1. **Dwuwymiarowa transformacja przez podobieństwo (4 parametry). Dwuwymiarowa transformacja konforemna. Rozwiązanie ścisłe.**

   Cechy: zachowuje kształty

   Zastosowania: pomiary z różnych epok pomiarowych są transformowane do jednolitego układu współrzędnych.

   Cechy:

   a) Zmiana skali – jednolita w obu kierunkach – 1 parametr
   b) Obrót – aby osie układów były || - 1 parametr
   c) Translacja - aby początki układów się pokrywały – 2 parametry

A. Zmiana Skali:

   $x'=Sx$

   $y'=Sy$ gdzie x',y'- współrzędne przeskalowane, S – współczynnik skali

B. Obrót (zgodnie z ruchem wskazówek zegara dla układów fotogrametrycznych [y na N]):

   $X'=x'\cos\theta - y'\sin\theta$

   $Y'=x'\sin\theta + y'\cos\theta$

C. Translacja:

   $X=X'+T_x$

   $Y=Y'+T_y$

   Zatem koniunkcja warunków A i B i C:

   $X=(S*\cos\theta)\, x - (S*\sin\theta)y + T_x$

   $Y= (S*\sin\theta)\, x + (S*\cos\theta)y + T_y$

   Niech:

   $(S*\cos\theta)=a$; $(S*\sin\theta)=b$; $T_x=c$; $T_y=d$

   Zatem:

   $ax-by+c=X+v_x$

   $bx+ay+d=Y+v_y$

   Więc jeżeli punktami dostosowania w zbiorze danych są punkty np. 100, .., n to otrzymujemy:

   $ax_{100}-by_{100}+c=X_{100}+v_{x100}$

   $bx_{100}+ay_{100}+d=Y_{100}+v_{y100}$

   ....

   ....

   $ax_n-by_n+c=X_n+v_{xn}$

   $bx_n+ay_n+d=Y_n+v_{yn}$

   zatem układ liniowy AX=L+V gdzie

$$A = \begin{bmatrix} x_{100} & -y_{100} & 1 & 0 \\ y_{100} & x_{100} & 0 & 1 \\ \cdots & \cdots & \cdots & \cdots \\ x_n & -y_n & 1 & 0 \\ y_n & x_n & 0 & 1 \end{bmatrix};\ X = \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix}; L = \begin{bmatrix} X_{100} \\ Y_{100} \\ \cdots \\ X_n \\ Y_n \end{bmatrix}; V = \begin{bmatrix} V_{x100} \\ V_{Y100} \\ \cdots \\ V_{Xn} \\ V_{Yn} \end{bmatrix}$$

$Więc: \theta = tan^{-1}\frac{b}{a}\ oraz\ S = \frac{a}{cos\theta}$

2. **Dwuwymiarowa transformacja afiniczna (6 parametrów) – osobny współczynnik skali dla każdej z osi. Rozwiązanie ścisłe.**

   Zastosowania np. w fotogrametrii dla orientacji wewnętrznej – transformacja układu zdjęcia (arbitralnie przyjętego) do układu współrzędnych kamery

   $ax+by+c=X+v_x$
   $dx+ey+f=Y+v_y$

   jest to układ liniowy zatem konieczne są <u>co najmniej 3 punkty dostosowania</u>
   Więc jeżeli punktami dostosowania w zbiorze danych są punkty np. 100, .., n to otrzymujemy:

   $ax_{100}+by_{100}+c=X_{100}+v_{x100}$
   $dx_{100}+ey_{100}+f=Y_{100}+v_{y100}$
   ….
   ….
   $ax_n+by_n+c=X_n+v_{xn}$
   $dx_n+ey_n+f=Y_n+v_{yn}$

   Zatem w postaci macierzowej

$$A = \begin{bmatrix} x_{100} & y_{100} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{100} & y_{100} & 1 \\ x_{101} & y_{101} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_{101} & y_{101} & 1 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ x_n & y_n & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x_n & y_n & 1 \end{bmatrix};\ X = \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix}; L = \begin{bmatrix} X_{100} \\ Y_{100} \\ X_{101} \\ Y_{101} \\ \cdots \\ X_n \\ Y_n \end{bmatrix}; V = \begin{bmatrix} V_{x100} \\ V_{Y100} \\ V_{x101} \\ V_{Y101} \\ \cdots \\ V_{Xn} \\ V_{Yn} \end{bmatrix}$$

