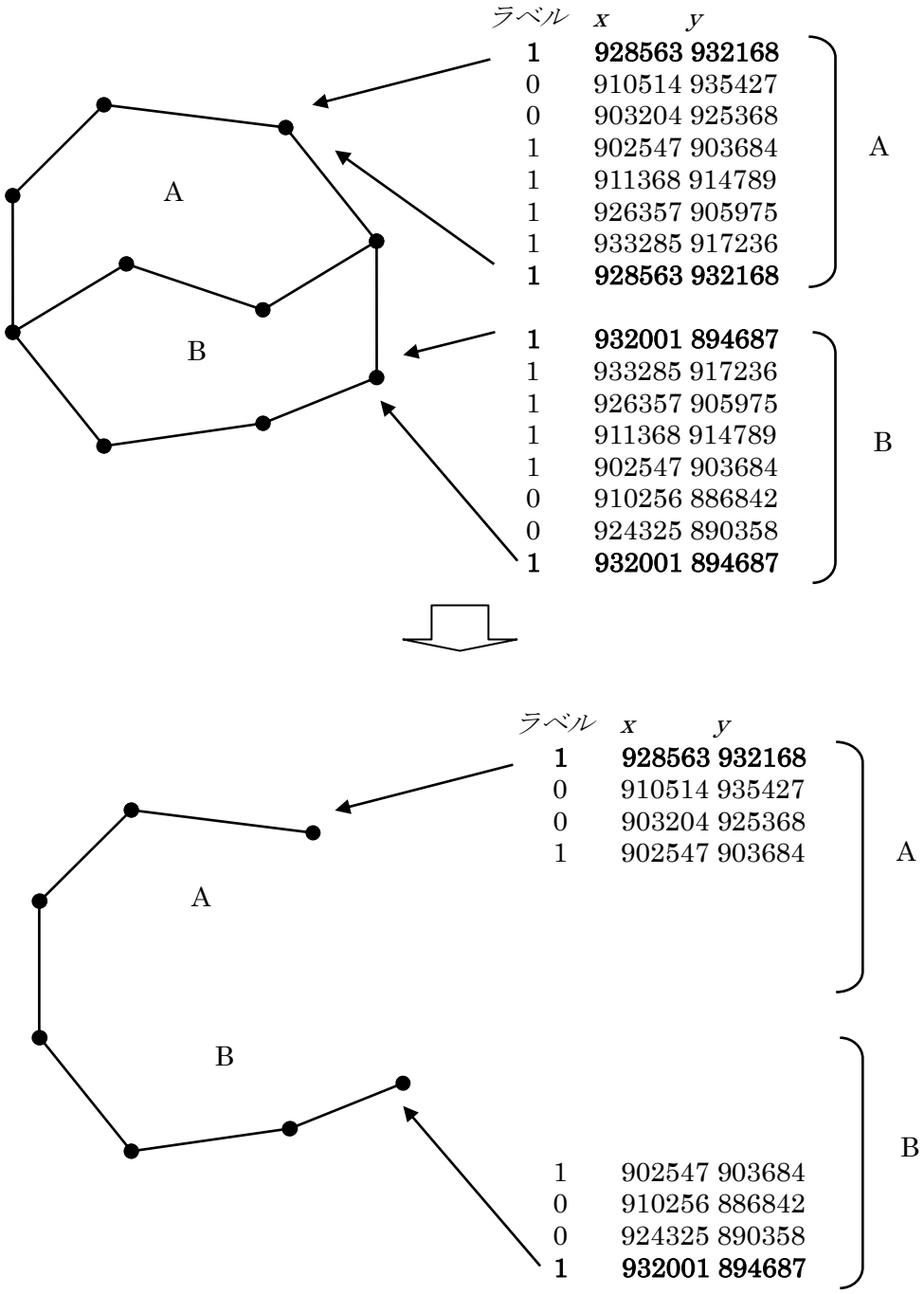


図 4.4.6：ポリゴン型データの場合

図 4.4.6 のデータはポリゴン型のデータとなっている。なので、ポリゴン A、ポリゴン B 共に最初と最後のデータが同じになっている。このようなデータでポリゴンデータの併合を行うと必要な点まで削除してしまう。しかし、これをラインストリング型のデータにすると併合出来る（図 4.4.7）。

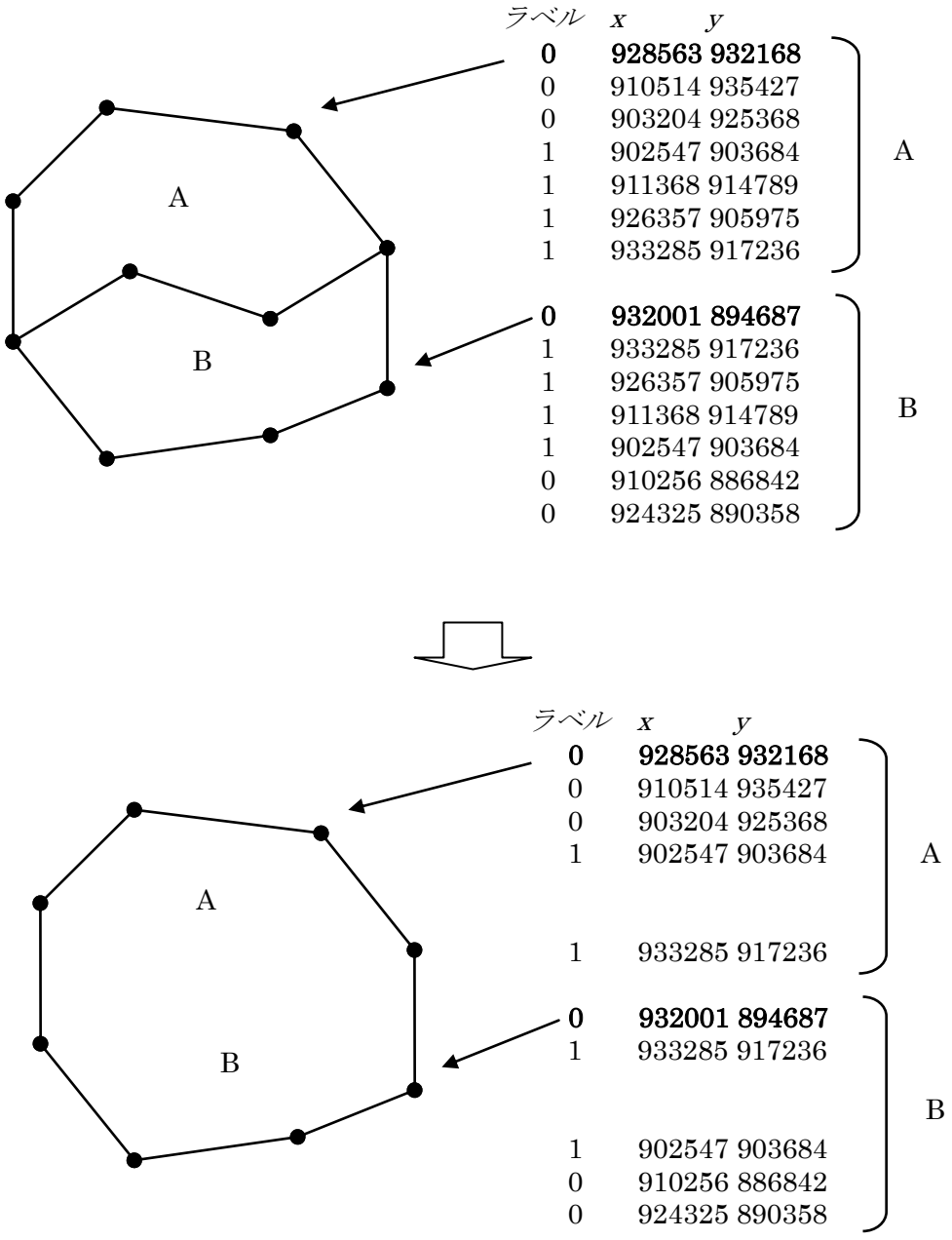
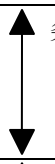


