

- [Главная](#)
- [Вопросы и ответы](#)

# Полиномиальные преобразования - примеры реализации

Математические выкладки решения задачи, применяемой при привязке данных

[Обсудить в форуме](#) Комментариев — 16

При операции географической привязки данных, то есть перевода данных из локальной системы координат в географическую или прямоугольную, сам пересчет обычно происходит "за сценой" и его особенности часто понятны пользователю только интуитивно. Эта статья использует [математические выкладки](#) алгоритма и показывает их реализацию, с помощью различных инструментов. Основная цель - быстрая интеграция кода в свое программное обеспечение и просто лучшее понимание процесса привязки.

Так как одно из наиболее часто используемых преобразований при привязке - полиномиальное преобразование 2-й степени, мы иллюстрируем наши расчёты на его примере. Вычисления для аффинного преобразования (оно же полиномиальное преобразование 1-й степени) выполняются аналогичным образом, с меньшим количеством коэффициентов.

## Оглавление

1. [Математика](#)
2. [Тестовый набор данных](#)
3. [Пример пересчета в Excel](#)
4. [Пример пересчета в R](#)
5. [Листы расчётов для MathCad](#)
6. [Процедура для пересчета на Delphi](#)
7. [Процедура для пересчета на C#](#)



### Математика

Математические выкладки описывающие аналитическое решение задачи приводятся [в отдельной статье](#).



### Тестовый набор данных

Для дальнейших пересчетов, зададим тестовый набор данных, на котором будем демонстрировать корректность вычислений. Данный пример иллюстрирует преобразование из локальной системы координат в прямоугольную (спроектированную), но на месте конечных координат могут быть и географического координаты долгота/широта.

Точка	x	y	x'	y'
1	83.786	-36.107	557124.596	5479746.857
2	109.929	-582.929	564344.898	5376737.207
3	1038.000	-434.786	646174.994	5421503.083
4	539.107	-694.036	603772.500	5363472.000

5	831.036	-352.000	626857.500	5433468.000
---	---------	----------	------------	-------------

6	632.786	-219.107	607905.000	5455042.500
---	---------	----------	------------	-------------

Сделаем нашей тестовой точкой точку с координатами:  $x = 500$  и  $y = -300$ . Используя возможности ERDAS IMAGINE получим его вариант предсказания: 596703.345492103, 5437370.8467262

Так же, используя тот же инструмент ERDAS IMAGINE, посмотрим вычисленные им коэффициенты преобразования. Примечание: по какой-то причине, ERDAS IMAGINE показывает коэффициенты не прямой, а обратной задачи, поэтому при вычислении например в Excel, нужно будет поменять местами  $x, y$  и  $x', y'$ . Итак, для нашего тестового набора данных коэффициенты рассчитанные ERDAS IMAGINE следующие:



#### Пример пересчета в Excel

Для быстрой проверки можно воспользоваться [данной таблицей в MS Excel](#), которая использует вышеприведенные расчеты для вычисления трансформации и предсказания новых координат. С помощью этой таблицы, используем тестовый набор данных и сравним коэффициенты преобразования с полученными в ERDAS IMAGINE:

	Коэффициент a	b
0	- 15071.583923079	-24679.2116161184
1 (X)	0.01203458721463060	-0.0017086683749
2 (Y)	0.00267941522144888	0.0040310417433
3 (X <sup>2</sup> )	-0.00000000027645868	0.0000000001492
4 (XY)	-0.00000000011992843	0.0000000000882

5 (Y<sup>2</sup>)      -0.00000000019248837    0.0000000001062

Результат - не отличается от результата ERDAS IMAGINE.



### Пример пересчета в R

Зададим исходные данные:

```
x = c(83.786,109.929,1038.000,539.107,831.036,632.786)
y = c(-36.107,-582.929,-434.786,-694.036,-352.000,-219.107)
x2 = c(557124.596,564344.898,646174.994,603772.500, 626857.5, 607905.000)
y2 = c(5479746.857,5376737.207,5421503.083,5363472.000,5433468.000,5455042.500)
```

Построим матрицу:

```
mat = matrix( c(1, 1, 1, 1, 1, 1, x, y, x^2, x*y, y^2), nrow = 6, ncol = 6)
```

И решим прямую задачу (x,y -> x2,y2) для нахождения коэффициентов a и b:

```
an = solve(mat, x2)
bn = solve(mat, y2)
```

