Конформное преобразование

Обсудить в форуме Комментариев — 22

Эта страница опубликована в основном списке статей сайта
по адресу http://gis-lab.info/qa/helmert2d.html

Конформное преобразование на плоскости широко используется в геодезии при создании местных координатных систем на небольшие территории, ограниченные размерами населённого пункта.

Содержание

Введение

Следующие формулы связывают координаты точек *x*, *y*, заданные в местной системе координат (МСК), и координаты *X*, *Y*, заданные в государственной системе координат (ГСК):

$$X = X_0 + m\left[(x - x_0)\cos\theta - (y - y_0)\sin\theta\right]$$
$$Y = Y_0 + m\left[(x - x_0)\sin\theta + (y - y_0)\cos\theta\right]$$

где *m* — масштабный множитель, ϑ — угол разворота, X_0, Y_0, x_0, y_0 — координаты одного из геодезических пунктов в ГСК и МСК, как правило. Этот набор параметров называется «ключ».

Исходный материал для определения ключа — пары координат пунктов геодезической сети, полученные из независимого уравнивания одних и тех же измерений в МСК и в ГСК[1]. В зависимости от класса пунктам (вернее, соответствующим парам уравнений) назначаются веса *p*.

Алгоритм нахождения параметров

Конформное преобразование представляется следующей математической моделью:

$$X' = a_0 + a_1 x - b_1 y$$
$$Y' = b_0 + b_1 x + a_1 y$$

Определению подлежат четыре параметра: a_0, b_0, a_1, b_1,.

Очевидно, конформное преобразование является частным случаем аффинного.

Шаг 1: вычисление взвешенных средних

$$x_c = \frac{\sum_{i=1}^{n} p_i x_i}{\sum_{i=1}^{n} p_i} \quad y_c = \frac{\sum_{i=1}^{n} p_i y_i}{\sum_{i=1}^{n} p_i} \quad X_c = \frac{\sum_{i=1}^{n} p_i X_i}{\sum_{i=1}^{n} p_i} \quad Y_c = \frac{\sum_{i=1}^{n} p_i Y_i}{\sum_{i=1}^{n} p_i}$$

Шаг 2: перенос осей в центр масс

$$\Delta x_i = x_i - x_c \quad \Delta y_i = y_i - y_c \quad \Delta X_i = X_i - X_c \quad \Delta Y_i = Y_i - Y_c$$

Шаг 3: вычисление a_1 и b_1

$$S_1 = \sum_{i=1}^{n} p_i \Delta X_i \Delta x_i \qquad S_2 = \sum_{i=1}^{n} p_i \Delta Y_i \Delta y_i$$

$$S_3 = \sum_{i=1}^{n} p_i \Delta Y_i \Delta x_i \qquad S_4 = \sum_{i=1}^{n} p_i \Delta X_i \Delta y_i$$

$$S_5 = \sum_{i=1}^{n} p_i \left(\Delta x_i^2 + \Delta y_i^2 \right)$$

$$a_1 = \frac{S_1 + S_2}{S_5} \qquad b_1 = \frac{S_3 - S_4}{S_5}$$

Шаг 4: вычисление a_0 и b_0

$$a_0 = X_c - a_1 x_c + b_1 y_c$$
$$b_0 = Y_c - b_1 x_c - a_1 y_c$$

Шаг 5: вычисление невязок

$$v_{xi} = X_i - X_i' \quad v_{yi} = Y_i - Y_i'$$

Невязки позволяют выявить точки, плохо укладывающиеся в полученную модель. Классическая процедура удаляет такие «отлетающие» точки, после чего вычисление параметров повторяется без них. Робастные процедуры переназначают веса уравнениям в соответствии с невязками, и повторное вычисление повторяется с полным набором точек при том, что «отлетающие» влияют на результат незначительно.

Кроме того, невязки необходимы для оценки точности измерений и результатов.

Шаг 6: вычисление ключа

Вычислим масштабный множитель и угол разворота:

$$m = \sqrt{a_1^2 + b_1^2} \qquad \theta = \operatorname{arc\,tg}\frac{b_1}{a_1}$$

Выберем *j*-й пункт с малыми невязками по возможности в середине массива точек. Его координаты в обеих системах X_j, Y_j, x_j, y_j становятся параметрами X_0, Y_0, x_0, y_0.