図 4.4.7：ラインストリング型の場合

以上の理由により事前にポリゴン型のデータをラインストリング型のデータに変換した。

4.4.2. 擬似言語

ポリゴンデータ併合アルゴリズムの擬似言語を示す。なお、擬似言語内に出てくる図形は以下のような規則に基づいている。

記述形式	説明
○	手続き、変数などの名前、型などを宣言する。
・変数←式	変数に式の値を代入する。
	単岐選択処理を示す。 条件式が真の時は処理を実行する。
	双岐選択処理を示す。 条件式が真の時は処理１を実行し、偽の時は処理２を実行する。
	前判定繰り返し処理を示す。 条件式が真の間、処理を繰り返し実行する。
	後判定繰り返し処理を示す。 処理を実行し、条件が真の間、処理を繰り返し実行する。

１．全データに数字を付ける。

○プログラム名：x,y 座標データに数字を付ける
○整数型：Max /*データの総数*/
○整数型：x,y
○文字列型：lines /*ファイルから x,y 座標を読み込む変数*/
○整数型：i,j,n,k
○整数型：flag /*重複している点が存在するときフラグを立てる変数*/
○整数型：poly[2][Max] /*x,y 座標を格納する配列*/, tag[Max] /*各データに数字を付ける配列*/
○文字列型：data /*河川流域データ*/
○手続き：ReadF(a) /*ファイルから変数 a に文字列を入力*/
○手続き：WriteF(a) /*変数 a の内容をファイルへ出力*/
○手続き：length(a) /*変数 a の長さを格納*/
○手続き：ReadD(a) /*変数 a を読み込む*/

/*データを読み取る*/
・i ← 0
■(lines = ReadF(data)) ≠ null
　・n ← length(data)
　▲n ≠ 0
　　・x ← ReadD(data) /*data から x 座標を読み込む*/
　　・y ← ReadD(data) /*data から y 座標を読み込む*/
　　・poly[0][i] ← x
　　・poly[1][i] ← y

```
    ・i ← i + 1
  ――――
    ・i ← i + 1
  ▼
■

/*数字を付ける*/
・k ← i
・flag ← 0
■i : 0, i<k, 1
  ■j : i+1, j<k, 1
    ▲poly[0][i] = poly[0][j] and poly[1][i] = poly[1][j] and poly[0][i] ≠ 0 and poly[1][i] ≠ 0
      ・flag ← flag + 1
      ・tag[j] ← flag
      ・tag[i] ← flag
    ▼
    ・flag ← 0 /*フラグをリセット*/
  ■
■

/*ファイルに書き出す*/
■i : 0, i<k, 1
  ▲poly[0][i] ≠ 0 and poly[1][i] ≠ 0
    ・WriteF(tag[i] + "¥t" + poly[0][i] + "¥t" + poly[1][i] + "¥n") /*タグと座標を出力*/
  ――――
    ・WriteF("¥n") /*改行を出力*/
  ▼
■
```

２．データを削除する。

```
○プログラム名：ポリゴンデータの併合
○整数型：Max /*データの総数*/
○整数型：t,x,y /*t はタグの値を格納する変数*/
○文字列型：lines /*ファイルから x,y 座標を読み込む変数*/
○整数型：i,j,m,n,k,l
○整数型：a,b,c
○整数型：poly[2][Max] /*x,y 座標を格納する配列*/, tag[Max] /*各データに数字を付ける配列
*/
○文字列型：data /*タグと座標データ*/
○手続き：ReadF(a) /*ファイルから変数 a に文字列を入力*/
○手続き：WriteF(a) /*変数 a の内容をファイルへ出力*/
○手続き：length(a) /*変数 a の長さを格納*/
○手続き：ReadD(a) /*変数 a を読み込む*/
```

```
/*データを読み取る*/
・i ← 0
■(lines = ReadF(data)) ≠ null
 ・n ← length(data)
 ▲n ≠ 0
  ・t ← ReadD(data) /*data からタグを読み込む*/
  ・x ← ReadD(data) /*data から x 座標を読み込む*/
  ・y ← ReadD(data) /*data から y 座標を読み込む*/
  ・poly[0][i] ← x
  ・poly[1][i] ← y
  ・i ← i + 1
 ――――
  ・l ← 0 /*ポリゴンの塊の最後*/
  ■k : 0, k<i, 1
   ・l ← i−1
   ・a ← p + 1
   ・b ← p
   ・c ← p – 1
   ▲k = 0 /*始点のときは、点 0,1,N－１の３点で処理する*/
    ・a ← 1
    ・b ← 0
    ・c ← i – 1
   ▼
   ▲k = i – 1 /*終点のときは、点 0,N-2,N-1 の３点で処理する*/
    ・a ← 0
    ・b ← i – 1
    ・c ← i – 2
   ▼
   ▲tag[a] ≧ 1 and tag[b] ≧ 1 and tag[c] ≧ 1 /*削除対象の座標を－１にする*/
    ・poly[0][b] ← −1
    ・poly[1][b] ← −1
   ▼
   ▲k = 0 and poly[0][k] ≠ −1 and poly[1][k] ≠ −1
    ・flag ← flag + 1
   ▼
   ▲k = l and poly[0][k] ≠ −1 and poly[1][k] ≠ −1
    ・flag ← flag + 1
   ▼
  ■

  /*データを並び替える*/
  ▲flag = 2
   ・k ← 0
   ■poly[0][k] ≠ −1 and poly[1][k] ≠ −1 /*一時的にデータを保存しておく*/
```

```
        ・tmp[0][k] ← poly[0][k]
        ・tmp[1][k] ← poly[1][k]
        ・k ← k + 1
      ■
      ・m ← 0
      ■j : 0, j<i, 1
        ・poly[0][j] ← poly[0][j + k]
        ・poly[1][j] ← poly[1][j + k]
        ▲j ≧ i − k
          ・poly[0][j] ← tmp[0][m]
          ・poly[1][j] ← tmp[1][m]
          ・m ← m + 1
        ▼
      ■
    ▼

    /*ファイルに書き込む*/
    ■k : 0, k<i, 1
      ▲poly[0][k] ≠ −1 and poly[1][k] ≠ −1
        ・WriteF(poly[0][k] + "¥t" + poly[1][k] + "¥t" + "¥n")
      ▼
    ■
    ・WriteF("¥n")
    ・i ← 0
    ・flag ← 0
  ▼
■
```

