DATA.ZERO

WP2 Training Seminar

3D Characterisation of cylindrical runout

Luke Todhunter

DAT4.Zero has received funding from the European Union's Horizon 2020
Research and Innovation programme under Grant Agreement No. **958363**

DATA.ZERO

Runout optimized pressing of head shaft and chuck[A2.3]

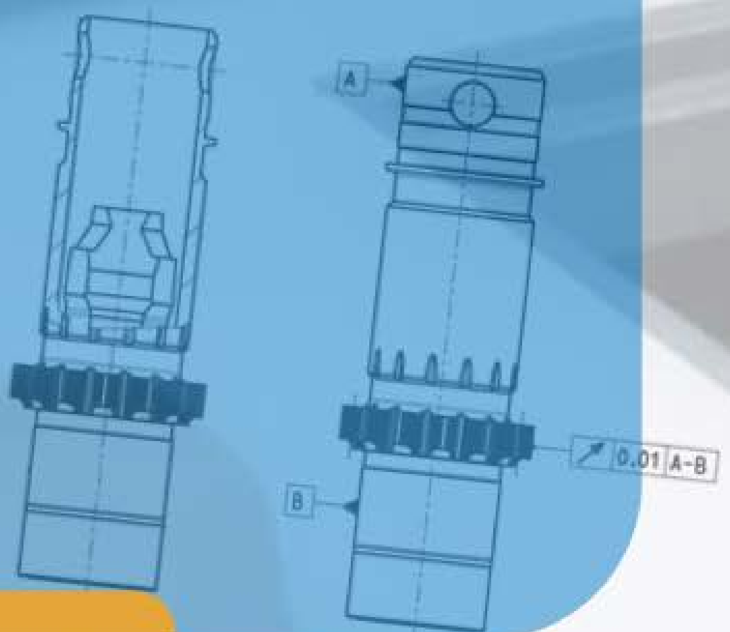
DATA.ZERO

Goal / motivation
- Runout oriented assembly
- Different parts -both with runout from 0 to 0,01mm- become one subassembly with maximum runout of 0,01mm
- Zero defect assembly

Status quo
- [...]
- Only similarity: measuring of imbalance

Details
- [...]

Challenges
- Multi references of different parts
- Drill impact vs runout of gears
- Three-dimensional wobble
- Use of the data for assembly



Data
- Q-DAS connection
- Use in automation

Context

- User story 4a - Dentsply
  - Optimise assembly to minimise overall runout
- Requirement 1:
  - A good understanding of the angular direction of runout error

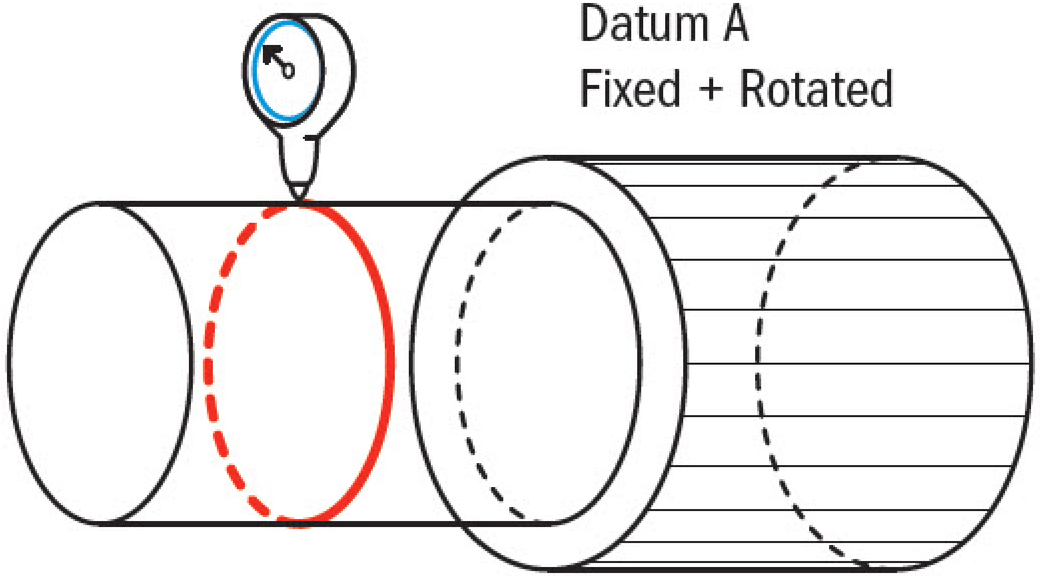
Context

- User story 4a - Dentsply
  - Optimise assembly to minimise overall runout
- Requirement 1:
  - A good understanding of the angular direction of runout error

DATA.ZERO

Traditional runout

Datum A
Fixed + Rotated

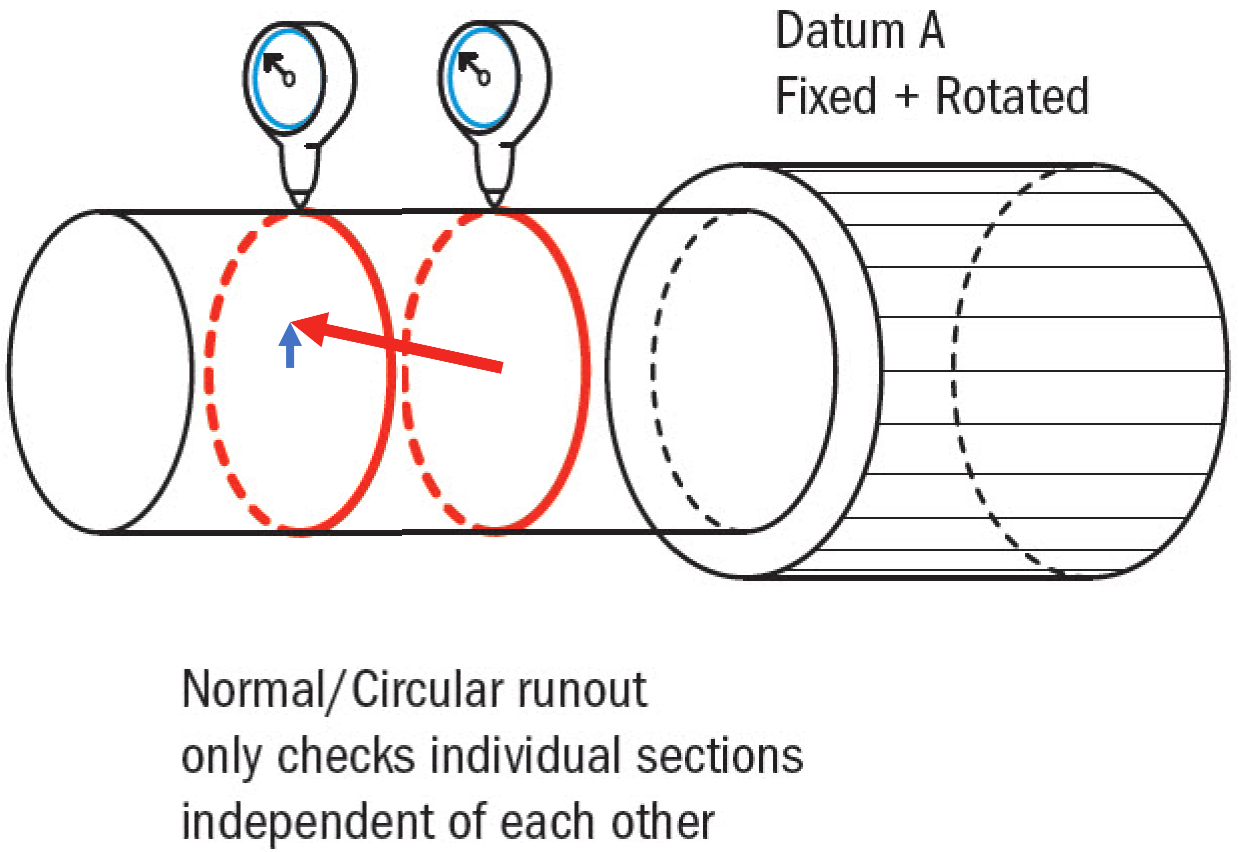


Normal/Circular runout
only checks individual sections
independent of each other

A singular magnitude – **No directional information**!

Mainly identifies **Roundness variation**

3D axial runout



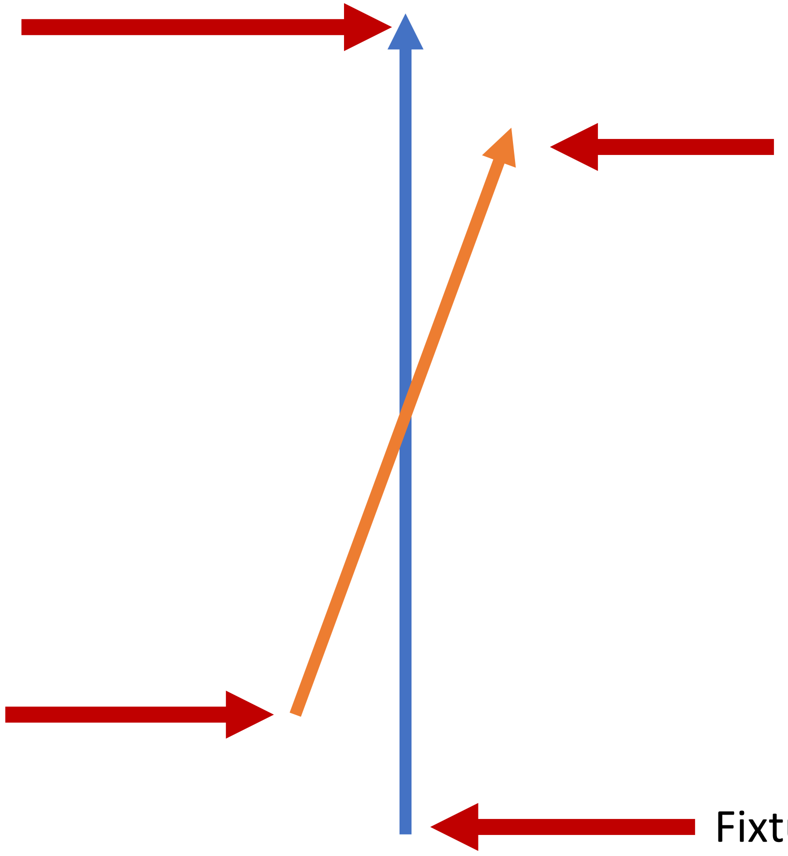
Aim: Magnitude and direction
- **Runout Vector**