Или обратную задачу (x,y <- x2,y2), для этого нам также понадобится переопределить матрицу:

```
matinv <- matrix( c(1, 1, 1, 1, 1, 1, x2, y2, x2^2, x2*y2, y2^2), nrow = 6, ncol = 6)
aninv = solve(matinv, x)
bninv = solve(matinv, y)
```

Результатом прямой задачи будут следующие наборы коэффициентов, для прямого преобразования:

```
1 5.493158e+05 8.948879e+01 -8.564444e+00 2.322324e-04 2.727497e-04 6.293799e-04
1 5.485053e+06 1.809379e+01 1.888970e+02 -2.225621e-04 -3.448929e-04 -6.190561e-04
```

Создадим тестовую точку и проверим результат:

```
testpoint = c(500, -300)
xpred = an1 + an2* testpoint1 + an3* testpoint2 + an4* testpoint1^2 + an5*
testpoint1*testpoint2 + an6*testpoint2^2
ypred = bn1 + bn2* testpoint1 + bn3* testpoint2 + bn4* testpoint1^2 + bn5*
testpoint1*testpoint2 + bn6*testpoint2^2
```

Наш результат:

```
596703.3
5437371.0
```

Что отличается от результатов ERDAS лишь на доли метра. Что и требовалось доказать.



### Листы расчётов для MathCad

Файлы представляют из себя лист расчётов для среды MathCad (версия 11 и выше), где на примере показано как можно трансформировать координаты с использованием полиномиальных преобразований 1-го и 2-го порядков ([скачать](#)). Прислал Александр Г.



### Процедура для пересчета на Delphi

Для расчетов также понадобится дополнительная [библиотека](#) для осуществления операций с матрицами.

```

map:record
    pnts:array of record
        xr,yr,xg,yg:extended;
географические координаты
    end;
    coeff_x:array of extended;
    coeff_y:array of extended;
    end;

function getx(x,y:extended):extended;var m:word;
begin
    if length(main.map.coeff_x)>0 then
getx:=main.map.coeff_x1+main.map.coeff_x2*x+main.map.coeff_x3*y+main.map.coeff_x4*x*x+m
ain.map.coeff_x5*x*y+main.map.coeff_x6*y*y else getx:=-1;
    end;
function gety(x,y:extended):extended;
begin
    if length(main.map.coeff_y)>0 then
gety:=main.map.coeff_y1+main.map.coeff_y2*x+main.map.coeff_y3*y+main.map.coeff_y4*x*x+m
ain.map.coeff_y5*x*y+main.map.coeff_y6*y*y else gety:=-1;
    end;

procedure map_calculate_ceeff;var aa:TReal2DArray;var i,j:byte;
begin
    // вычисление коэффициентов полинома
    //-----
    // заполнение обратной матрицы
    setlength(aa,7,7);
    for i:=1 to 6 do begin
        aa1,i:=1;
        aa2,i:=main.map.pntsi-1.xg;
        aa3,i:=main.map.pntsi-1.yg;
        aa4,i:=main.map.pntsi-1.xg*main.map.pntsi-1.xg;
        aa5,i:=main.map.pntsi-1.xg*main.map.pntsi-1.yg;
        aa6,i:=main.map.pntsi-1.yg*main.map.pntsi-1.yg;
    end;
    // вычисление обратной матрицы
    if Inverse(aa,6)=false then begin main.show_err('Ошибка. Необходим другой набор
контрольных точек. ');exit;end;

    // умножение с обратной матрицей X
    setlength(main.map.coeff_x,7);for j:=1 to 6 do main.map.coeff_xj:=0;
    for i:=1 to 6 do for j:=1 to 6 do
main.map.coeff_xi:=main.map.coeff_xi+aaj,i*main.map.pntsj-1.xr;
    // умножение с обратной матрицей Y
    setlength(main.map.coeff_y,7);for j:=1 to 6 do main.map.coeff_yj:=0;
    for i:=1 to 6 do for j:=1 to 6 do
main.map.coeff_yi:=main.map.coeff_yi+aaj,i*main.map.pntsj-1.yr;
end;

```



### Процедура для пересчета на C#

Для расчетов также понадобится дополнительная библиотека для осуществления операций с матрицами. Библиотеку LU-декомпозиции для решения системы, а так же пример программы можно скачать [здесь](#). Прислал Deshchenko Sergey.

```

using System;

namespace Transformation
{
    class Program
    {
        static void Main()
        {
            double x1 = { 83.786, 109.929, 1038.000, 539.107, 831.036, 632.786 };