以上のアルゴリズムによりポリゴンデータの併合を行った。結果が図 4.4.7 である。

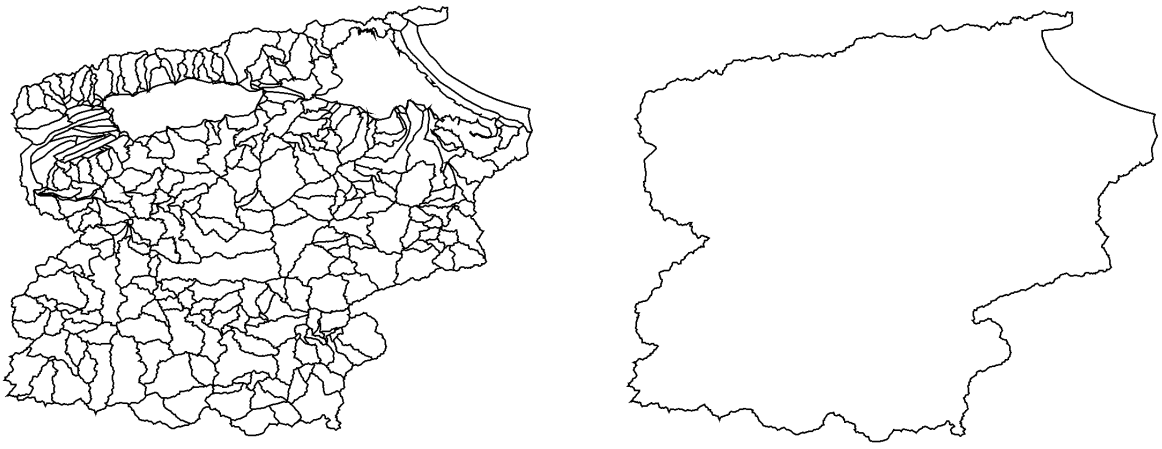
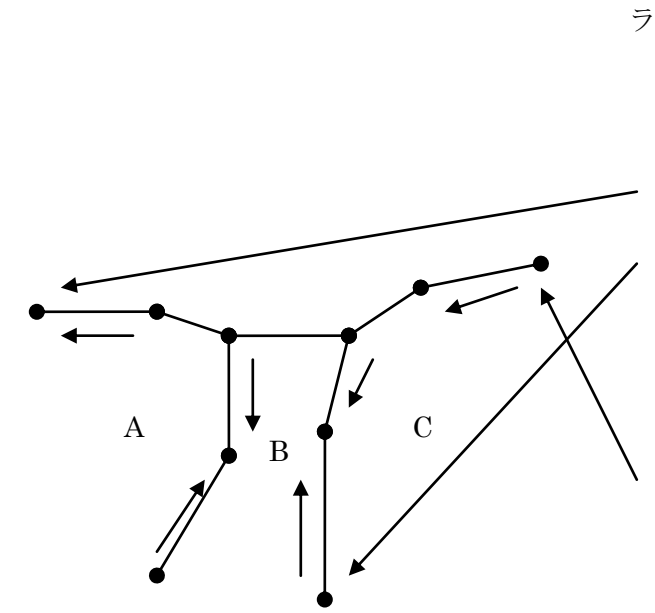
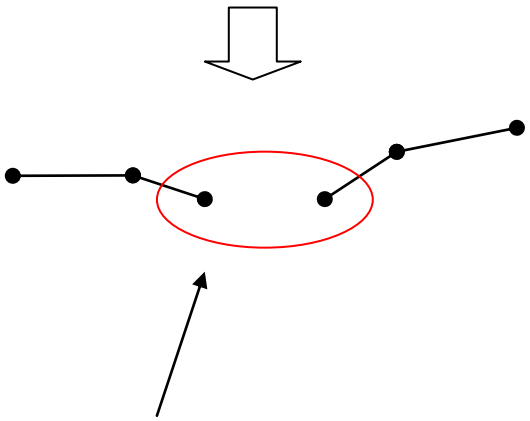


図 4.4.7：併合結果

なお、この方法では図 4.4.8 の場合ポリゴンデータを併合できない欠点が存在するが、今回対象としたデータでは、そのような測量データは原則として存在していない。



	ラベル	x	y
A	1	905214	904532
	1	907563	1003210
	1	906357	1105931
	0	902014	1108364
	0	882156	1107647
B	1	1032540	902687
	1	1013690	1005310
	1	1039860	1108963
	1	906357	1105931
	1	907563	1003210
	1	905214	904532
C	0	1206951	1208974
	0	1163147	1203204
	1	1039860	1108963
	1	1013690	1005310
	1	1032540	902687



データのつながりがなくなる。

	ラベル	x	y
A	1	906357	1105931
	0	902014	1108364
	0	882156	1107647
B			
C	0	1206951	1208974
	0	1163147	1203204
	1	1039860	1108963

図 4.4.8：ポリゴンデータ併合の欠点

図 4.4.8 の A,B,C は３つの小流域であると定義する。ポリゴンデータ併合のアルゴリズムを用いると B のデータが全てなくなってしまい、最外周のデータがつながらなくなってしまう。