3. **Dwuwymiarowa transformacja rzutowa (przekształcenie rzutowe). Transformacja ośmioparametrowa. Rozwiązanie ścisłe.**

Zastosowanie: transformacja dwuwymiarowego układu współrzędnych na inny **NIERÓWNOLEGLY** układ współrzędnych.

$$X = \frac{a_1 x + b_1 y + c_1}{a_3 x + b_3 y + 1}$$
$$Y = \frac{a_2 x + b_2 y + c_2}{a_3 x + b_3 y + 1}$$

Zauważ, że jeżeli $a_3 = b_3 = 0$ to jest to transformacja afiniczna. Musimy wyznaczyć 8 parametrów zatem konieczne są co najmniej 4 punkty dostosowania.
Ponieważ równanie nie jest nieliniowe musimy je zlinearyzować.

PRZYPOMNNIENIE {

LSF/MNK dla systemów **nieliniowych**

Algorytm ogólny:

1. Zapis rozwinięcia każdego równania w szereg Tylora
2. Wybór wartości niewiadomych (przybliżenie)
3. Obliczenie 1 wyrównania – uzyskujemy poprawki do niewiadomych
4. Poprawienie wartości przybliżonych o poprawki
5. Powtarzanie kroków 3 i 4 aż poprawki << kryterium

Ogólnie:

JX=K+V; J-jakobian współczynnika równań zlinearyzowanych

$$J = \begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \cdots & \frac{\partial F_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial F_m}{\partial x_1} & \cdots & \frac{\partial F_m}{\partial x_n} \end{bmatrix}; x = \begin{bmatrix} dx_1 \\ \vdots \\ dx_n \end{bmatrix}$$
$$K = \begin{bmatrix} L_1 - F_1(x_1, x_2, \dots, x_n) \\ \vdots \\ L_m - F_m(x_1, x_2, \dots, x_n) \end{bmatrix}; V = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}$$

Zatem:

$X=(J^T J)^{-1} J^T K= N^{-1} J^T K$

Dla równań wagowanych:

WJK=WK

$X=(J^T WJ)^{-1} J^T WK= N^{-1} J^T WK$

};

W naszym przypadku otrzymujemy:

$$\begin{bmatrix} (\frac{\partial X}{\partial a_1})_0 & (\frac{\partial X}{\partial b_1})_0 & (\frac{\partial X}{\partial c_1})_0 & 0 & 0 & 0 & (\frac{\partial X}{\partial a_3})_0 & (\frac{\partial X}{\partial b_3})_0 \\ 0 & 0 & 0 & (\frac{\partial Y}{\partial a_2})_0 & (\frac{\partial Y}{\partial b_2})_0 & (\frac{\partial Y}{\partial c_2})_0 & (\frac{\partial Y}{\partial a_3})_0 & (\frac{\partial Y}{\partial b_3})_0 \end{bmatrix} \begin{bmatrix} da_1 \\ db_1 \\ dc_1 \\ da_2 \\ db_2 \\ dc_2 \\ da_3 \\ db_3 \end{bmatrix} = \begin{bmatrix} X - X_0 \\ Y - Y_0 \end{bmatrix}$$

PRZYPOMNIENIE 2 {

$Niech\ y = u\big(v(x)\big)$

Przykład $\left(\frac{1}{x}\right)' = -\left(\frac{1}{x^2}\right)\frac{\partial y}{\partial x}\ bo\ u'(v)v'(x)$

};

Zatem:

$$\frac{\partial X}{\partial a_1} = \frac{X}{a_3x + b_3 + 1}\ \frac{\partial X}{\partial b_1} = \frac{Y}{a_3x + b_3 + 1}\ \frac{\partial X}{\partial c_1} = \frac{1}{a_3x + b_3 + 1}$$

$$\frac{\partial Y}{\partial a_2} = \frac{X}{a_3x + b_3 + 1}\ \frac{\partial Y}{\partial b_2} = \frac{Y}{a_3x + b_3 + 1}\ \frac{\partial Y}{\partial c_2} = \frac{1}{a_3x + b_3 + 1}$$

$$\frac{\partial X}{\partial a_3} = -\frac{a_1x + b_1y + c_1}{(a_3x + b_3 + 1)^2}\ x$$

$$\frac{\partial X}{\partial b_3} = -\frac{a_1x + b_1y + c_1}{(a_3x + b_3 + 1)^2}\ y$$

