ACTS Bevelled Cylinder Implementation Update

Sakib Rahman, University of Manitoba
18 August 2022

Boundary Condition Update For Bevelled Cylinders

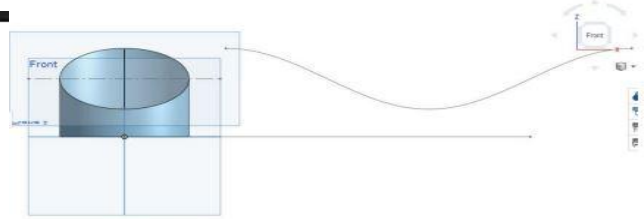
Checking validity of changes

Need two things:

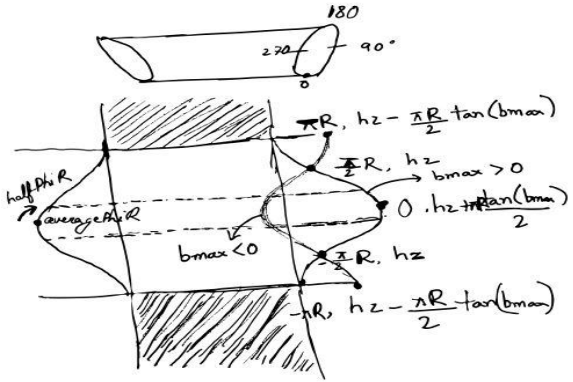
1) Boundary checks work for ACTS bevelled surface

2) Conversion from DD4hep/TGeo cut tubes to ACTS bevelled surface works

*Create a standalone DD4hep example based on ODD for quick check



Updated plugin to force n_y=0 for TGeoCutTube. Then it can be parametrized as



```
bool Acts::CylinderBounds::inside3D(const Vector3& position,
                                    const BoundaryCheck& bcheck) const {
  // additional tolerance from the boundary check if configred
  bool checkAbsolute = bcheck.m_type == BoundaryCheck::Type::eAbsolute;

  // this fast check only applies to closed cylindrical bounds
  double addToleranceR =
      (checkAbsolute && m_closed) ? bcheck.m_tolerance[0] : 0.;
  double addToleranceZ = checkAbsolute ? bcheck.m_tolerance[1] : 0.;
  // check if the position compatible with the radius
  if ((s_onSurfaceTolerance + addToleranceR) <=
      std::abs(perp(position) - get(eR))) {
    return false;
  } else if (checkAbsolute && m_closed) {
    double bevelMinZ = get(eBevelMinZ);
    double bevelMaxZ = get(eBevelMaxZ);

    double addedMinZ =
        bevelMinZ != 0. ? position.y() * std::sin(bevelMinZ) : 0.;
    double addedMaxZ =
        bevelMinZ != 0. ? position.y() * std::sin(bevelMaxZ) : 0.;

    return ((s_onSurfaceTolerance + addToleranceZ + get(eHalfLengthZ) +
             addedMinZ) >= position.z()) &&
           ((s_onSurfaceTolerance + addToleranceZ + get(eHalfLengthZ) +
             addedMaxZ) <= position.z());
  }
  // detailed, but slower check
  Vector2 lpos(detail::radian_sym(phi(position) - get(eAveragePhi)),
               position.z());
  return bcheck.transformed(jacobian())
      .isInside(lpos, Vector2(-get(eHalfPhiSector), -get(eHalfLengthZ)),
                Vector2(get(eHalfPhiSector), get(eHalfLengthZ)));
}
```

```
bool Acts::CylinderBounds::inside3D(const Vector3& position,
                                    const BoundaryCheck& bcheck) const {
  // additional tolerance from the boundary check if configred
  bool checkAbsolute = bcheck.m_type == BoundaryCheck::Type::eAbsolute;

  // fast check
  double addToleranceR = checkAbsolute ? bcheck.m_tolerance[0] : 0.;
  double addToleranceZ = checkAbsolute ? bcheck.m_tolerance[1] : 0.;
  // check if the position compatible with the radius
  if ((s_onSurfaceTolerance + addToleranceR) <= std::abs(perp(position) - get(eR))) {
    return false;
  } else if (get(eBevelMinZ)==0 && get(eBevelMaxZ)==0 &&
             ((s_onSurfaceTolerance + addToleranceZ) <= (std::abs(position.z()) - get(eHalfLengthZ)))) {
    // fast check on z position for nominal cylinder
    return false;
  } else {
    // detailed but slow check
    return inside({get(eR)*phi(position), position.z()}, bcheck);
  }

}
```