Mainly identifies **Axial variation**

DATA.ZERO

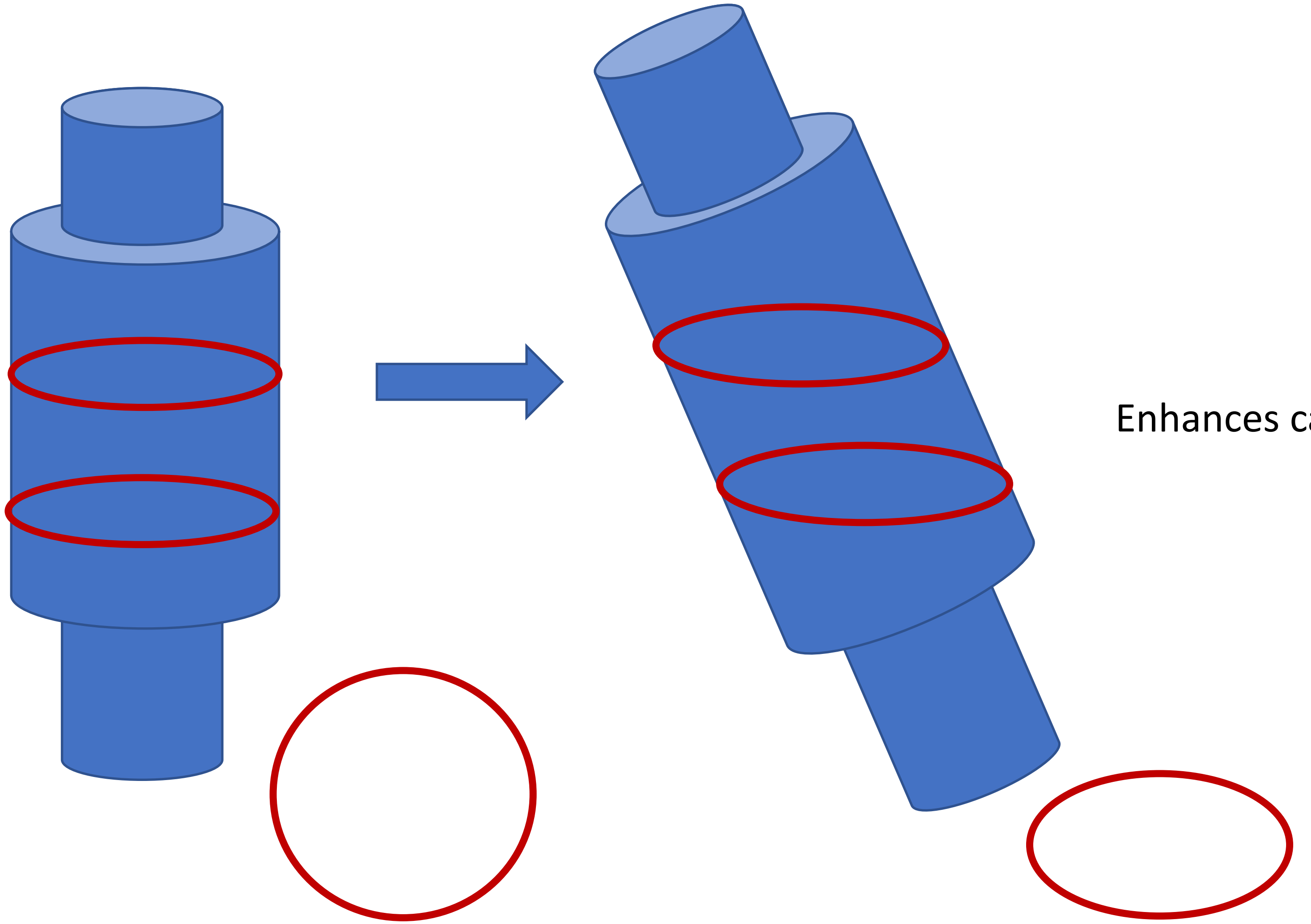
Axial runout - Process

Capture measurement data that contains axis information

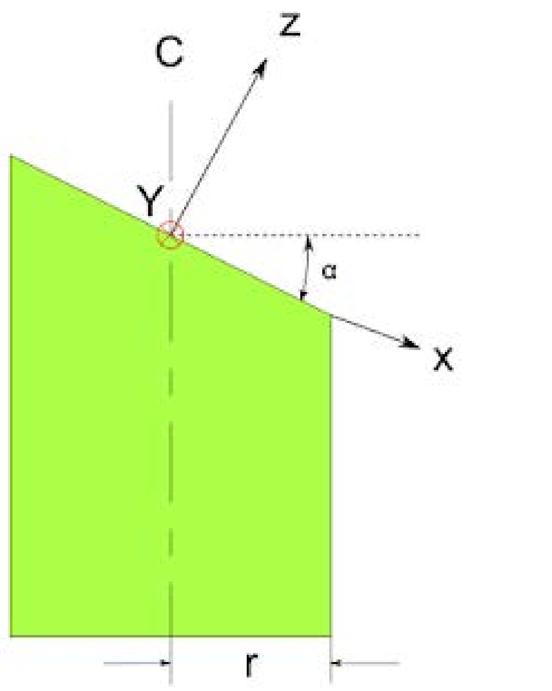


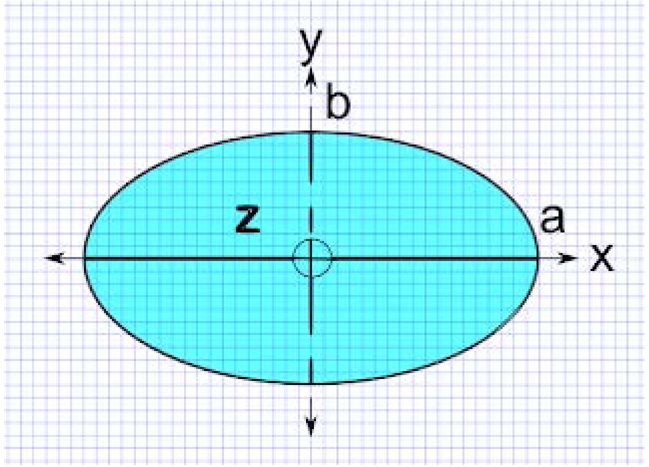
Elliptical fitting

A better method for characterising relative angle

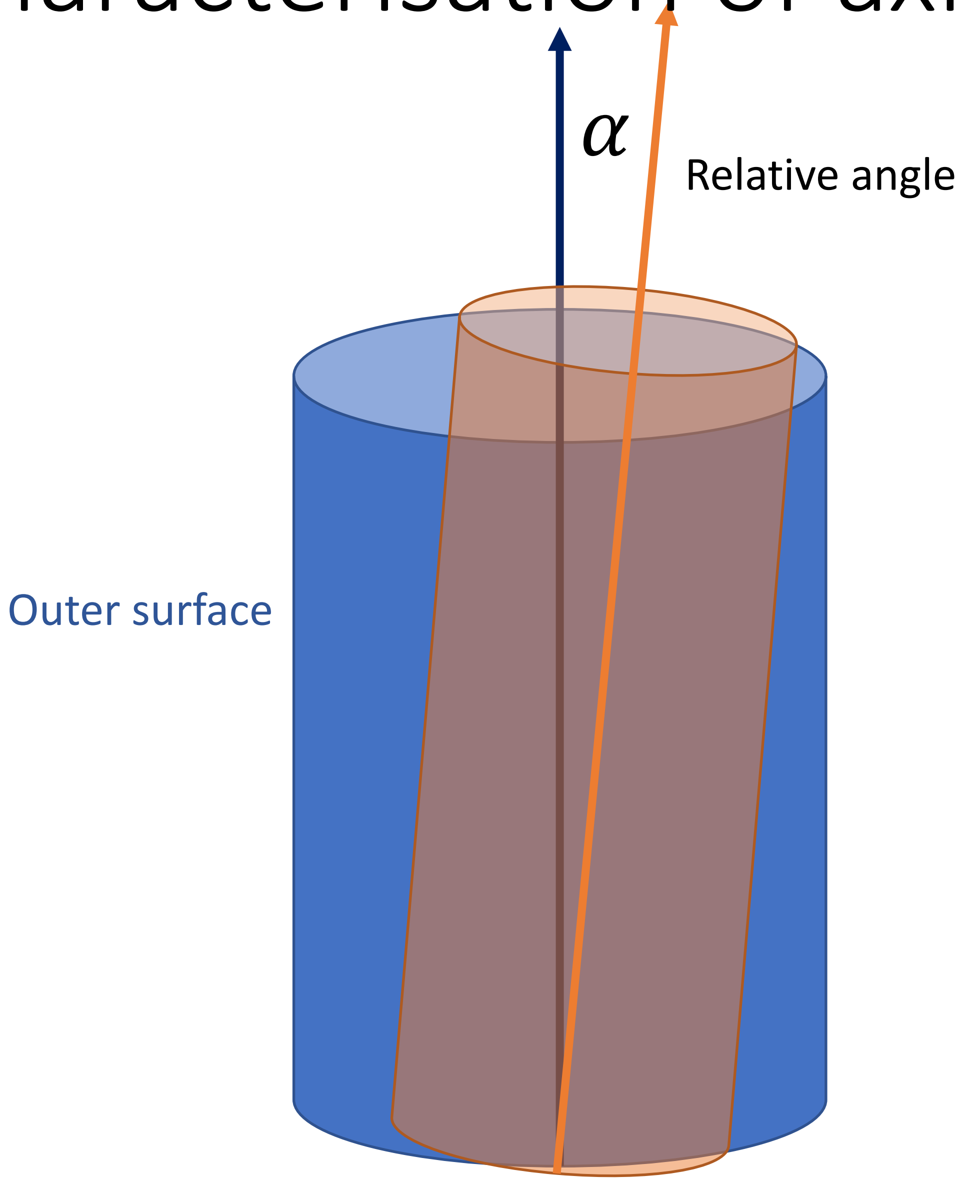


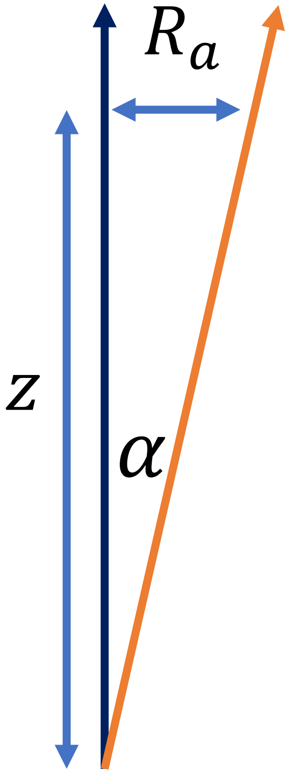
Enhances calculation of axial tilt





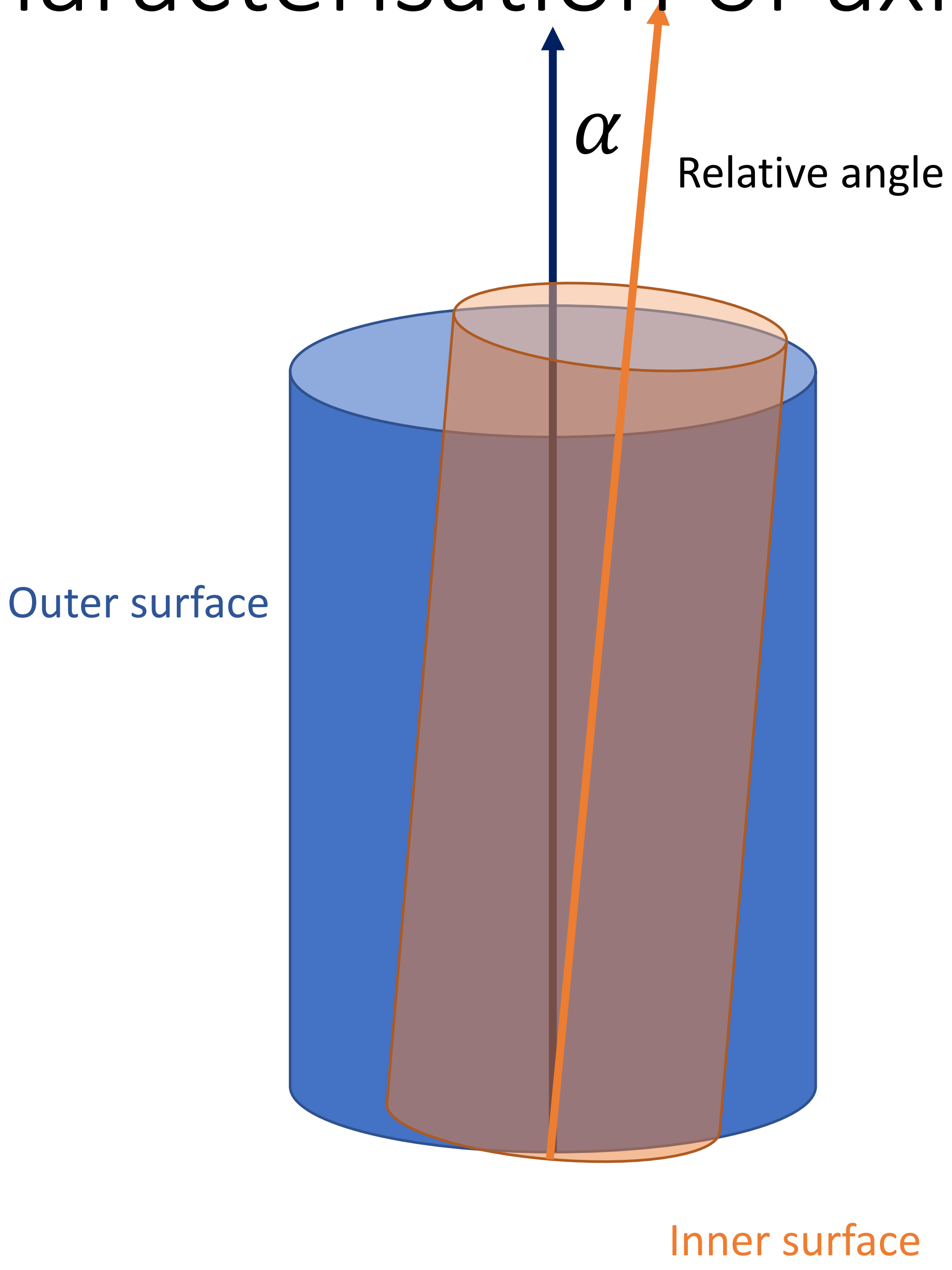
Characterisation of axial runout

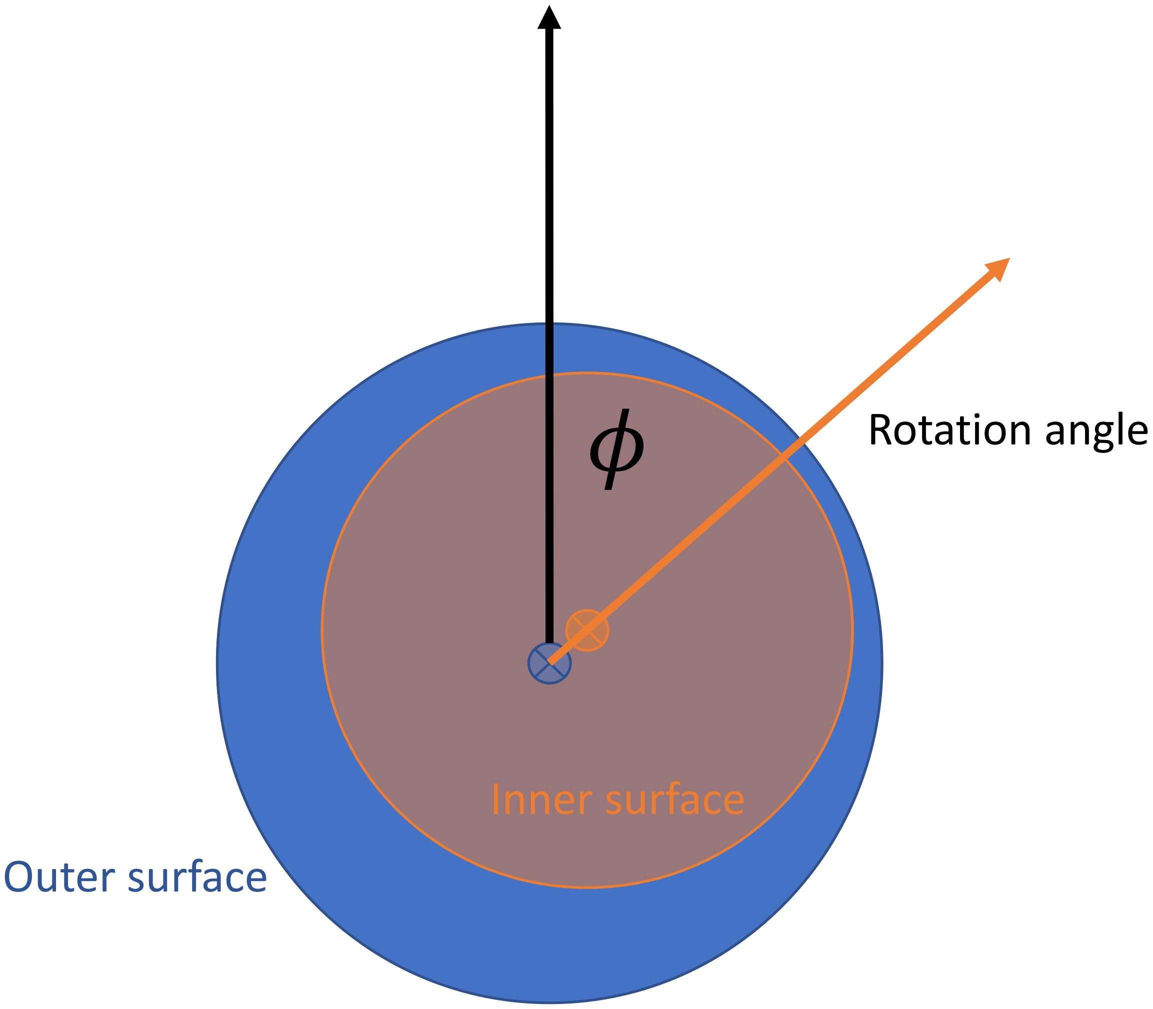




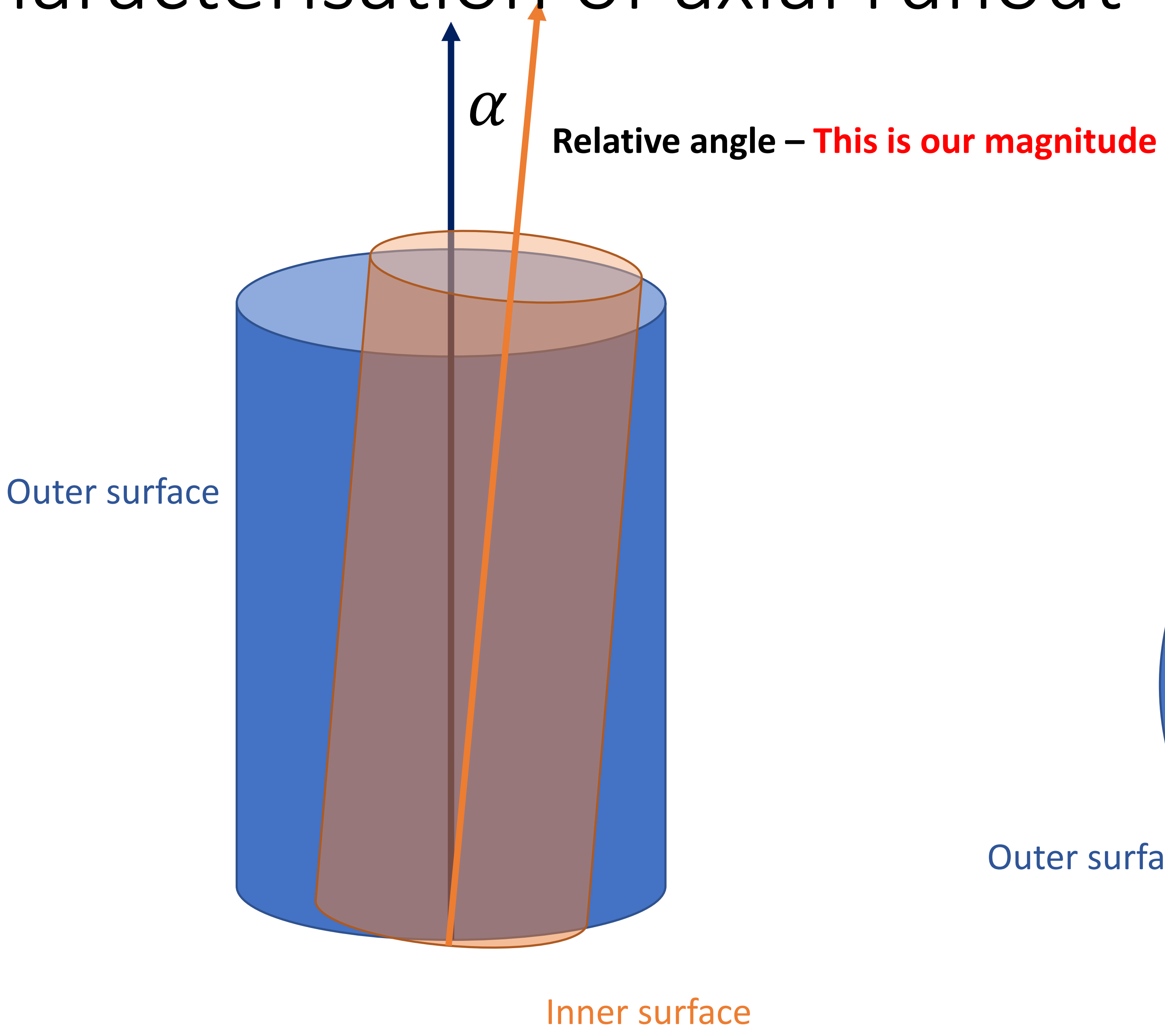
$$Axial\ runout\ magnitude\ R_a = z \tan \alpha$$

