Applying Compensation Map Data

Table of Contents

Applying Compensation Map Data

Purpose

This article is written as a practical example of how to interpret compensation data from a coordinate measuring machine and create a correction value and tool coordinate system for any position in the measuring volume. This information can be used to build a working model for error map correction and can be useful for developing other utilities to allow interpretation or correction based on measurement data.

Description

The compensation map is simply a table of data for each axis containing known geometric errors at different points along the axis. Since most CMM's have a minimum of three axis a typical compensation map will have a minimum of three tables. The following is an example of a compensation map table for the X, Y, and Z axis:

X Axis

Position	Lx	Ly	Lz	Rx	Ry	Rz
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
100.0000	0.0050	0.0100	0.0150	0.0200	0.0250	0.0300
200.0000	0.0100	0.0200	0.0300	0.0400	0.0500	0.0600
300.0000	0.0150	0.0300	0.0450	0.0600	0.0750	0.0900
400.0000	0.0200	0.0400	0.0600	0.0800	0.1000	0.1200

Y Axis

Position	Lx	Ly	Lz	Rx	Ry	Rz
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
200.0000	0.0030	0.0400	0.0060	0.0070	0.0080	0.0090
400.0000	0.0060	0.0800	0.0120	0.0140	0.0160	0.0180
600.0000	0.0090	0.1200	0.0180	0.0210	0.0240	0.0270
800.0000	0.0120	0.1600	0.0240	0.0280	0.0320	0.0360

Z Axis

Position	Lx	Ly	Lz	Rx	Ry	Rz
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
-50.0000	0.0130	0.0110	0.0120	0.0130	0.0140	0.0150
-100.0000	0.0260	0.0220	0.0240	0.0260	0.0280	0.0300
-150.0000	0.0390	0.0330	0.0360	0.0390	0.0420	0.0450
-200.0000	0.0520	0.0440	0.0480	0.0520	0.0560	0.0600

Applying Compensation Map Data

The parameters Lx, Ly, Lz represent the linear errors and Rx, Ry, Rz represent the angular errors for each axis. In the above tables the linear values are in millimeters and the angular values are in millimeters per meter.

There are many variations to the naming convention of the compensation map data. The names Roll, Pitch, Yaw, Horizontal Straightness, Vertical Straightness, and Scale Error are frequenty used but can be ambiguous in some cases (example, definition of pitch or yaw for the Z axis) so using more specific label names is preferred.

In the example data Lx, Ly, Lz represents the linear error in the X,Y, and Z axis respectively and Rx, Ry, Rz is the rotation error around the X, Y, and Z axis. When each parameter is combined with a specific axis the names will take on a new meaning:

X Axis

Lx represents the scale error (error of the X axis in the X direction) - Lxx
Ly represents the horizontal straightness error (error of the X axis in the Y direction) - Lxy
Rz represents the X axis yaw error (rotation error of the X axis around the Z axis) - Rxz

Y Axis

Lx represents the horizontal straightness error (error of the Y axis in the X direction) - Lyx
Ly represents the scale error (error of the Y axis in the Y direction) - Lyy
Rz represents the Y axis yaw error (rotation error of the Y axis around the Z axis) - Ryz

Kinematic Chain

The kinematic chain describes the order that the axis are physically connected to each other. In order to interpret the compensation data properly it is necessary to apply the correction data from the compensation map in this order. For a typical CMM where the Y axis is front to back and moves along the workpiece table, X from left to right and is connected to the Y axis, and Z is vertical and connected to the X axis the kinematic chain would be YXZ.

The common axis convention for vertical arm machines is YXZ or XYZ. For horizontal arms it is usually XZY. These have become defacto axis conventions but any axis convention is possible.

Data Interpolation

The data in the compensation tables is provided at specific points along the axis. In almost all cases it is necessary to get tables values at positions other than those listed in the compensation map. In order to have a correction value for any position along the axis it is necessary to interpolate the table data.