第５章　　河川座標データの追加

5.1.　概要

河川流域データとして河川座標のデータを追加した。使用したデータは国土地理院の国土数値情報ダウンロードサービス[9]より JPGIS 準拠データをダウンロードした。

JPGIS とは、正式名称を「地理情報標準プロファイル（Japan Profile for Geographic Information Standards)」といい、日本国内における地理情報の標準規格である。

5.2.　フォーマット

河川座標データのデータ構造を示す。

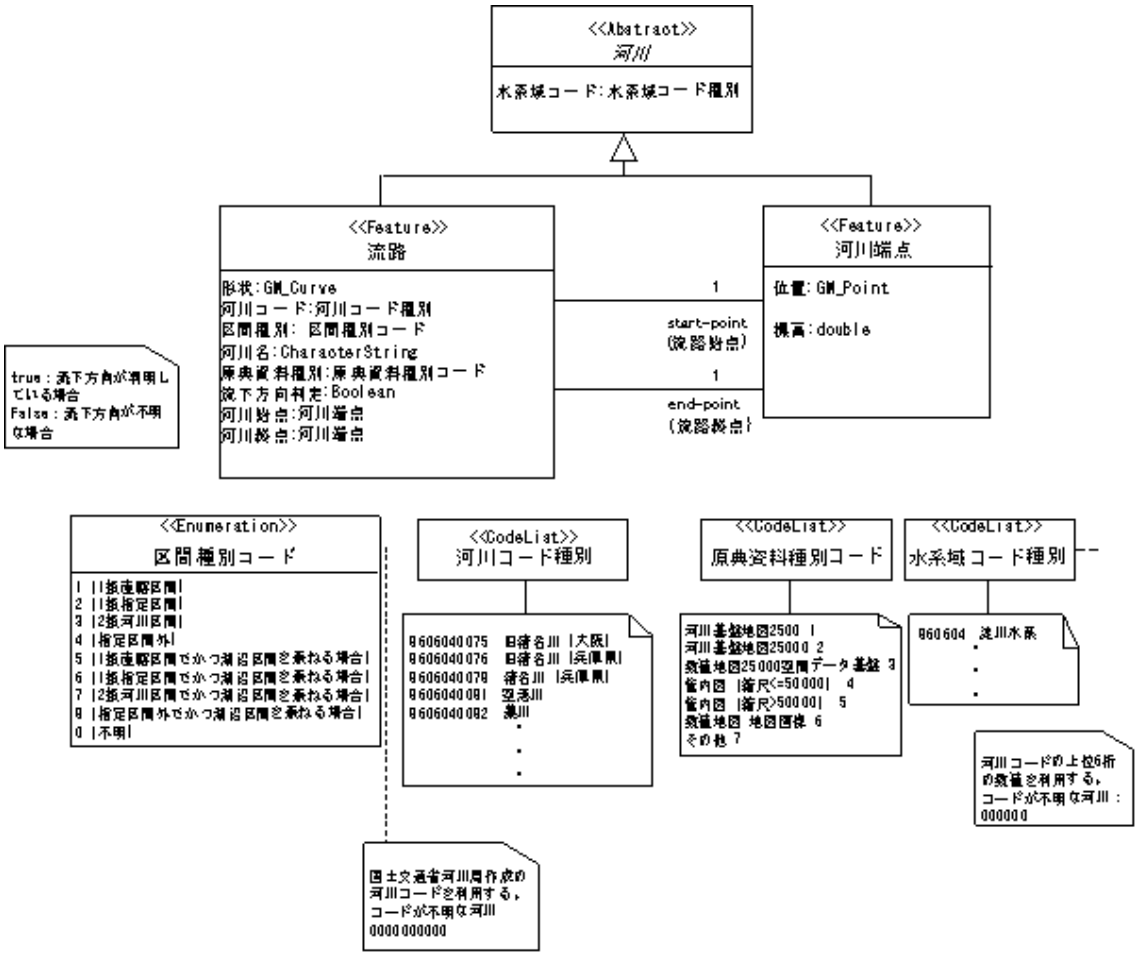


図 5.1.1：データ構造

XML ファイルを示す（図 5.1.2)。

```
…
<ksj:GB02 id="r-1">
                                                    <ksj:WSC
codeSpace="WaterSystemCodeCd.xml">320000</ksj:WSC>
                                                    <ksj:LOC idref="c-1"/>
```

```
                                                    <ksj:RIC
codeSpace="RiverCodeCd.xml">3200000000</ksj:RIC>
                                                    <ksj:COP>0</ksj:COP>
                                                    <ksj:RIN>名称不明</ksj:RIN>
                                                    <ksj:SOS idref="gb03_3200002"/>
                                                    <ksj:EOS idref="gb03_3200001"/>
                                                    <ksj:ODC
codeSpace="OriginalDataCodeCd.xml">3</ksj:ODC>
                                                    <ksj:DFD>1</ksj:DFD>
                                                    <ksj:SOR idref="gb03_3200002"/>
                                                    <ksj:EOR idref="gb03_3200001"/>
                                            </ksj:GB02>
…
<ksj:GB03 id="gb03_3200841">
                                                    <ksj:WSC
codeSpace="WaterSystemCodeCd.xml">870704</ksj:WSC>
                                                    <ksj:POS idref="p-841"/>
                                                    <ksj:ELE>7</ksj:ELE>
                                            </ksj:GB03>
…
```

図 5.1.2：XML ファイル

XML とは、文章やデータの意味や構造を記述するためのマークアップ言語の１つである。マークアップ言語とは、「タグ」と呼ばれる特定の文字列で地の文に情報の意味や構造、装飾などを埋め込んでいく言語のことである。

表 5.1.1 は河川座標の XML ファイルで使用されているタグの一覧である。

表 5.1.1：タグ一覧

クラス	属性・役割関連	型	タグ名	英語名
河川			GB01	River
	水系域コード	水系域コード種別	WSC	Water system code
流路			GB02	Passage
	形状	GM_Curve	LOC	Location
	河川コード	Code Type	RIC	River code
	区間種別	Code Type	COP	Classification of passage
	河川名	CharacterString	RIN	River name
	流路始点	idref	SOS	Start of Stream
	流路終点	idref	EOS	End of Stream
	原点資料種別	Code Type	ODC	Original data code
	流下方判定	Bollean	DFD	Determination of flow direction
	河川始点	idref	SOR	Start of River
	河川終点	idref	EOR	End of River