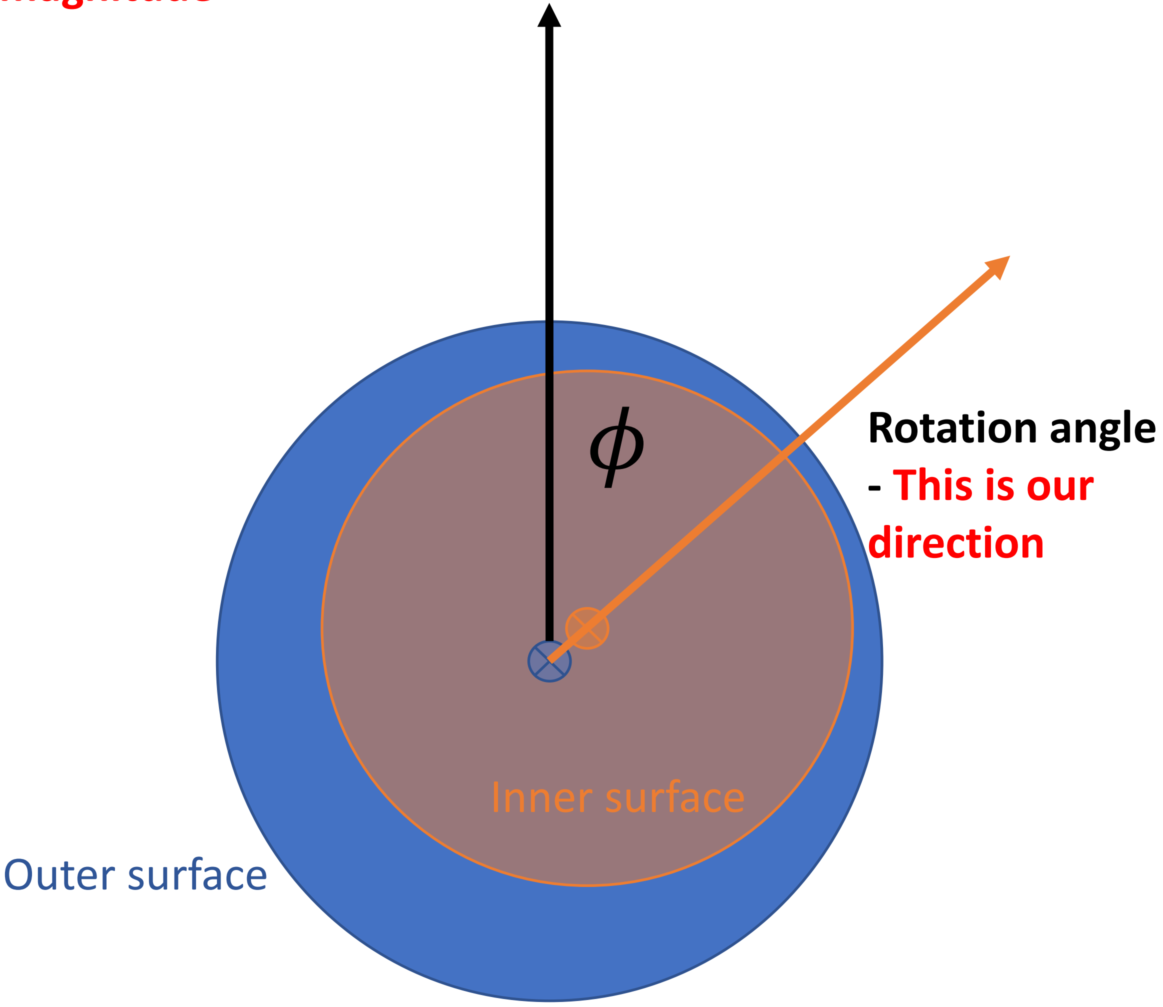
Characterisation of axial runout



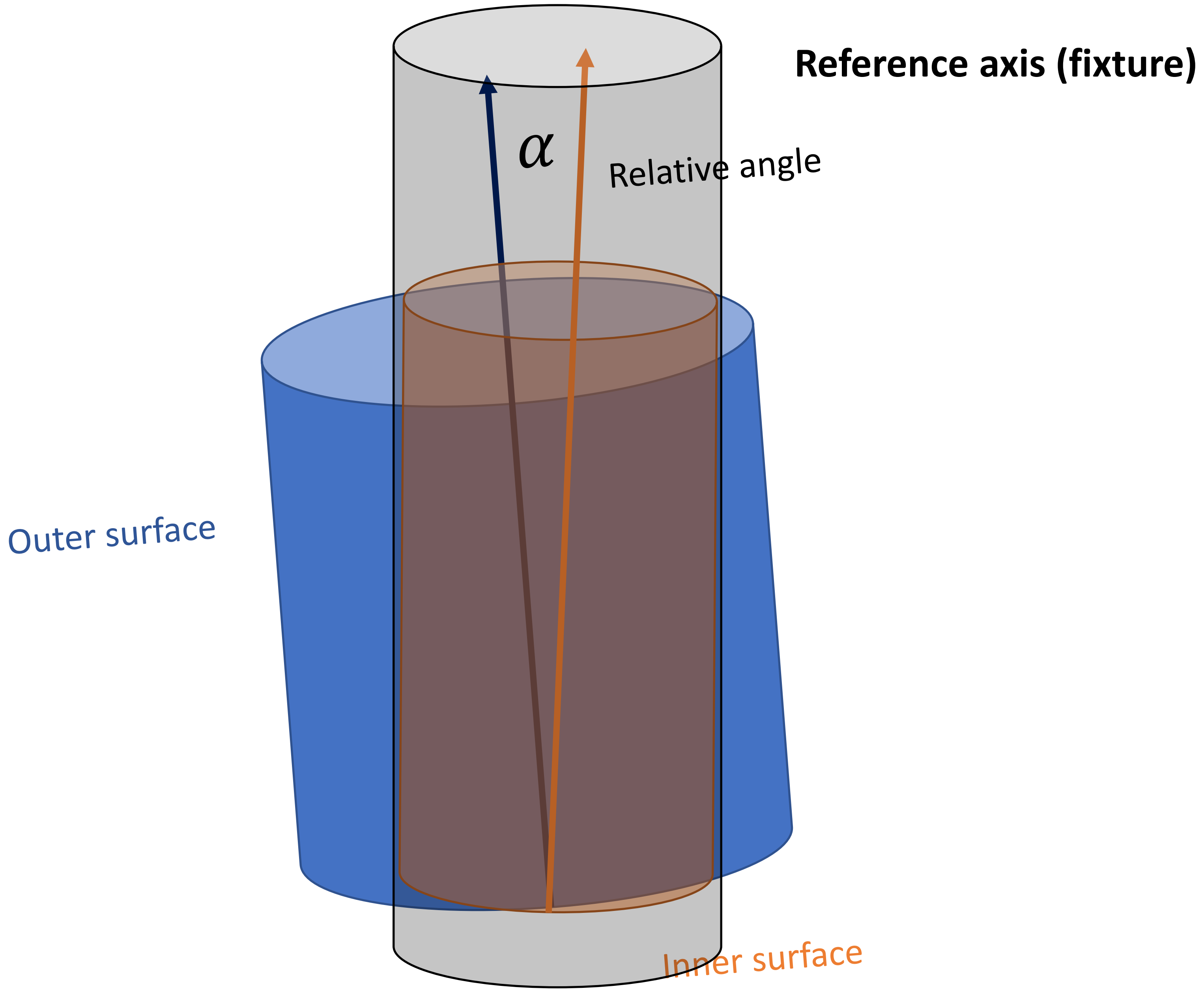


Characterisation of axial runout



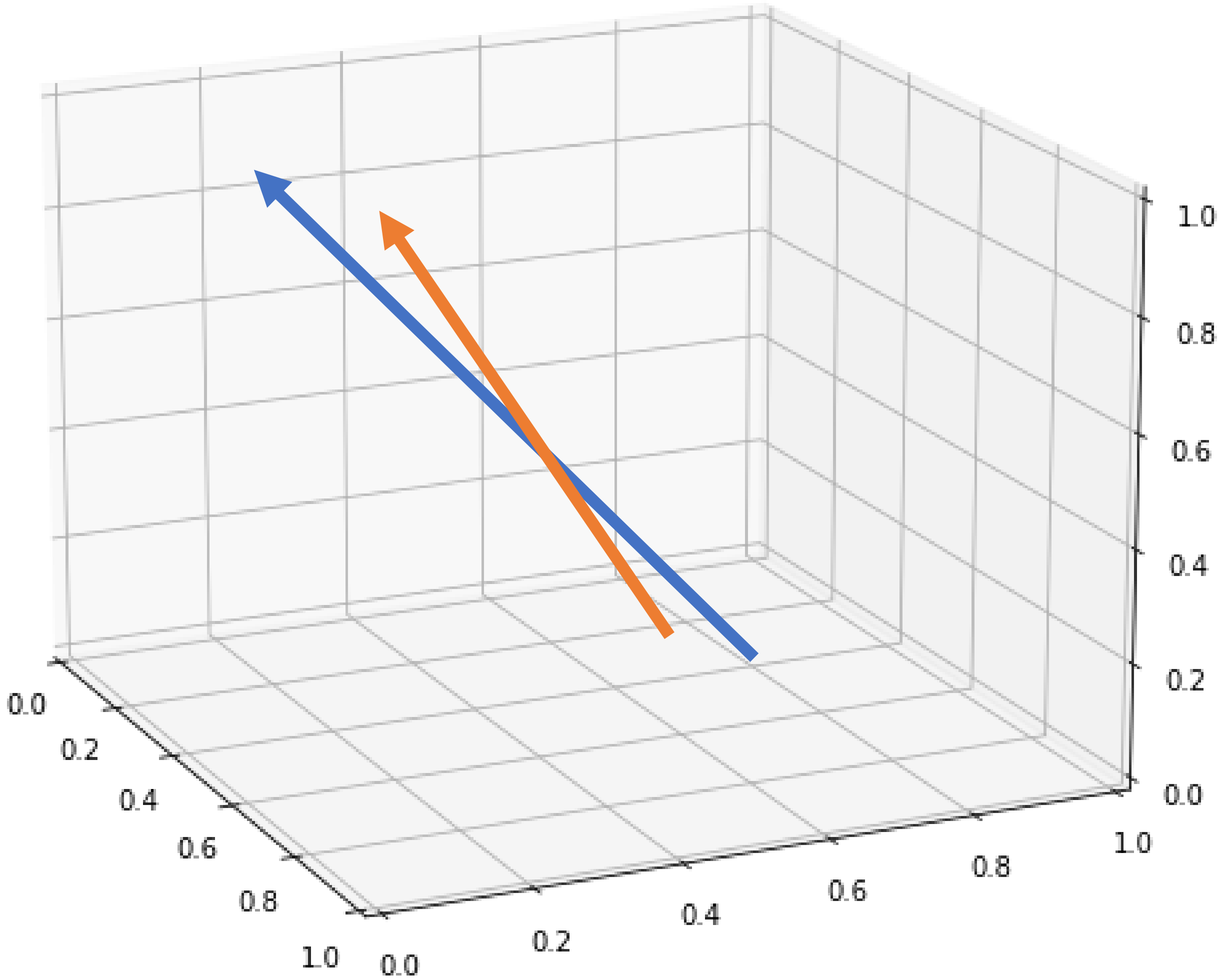


Characterisation of axial runout



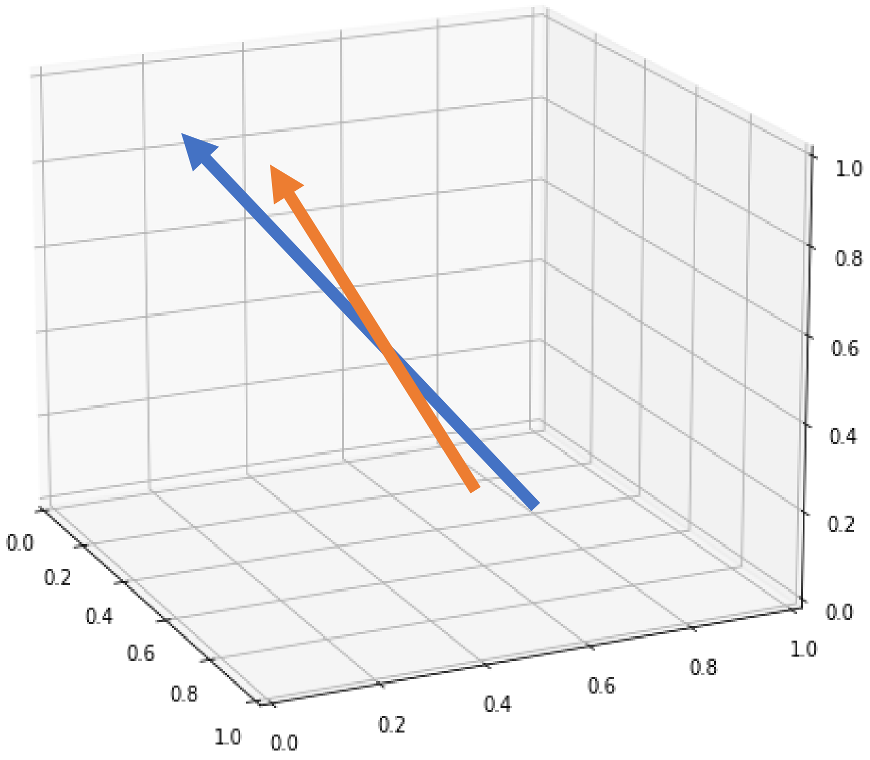
Axial runout - Process

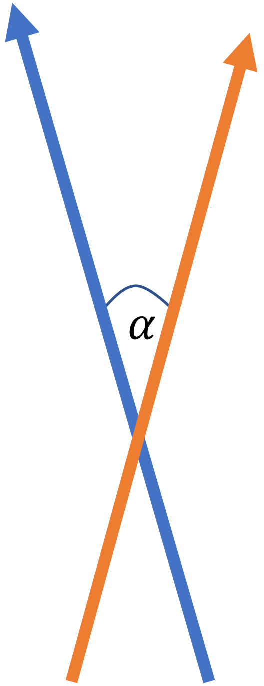
Define vectors for each measurement pair (reference and part)



Axial runout - Process

Relative angle: Calculate angle between vectors