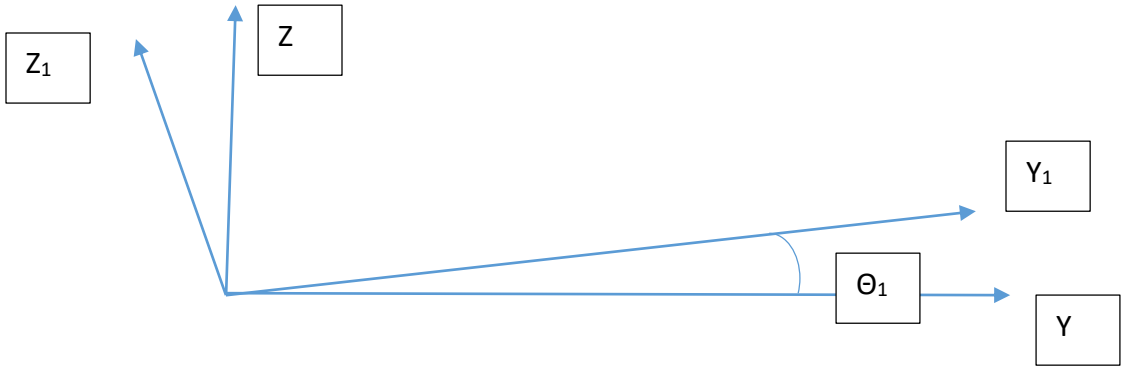
$$\frac{\partial Y}{\partial a_3} = -\frac{a_2x + b_2y + c_2}{(a_3x + b_3 + 1)^2}\ x$$

$$\frac{\partial Y}{\partial b_3} = -\frac{a_2x + b_2y + c_2}{(a_3x + b_3 + 1)^2}\ y$$

4. Trójwymiarowa transformacja konforemna. Transformacja siedmioparametrowa. Rozwiązanie ścisłe.

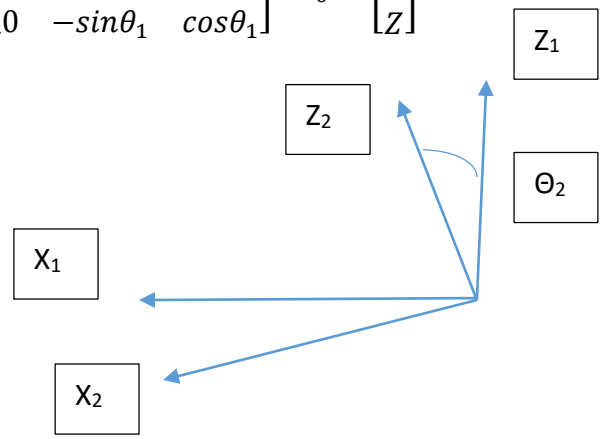
Zastosowanie: fotogrametria, skaning, GNSS.

Macierz obrotów to trzy postępujące po sobie macierze obrotów 2D wokół osi x,y,z. Kąty Θ są skierowane przeciwnie do ruchu wskazówek zegara w rzucie 2D.



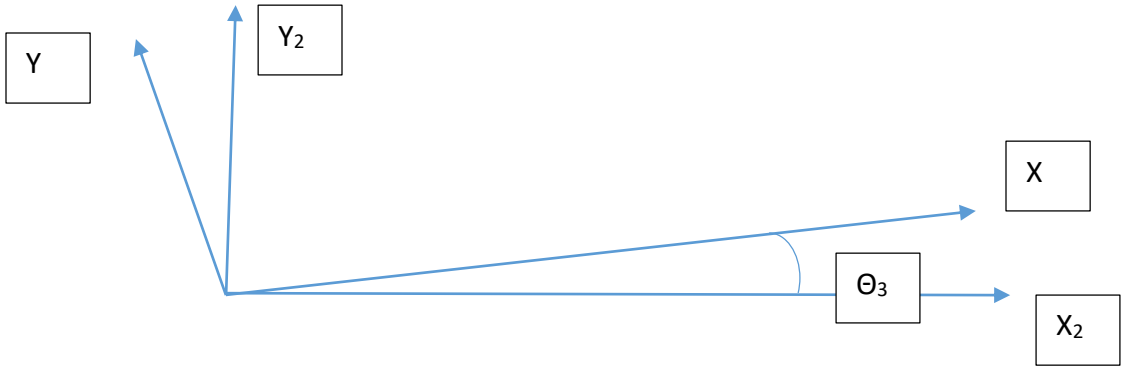
$X_1=R_1X_0$

$$X_1 = \begin{bmatrix} x_1 \\ y_1 \\ z_1 \end{bmatrix}; R_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\theta_1 & sin\theta_1 \\ 0 & -sin\theta_1 & cos\theta_1 \end{bmatrix}; X_0 = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$