河川端点			GB03	Edge point of river
	位置	GM_Point	POS	Position
	標高	Double	ELE	Elevation

本研究では、河川座標データとして「水系域コード」「河川コード」「河川名」「河川座標」を使用する。水系域コードとは５桁とし、上２桁は管轄地建番号、下３桁は全国の通しの一連番号である（図 5.1.3)。河川コードとは２万５千分の１計測基図から流路を移写した 20 万分の１地勢図上で、水系単位に３桁の数字で表 5.1.1 の基準に基づいて設定した数字である。

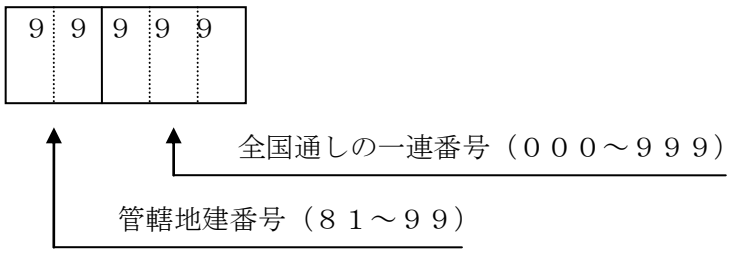


図 5.1.3：水系域コード

表 5.1.2：河川コード

	１次河川	２次河川	３次河川
一級河川および主要２級河川	００１～００９	０１１～０９９	１１１～９９９
その他複合水系域	００１～０２９	０３１～１９９	２０１～９９９

5.3. データの抽出

PostgreSQL へのデータの格納をより簡易にするため、国土地理院の提供する「国土数値情報データ変換ツール」[10]を用いて shape ファイル形式へ変換を行った。shape ファイルとは ESRI 社[11]の提唱する公開されたベクタデータの業界標準フォーマットである。多くの GIS ソフトウェアで利用が可能であり、本研究で使用する PostgreSQL も対応している。なお、対応する測地系はもともと世界測地系である。

5.4. DB 格納

「５．３．データの抽出」で抽出した Shape ファイル形式のデータを DB に格納した。その手順を以下に示す。

１．コマンドプロンプトを起動してデータベースを作成する。
２．データベースにシェープファイルを投入する。

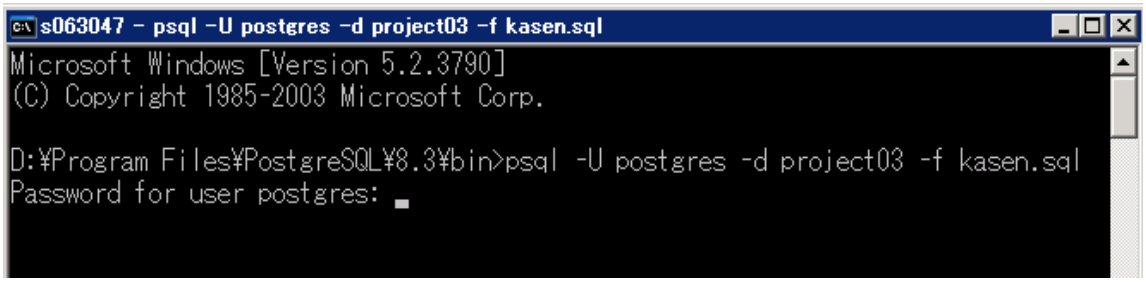


図 5.4.1：shape ファイルの投入

図 5.4.1 で「-U」はユーザ名、「-d」は DB 名、「-f」はファイル名を示すオプションであり、それぞれ「postgres・project03・kasen.sql」となっている。

上記の方法でテーブルを作成したのでそのテーブルの構成について以下に示す。

表 5.3.1：DB 内での河川座標のテーブル

Table_Name：kasen					
gid:serial 型	id:integer 型	水系域コード: text 型	河川コード: text 型	河川名: text 型	the_geom: Geometry 型

表 5.3.1 は河川座標のテーブルの構成である。Serial 型の gid はシーケンスである。シーケンスとは、連番を発生させるオブジェクトである。シーケンスを作成する構文は、以下の通りである。

```
create sequence  シーケンス名  [increment [by]  増加分]
      [minvalue  最小値] [maxvalue  最大値]
      [start  開始点];
```

オプションを指定しなかった場合、連番の値は 1 から始まり、 1 ずつ増加する。

Geometry 型の the_geom は MultiLineString である。

第６章　　測地系の変更

6.1.　概要

測地系とは地球上の位置を経度・緯度で表すための基準であり、地球の形にもっとも近い回転楕円体で定義されている。今回使用した流域境界データの測地系は日本測地系である。しかし、Google Earth の測地系は世界測地系であるので、測地系を日本測地系から世界測地系へと変更を行う必要がある。この方法は文献[12]によるものである。

6.2.　絶対座標化

流域境界データは２次メッシュ単位に左下を（０，０）、右上を（１００００００，１０００００）とする正規化された相対座標で格納されている。そのデータを島村がメッシュを考慮して絶対座標化した。絶対座標とは原点を設定し、そこからの距離によって位置を表す座標である。

まず、絶対座標化について説明する。絶対座標化について図 6.2.1 の A という２次メッシュファイル（A）の代表点の座標は、（５００００，５００００）であることが分かる。また、このＡが、図 6.2.1 の複数の２次メッシュファイルの中で存在しているときの代表点は、ファイルの書式形式上では（５００００，５００００）という座標で記載されているが、このままの状態では、複数の２次メッシュファイルから構成されている流域境界では、正確な位置を表すことが出来ない。したがって、この流域境界の正確な座標を B ファイルの左下（０，０）を原点として、右上の（３０００００，２０００００）の範囲内で表すと、その代表点は（２５００００，１５００００）という新たな座標に変換することが出来る。このような処理が絶対座標化である。

