Задачи на сфере: прямая геодезическая задача

Обсудить в форуме Комментариев — 21

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

Прямая геодезическая задача — это нахождение положения точки по координатам исходного пункта и значениям начального направления и расстояния.

Содержание

Общие положения

В качестве модели Земли принимается сфера с радиусом R, равным среднему радиусу земного эллипсоида. Аналогом прямой линии на плоскости является геодезическая линия на поверхности. На сфере геодезическая линия — дуга большого круга.

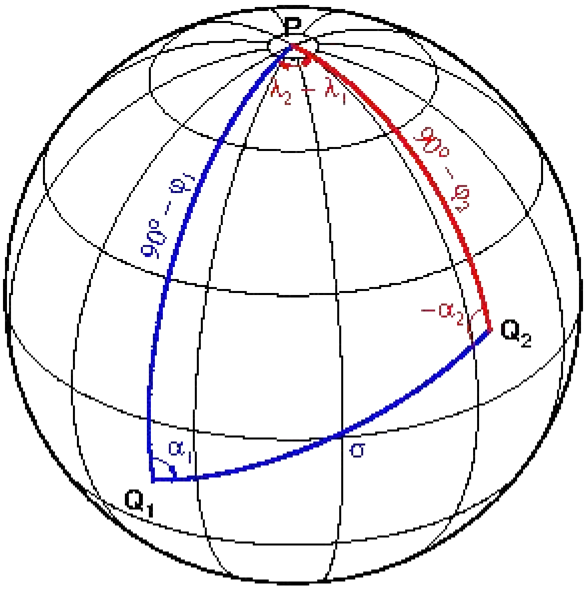
Введём следующие обозначения:

- φ — географическая широта,
- λ — географическая долгота,
- α — азимут дуги большого круга,
- σ — сферическое расстояние (длина дуги большого круга, выраженная в долях радиуса шара).

Линейное расстояние по дуге большого круга s связано со сферическим расстоянием σ формулой $s = R\,\sigma$.

Прямая и обратная геодезические задачи являются важными элементами более сложных геодезических задач.

Постановка задачи



Прямая геодезическая задача

Исходные данные координаты пункта Q_1, начальное направление и расстояние на сфере — $\varphi_1, \lambda_1, \alpha_1, \sigma$.
Определяемые величины координаты пункта Q_2 — φ_2, λ_2.

На рисунке синим цветом выделены заданные элементы сферического треугольника, красным цветом неизвестные.

Алгоритм

Существует великое множество подходов к решению поставленной задачи. Рассмотрим простой и надёжный векторный метод.

Последовательность решения:

1. преобразовать углы (90° − σ) и (180° − α_1) в декартовы координаты,
2. развернуть координатные оси вокруг оси *Y* на угол (φ_1 − 90°),
3. развернуть координатные оси вокруг оси *Z* на угол $-\lambda_1$,
4. преобразовать декартовы координаты в сферические.

Если третий пункт пропустить, на выходе вместо долготы λ_2 получится разность долгот ($\lambda_2 - \lambda_1$), что упростит алгоритм. Останется только прибавить долготу первого пункта. С другой строны, благодаря третьему пункту долгота λ_2 всегда будет в диапазоне −180°, +180°.

Пример реализации алгоритма в виде функции языка Си:

```
/*
 * Решение прямой геодезической задачи
 *
 * Аргументы исходные:
 *     pt1  - {широта, долгота} точки Q1
 *     azi  - азимут начального направления
 *     dist - расстояние (сферическое)
 *
 * Аргументы определяемые:
 *     pt2  - {широта, долгота} точки Q2
 */
void SphereDirect(double pt1[], double azi, double dist, double pt2[])
{
  double pt[2], x[3];

  pt[0] = M_PI_2 - dist;
  pt[1] = M_PI - azi;
  SpherToCart(pt, x);                 // сферические -> декартовы
```

```
  Rotate(x, pt1[0] - M_PI_2, 1);       // первое вращение
  Rotate(x, -pt1[1], 2);               // второе вращение
  CartToSpher(x, pt2);                 // декартовы -> сферические

  return;
}
```

Следует заметить, что прямая и обратная задача математически идентичны, и алгоритмы их решения зеркально отражают друг друга.