Data interpolation is done as a linear interpolation between adjacent map steps. Methods such as using a high order interpolating or approximating spline will not produce good results the spline will fit through (or approximate) the provided data trying to maintain a smooth shape.

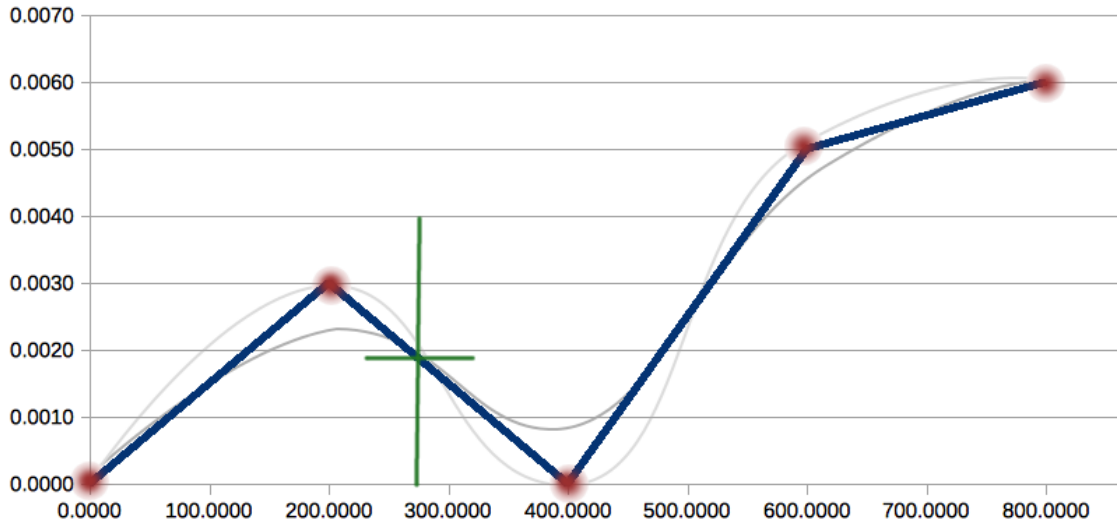

Illustration 1: Interpolation example to extract data at position 275.000 mm. Linear interpolation in blue, interpolated and approximated spline fits shown in grey which should not be used. The red dots represent data from the compensation table.

This function as shown in the following example preforms the interpolation is useful for any implementation of this method:

```
First_Comp_Axis.Evaluate(Y_Position,&Lx,&Ly,&Lz,&Rx,&Ry,&Rz);
```

where:

First_Comp_Axis = Class representing the first compensation axis in the kinematic chain
Y_Position = Position along the Y axis
&Lx, &Ly,...&Rz = Variables that are updated with the interpolated values at Y_Position.

Although the implementation of this function is not shown it is not difficult to write (left as a problem for the programmer to solve).

Application of Compensation Parameters

The following is a step by step process to apply compensation data from the example compensation map. The kinematic order of the axis is assumed to by YXZ.

The input value is the uncompensated point of 200,600,-150 in XYZ.

From the compensation table the compensation table errors at these positions in the map are as follows:

Axis	Position	Lx	Ly	Lz	Rx	Ry	Rz
Y	600.0000	0.0090	0.1200	0.0180	0.0210	0.0240	0.0270
X	200.0000	0.0100	0.0200	0.0300	0.0400	0.0500	0.0600
Z	-150.0000	0.0390	0.0330	0.0360	0.0390	0.0420	0.0450

The calculations are performed in the order of the kinematic chain (Y, then X, then Z). The compensation value is a running sum of the errors as the different components are applied. The

final compensated location is the sum of the uncompensated point and the final sum of the compensation errors.

