Update on track seeding in EICRecon

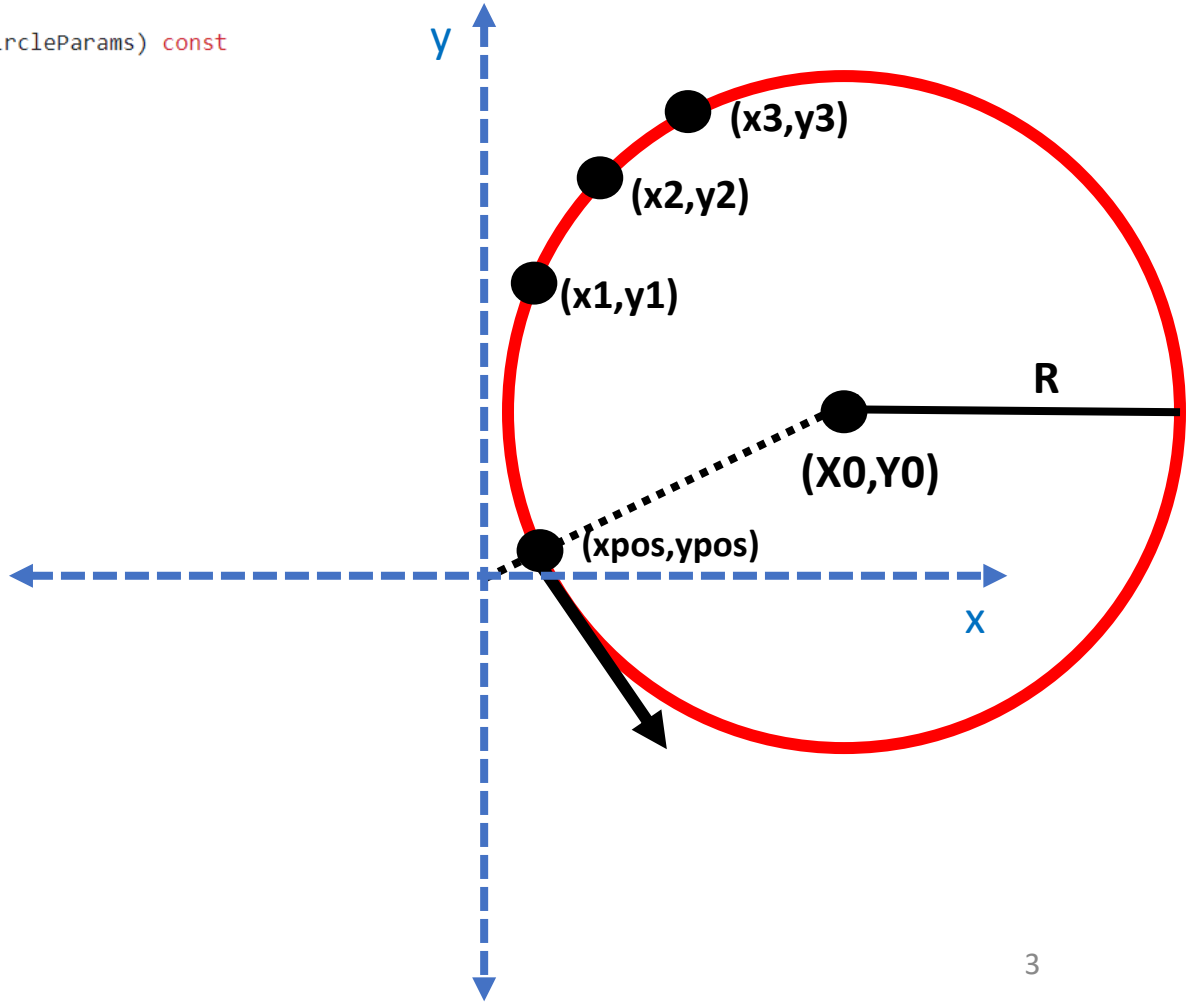
Barak Schmookler

Current status

- We want to have the real seeding + CKF included by default in our main EICRecon branch. We have some pull requests to do this: 711 (merged), 804 (under review).
- One reason for the updated pull request is that we observed some seeds being returned where the seed fit parameters are set to *NaN*. And we wanted to resolve this before merging the code into the main branch.
- There are two types of seeds which cause issues, and I'll give details on the following slides. I found the fist type occurred in about 30/10,000 single muon events. The second type occurred in about 2 / 10,000 single muon events.