Пример вычисления параметров

Даны координаты четырёх пунктов *x*, *y* в исходной системе, *X*, *Y* в конечной системе с весами *p*:

i	x_i	y_i	X_i	Y_i	p_i
1	1334.71	285.94	83477.64	87377.60	1.0
2	563.67	−5197.34	82557.14	81916.51	1.0
3	4444.27	1153.79	86610.19	88160.39	1.0
4	−252.07	2881.90	81962.05	90016.34	1.0
				$\sum p =$	4.0

i	$p_i x_i$	$p_i y_i$	$p_i X_i$	$p_i Y_i$
1	1334.71	285.94	83477.64	87377.60
2	563.67	−5197.34	82557.14	81916.51
3	4444.27	1153.79	86610.19	88160.39
4	−252.07	2881.90	81962.05	90016.34
∑ =	6090.58	−875.71	334607.02	347470.84

Взвешенные средние:

x_c	y_c	X_c	Y_c
1522.645	−218.9275	83651.755	86867.71

Перенос осей в центр масс:

i	Δx_i	Δy_i	ΔX_i	ΔY_i
1	−187.935	504.8675	−174.115	509.89
2	−958.975	−4978.4125	−1094.615	−4951.20
3	2921.625	1372.7175	2958.435	1292.68
4	−1774.715	3100.8275	−1689.705	3148.63

i	$p_i \Delta x_i \Delta X_i$	$p_i \Delta y_i \Delta Y_i$	$p_i \Delta x_i \Delta Y_i$	$p_i \Delta y_i \Delta X_i$	$p_i (\Delta x_i^2 + \Delta x_i^2)$
1	32722.3	257426.9	−95826.2	−87905.0	290210.8
2	1049708.4	24649116.0	4748077.0	5449445.0	25704224.1
3	8643437.7	1774484.5	3776726.2	4061095.5	10420246.0
4	2998744.8	9763358.5	−5587920.9	−5239483.7	12764744.5
∑ =	12724613.2	36444385.8	2841056.2	4183151.8	49179425.3
	S_1	S_2	S_3	S_4	S_5

Параметры конформного преобразования:

a_1	b_1
0.99978799	−0.02728978

a_0	b_0
82135.407	87128.144

Невязки:

i	v_{xi}	v_{yi}
1	0.002	0.001
2	0.016	−0.013
3	−0.032	−0.016
4	0.013	0.028

Масштаб и разворот:

m	ϑ
1.00016037	−1°33′48.72″

Сконструируем ключ на основе первого геодезического пункта:

X_0	Y_0	x_0	y_0	m	ϑ
83477.64	87377.60	1334.71	285.94	1.00016037	−1°33′48.72″

Пример программной реализации

Вот листинг простой утилиты на языке C:

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int main(int argc, char *argv[])
{
  char buf[1024], name[32];
  double x[2], y[2], wgt;
  double xc[2], yc[2];
  double dx[2], dy[2], dz[2];
  double a[2][2], scale, rotation;
  double s[5];
  int i;
  FILE *fp0, *fp1;

  if (argc < 4) {
    printf("usage: findkey input-file key-file var-file\n");
    exit(EXIT_FAILURE);
  }

  if ((fp0 = fopen(argv[1], "r")) == NULL) {
    printf("can't open %s\n", argv[1]);
    exit(EXIT_FAILURE);
  }

  /* подсчитать сумму координат */
  for (i = 0; i < 5; i++)
    s[i] = 0.;
  while (fgets(buf, 1024, fp0) != NULL) {
    sscanf(buf, "%s %lf %lf %lf %lf %lf",
           name, &x[0], &x[1], &y[0], &y[1], &wgt);
    s[0] += x[0] * wgt;
    s[1] += x[1] * wgt;
    s[2] += y[0] * wgt;
    s[3] += y[1] * wgt;
    s[4] += wgt;
  }
  rewind(fp0);

  /* найти центр масс */
  for (i = 0; i < 2; i++) {
```

```
  xc[i] = s[i] / s[4];
  yc[i] = s[2 + i] / s[4];
}

/* подсчитать сумму произведений */
for (i = 0; i < 5; i++)
  s[i] = 0.;
while (fgets(buf, 1024, fp0) != NULL) {
  sscanf(buf, "%s %lf %lf %lf %lf %lf",
         name, &x[0], &x[1], &y[0], &y[1], &wgt);
  /* вычислить разности */
  dx[0] = x[0] - xc[0];
  dx[1] = x[1] - xc[1];
  dy[0] = y[0] - yc[0];
  dy[1] = y[1] - yc[1];
  /* суммировать */
  s[0] += dx[0] * dy[0] * wgt;
  s[1] += dx[1] * dy[1] * wgt;
  s[2] += dx[0] * dy[1] * wgt;
  s[3] += dx[1] * dy[0] * wgt;
  s[4] += (dx[0] * dx[0] + dx[1] * dx[1]) * wgt;
}
rewind(fp0);

/* найти первичные параметры */
a[1][0] = (s[0] + s[1]) / s[4];
a[1][1] = (s[2] - s[3]) / s[4];
a[0][0] = yc[0] - a[1][0] * xc[0] + a[1][1] * xc[1];
a[0][1] = yc[1] - a[1][1] * xc[0] - a[1][0] * xc[1];