$X_2=R_2X_1$

$$X_2 = \begin{bmatrix} x_2 \\ y_2 \\ z_2 \end{bmatrix}; R_2 = \begin{bmatrix} cos\theta_2 & 0 & -sin\theta_2 \\ 0 & 1 & 0 \\ sin\theta_2 & 0 & cos\theta_2 \end{bmatrix};$$



$X=R_3X_2$

$$X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$$

$$R_3 = \begin{bmatrix} cos\theta_3 & sin\theta_3 & 0 \\ -sin\theta_3 & cos\theta_3 & 0 \\ 0 & 0 & 1 \end{bmatrix};$$

Zatem:

$X=R_3R_2R_1X_0=RX$

Niech

$$R = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}$$

Więc z iloczynu macierzy dla obrotów dwuwymiarowych otrzymujemy

$$r_{11} = cos\theta_2 cos\theta_3$$

$$r_{12} = sin\theta_1 sin\theta_2 cos\theta_3 + cos\theta_1 sin\theta_3$$

$$r_{13} = -cos\theta_1 sin\theta_2 cos\theta_3 + sin\theta_1 sin\theta_3$$

$$r_{21} = -cos\theta_2 sin\theta_3$$

$$r_{22} = -sin\theta_1 sin\theta_2 sin\theta_3 + cos\theta_1 cos\theta_3$$

$$r_{23} = cos\theta_1 sin\theta_2 sin\theta_3 + sin\theta_1 cos\theta_3$$

$$r_{31} = sin\theta_2$$

$$r_{32} = -sin\theta_1 cos\theta_2$$

$r_{33} = cos\theta_1 cos\theta_2$

Obserwacja: Macierz obrotów R jest ortogonalna.

Implikacja obserwacji: Odwrotność macierzy R jest równa jej transpozycji bo $RR^T=I$

Implikacja obserwacji numer dwa: możemy ją pomnożyć przez skalę oraz dodać translację.

Zatem:

$X = S(r_{11}x + r_{21}y + r_{31}z) + T_x$

$Y = S(r_{12}x + r_{22}y + r_{32}z) + T_y$

$Z = S(r_{13}x + r_{23}y + r_{33}z) + T_z$

Obserwacja: konieczne są **minimum** dwa punkty ze współrzędnymi x i y oraz trzy ze znaną współrzędną z bo do wyznaczenia jest siedmioelementowy wektor niewiadomych $(S, \theta_1, \theta_2, \theta_3, T_x, T_y, T_z)$.

Czyli układ równań przybiera postać:

$$\begin{bmatrix} (\frac{\partial X}{\partial S})_0 & 0 & (\frac{\partial X}{\partial \theta_2})_0 & (\frac{\partial X}{\partial \theta_3})_0 & 1 & 0 & 0 \\ (\frac{\partial Y}{\partial S})_0 & (\frac{\partial Y}{\partial \theta_1})_0 & (\frac{\partial Y}{\partial \theta_2})_0 & \left(\frac{\partial Y}{\partial \theta_3}\right)_0 & 0 & 1 & 0 \\ (\frac{\partial Z}{\partial S})_0 & (\frac{\partial Z}{\partial \theta_1})_0 & (\frac{\partial Z}{\partial \theta_2})_0 & \left(\frac{\partial Z}{\partial \theta_3}\right)_0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} dS \\ d\theta_1 \\ d\theta_2 \\ d\theta_3 \\ dT_x \\ dT_y \\ dT_Z \end{bmatrix} = \begin{bmatrix} X - X_0 \\ Y - Y_0 \\ Z - Z_0 \end{bmatrix}$$

gdzie:

$\frac{\partial X}{\partial S} = r_{11}x + r_{21}y + r_{31}z;$

$\frac{\partial Y}{\partial S} = r_{12}x + r_{22}y + r_{32}z;$

$\frac{\partial Z}{\partial S} = r_{13}x + r_{23}y + r_{33}z;$

$\frac{\partial Y}{\partial \theta_1} = -S(r_{13}x + r_{23}y + r_{33}z);$

$\frac{\partial Z}{\partial \theta_1} = S(r_{12}x + r_{22}y + r_{32}z);$

$\frac{\partial X}{\partial \theta_2} = S(-x\ sin\theta_2\ cos\theta_3 + y\ sin\theta_2\ sin\theta_3 + z\ cos\theta_2);$