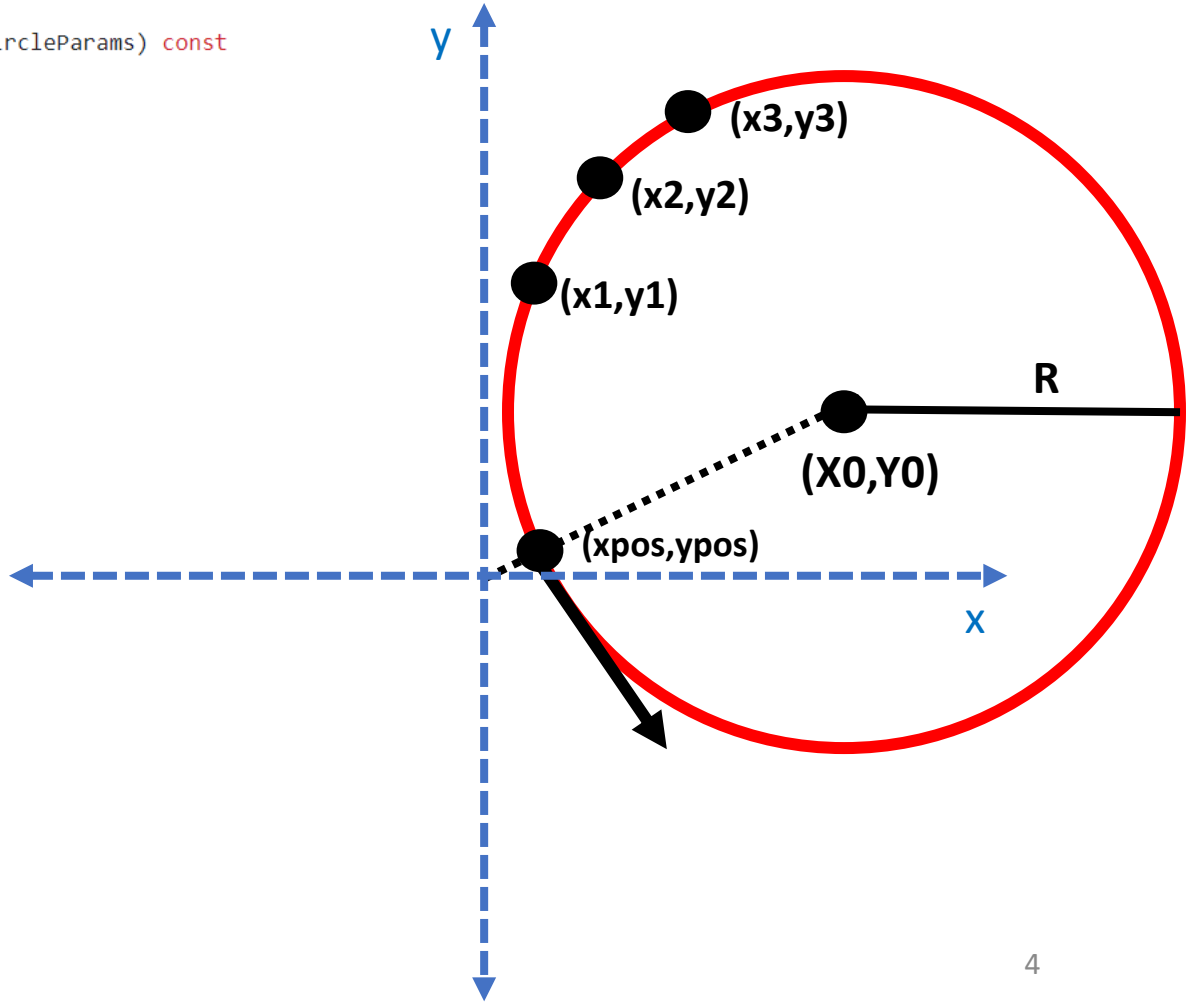
Both issues occur in the *findRoot* function

```
134 std::pair<float, float> eicrecon::TrackSeeding::findRoot(std::tuple<float,float,float>& circleParams) const
135 {
136   const float R = std::get<0>(circleParams);
137   const float X0 = std::get<1>(circleParams);
138   const float Y0 = std::get<2>(circleParams);
139   const double miny = (std::sqrt(square(X0) * square(R) * square(Y0) + square(R)
140                       * pow(Y0,4)) + square(X0) * Y0 + pow(Y0, 3))
141     / (square(X0) + square(Y0));
142
143   const double miny2 = (-std::sqrt(square(X0) * square(R) * square(Y0) + square(R)
144                       * pow(Y0,4)) + square(X0) * Y0 + pow(Y0, 3))
145     / (square(X0) + square(Y0));
146
147   const double minx = std::sqrt(square(R) - square(miny - Y0)) + X0;
148   const double minx2 = -std::sqrt(square(R) - square(miny2 - Y0)) + X0;
149
150   /// Figure out which of the two roots is actually closer to the origin
151   const float x = ( std::abs(minx) < std::abs(minx2)) ? minx:minx2;
152   const float y = ( std::abs(miny) < std::abs(miny2)) ? miny:miny2;
153   return std::make_pair(x,y);
154 }
```