/* найти вторичные параметры */
scale = hypot(a[1][0], a[1][1]);
rotation = atan2(a[1][1], a[1][0]);

/* вывести параметры в файл ключа */
if ((fp1 = fopen(argv[2], "w")) == NULL) {
  printf("can't create %s\n", argv[2]);
  exit(EXIT_FAILURE);
}
fprintf(fp1, "%.3f\n", a[0][0]);
fprintf(fp1, "%.3f\n", a[0][1]);
fprintf(fp1, "%.12f\n", a[1][0]);
fprintf(fp1, "%.12f\n", a[1][1]);
fprintf(fp1, "%.12f\n", scale);
fprintf(fp1, "%+.10f\n", rotation * 180. / M_PI);
fclose(fp1);

/* вывести данные вместе с невязками */
if ((fp1 = fopen(argv[3], "w")) == NULL) {
  printf("can't create %s\n", argv[3]);
  exit(EXIT_FAILURE);
}
while (fgets(buf, 1024, fp0) != NULL) {
  sscanf(buf, "%s %lf %lf %lf %lf %lf",
         name, &x[0], &x[1], &y[0], &y[1], &wgt);
  /* "наблюдённые" dx, dy */
  dx[0] = x[0] - xc[0];
  dx[1] = x[1] - xc[1];
  dy[0] = y[0] - yc[0];
  dy[1] = y[1] - yc[1];
  /* "вычисленные" dy */
  dz[0] = a[1][0] * dx[0] - a[1][1] * dx[1];
  dz[1] = a[1][1] * dx[0] + a[1][0] * dx[1];
  fprintf(fp1, "%s %.3f %.3f %.3f %.3f %g %.3f %.3f\n",
          name, x[0], x[1], y[0], y[1], wgt, dy[0] - dz[0], dy[1] - dz[1]);
```

```
  }
  fclose(fp1);

  fclose(fp0);

  return 0;
}
```

Сохраним код в файл **findkey.c**. Исполняемый модуль можно создать, например, компилятором **gcc**:

```
$ gcc -o findkey findkey.c -lm
```

Для MS Windows можно загрузить уже скомпилированную программу.

Утилита **findkey** запускается в командной строке с именами трёх файлов в качестве аргументов: входной файл данных, выходные файл параметров и файл невязок.

Подготовим файл данных **data.dat**, структура которого соответствует таблице исходных данных, т. е. содержит в колонках названия пунктов, входные координаты *x*, *y*, выходные координаты *X*, *Y*, веса:

```
1 1334.71   285.94 83477.64 87377.60 1.0
2  563.67 -5197.34 82557.14 81916.51 1.0
3 4444.27  1153.79 86610.19 88160.39 1.0
4 -252.07  2881.90 81962.05 90016.34 1.0
```

После запуска программы

```
$ findkey data.dat key.dat var.dat
```

получим параметры **key.dat**

```
82135.407
87128.144
0.9997879942
-0.0272897781
1.0001603698
-1.56353244
```

и невязки **var.dat**

```
1 1334.710 285.940 83477.640 87377.600 1 0.002 0.001
2 563.670 -5197.340 82557.140 81916.510 1 0.016 -0.013
3 4444.270 1153.790 86610.190 88160.390 1 -0.032 -0.016
4 -252.070 2881.900 81962.050 90016.340 1 0.013 0.028
```

Заключение

Положенные в основу статьи формулы называются в учебниках геодезии «неполными» в противоположность «полным». Дело в том, что при большом удалении объекта от осевого меридиана исходной проекции Гаусса-Крюгера возникает значительный градиент масштаба изображения в направлении запад-восток. Чтобы компенсировать возникающие при этом специфические искажения конформного отображения, в «полные» выражения добавляют необходимые члены разложений формул проекций. Разумеется, такой подход несовершенен, как любые ограниченные разложения. В статье Добавление местной координатной системы в GIS в качестве альтернативы предлагается переход от ГСК к проекции с нулевым градиентом масштаба в центре объекта или вблизи него, что делает «полные» формулы ненужными.

Примечания

↑ ГКИНП (ОНТА)-01-271-03 Руководство по созданию и реконструкции городских геодезических сетей с использованием спутниковых систем ГЛОНАСС/GPS.

Ссылки

- Convert Local Coordinate Systems to Standard Coordinate Systems, McCoy J., 2012
- Coordinate transformations in surveying and mapping, Deakin R.E., 2004
- Coordinate transformations, Knippers R., 2009
- Аффинные преобразования - математика
- Добавление местной координатной системы в GIS
- Полиномиальные преобразования
- Полиномиальные преобразования - примеры реализации
- Среднеквадратичная ошибка (RMSE)

Обсудить в форуме Комментариев — 22

Последнее обновление: 2014-09-11 14:55

Дата создания: 9.03.2013
Автор(ы): ErnieBoyd