**Description:**

|  |
| --- |
| Rotation vectors is different from the PolyScope and port 30003 – see example below    **The reason is:**  In PolyScope we scale the rotation vector so that it looks more stable and doesn't flicker so much. The data on port 30003 is unscaled, but when scaled it corresponds to the GUI values shown in the customer's screenshot.  Attached/below is a Python program that exemplifies the scaling. When run it produces the following results:  PolyScope SCALED value: [0.0012193680503253582, -3.166495598686568, -0.03951768623096099]  PolyScope SCALED value: [2.4759166894662425, -5.364486160510192, 1.6506111263108283]  The first vector is the default startup position when the URControl is simulated. Note, the input and scaled vector are pretty similar, but not the same.  The second vector is based on the customer's data.  from math import \*  v\_init=[-0.0012, 3.1162, 0.03889]  v=[-0.06, 0.13, -0.04]  def length(v):  return sqrt(pow(v[0],2)+pow(v[1],2)+pow(v[2],2))  def norm(v):  l=length(v)  norm=[v[0]/l, v[1]/l, v[2]/l]  return norm  def \_polyscope(x,y,z):  if ( (abs(x) >= 0.001 and x < 0.0) or (abs(x) < 0.001 and abs(y) >= 0.001 and y < 0.0) or (abs(x) < 0.001 and abs(y) < 0.001 and z < 0.0) ):  scale = 1 - 2\*pi / length([x,y,z])  ret = [scale\*x, scale\*y, scale\*z]  print "PolyScope SCALED value: ", ret  return ret  else:  ret = [x,y,z]  print "PolyScope value: ", ret  return ret  def polyscope(v):  return \_polyscope(v[0], v[1], v[2])  polyscope(v\_init)  polyscope(v) |