Initial value of the correction sum is zero:

```
Correction Sum: 0.0000, 0.0000, 0.0000
```

Matrix Class for Angular Correction

The rotation is done using a matrix data type that allows for rotations around specific vectors. The rotation angle is in radians (equivalent to mm/mm). The matrix type is referenced as a class name 'mat' and is described below:

```
Default values.  Called identity matrix
I: 1.000000000 0.000000000 0.000000000
J: 0.000000000 1.000000000 0.000000000
K: 0.000000000 0.000000000 1.000000000

mat.X(); // returns the current direction of the X axis
mat.Y(); // returns the current direction of the X axis
mat.Z(); // returns the current direction of the X axis
mat.Rotate(v,a);    // rotates the matrix by angle 'a' around the axis 'v'


Rotate matrix by 30 degrees (0.5236 radians) around the Z axis
mat.Rotate(mat.Z(),0.52359877559831);

Result of rotating matrix by 30 degrees (0.5236 radians) around the Z axis
I: 0.866025404 0.500000000 0.000000000
J: -0.500000000 0.866025404 0.000000000
K: 0.000000000 0.000000000 1.000000000
```

To find the location of a point relative to the axis of the matrix it is only necessary to multiply the two items:

```
new_point = mat * point

where
  point = 100,0,300
  new_point = 86.602540378, 50.000000000, 300.000000000
```

The 'new_point' can be described as the location of the point from an orthogonal coordinate system knowing that the X axis is rotated 30 degrees from the Y axis.

Linear Correction for Y Axis

The linear errors from the compensation map for the first axis of the kinematic chain is added directly to the correction sum.

```
Linear Correction at Y=600: 0.0090, 0.1200, 0.0180
Correction Sum: 0.0000+0.0090, 0.0000+0.1200, 0.0000+0.0180
Correction Sum: 0.0090, 0.1200, 0.0180
```

Angular Correction from Y Axis

The angular errors from the compensation map for the first axis of the kinematic chain are used to define the direction of the second axis. The idea is to convert the known axis direction into a position that would be orthogonal to the first axis.

```
Angular Correction at Y=600: 0.0210, 0.0240, 0.0270
Position of the X axis: 200.000
```

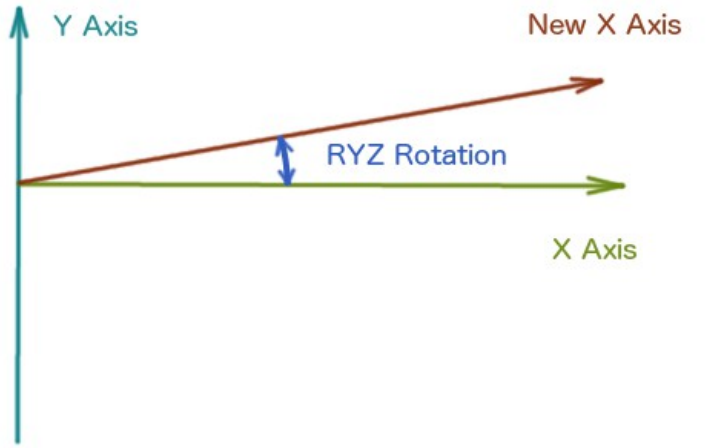

Illustration 2: Calculation showing effect of rotation RYZ on direction of X axis. Direction of X axis is defined in a matrix variable.

The angular errors are not cumulative. The correction is applied based on the absolute error at the compensation position and not based on the error from the previous position. This effect of this method will appear as a straightness error and is compensated separately and can even be estimated based on how the angular error is calculated.

```
Initial value of variable mat:
I: 1.000000000 0.000000000 0.000000000
J: 0.000000000 1.000000000 0.000000000
K: 0.000000000 0.000000000 1.000000000

mat.Rotate(mat.X(),0.0000210); // rotate around X axis 0.021 mm/m
mat.Rotate(mat.Y(),0.0000240); // rotate around Y axis 0.024 mm/m
mat.Rotate(mat.Z(),0.0000270); // rotate around Z axis 0.027 mm/m

Result of rotations:
I: 0.999999999 -0.000026999 0.000024001
I: 0.000026999 0.999999999 -0.000020999
K: -0.000024001 0.000020999 0.999999999
```