$\frac{\partial Y}{\partial \theta_2} = S(x\ sin\theta_1\ cos\theta_2 cos\theta_3 - y\ sin\theta_1\ cos\theta_2 sin\theta_3 + z\ sin\theta_1 sin\theta_2);$

$\frac{\partial Z}{\partial \theta_2} = S(-x\ cos\theta_1\ cos\theta_2 cos\theta_3 + y\ cos\theta_1\ cos\theta_2 sin\theta_3 - z\ cos\theta_1 sin\theta_2);$

$$\frac{\partial X}{\partial \theta_3} = S(r_{21}x - r_{11}y);$$

$$\frac{\partial Y}{\partial \theta_3} = S(r_{22}x - r_{12}y);$$

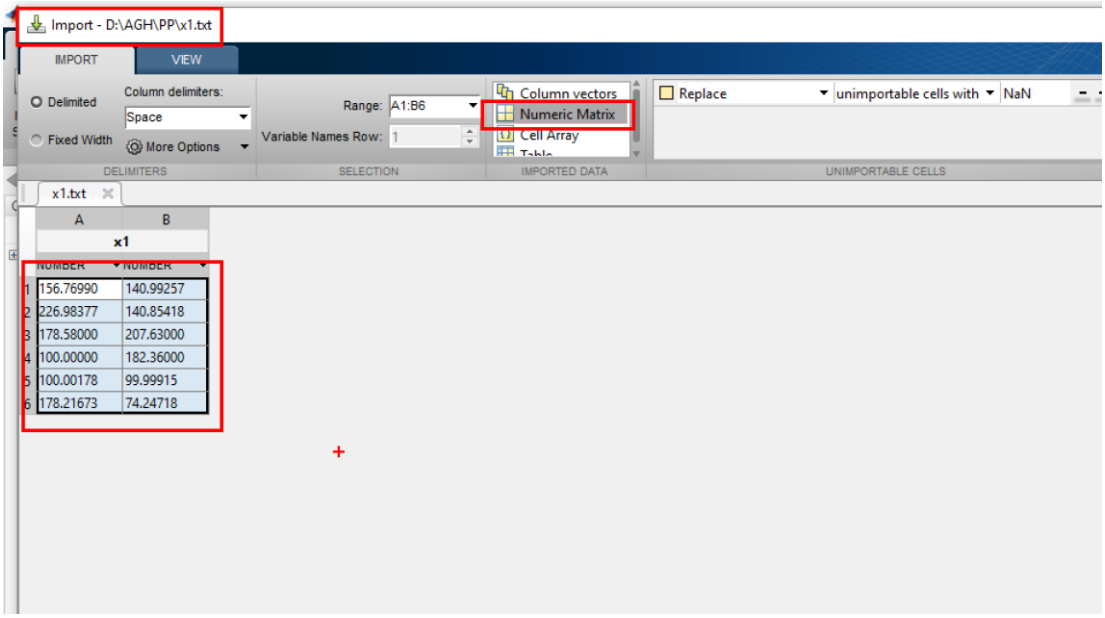
$$\frac{\partial Z}{\partial \theta_3} = S(r_{23}x - r_{13}y);$$

Dane są dwa zbiory punktów. Numeracja domyślna zgodnie z kolejnością

156.769 140.992
226.983 140.854
178.580 207.630
100.000 182.360
100.001 99.999
178.216 74.247