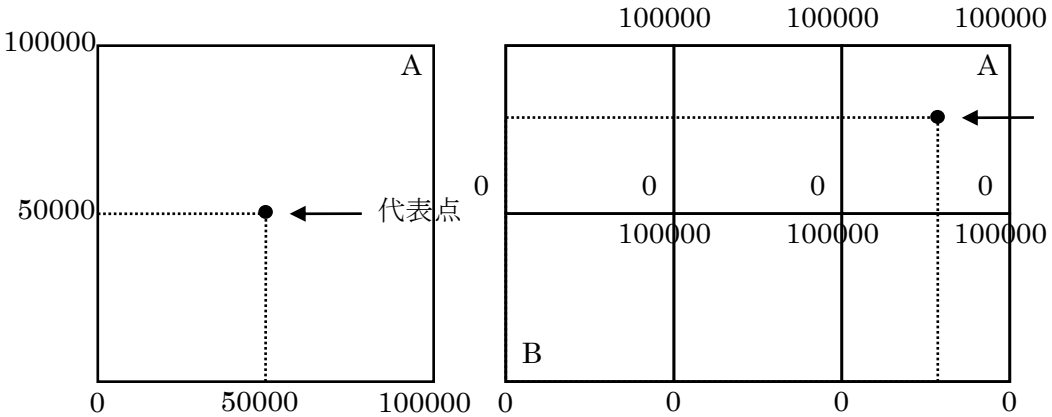


図 6.2.1：絶対座標値化

6.3.　地域メッシュポリゴン

流域境界データの測地系を変更するためには、まず、１km×１km 四角形の Polygon 型図形の世界測地系の経度緯度を求めなければならない。この図形は第３次メッシュの左下の経度緯度を求めることで生成できる。

３次メッシュコードから座標を求めるには、メッシュコードの構成を知る必要がある。これについては「３．２．地域メッシュ」で述べたように、３次メッシュコードというのは上

４桁が１次メッシュコード、次の２桁が２次メッシュコード、残りの２桁が３次メッシュコードである。１次メッシュコードというのは全国の地域を１度毎の経線と３分の２度毎の緯線によって分割したものである。２次メッシュコードというのは１次メッシュを８等分し、０～７の番号を付けたものである。３次メッシュコードというのは２次メッシュを 10 等分し、0～9の番号を付けたものである。このことから、２次メッシュは経度を８分の１度毎、緯度を 12 分の１度毎に分割したものであり、３次メッシュは経度を 80 分の１度毎、緯度を 120 分の１度毎に分割したものだといえる。これらのことを踏まえて例を用いて説明する。

例：第３次メッシュコード「52324646」の緯度経度を求める。

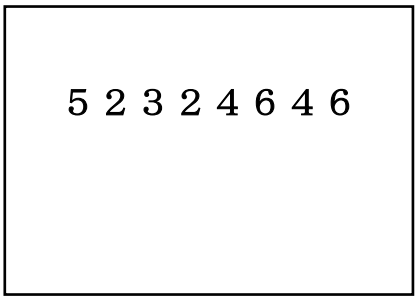


図 6.3.1：第３次メッシュの例

上４桁が１次メッシュコード，次の２桁が２次メッシュコード，残りの２桁が３次メッシュコードなので経度緯度は、

緯度＝（５２×２／３）＋（４／１２）＋（４／１２０）＝　３５．０３３…
緯度＝（３２＋１００）＋（６／８）＋（６／８０）　　＝１３２．８２５

この方法を用い、３次メッシュコードを経度緯度座標に変換する。しかし、この経度緯度は日本測地系の経度緯度なので、世界測地系に変換する必要がある。これについては、後述する。

6.4.　日本測地系から世界測地系へ座標変換

日本測地系を世界測地系に変換するには、３次メッシュコードと日本測地系の経度緯度が必要となる。流域境界データの場合について説明する。流域境界データは「６．２．絶対座標化」で述べた方法により、１次メッシュコード「５２３２」を原点とした絶対座標になっている（図 6.4.1)。その絶対座標化した座標を調べ、まず、２次メッシュコードを求め、次に３次メッシュを求める。その変換方法は

１．座標値が 1000000 以上の場合は上２桁を調べその値が
　１０→２　１１→３　１２→４　１３→４　１４→５　１５→６　１６→７
のように２次メッシュコードの下２桁を求める。

２．座標値が 1000000 未満の場合は上１桁を調べその値が
　０→０　１→１　２→２　３→３　４→４　５→５　６→６　７→７　８→０　９→１

のように２次メッシュコードの下２桁を求める。なお、８と９の場合は１次メッシュコードがｘならば下から１桁目が、ｙならば下から３桁目が１つプラスされる。

３.座標値の下５桁を読み取り下５桁が、０～０９９９９ならば３次メッシュコードは０、１００００～１９９９９ならば１のようにして３次メッシュコードを求める。

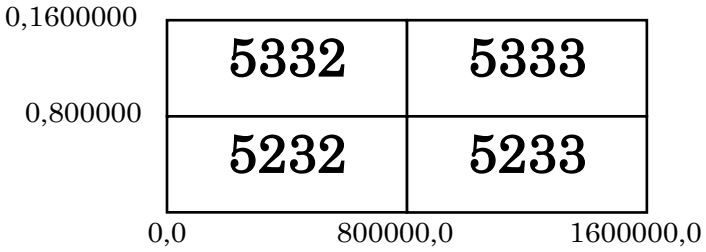
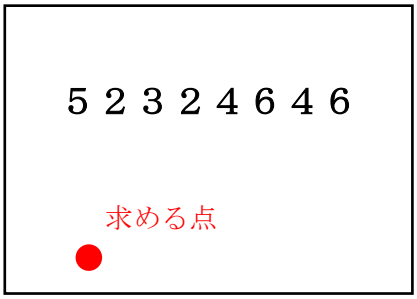


図 6.4.1：１次メッシュコードと絶対座標値

次に、例を示す。（座標値「３２８９２０，１０２３７２１」の場合）
・図 6.4.1 より１次メッシュコードは５３３２である。
・ｘ座標の上１桁目が３であり、ｙ座標の上２桁が１０なので２次メッシュコードは５３３２２３である。
・ｘ座標の下５桁は２８９２０であり、ｙ座標の下５桁は２３７２１であるので３次メッシュコードは５２３２２３２２である。

この３次メッシュコードから「６．３．地域メッシュポリゴン」で述べた方法により、日本測地系の３次メッシュコード左下の経度緯度を求める。この求めた日本測地系の経度緯度を国土地理院測地部が配布している変換パラメータ（URL：http://vldb.gsi.go.jp/sokuchi/coordinates/localtrans.html）[1]を用いて変換する。このパラメータは１行毎に３次メッシュコードとその３次メッシュコードの左下の日本測地系経度緯度を世界測地系に変換する補正量が格納されている。

実際の変換方法は「６．３．地域メッシュポリゴン」と同様に、求める座標値を含む３次メッシュを調べ、そのメッシュコードに対する補正量（注１）を、変換パラメータファイルをサーチして、求める（図参照）。補正量は左下の点の補正量なので、求める点が左下の点でない場合はバイリニア補間法（注２）を使用する。