The effect of the angular correction is the difference between where the point would be at the nominal X axis direction compared to the location of the point from the rotated X axis.

```
Nominal X Position: 200,0,0
Rotated X Position: 199.999999870 0.005399899 -0.004800113
Angular correction: -0.000000130 0.005399899 -0.004800113

Correction Sum: 0.0090+(-0.000000130),0.1200+0.005399899,0.0180+(-0.004800113)
Correction Sum: 0.0090, 0.1254, 0.0132
```

Linear Correction for X Axis

The linear errors from the compensation map for the second axis of the kinematic chain is added

directly to the correction sum.

```
Linear Correction at X=200: 0.0100, 0.0200, 0.0300
Correction Sum: 0.0090+0.0100, 0.1254+0.0200, 0.0132+0.0300
Correction Sum: 0.0190, 0.1454, 0.0432
```

Angular Correction from X Axis

The angular errors from the compensation map for the second axis of the kinematic chain are used to define the direction of the third axis in the kinematic chain.

```
Angular Correction at X=200: 0.0400, 0.0500, 0.0600
Position of the Z axis: -150.000

Current rotations:
I: 0.999999999 -0.000026999 0.000024001
J: 0.000026999 0.999999999 -0.000020999
K: -0.000024001 0.000020999 0.999999999

mat.Rotate(mat.X(),0.0000400); // rotate around X axis 0.040 mm/m
mat.Rotate(mat.Y(),0.0000500); // rotate around Y axis 0.050 mm/m
mat.Rotate(mat.Z(),0.0000600); // rotate around Z axis 0.060 mm/m

Result of rotations:
I: 0.999999993 -0.000086994 0.000074007
J: 0.000086998 0.999999994 -0.000060991
K: -0.000074002 0.000060998 0.999999995
```

The effect of the angular correction is the difference between where the point would be at the nominal Z axis direction compared to the location of the point from the rotated Z axis.

```
Nominal Z Position: 0,0,-150
Rotated Z Position: -0.011101106 0.009148676 -149.999999310
Angular correction: -0.011101106 0.009148676 0.000000690

Correction Sum: 0.0190+(-0.011101106),0.1454+0.009148676,0.0432+0.000000690
Correction Sum: 0.0079, 0.1545, 0.0432
```

Linear Correction for Z Axis

The linear errors from the compensation map for the third axis of the kinematic chain is added directly to the correction sum.

```
Linear Correction at Z=-150: 0.0390, 0.0330, 0.0360
Correction Sum: 0.0079+0.0390, 0.1545+0.0330, 0.0432+0.0360
Correction Sum: 0.0469, 0.1875, 0.0792
```

This value represents the final corrected value at the bottom of the Z axis. To compensate to the probe stylus the residual directions at the bottom of the Z are used.

Angular Correction from Z Axis

The angular errors from the compensation map for the third axis of the kinematic chain are used to define the direction of the probe stylus.

```
Angular Correction at Z=-150: 0.0390, 0.0420, 0.0450

Current rotations:
```

```
I: 0.999999993 -0.000086994 0.000074007
J: 0.000086998 0.999999994 -0.000060991
K: -0.000074002 0.000060998 0.999999995

mat.Rotate(mat.X(),0.0000390); // rotate around X axis 0.040 mm/m
mat.Rotate(mat.Y(),0.0000420); // rotate around Y axis 0.050 mm/m
mat.Rotate(mat.Z(),0.0000450); // rotate around Z axis 0.060 mm/m

Result of rotations:
I: 0.999999985 -0.000131988 0.000116013
J: 0.000132000 0.999999986 -0.000099984
K: -0.000116000 0.000100000 0.999999988
```

This final set of vectors represents the sum of all the rotations throughout the kinematic axis. A probe or other instrument mounted at the bottom of the Z would actually be oriented adjusted by the final rotation values.