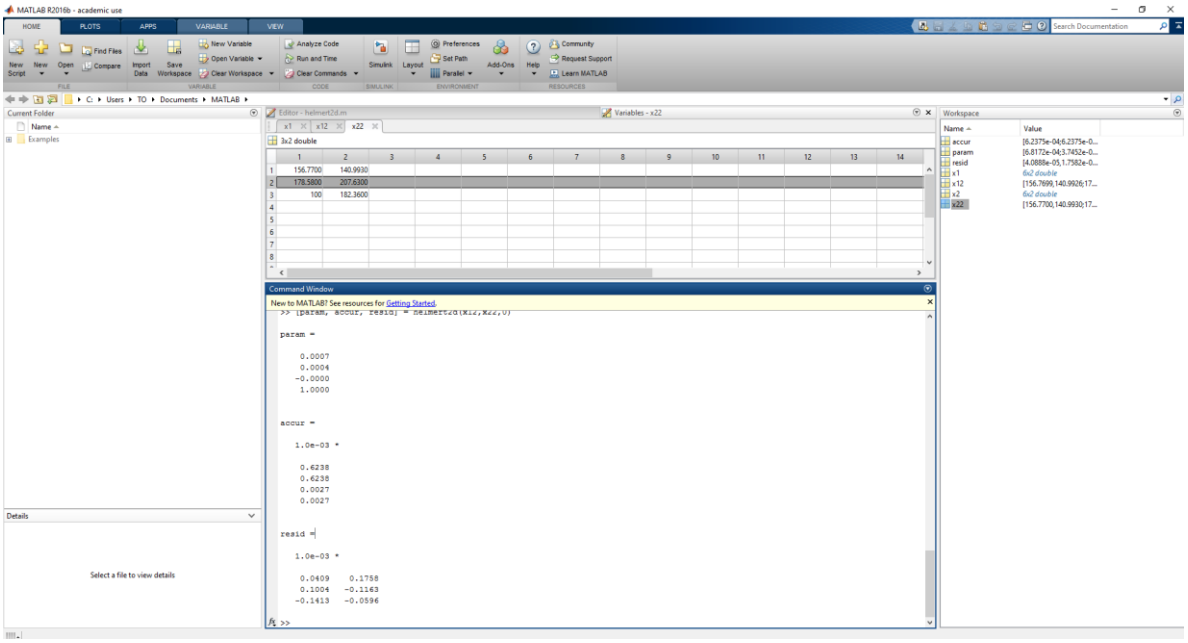
oraz

156.770 140.993
226.985 140.868
178.580 207.630
100.000 182.360
99.990 100.008
178.217 74.251



Jako punkty dostosowania wybieramy te o numerach 1,3,4. Uruchamiamy funkcję

```
>> [param, accur, resid] = helmert2d(x12,x22,0)

param =
    0.0007
    0.0004
   -0.0000
    1.0000

accur =
  1.0e-03 *

    0.6238
    0.6238
    0.0027
    0.0027

resid =
  1.0e-03 *

    0.0409    0.1758
    0.1004   -0.1163
   -0.1413   -0.0596
```



Następnie liczymy transformacji układu wtórnego to pierwotnego.

>> xy=d2trafo(x2,param,0)

xy =

156.7701 140.9933
226.9849 140.8684
178.5799 207.6301
100.0001 182.3601
99.9903 100.0083
178.2171 74.2515

>> x1-xy //liczymy róznice współrzednych

ans =

-0.0002 -0.0007
-0.0011 -0.0142
0.0001 -0.0001
-0.0001 -0.0001
0.0115 -0.0091
-0.0004 -0.0043