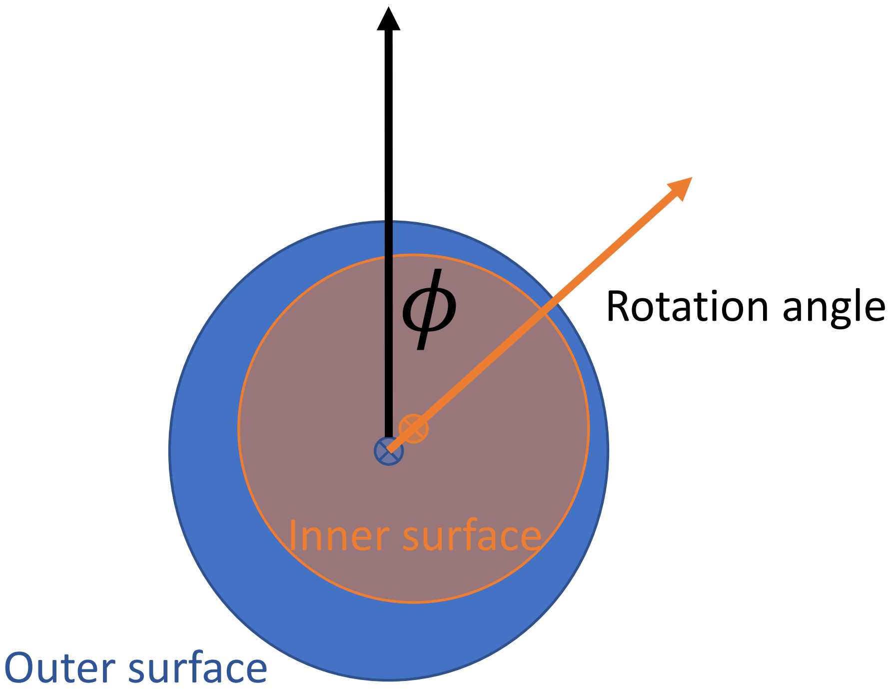


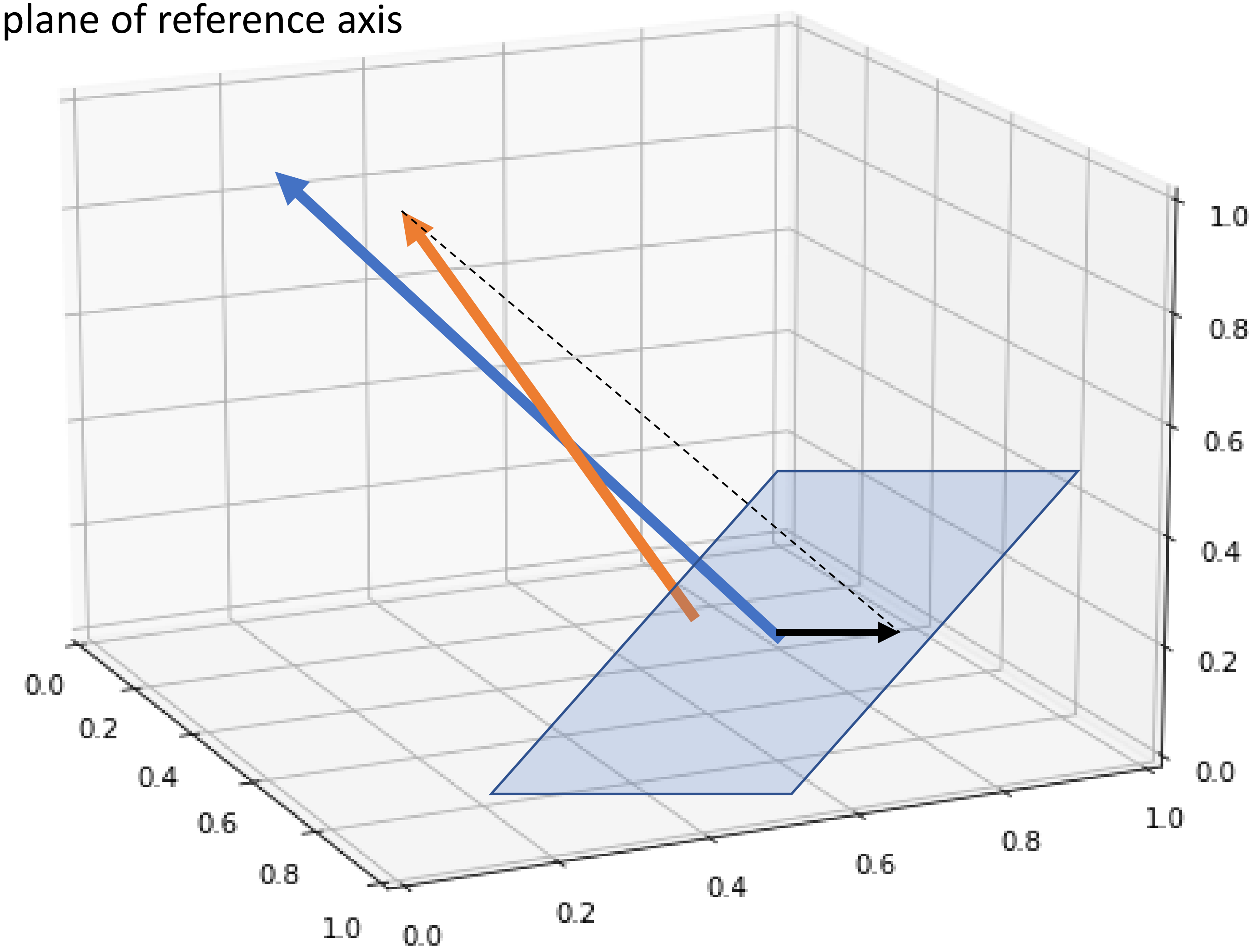


$$\alpha = \arccos \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{|\boldsymbol{a}||\boldsymbol{b}|}$$

Axial runout - Process

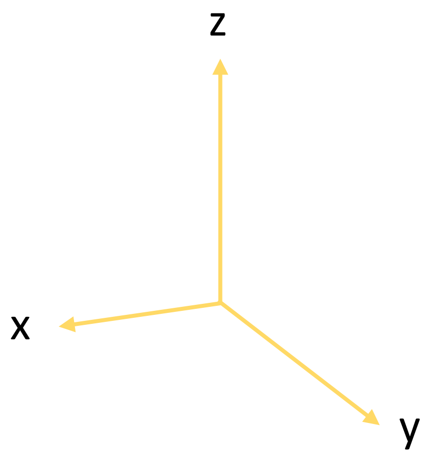
Rotation angle: Project part axis onto normal plane of reference axis

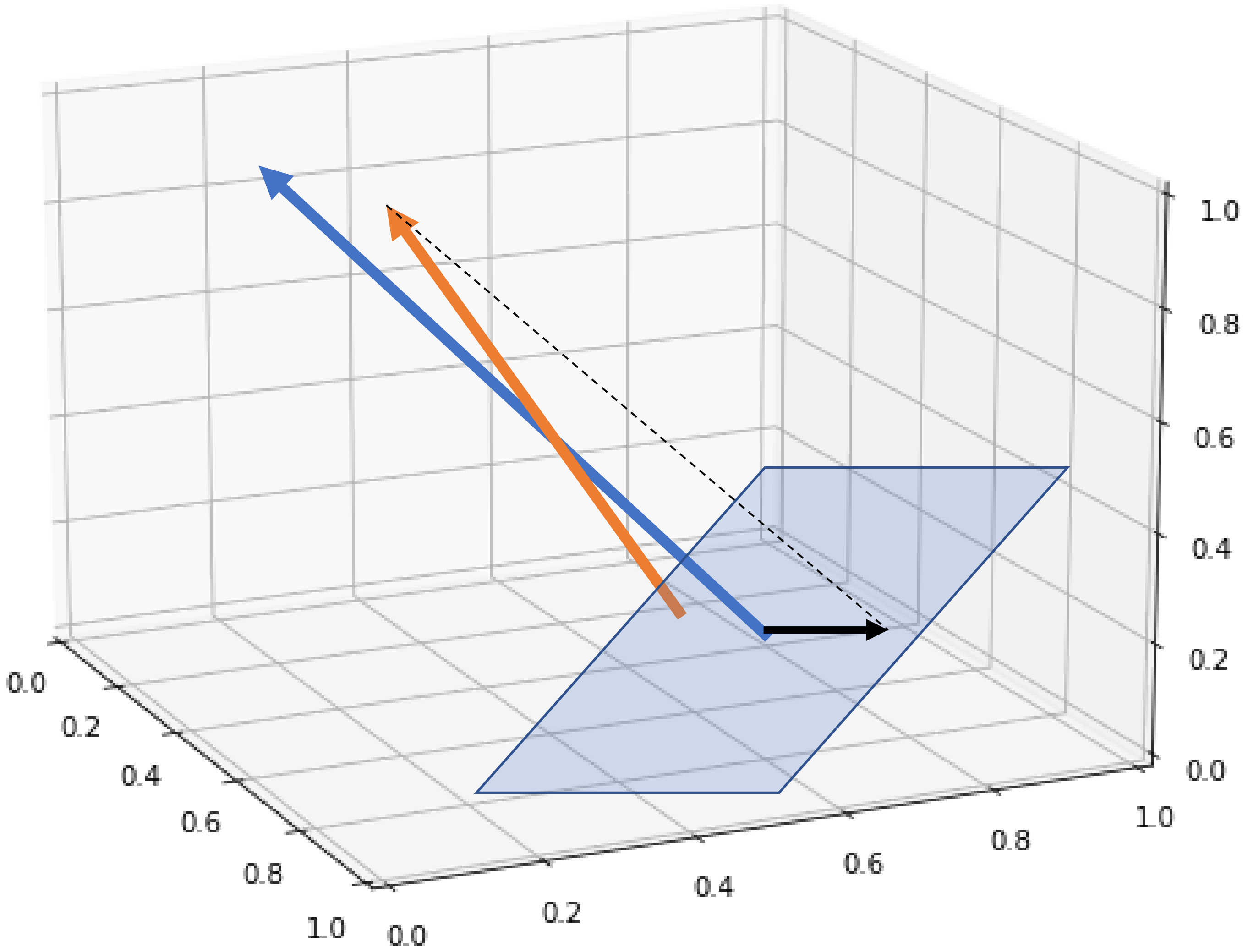




Axial runout - Process

Rotation angle: Calculate angle between part projection and y-axis projection (ensuring consistent direction)