3 dimensions but only two tolerance variables in bcheck. Use m_tolerance[0] for both R and RPhi?

Call the 2D inside function to avoid redundancy

Updating Unit Tests

Unresolved issue:

1) The bevel min test is failing now. Could be a result of changed reference frame to align with central axis. Or could be a bug in new implementation.
2) Add cases for partial coverage in phi in the bevelled case for generality.

```
Running 5 test cases...
/__w/acts/acts/Tests/UnitTests/Core/Surfaces/CylinderBoundsTests.cpp(126): error: in "Surfaces/CylinderBoundsProperties": check cylinderBoundsBeveledObject.inside( withinBevelMin, trueBoundaryCheckWithLessTolerance) has failed

*** 1 failure is detected in the test module "ActsUnitTestCylinderBounds"
```

```
BOOST_AUTO_TEST_CASE(CylinderBoundsProperties) {
  // CylinderBounds object of radius 0.5 and halfz 20
  double nominalRadius{0.5};
  double nominalHalfLength{20.};
  double halfphi(M_PI / 4.0);
  double averagePhi(0.0);
  double bevelMinZ(M_PI / 4);
  double bevelMaxZ(M_PI / 6);
  CylinderBounds cylinderBoundsObject(nominalRadius, nominalHalfLength);
  CylinderBounds cylinderBoundsSegment(nominalRadius, nominalHalfLength,
                                       halfphi, averagePhi);
  CylinderBounds cylinderBoundsBeveledObject(nominalRadius, nominalHalfLength,
                                             M_PI, 0., bevelMinZ, bevelMaxZ);

  /// test for type()
  BOOST_CHECK_EQUAL(cylinderBoundsObject.type(), SurfaceBounds::eCylinder);

  /// test for inside(), 2D coords are r or phi ,z? : needs clarification
  const Vector2 origin{0., 0.};
  const Vector2 atPiBy2{M_PI / 2., 0.0};
  const Vector2 atPi{M_PI, 0.0};
  const Vector2 beyondEnd{0, 30.0};
  const Vector2 unitZ{0.0, 1.0};
  const Vector2 unitPhi{1.0, 0.0};
  const Vector2 withinBevelMin{0.5, -20.012};
  const Vector2 outsideBevelMin{0.5, -40.};
  const BoundaryCheck trueBoundaryCheckWithTolerance(true, true, 0.1, 0.1);
  const BoundaryCheck trueBoundaryCheckWithLessTolerance(true, true, 0.01,
                                                         0.01);
  BOOST_CHECK(
      cylinderBoundsObject.inside(atPiBy2, trueBoundaryCheckWithTolerance));
  BOOST_CHECK(
      !cylinderBoundsSegment.inside(unitPhi, trueBoundaryCheckWithTolerance));
  BOOST_CHECK(
      cylinderBoundsObject.inside(origin, trueBoundaryCheckWithTolerance));

  BOOST_CHECK(!cylinderBoundsObject.inside(withinBevelMin,
                                           trueBoundaryCheckWithLessTolerance));
  BOOST_CHECK(cylinderBoundsBeveledObject.inside(
      withinBevelMin, trueBoundaryCheckWithLessTolerance));
  BOOST_CHECK(!cylinderBoundsBeveledObject.inside(
      outsideBevelMin, trueBoundaryCheckWithLessTolerance));
```