補正量

	dB（分）緯度	dL（分）経度
52324646	11.43329	-9.26215

図 6.4.2：測地系の変更

注１）補正量というのは旧座標値の準拠する楕円体で新座標値を計算し、その経度差、緯度差を求め、これらの格子点の座標差に旧座標値を加え、格子点の新座標値（ベッセル楕円体）を求め、楕円体高と合わせて世界測地系に変換し、この座標から旧座標値を引いた格子点の座標変換パラメータのことである。

注２）バイリニア補間法とは求める点の値を、周りの 4 つの格子点の値を 2 つのペアに分け、それぞれの線形補間を行、さらに出てきた点で再び線形補間を行うことによって求める方法である。

具体的には，下図を使って説明する。この図において，Z00 は点（０，０）、Z01 は点（０，１），Z10 は点（１，０）、Z11 は点（１，１）での既知の値とし、点（x，y）での値 Z を求める。

最初に Z00 と Z10 を使って、点（x，0）での値 Z0 を次式で求める。

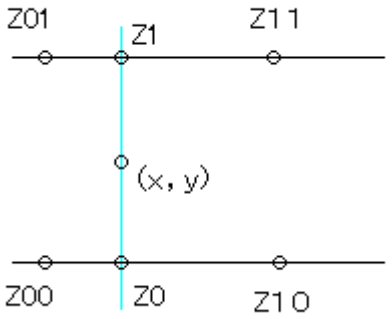
$Z0 = x \times Z10 + (1-x) \times Z00$

次に Z01 と Z11 を使って、点（x，1）での値 Z1 を次式で求めまる。

$Z1 = x \times Z11 + (1-x) \times Z01$

最後に Z0 と Z1 を使って、点（x，y）での値 Z を次式で求める。

$Z = y \times Z1 + (1-y) \times Z0$



6.5. DB 格納

DB に流域境界データを格納する。

表 6.5.1 : DB 内での流域境界のテーブル

Table_name : ryuiki			
gid:serial 型	id:integer 型	水系域コード:text 型	the_geom:Geometry 型

表 6.5.1 は流域境界のテーブルの構成である。Geometry 型の the_geom は Polygon である。

第７章　　表示システム

このシステムは河川流域データを KML ファイルにすることで Google Earth 上に描画し、表示するシステムである。以下にシステムの流れを説明する。

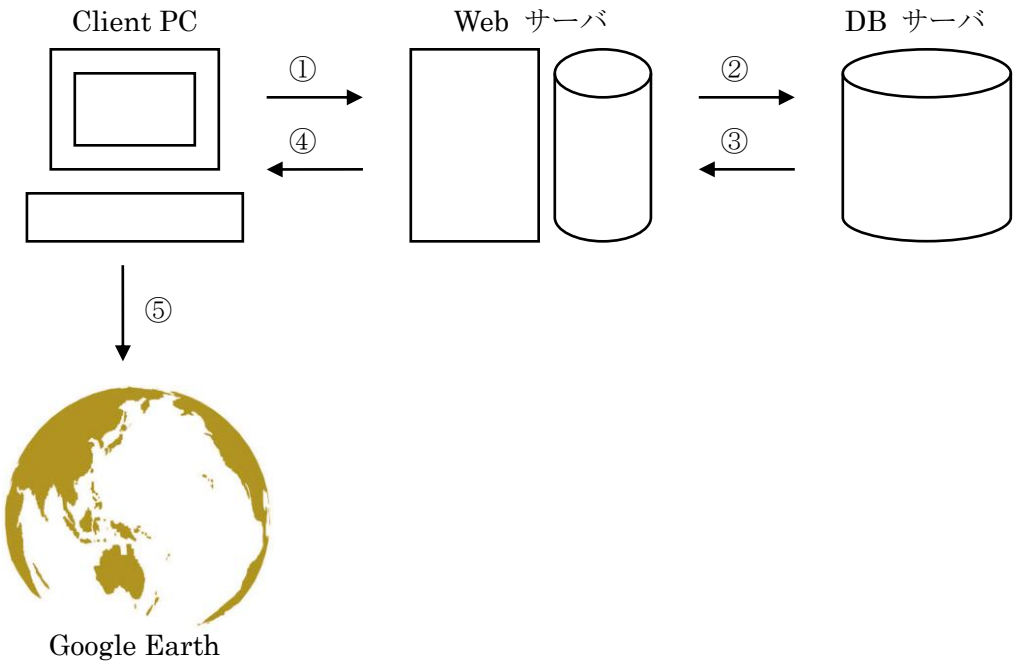


図 7.1.1：表示システムの説明

① ユーザがブラウザを使用して Web サーバに対し「河川流域データを表示する」というリクエストを送る。
② Web サーバではリクエストを受け取り、サーブレットを起動させる。その後サーブレットを用いて DB サーバに対して SQL 文を送る。
SQL 文：[select AsText(the_geom) from ryuiki;]
③ DB サーバでは SQL 文に従いデータを検索し、その結果を Web サーバに送る。
④ Web サーバは検索結果を元に KML ファイルを作成し、KML ファイルをクライアントに送る。
⑤ ユーザは受け取った KML ファイルを元に Google Earth を用いて河川流域データを Google Earth 上に描画し、表示する。

KML（Keyhole Markup Language）とは Google Earth などで地理情報を表示されるために使われているファイルフォーマットであり、KML はタグベース構造で XML を基にしている。さらに、KML を ZIP 圧縮することで KMZ となる。

KML は HTML や XML と似たような方法で処理されているので、Google Earth は KML のブラウザ機能を果たす。

```
<?xml version='1.0' encoding='UTF-8'?>
<kml xmlns='http://earth.google.com/kml/2.2' xmlns:gx='http://www.google.com/kml/ext/2.2'
xmlns:kml='http://www.opengis.net/kml/2.2' xmlns:atom='http://www.w3.org/2005/Atom'>
<Document>
  <name>dairyuiki KML</name>
  <description>hikawa dairyuiki kml</description>
  <Style id='black'>
    <LineStyle id='black2_line'>
      <color>ff000000</color>
    </LineStyle>
    <PolyStyle>
      <fill>0</fill>
    </PolyStyle>
  </Style>
  <StyleMap id='black0'>
    <Pair>
      <key>normal</key>
      <styleUrl>#black2</styleUrl>
    </Pair>
    <Pair>
      <key>highlight</key>
      <styleUrl>#black2</styleUrl>
    </Pair>
  </StyleMap>
  <Style id='black1'>
    <LineStyle id='black2_line'>
      <color>ff000000</color>
    </LineStyle>
  </Style>
  <Style id='black2'>
    <LineStyle id='black2_line'>
      <color>ff000000</color>
      <width>3</width>
    </LineStyle>
    <PolyStyle>
      <fill>0</fill>
    </PolyStyle>
  </Style>
  <Folder>
    <name>hikawa dairyuiki kml</name>
    <description>daihikawa ryuiki kml</description>
      <Placemark>
        <name>1</name>
        <styleUrl>#black0</styleUrl>
        <Polygon>
          <outerBoundaryIs>
            <LinearRing>
```

```
                <coordinates>
                  132.897707030556,35.1034798055556
                  …
                  132.8956533,35.1046588722222
                  132.8964033,35.1040822055555
                  132.897707030556,35.1034798055556
                </coordinates>
              </LinearRing>
            </outerBoundaryIs>
          </Polygon>
        </Placemark>
  </Folder>
</Document>
</kml>
```