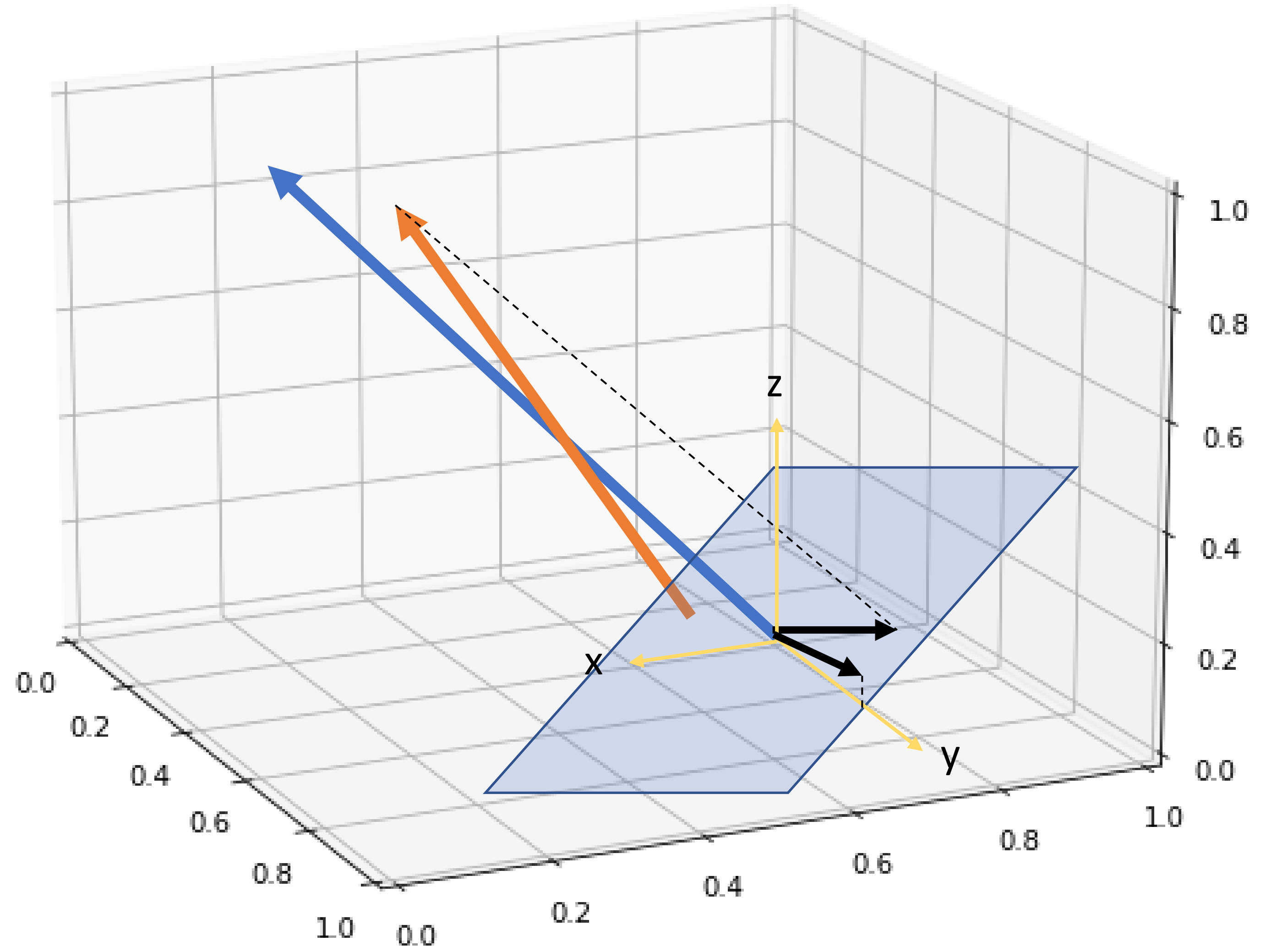




DATA.ZERO

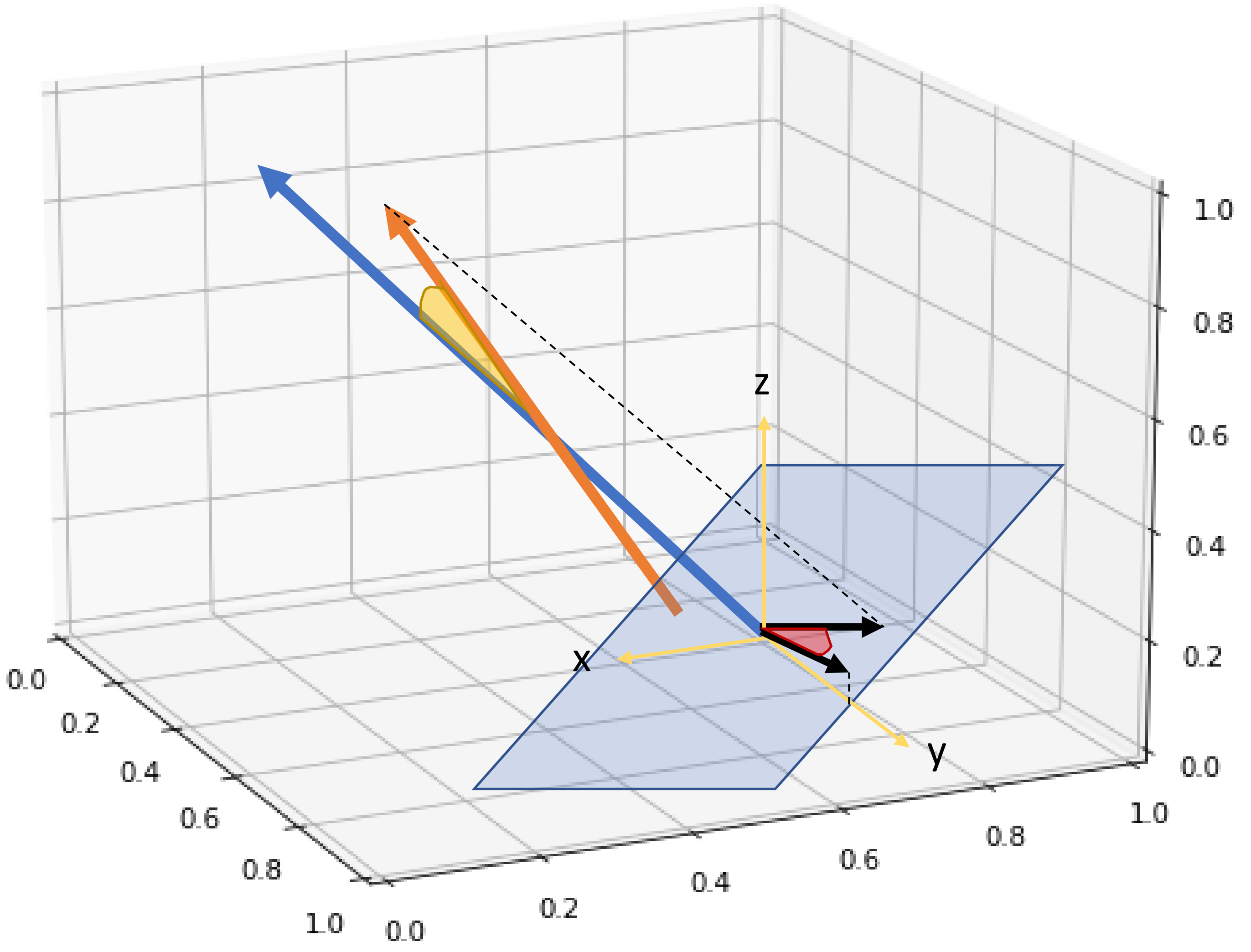
Axial runout - Process

Rotation angle: Calculate angle between part projection and y-axis projection (ensuring consistent direction)



Axial runout - Process

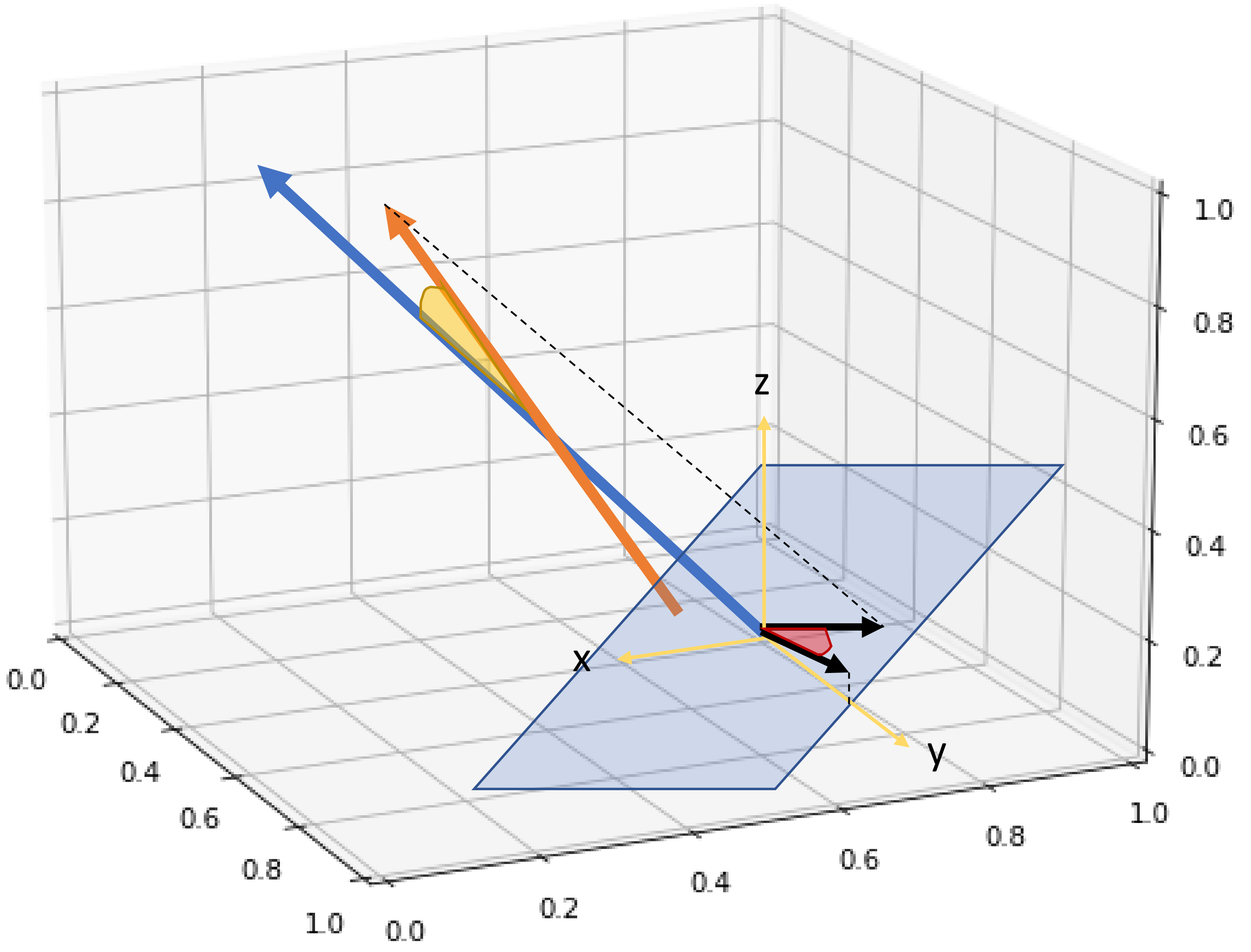
Rotation angle: Calculate angle between part projection and y-axis projection (ensuring consistent direction)



Axial runout - Process

Rotation angle: Calculate angle between part projection and y-axis projection (ensuring consistent direction)

Runout vector obtained!



Model validation

- How well does this model extract the correct angles from measurement data?
- Under what scenarios does the model fall apart?

- Ideally – this would be performed by comparing the model predictions to known angle values taken via real measurement
  - But this is the problem we're currently solving – there are no known angles!

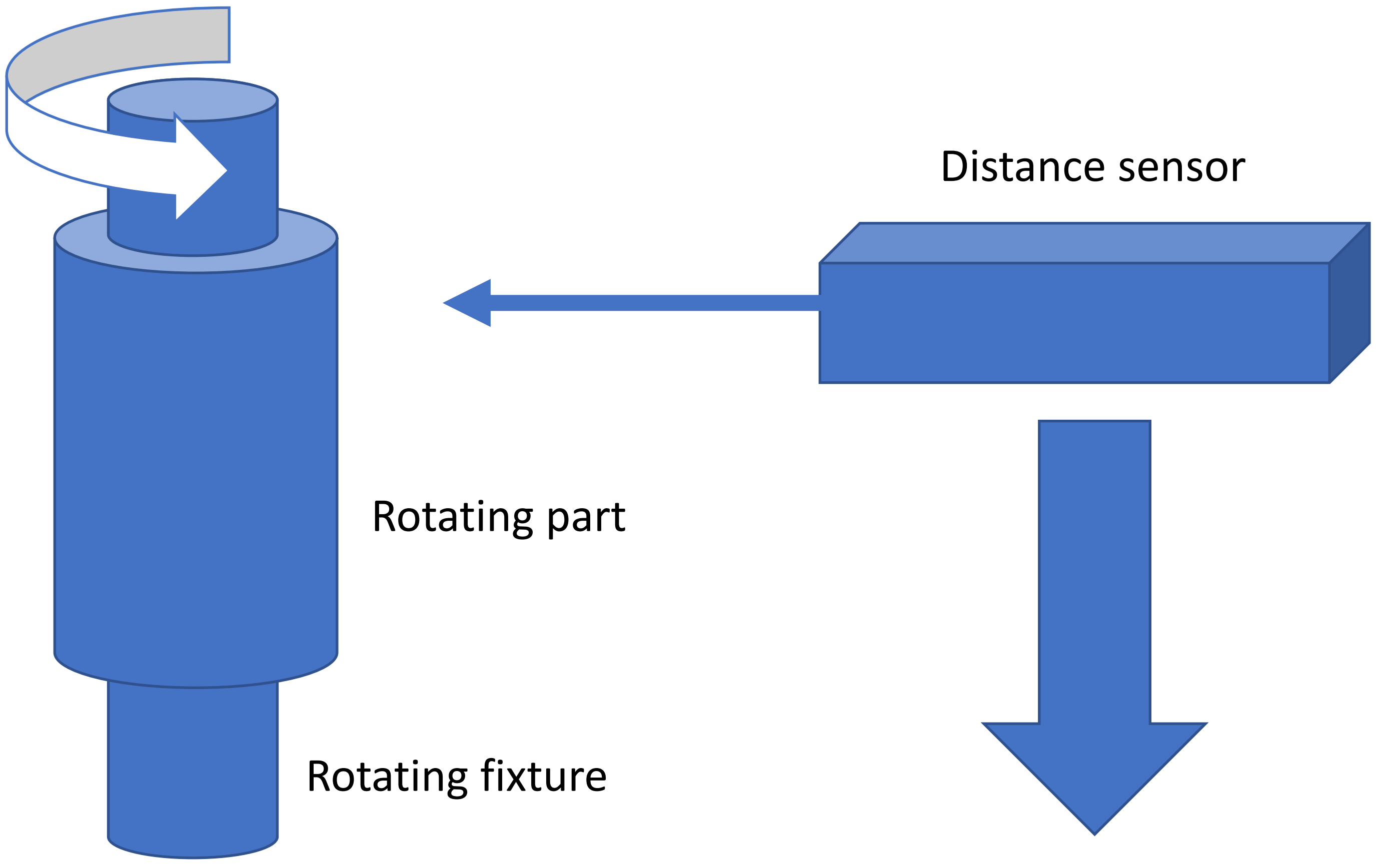
Model validation

- How well does this model extract the correct angles from measurement data?
- Under what scenarios does the model fall apart?