Преобразование сферических координат в декартовы

$$x = \cos\varphi\cos\lambda$$
$$y = \cos\varphi\sin\lambda$$
$$z = \sin\varphi$$

В данном случае в качестве сферических координат φ, λ подставим углы (90° − σ), (180° − α_1).

Реализация на Си:

```
/*
 * Преобразование сферических координат в вектор
 *
 * Аргументы исходные:
 *      y - {широта, долгота}
 *
 * Аргументы определяемые:
 *      x - вектор {x, y, z}
 */
void SpherToCart(double y[], double x[])
{
  double p;

  p = cos(y[0]);
  x[2] = sin(y[0]);
  x[1] = p * sin(y[1]);
  x[0] = p * cos(y[1]);

  return;
}
```

Вращение вокруг оси

Представим оператор вращения вокруг оси *X* на угол ϑ в следующем виде:

$$x' = x$$
$$y' = y\cos\theta + z\sin\theta$$
$$z' = -y\sin\theta + z\cos\theta$$

Операторы вращения вокруг осей *Y* и *Z* получаются перестановкой символов.

Реализация вращения вокруг *i*-ой координатной оси на Си:

```
/*
 * Вращение вокруг координатной оси
 *
 * Аргументы:
 *      x - входной/выходной 3-вектор
 *      a - угол вращения
 *      i - номер координатной оси (0..2)
 */
```

```
void Rotate(double x[], double a, int i)
{
  double c, s, xj;
  int j, k;

  j = (i + 1) % 3;
  k = (i - 1) % 3;
  c = cos(a);
  s = sin(a);
  xj = x[j] * c + x[k] * s;
  x[k] = -x[j] * s + x[k] * c;
  x[j] = xj;

  return;
}
```

Преобразование декартовых координат в сферические

$$\lambda = \operatorname{arctg} \frac{y}{x}$$

$$\varphi = \operatorname{arctg} \frac{z}{\sqrt{x^2 + y^2}}$$

В данном случае в роли сферических координат φ, λ окажутся φ_2, λ_2.

Реализация на Си:

```
/*
 * Преобразование вектора в сферические координаты
 *
 * Аргументы исходные:
 *     x - {x, y, z}
 *
 * Аргументы определяемые:
 *     y - {широта, долгота}
 *
 * Возвращает:
 *     длину вектора
 */
double CartToSpher(double x[], double y[])
{
  double p;

  p = hypot(x[0], x[1]);
  y[1] = atan2(x[1], x[0]);
  y[0] = atan2(x[2], p);

  return hypot(p, x[2]);
}
```

Пример программной реализации

Исходники вышеприведённых функций можно найти в архиве Sph.zip в файле **sph.c**. Кроме того, в файл **sph.h** включены следующие определения:

```
#define A_E 6371.0                          // радиус Земли в километрах
#define Degrees(x)  (x * 57.29577951308232)   // радианы -> градусы
#define Radians(x)  (x / 57.29577951308232)   // градусы -> радианы
```

Теперь напишем программу, которая обращается к функции SphereDirect для решения прямой задачи:

```
#include <stdio.h>
#include <stdlib.h>
#include "sph.h"
```

```
int main(int argc, char *argv[])
{
  char buf[1024];
  double pt1[2], pt2[2];
  double lat1, lon1, azi1, dist, azi2;

  while (fgets(buf, 1024, stdin) != NULL) {
    sscanf(buf, "%lf %lf %lf %lf", &lat1, &lon1, &azi1, &dist);
    pt1[0] = Radians(lat1);
    pt1[1] = Radians(lon1);
    SphereDirect(pt1, Radians(azi1), dist / A_E, pt2);        // Решение прямой задачи
    SphereInverse(pt2, pt1, &azi2, &dist);             // Вычисление обратного азимута
    printf("%f\t%f\t%f\n", Degrees(pt2[0]), Degrees(pt2[1]), Degrees(azi2));
  }
  return 0;
}
```

В архиве Sph.zip этот код находится в файле **dir.c**. Создадим исполняемый модуль **dir** компилятором **gcc**:

```
$ gcc -o dir dir.c sph.c -lm
```

Впрочем, в архиве есть **Makefile**. Для MS Windows готовую программу **dir.exe** можно найти в архиве Sph-win32.zip.