Przykłady implementacji autorstwa dr Peter Wasmeiera - Technische Universität München
https://www.mathworks.com/matlabcentral/fileexchange/9696-geodetic-transformations

```
function [tp,ac,tr]=helmert2d(datum1,datum2,WithOutScale,NameToSave)

% HERLMERT2D    overdetermined cartesian 2D similarity transformation
("Helmert-Transformation")
%
% [param, accur, resid] = helmert2D(datum1,datum2,DontUseScale,SaveIt)
%
% Inputs:  datum1  n x 2 - matrix with coordinates in the origin datum (x y)
%                  datum1 may also be a file name with ASCII data to be
processed. No point IDs, only
%                  coordinates as if it was a matrix.
%
%          datum2  n x 2 - matrix with coordinates in the destination datum (x
y)
%                  datum2 may also be a file name with ASCII data to be
processed. No point IDs, only
%                  coordinates as if it was a matrix.
%                  If either datum1 and/or datum2 are ASCII files, make sure
that they are of same
%                  length and contain corresponding points. There is no auto-
assignment of points!
%
%    DontUseScale  if this is not 0, do not calculate scale factor but set it
to the inputted value
%                  Default: 0 (Use scale)
%
%          SaveIt  string with the name to save the resulting parameters in
Transformations.mat.
%                  Make sure only to use characters which are allowed in
Matlab variable names
%                  (e.g. no spaces, umlaute, leading numbers etc.)
%                  If left out or empty, no storage is done.
%                  If the iteration is not converging, no storage is done and
a warning is thrown.
%                  If the name to store already is existing, it is not
overwritten automatically.
%                  To overwrite an existing name, add '#o' to the name, e.g.
'wgs84_to_local#o'.
%
% Outputs:  param  4 x 1 Parameter set of the 2D similarity transformation
%                      2 translations (x y) in [Unit of datums]
%                      1 rotation (ez) in [rad]
%                      1 scale factor
%
%           accur  4 x 1 accuracy of the parameters (or 3 x 1 if scale factor
is set to be 1)
%
%           resid  n x 2 - matrix with the residuals datum2 - f(datum1,param)
%
% Used to determine transformation parameters e.g. for cadastral purposes
(transforming local system
```

```
% to 2D mapping system) when at least 2 identical points in both datum systems
are known.
% Parameters can be used with d2trafo.m

% 09/12/11 Peter Wasmeier - Technische Universität München
% p.wasmeier@bv.tum.de
% 04/17/15 Andrea Pinna - University degli Studi di Cagliari
% andrea.pinna@unica.it

%% Argument checking and defaults

if nargin<4
    NameToSave=[];
else
    NameToSave=strtrim(NameToSave);
    if strcmp(NameToSave,'#o')
        NameToSave=[];
    end
end

if nargin<3 || isempty(WithOutScale)
    WithOutScale=0;
end

% Load input file if specified
if ischar(datum1)
    datum1=load(datum1);
end
if ischar(datum2)
    datum2=load(datum2);
end

if (size(datum1,1)==2)&&(size(datum1,2)~=2)
    datum1=datum1';
end
if (size(datum2,1)==2)&&(size(datum2,2)~=2)
    datum2=datum2';
end

s1=size(datum1);
s2=size(datum2);
if any(s1~=s2)
    error('The datum sets are not of equal size')
elseif any([s1(2) s2(2)]~=[2 2])
    error('At least one of the datum sets is not 2D')
elseif any([s1(1) s2(1)]<2)
    error('At least 2 points in each datum are necessary for calculating')
end

%% Approcimation values and adjustment

naeh=[0 0 0 1];
if WithOutScale
    naeh(4)=WithOutScale;
end
```

```
WertA=[1e-8 1e-8];
zaehl=0;

x0=naeh(1);
y0=naeh(2);
ez=naeh(3);
m=naeh(4);

tp=[x0 y0 ez m];

% This will only be used, if point accuracies were given - not used yet
% Qbb=eye(2*s1(1));

while(1)
    A=zeros(2*s1(1),4);
    w=zeros(2*s1(1),1);

    % Compute sine and cosine of angle - speeds up computation
    cosez = cos(ez);
    sinez = sin(ez);

    A(1:2:end,1)=1;
    A(2:2:end,2)=1;

    A(1:2:end,3)=m*(-sinez*datum1(:,1)+cosez*datum1(:,2));
    A(1:2:end,4)=cosez*datum1(:,1)+sinez*datum1(:,2);
    A(2:2:end,3)=m*(-cosez*datum1(:,1)-sinez*datum1(:,2));
    A(2:2:end,4)=-sinez*datum1(:,1)+cosez*datum1(:,2);

    w(1:2:end,1)=datum2(:,1)-x0-m*(cosez*datum1(:,1)+sinez*datum1(:,2));
    w(2:2:end,1)=datum2(:,2)-y0-m*(-sinez*datum1(:,1)+cosez*datum1(:,2));

    if WithOutScale
        A=A(:,1:3);
    end

    warning off;
    r=size(A,1)-size(A,2);
    Pbb = speye(2*s1(1));   % Accuracy of points is not used yet
    APbbA = A'*Pbb*A;
    deltax=APbbA\(A'*Pbb*w);
    v=A*deltax-w;
    sig0p=sqrt((v'*Pbb*v)/r);
    Kxxda=sig0p^2*inv(APbbA);