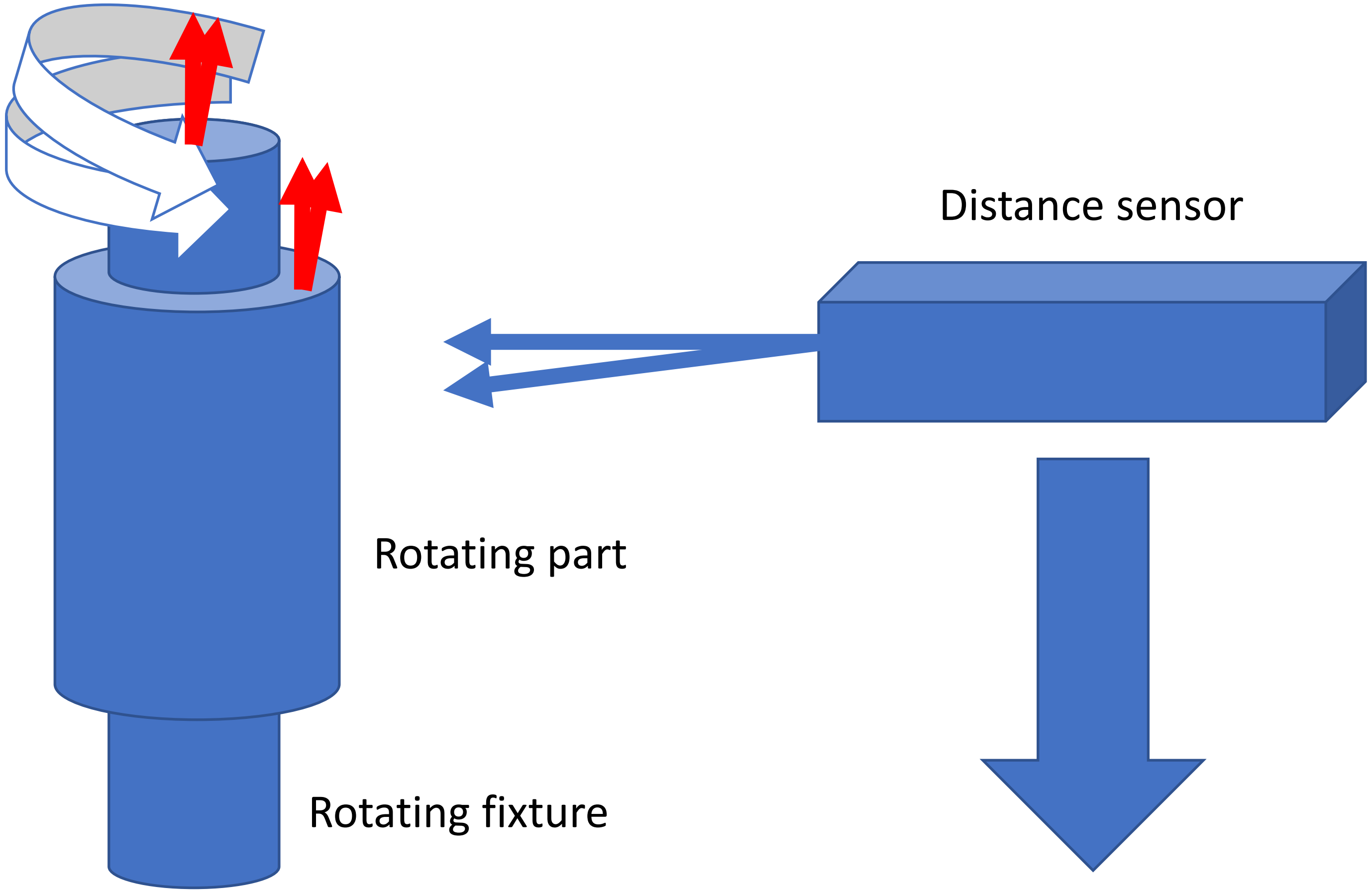
- Ideally – this would be performed by comparing the model predictions to known angle values taken via **real measurements**
  - But this is the problem we're currently solving – there are no known angles!

- Solution: Comparing the model predictions to known angles taken via **virtual measurements**

Physical measurement setup



Physical measurement setup



DATA.ZERO

Virtual instrument

- Objects:
  - Distance sensor(position, orientation)
  - Fixture/inner part surface(radius, position, orientation)
  - Part outer surface(radius, position, orientation)

- Actions:
  - Capture distance from sensor to object
  - Rotate fixture about central axis
  - Rotate part about fixture axis
  - Repeat for 360 degrees
  - Move sensor to next position
  - Repeat for all required measurements

```
def FullMeasurement(sensor, part, fixture, measurementID):
    mitaka.ZShift(0.025)
    fixtureTop = MeasurementSystem(sensor, fixture)
    fixtureTop.Capture(rotationAxis)
    fixtureTop.SavePoints('fixtureSim_top_' + str(measurementID))

    mitaka.ZShift(-0.003)
    partTop = MeasurementSystem(sensor, part)
    partTop.Capture(rotationAxis)
    partTop.SavePoints('sim_top_' + str(measurementID))

    mitaka.ZShift(-0.008)
    partBottom = MeasurementSystem(sensor,part)
    partBottom.Capture(rotationAxis)
    partBottom.SavePoints('sim_bottom_' + str(measurementID))

    mitaka.ZShift(-0.005)
    fixtureBottom = MeasurementSystem(sensor, fixture)
    fixtureBottom.Capture(rotationAxis)
    fixtureBottom.SavePoints('fixtureSim_bottom_' + str(measurementID))
    mitaka.ZShift(-0.009)
```

Virtual instrument

- Validation process:
  - Define a large set of simulated parts with varying levels of runout
  - Obtain measurement datasets using the virtual instrument
  - Feed the measurement data into the model
  - **Compare model predictions to known angles**

```
def FullMeasurement(sensor, part, fixture, measurementID):
    mitaka.ZShift(0.025)
    fixtureTop = MeasurementSystem(sensor, fixture)
    fixtureTop.Capture(rotationAxis)
    fixtureTop.SavePoints('fixtureSim_top_' + str(measurementID))

    mitaka.ZShift(-0.003)
    partTop = MeasurementSystem(sensor, part)
    partTop.Capture(rotationAxis)
    partTop.SavePoints('sim_top_' + str(measurementID))

    mitaka.ZShift(-0.008)
    partBottom = MeasurementSystem(sensor,part)
    partBottom.Capture(rotationAxis)
    partBottom.SavePoints('sim_bottom_' + str(measurementID))

    mitaka.ZShift(-0.005)
    fixtureBottom = MeasurementSystem(sensor, fixture)
    fixtureBottom.Capture(rotationAxis)
    fixtureBottom.SavePoints('fixtureSim_bottom_' + str(measurementID))
    mitaka.ZShift(-0.009)
```

DATA.ZERO

Model validation

- A total of 900 permutations were created with a variety of relative and rotation angles
- Each measured four times as per measurement protocol
- Input into model…

```
theta = [0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
phi = [0, 0.1, 1, 10, 100]
r = 1.6
```

Fixture top
Part top
Part bottom
Fixture bottom

Model validation

- A total of 900 permutations were

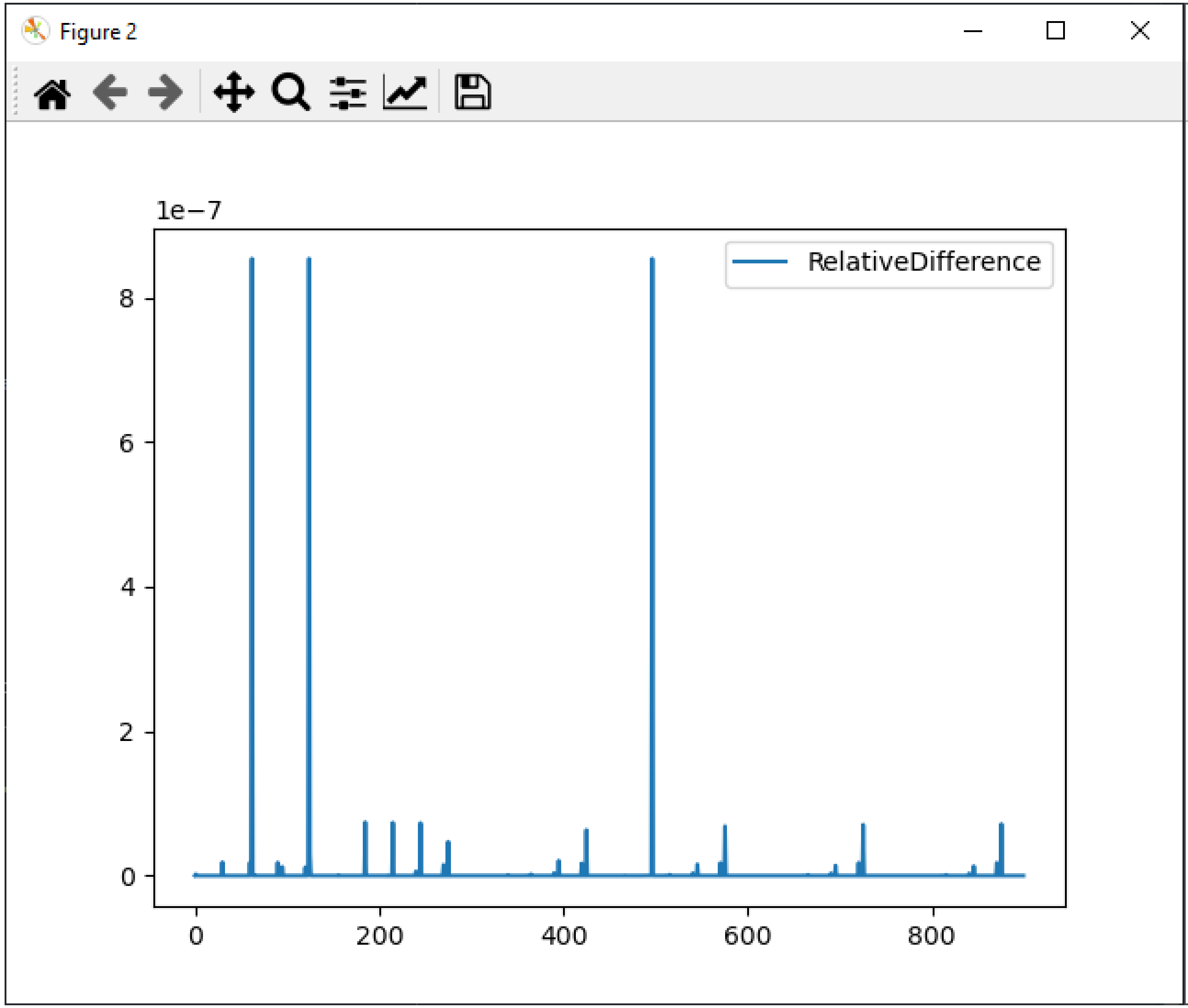
```
theta = [0.1, 0.2, 0.5, 1.0, 2.0, 5.0]
```