Программа читает данные из стандартного ввода консоли и отправляет результаты на стандартный вывод. Для чтения и записи файлов используются символы перенаправления потока «>» и «<» соответственно. Из каждой строки ввода программа считывает координаты первого пункта φ_1, λ_1, начальный азимут α_1 в градусах и расстояние s в километрах; решает прямую задачу; записывает в строку вывода координаты второго пункта φ_2, λ_2 и обратный азимут α_2 в градусах.

Создадим файл **dir.dat**, содержащий одну строку данных:

```
30 0 44.804060 5001.1309
```

После запуска программы

```
$ dir < dir.dat
```

получим φ_2, λ_2, α_2:

```
52.000000 54.000001 262.415109
```

В архиве Sph-py.zip находятся скрипты на языке Питон. Выполнение скрипта в командной консоли:

```
$ python dir.py dir.dat
```

Решение прямой задачи средствами PROJ.4

В пакет PROJ.4 входит программа **geod**, предназначенная для решения прямых и обратных геодезических задач на сфере. Так выглядит команда обработки файла **dir.dat**:

```
$ geod +a=6371000 -f "%f" +units=km dir.dat
```

Параметр +a определяет радиус сферы, -f — формат вывода угловых величин, +units — единица измерения расстояний. В итоге получим идентичный результат:

```
52.000000 54.000001 -97.584891
```

Различие значений α_2 на 360° объясняется тем, что **dir** выводит азимуты в диапазоне от 0° до 360°, а **geod** от −180° до +180°.

С помощью **geod** можно также расставить промежуточные точки вдоль геодезической линии либо по дуге малого круга на заданном расстоянии от исходного пункта. В обоих случаях нужно задать положение начальной точки параметрами +lat_1, +lon_1 и либо координаты второй точки +lat_2, +lon_1, либо расстояние и азимут ко второй точке +S, +A. За подробностями обращайтесь к документации.

Альтернативные методы

Большая часть других методов основана на сферической тригонометрии. Многие из них используют вычисление φ_2 или $(\lambda_2 - \lambda_1)$ по таким функциям, как синус, косинус или гаверсинус. Это приводит к неоднозначности результатов вблизи особых значений, когда производная функции равна нулю. Такие методы не могут считаться универсальными.

К наиболее надёжным относится следующий способ:

$$\xi = \cos\sigma\cos\varphi_1 - \sin\sigma\sin\varphi_1\cos\alpha_1$$

$$\eta = \sin\sigma\sin\alpha_1$$

$$\operatorname{tg}(\lambda_2 - \lambda_1) = \frac{\eta}{\xi}$$

$$\operatorname{tg}\varphi_2 = \frac{\sin\varphi_1\cos\sigma + \cos\varphi_1\sin\sigma\cos\alpha_1}{\xi\cos(\lambda_2 - \lambda_1) + \eta\sin(\lambda_2 - \lambda_1)}$$

В сферической тригонометрии углы и стороны должны быть в диапазоне 0, 180°. Алгоритмизация формул требует анализа и обработки случаев, когда входные величины не попадают в эти рамки.

Ссылки

- Вычисление расстояния и начального азимута между двумя точками на сфере
- Задачи на сфере: обратная геодезическая задача
- Задачи на сфере: угловая засечка
- Задачи на сфере: линейная засечка
- Краткий справочник по сферической тригонометрии
- man geod – PROJ.4
- Earth radius
- Степанов Н. Н. Сферическая тригонометрия

Обсудить в форуме Комментариев — 21

Последнее обновление: 2014-06-21 10:16

Дата создания: 11.03.2014

Автор(ы): ErnieBoyd