経度・緯度を記述

図 7.2.1：KML ファイルの例

図 7.2.1 のように HTML 文章のようにタグを用いることで線などの図形を表現することが出来る。

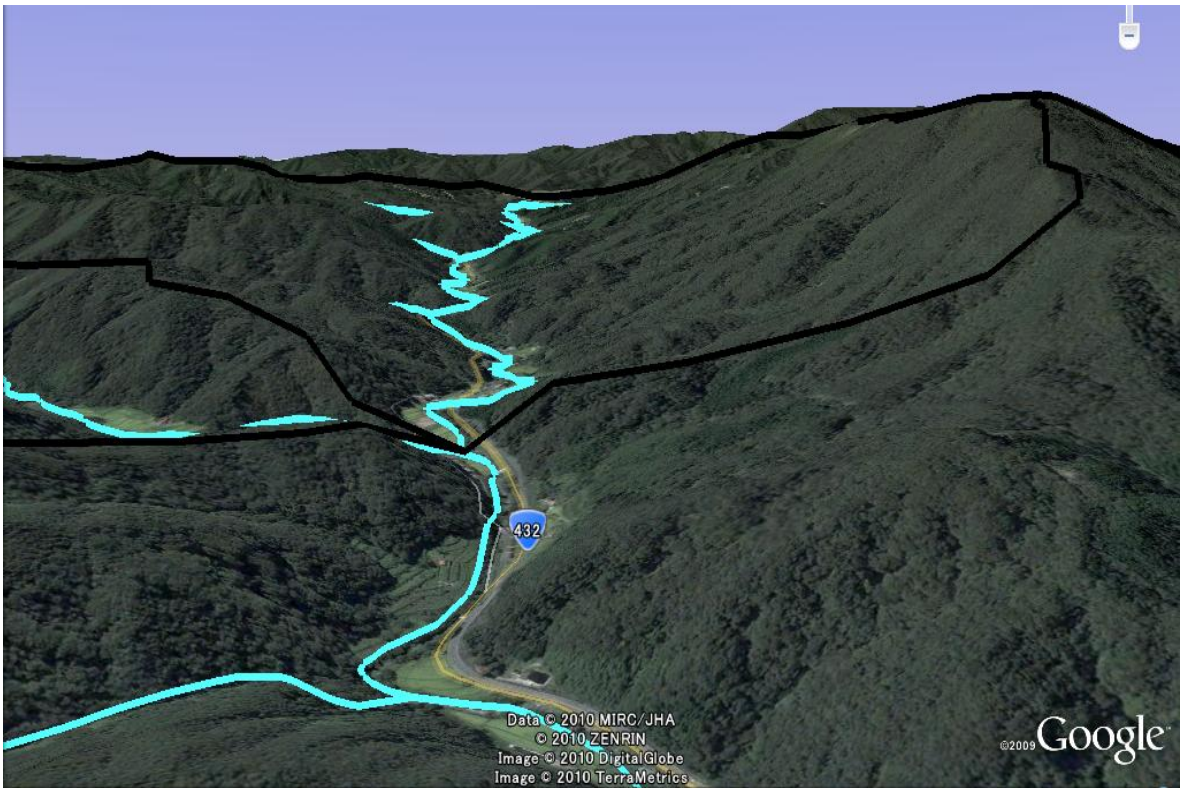


図 7.2.2：表示結果

図 7.2.2 では、黒い線が流域境界データを表しており、右上を見ると山の稜線に沿っていることが分かる。また、水色の線が河川座標データを表している。

第８章　終論

本研究では Google Earth を用いることで河川流域データを地図上に表示することが出来た。また、空間検索を目的とした流域境界データの最外周を抜き出すことが出来た。

今後は、気象情報や水位情報などのリアルタイムデータを随時取得しデータベースに格納、データの空間検索を行い、検索結果を河川流域データと同時に描画し、表示するシステムを作成していきたいと考えている。

謝辞

本研究にあたり、最後まで熱心な御指導をいただきました田中章司郎教授には心より御礼申し上げます。また、同じ研究室の宇垣貴裕さん両間成利さん李さん韋さんには本研究に関しまして、数々の御協力と御助言を頂きました。厚く御礼申し上げます。

なお、本論文、本研究で作成したプログラム及びデータ、並びに関連する発表資料等の全ての知的財産権を本研究の指導教官である田中章司郎教授に譲渡致します。

文献

[1] Google Earth
http://earth.google.com/intl/ja/

[2] Google Maps
http://maps.google.co.jp/maps

[3] PostgreSQL
http://www.postgresql.org/

[4] PostGIS
http://postgis.refractions.net/

[5] OGC
http://www.opengeospatial.org/

[6] 島村和義（2005）「ブラウザ環境における河川流域境界データ表示システムの試作」島根大学　総合理工学部　数理・情報システム学科　卒業論文

[7] 建設省国土地理院監修（財）日本地図センター編集・発行（平成４年７月１日）『数値地図ユーザーズガイド』

[8] 今井拓也（2006）「動的に描画点数を考慮した非同期数値地図表示システムの設計と実装」島根大学　総合理工学部　数理・情報システム学科　卒業論文

[9] 国土数値情報ダウンロードサービス
http://nlftp.mlit.go.jp/ksj/index.html

[10] 国土数値情報データ変換ツール
http://nlftp.mlit.go.jp/ksj/jpgis/jpgis_tool.html

[11] ESRI
http://www.esrij.com/

[12] 木村眞吾（2007）「目的別空間データベースサーバを用いた Web システムの設計と実装」島根大学　総合理工学部　数理・情報システム学科　卒業論文