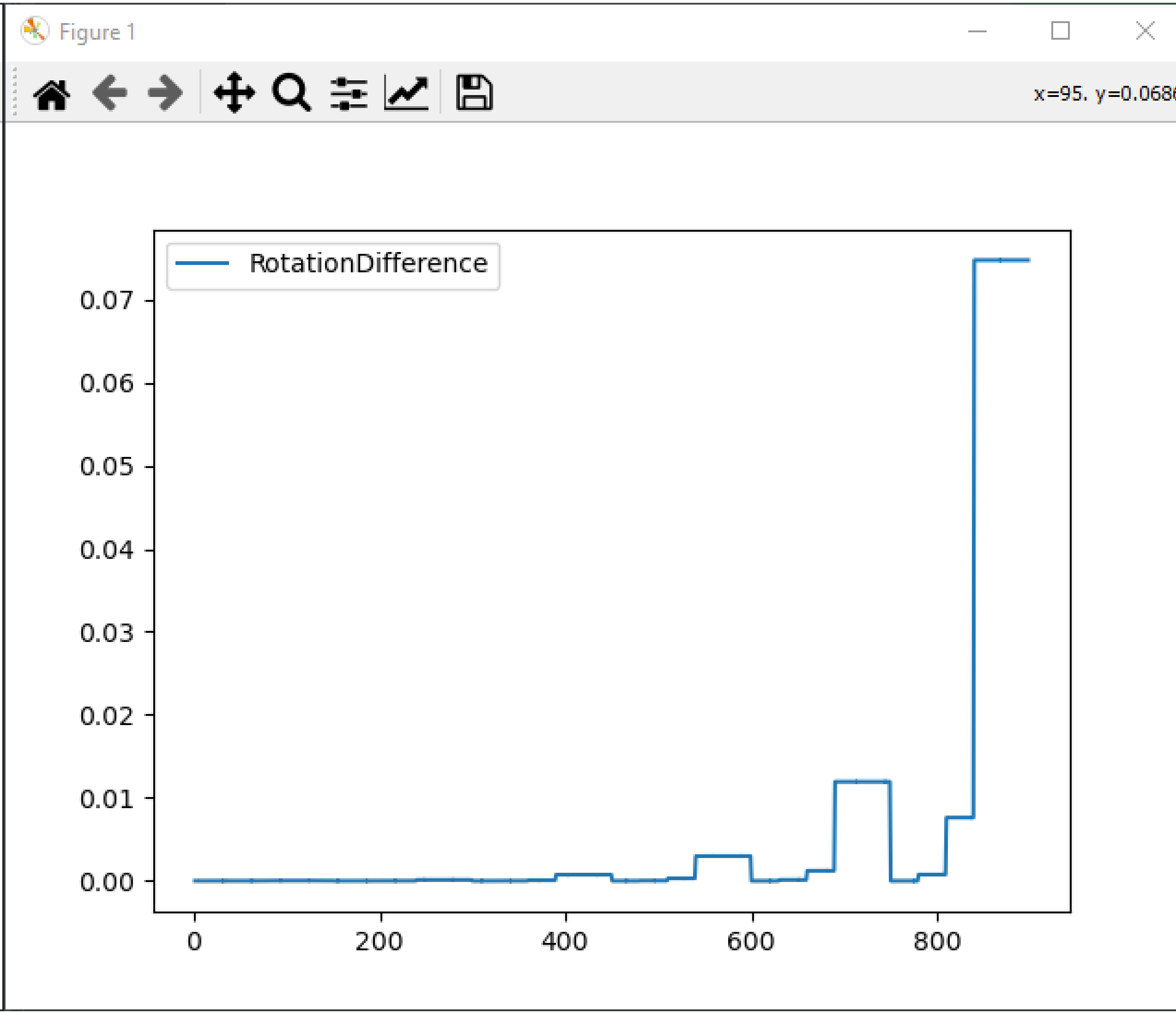
df - DataFrame

Index	Rotation Angle (Simulation)	Relative Angle (Simulation)	Rotation Angle (Measurement)	Relative Angle (Measurement)	FixtureXRotation	PartXRotation	FixtureZRotation	PartZRotatio	RotationDifference	RelativeDifference	ixture	Part
1	90.05	0.000174534	90.05	0.000174531	0.1	0.1	0	0.1	5.16063e-06	2.08823e-09	.1	1.1
2	90.5	0.00174531	90.5	0.00174531	0.1	0.1	0	1	2.78807e-07	2.08826e-10	.1	10.1
3	95	0.0174311	95	0.0174311	0.1	0.1	0	10	3.41815e-08	2.09088e-11	.1	100…
4	140	0.153209	140	0.153209	0.1	0.1	0	100	3.93067e-09	4.75778e-12	.1	100…
5	0	0.1	4.20159e-05	0.1	0.1	0.2	0	0	4.20159e-05	3.64467e-12	.1	0.2
6	0.199999	0.1	0.199999	0.1	0.1	0.2	0	0.1	1.33656e-10	7.28928e-12	.1	1.2
7	1.99969	0.10003	1.99969	0.10003	0.1	0.2	0	1	1.83187e-09	7.28707e-12	.1	10.2
8	19.7065	0.102994	19.7065	0.102994	0.1	0.2	0	10	2.58318e-11	7.07742e-12	.1	100…
9	124.374	0.238633	124.374	0.238633	0.1	0.2	0	100	7.28804e-10	1.52733e-12	.1	100…
10	0	0.4	4.12456e-10	0.4	0.1	0.5	0	0	4.12456e-10	1.82232e-12	.1	0.5
11	0.124999	0.4	0.124999	0.4	0.1	0.5	0	0.1	4.86245e-10	9.11171e-12	.1	1.5
12	1.24997	0.400019	1.24997	0.400019	0.1	0.5	0	1	3.91987e-10	3.64453e-12	.1	10.5
13	12.4763	0.401895	12.4763	0.401895	0.1	0.5	0	10	3.61686e-10	7.25503e-12	.1	100…
14	110.777	0.526654	110.777	0.526654	0.1	0.5	0	100	3.75934e-10	3.46023e-12	.1	100…
16	0.11111	0.9	0.11111	0.9	0.1	1	0	0.1	3.81539e-11	1.37692e-11	.1	1.1
17	1.11109	0.900017	1.11109	0.900017	0.1	1	0	1	2.28024e-11	6.88449e-12	.1	10.1

Format Resize Background color Column min/max Close

Model validation

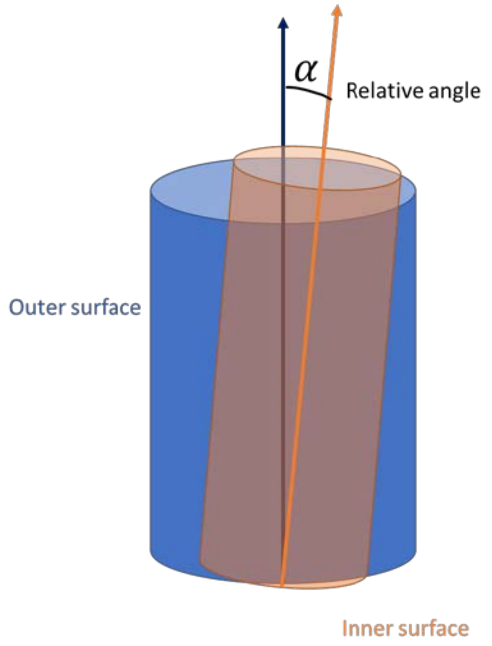


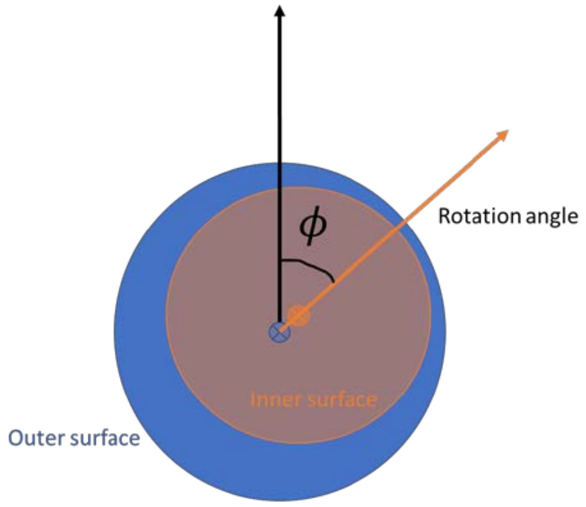


Model validation

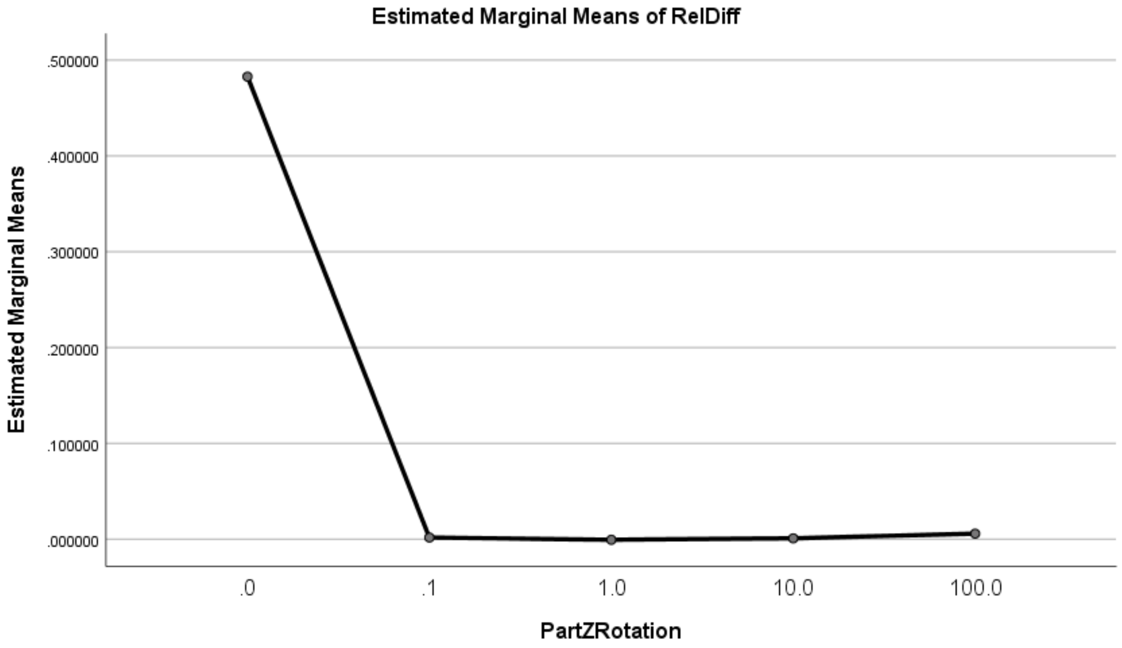
- Analysis of Variance (ANOVA) was used to identify significant factors contributing to model error for the two critical metrics:

- Relative angle (runout magnitude)
- Rotation angle (runout direction)