    ac=sqrt(diag(Kxxda));
    warning on;

    testv=sqrt((deltax(1)^2+deltax(2)^2)/2);
    testd=deltax(3);
    zaehl=zaehl+1;
    x0=x0+deltax(1);
    y0=y0+deltax(2);
    ez=ez+deltax(3);
    if ~WithOutScale && (m+deltax(4))>1e-15     % This condition is to prevent
numerical problems with m-->0
```

```
        m=m+deltax(4);
    end
    tp=[x0 y0 ez m]';
    if abs(testv) < WertA(1) && abs(testd) < WertA(2)
        break;
    elseif zaehl>1000
        warning('Helmert2D:Too_many_iterations','Calculation not converging
after 1000 iterations. I am aborting. Results may be inaccurate.')
        break;
    end
end

%% Transformation residuals
idz=zeros(s1);
idz(:,2)=tp(2)+tp(4)*(-sin(tp(3))*datum1(:,1)+cos(tp(3))*datum1(:,2));
idz(:,1)=tp(1)+tp(4)*(cos(tp(3))*datum1(:,1)+sin(tp(3))*datum1(:,2));

tr=datum2-idz;

%% Store data

if ~isempty(NameToSave)
    load Transformations;
    if zaehl>1000
        warning('Helmert2D:Results_too_inaccurate_to_save','Results may be
inaccurate and do not get stored.')
    elseif exist(NameToSave,'var') && length(NameToSave)>=2 &&
~strcmp(NameToSave(end-1:end),'#o')
        warning('Helmert2D:Parameter_already_exists',['Parameter set
',NameToSave,' already exists and therefore is not stored.'])
    else
        if strcmp(NameToSave(end-1:end),'#o')
            NameToSave=NameToSave(1:end-2);
        end
        if any(regexp(NameToSave,'\W')) || any(regexp(NameToSave(1),'\d'))
            warning('Helmert2D:Parameter_name_invalid',['Name ',NameToSave,'
contains invalid characters and therefore is not stored.'])
        else
            eval([NameToSave,'=num2str(tp'');']);
            save('Transformations.mat',NameToSave,'-append');
        end
    end
end


function xy=d2trafo(XY,p,dir,FileOut)

% D2TRAFO performs 2D-transformation with 4 parameters in either
transformation direction
%
% xy=d2trafo(XY,p,dir,FileOut)
%
% Inputs:   XY  nx2-matrix to be transformed. 2xn-matrices are allowed, be
careful with
%               2x2-matrices!
```

```
%               XY may also be a file name with ASCII data to be processed. No
point IDs, only
%               coordinates as if it was a matrix.
%
%            p  The vector of transformation parameters [dx dy ez s] with
%                 dx,dy = translations [unit of XY]
%                    ez = rotation in [rad]
%                     s = scale factor
%               p may also be a string with the name of a predefined
transformation stored in
%               Transformations.mat.
%
%          dir  the transformation direction.
%               If dir=0 (default if omitted or set to []), p are used as
given.
%               If dir=1, inverted p' is used to calculate the back-
transformation (i.e. if p was
%                  calculated in the direction Sys1 -> Sys2, p' is for Sys2 ->
Sys1).
%
%      FileOut  File to write the output to. If omitted, no output file is
generated.
%
% Output:   xy  nx2-matrix with the transformed coordinates.
%
% Systems need to be right-handed, i.e. [x y].
% Used for transforming plane cartesian coordinates from one system to
another, e.g. when changing
% from local 2D working system to cadastral mapping system.


% Author:
% Peter Wasmeier, Technical University of Munich
% p.wasmeier@bv.tum.de
% Dec 09, 2011

%% Do input checking , set defaults and adjust vectors

if nargin<4
    FileOut=[];
end
if nargin<3 || isempty(dir)
    dir=0;
elseif ~isscalar(dir)
    error('Parameter ''dir'' must be a scalar expression.')
end
if nargin<2
    error('Too few parameters for D2trafo execution. Check your inputs!')
end
if ischar(p)
    load Transformations;
    if ~exist(p,'var')
        error(['Transformation set ',p,' is not defined in Transformations.mat
- check your definitions!.'])
    elseif (length(p)~=4)
```

```
        error(['Transformation set ',p,' is of wrong size - check your
definitions!.'])
    end
    eval(['p=',p,';']);
end
if numel(p)~=4
    error('Parameter ''p'' must be a 1x4-vector!')
else
    p=p(:);
end

% Load input file if specified
if ischar(XY)
    XY=load(XY);
end

if (size(XY,1)~=2)&&(size(XY,2)~=2)
    error('Coordinate list XY must be a nx2-matrix!')
elseif (size(XY,1)==2)&&(size(XY,2)~=2)
    XY=XY';
end

%% Do the calculations

% number of coordinate pairs to transform
n=size(XY,1);

% Create rotation matrix
D=[cos(p(3)) sin(p(3));-sin(p(3)) cos(p(3))];

% Perform transformation
if ~dir
    xy=repmat(p(1:2),1,n)+p(4)*D*XY';
else
    xy=D'/p(4)*(XY'-repmat(p(1:2),1,n));
end
xy=xy';
%% Write output to file if specified

if ~isempty(FileOut)
    fid=fopen(FileOut,'w+');
    fprintf(fid,'%12.6f  %12.6f\n',xy');
    fclose(fid);
end
```