```

```

double y1 = { -36.107, -582.929, -434.786, -694.036, -352.000, -219.107 };
double x2 = { 557124.596, 564344.898, 646174.994, 603772.500, 626857.500,
607905.000 };
double y2 = { 5479746.857, 5376737.207, 5421503.083, 5363472.000,
5433468.000, 5455042.500 };
int n = 6;
int p = 6;

/*****
Входные параметры:
m - Матрица системы.
    Массив с нумерацией элементов 1..N, 1..N.
bForX - Правая часть для X.
bForY - Правая часть для Y.
    Массив с нумерацией элементов 1..N, 1..N.
n - Размерность системы.
p - Количество точек.
a - Решение системы с X.
b - Решение системы с Y.
*****/

double, m = new doublen + 1, n + 1;
double bForX = new doublen + 1;
double bForY = new doublen + 1;
double a = new doublen + 1;
double b = new doublen + 1;

for (int i = 0; i < p; i++)
{
    m1, 1 = n;
    m1, 2 = m2, 1 += x1i;
    m1, 3 = m3, 1 += y1i;
    m1, 4 = m4, 1 = m2, 2 += Math.Pow(x1i, 2);
    m1, 5 = m5, 1 = m2, 3 = m3, 2 += x1i * y1i;
    m1, 6 = m6, 1 = m3, 3 += Math.Pow(y1i, 2);
    m2, 4 = m4, 2 += Math.Pow(x1i, 3);
    m2, 5 = m5, 2 = m3, 4 = m4, 3 += Math.Pow(x1i, 2) * y1i;
    m2, 6 = m6, 2 = m3, 5 = m5, 3 += x1i * Math.Pow(y1i, 2);
    m3, 6 = m6, 3 += Math.Pow(y1i, 3);
    m4, 4 += Math.Pow(x1i, 4);
    m4, 5 = m5, 4 += Math.Pow(x1i, 3) * y1i;
    m4, 6 = m6, 4 = m5, 5 += Math.Pow(x1i, 2) * Math.Pow(y1i, 2);
    m5, 6 = m6, 5 += x1i * Math.Pow(y1i, 3);
    m6, 6 += Math.Pow(y1i, 4);

    bForX1 += x2i;
    bForX2 += x1i * x2i;
    bForX3 += y1i * x2i;
    bForX4 += Math.Pow(x1i, 2) * x2i;
    bForX5 += x1i * y1i * x2i;
    bForX6 += Math.Pow(y1i, 2) * x2i;
    bForY1 += y2i;
    bForY2 += x1i * y2i;
    bForY3 += y1i * y2i;
    bForY4 += Math.Pow(x1i, 2) * y2i;
    bForY5 += x1i * y1i * y2i;
    bForY6 += Math.Pow(y1i, 2) * y2i;
}
{
    System.Console.Write("Error! Degenerate matrix A!");
    System.Console.WriteLine();
    System.Console.ReadKey();
    return;
}
if (!linsolve.solvesystem(m, bForY, n, ref b))
{
    System.Console.Write("Error! Degenerate matrix A!");
}

```

```

        System.Console.WriteLine();
        System.Console.ReadKey();
        return;
    }
    double testPointX1 = 500;
    double testPointY1 = -300;
    double testPointX2 = a1 + a2 * testPointX1 + a3 * testPointY1 + a4 *
Math.Pow(testPointX1, 2) + a5 * testPointX1 * testPointY1 + a6 * Math.Pow(testPointY1,
2);
    double testPointY2 = b1 + b2 * testPointX1 + b3 * testPointY1 + b4 *
Math.Pow(testPointX1, 2) + b5 * testPointX1 * testPointY1 + b6 * Math.Pow(testPointY1,
2);

    System.Console.Write("X = " + testPointX2);
    System.Console.WriteLine();
    System.Console.Write("Y = " + testPointY2);
    System.Console.WriteLine();
    for (int i = 0; i < n; i++)
    {
        System.Console.WriteLine();
        System.Console.Write("a" + i + " = " + ai + 1);
    }
    System.Console.WriteLine();
    for (int i = 0; i < n; i++)
    {
        System.Console.WriteLine();
        System.Console.Write("b" + i + " = " + bi + 1);
    }
    System.Console.ReadKey();
}
}
}

```

[Обсудить в форуме](#) Комментариев — 16

## Ссылки по теме

- [Полиномиальные преобразования](#)
- [Среднеквадратичная ошибка \(RMSE\)](#)
- [Полиномиальные преобразования - математика](#)

Последнее обновление: May 01 2011

Дата создания: 02.03.2008

Автор(ы): [Максим Дубинин](#)