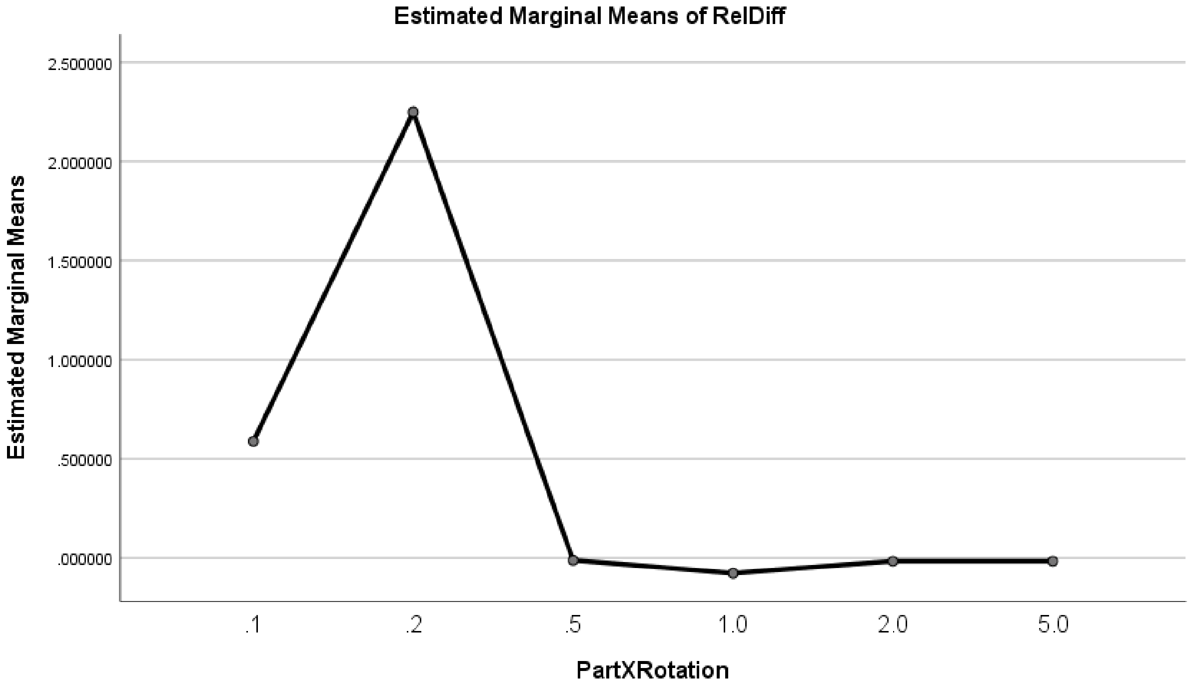




Relative Angle

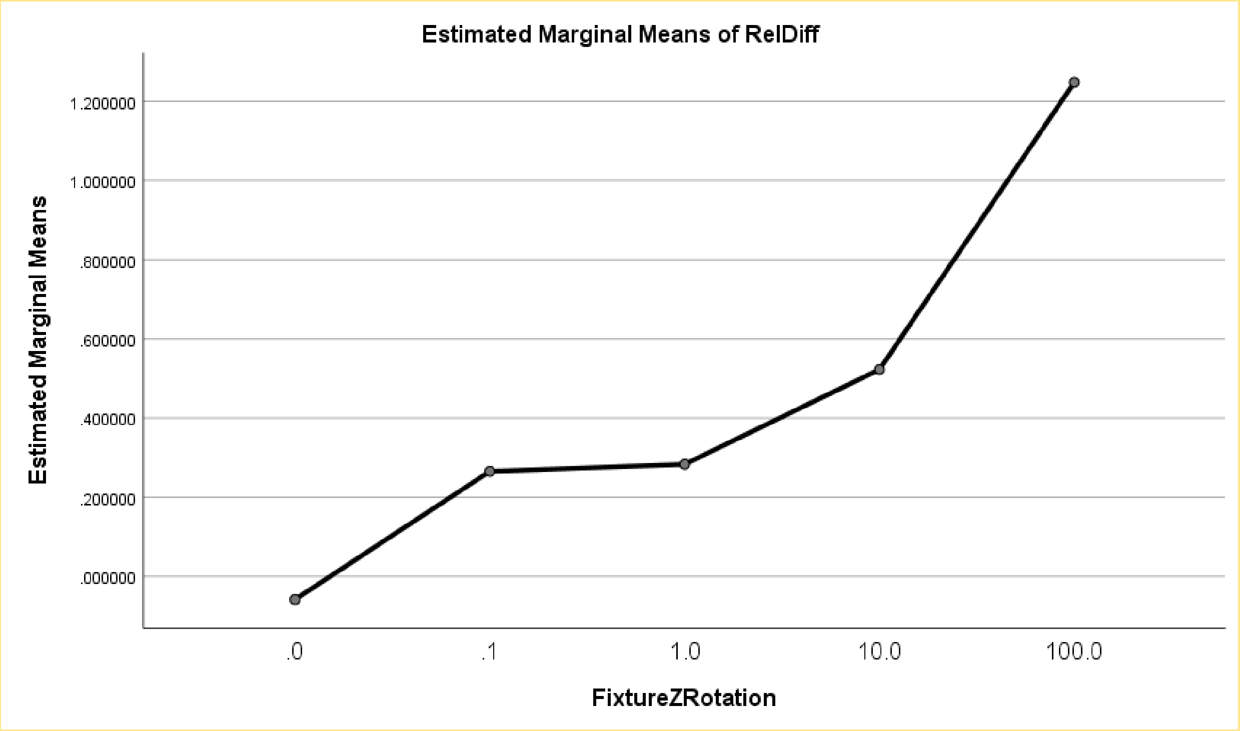


Errors occur when the part is well aligned to the rotation axis…



Relative angle

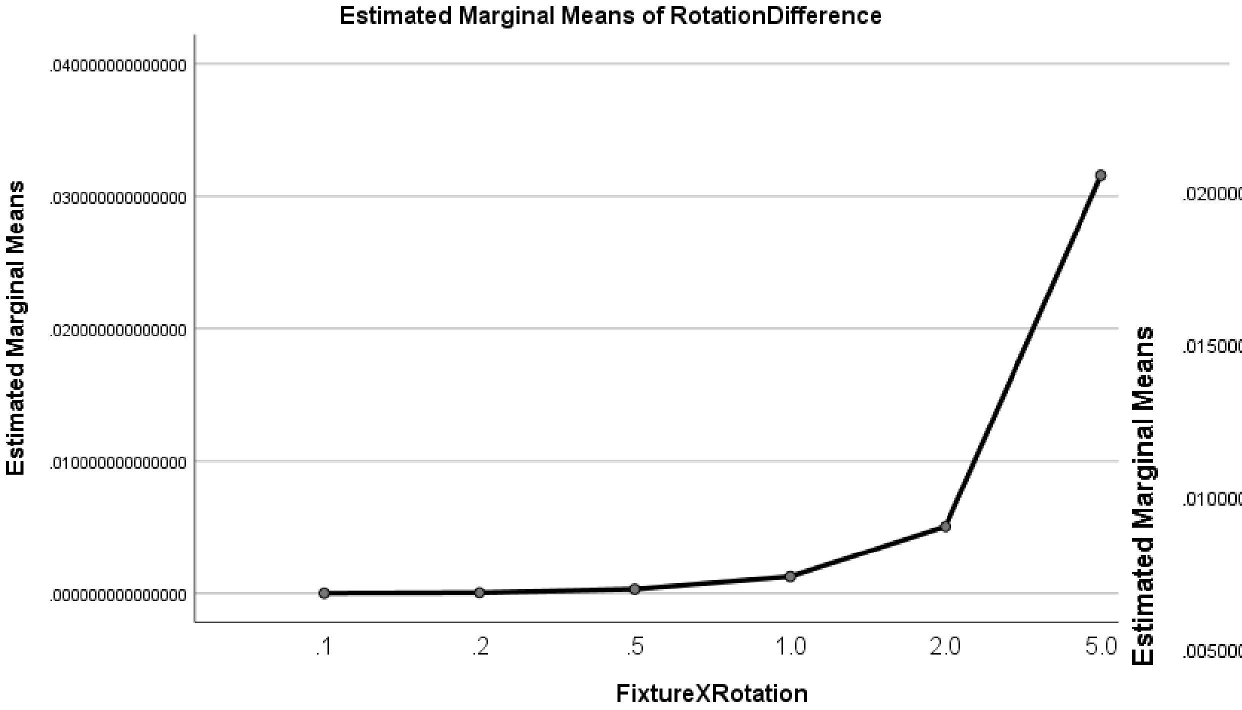
… and the fixture is poorly aligned!

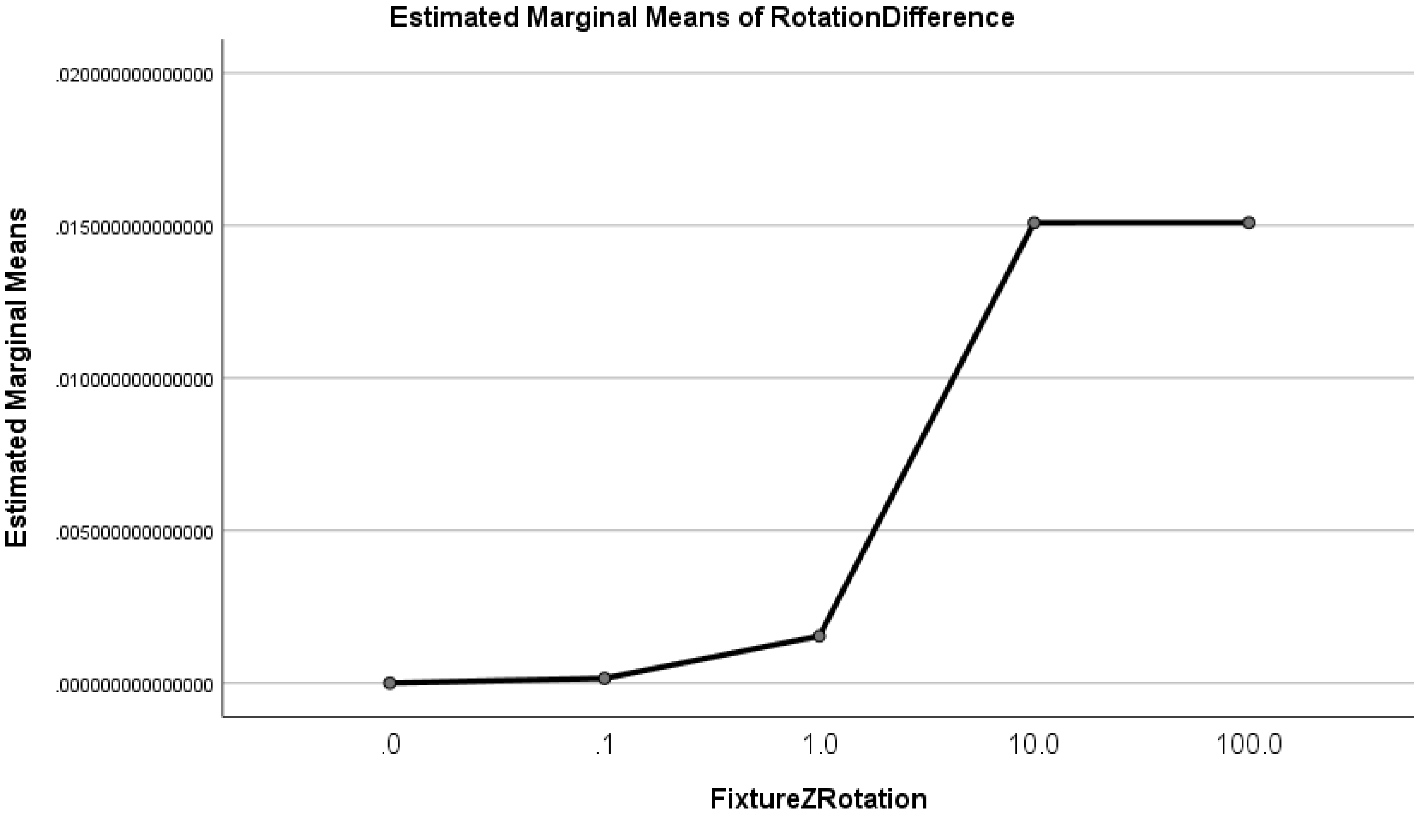


But, magnitudes are tiny (< 1e-6°), so not overwhelming measurement accuracy requirements

Rotation angle

Large misalignment of the fixture leads to large rotation angle errors (0.07°)

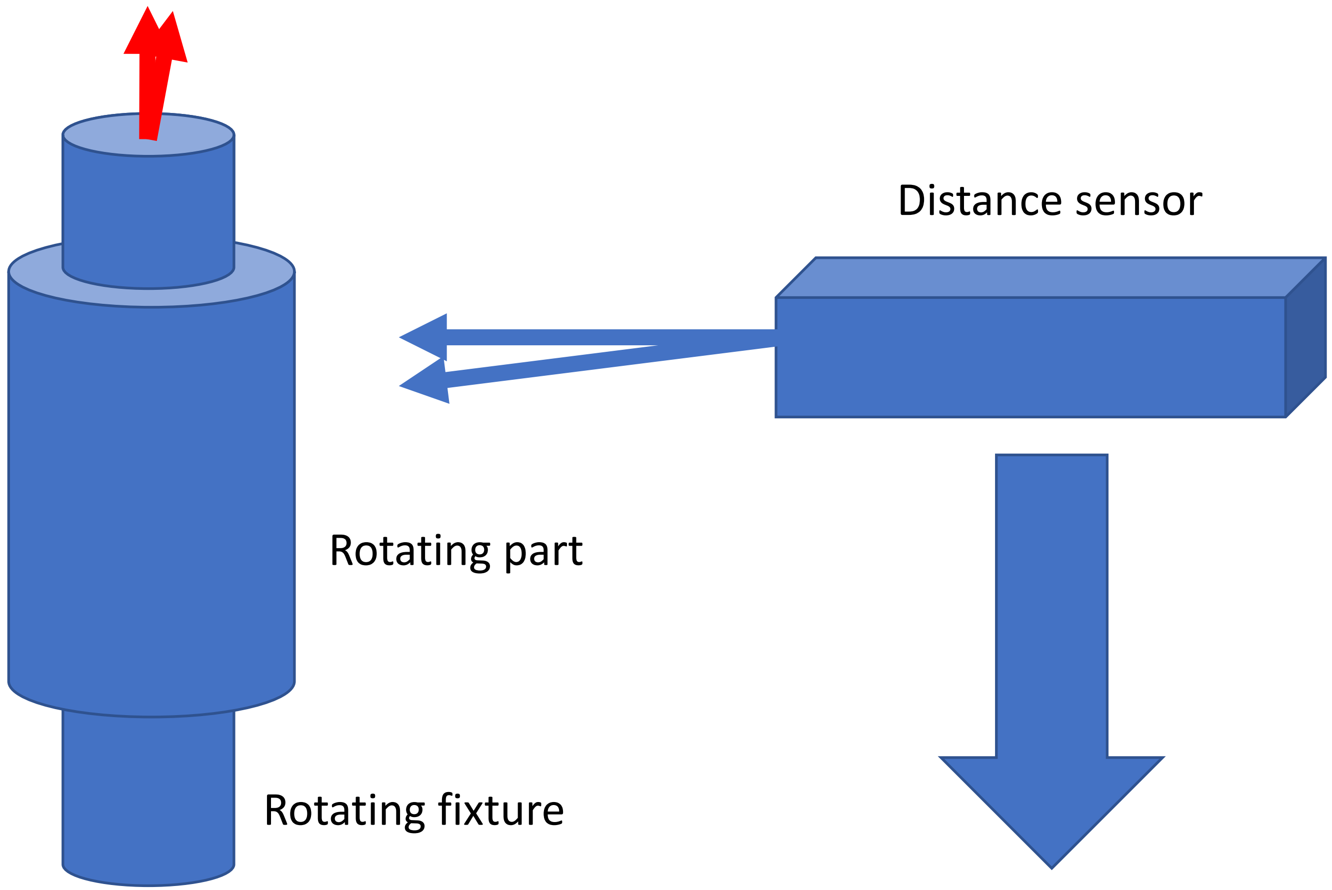




Model validation - conclusion

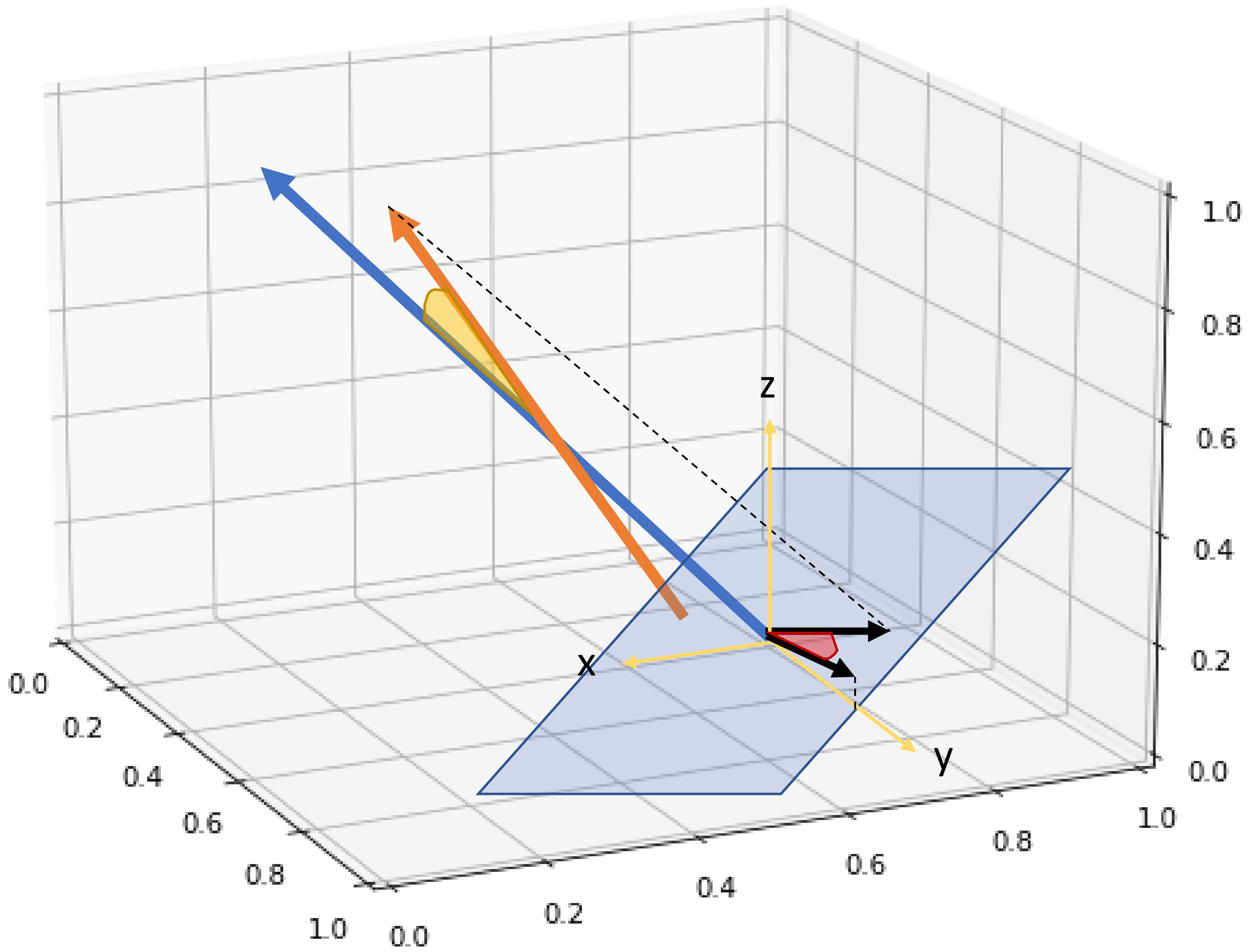
- For the model to be accurate for purpose:
  - the fixture/sensor pair should be perpendicularly aligned (90°),
  - And the fixture should be aligned with the rotation axis,

With total angular errors **smaller than 5°**



Conclusion

- Runout Vector (Magnitude and direction) characterisation has been developed
- Validation with virtual instrument



Thank you

DAT4.Zero has received funding from the European Union's Horizon 2020 Research and Innovation programme under Grant Agreement No. **958363**