Final Compensated Value

```
Initial XYZ Value: 200,600,-150
Correction Sum: 0.0469, 0.1875, 0.0792
Compensated XYZ value: 200+0.0469,600+0.1875,-150+0.0792

Compensated XYZ value: 200.0469, 600.1875, -149.9208

Orientation at bottom of Z axis
I: 0.999999985 -0.000131988 0.000116013
J: 0.000132000 0.999999986 -0.000099984
K: -0.000116000 0.000100000 0.999999988
```

Probe Tip Compensation

For a probe stylus connected to the bottom of the Z axis it is necessary to further offset along the IJK vectors at the bottom of the Z axis. This is done by multiplying the nominal XYZ probe tip offset by the final orientation matrix.

Squareness Correction

There are two known methods that are used to correct for squareness errors in machines. The squareness errors are corrections to the XY, YZ, and ZX in order to keep each pair of axis at a right angle to each other.

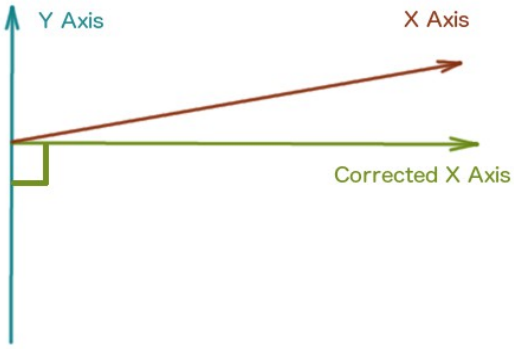

Illustration 3: Squareness error between the X and Y axis

Correction By Adjusting Straightness

This method changes the straightness of the axis by adding a gradient. For each position along the axis the straightness correction in the compensation map increases accordingly.

Correction By Adjusting Angular

This method is the preferred method for interpreting squareness errors. The squareness error is applied as a constant to all angular data. From the example shown in illustration 3 the squareness correction would be applied equally to all parameters of RYZ.

Vendor Specific Methods

All vendors that provides geometric compensation to their CMM product lines are sufficiently similar that a generic implementation would cover the majority. Some differences that have been observed are listed below:

- Rotation point for Z axis of a horizontal arm calculated from opposite end of zero point. This makes sense since the mechanical rotation and the software rotation point would be the same. Not all vendors do this.
- Arbitrary rotation point for all axis inside measurement volume. Some vendors calculate all rotation values from a point other than the map zero point. One advantage of this method is that the software corrections are smallest at the rotation point and grow as you move toward the edges of the machine volume.
- Rotation signs. Most vendors use the standard rule of counter clockwise representing a positive angular rotation but not all.
- Linear signs. Most vendors use linear correction signs that are the same as the error calculated from a measurement of 'laser – machine' (i.e. laser shows 100.010, machine says 100.000, then error is 0.010). Ths is not true for all vendors.

In order to better understand a vendor specific implementation it is necessary to perform tests to determine the amount of compensation calculated for various points in the machine volume and reproduce those corrections precisely. The preferred method is to have the option of driving the

machine to a specific location in the machine volume and switch on/off the compensation comparing the difference in position. Methods that involve measuring an artifact (i.e. measure the location of a calibration sphere with compensation on and off) can be error prone since the correction value will be opposite to the shift in the sphere position. This method can be used but should be carefully reviewed for mistakes.

Removing Compensation

Removing the compensation value from a point in the volume that is already compensated is a little more involved than calculating the compensation from an uncompensated point. Two methods can be used if necessary:

- Find the product of all compensation steps in the form of a single matrix. Inverting the product and multiplying by the compensated point will produce an uncompensated value.
- Calculated the correction with the assumption the compensated and uncompensated points are close to each other. Once the correction value is known the uncompensated point is the difference between the compensated and the correction. If this method is repeated two or three times in a loop the end result is very accurate even with very large correction values.