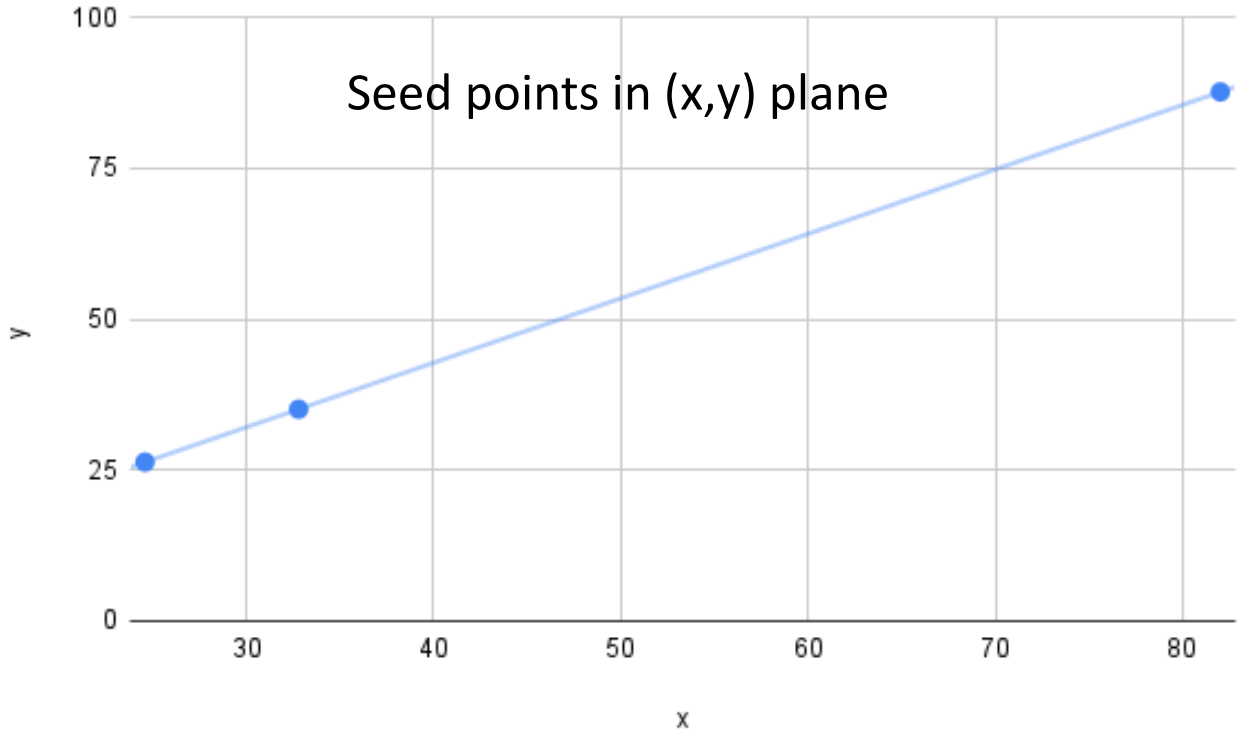
Issue 1 – circle parameters raised to 6th power

```
134  std::pair<float, float> eicrecon::TrackSeeding::findRoot(std::tuple<float,float,float>& circleParams) const
135  {
136    const float R = std::get<0>(circleParams);
137    const float X0 = std::get<1>(circleParams);
138    const float Y0 = std::get<2>(circleParams);
139    const double miny = (std::sqrt(square(X0) * square(R) * square(Y0) + square(R)
140                        * pow(Y0,4)) + square(X0) * Y0 + pow(Y0, 3))
141      / (square(X0) + square(Y0));
142
143    const double miny2 = (-std::sqrt(square(X0) * square(R) * square(Y0) + square(R)
144                        * pow(Y0,4)) + square(X0) * Y0 + pow(Y0, 3))
145      / (square(X0) + square(Y0));
146
147    const double minx = std::sqrt(square(R) - square(miny - Y0)) + X0;
148    const double minx2 = -std::sqrt(square(R) - square(miny2 - Y0)) + X0;
149
150    /// Figure out which of the two roots is actually closer to the origin
151    const float x = ( std::abs(minx) < std::abs(minx2)) ? minx:minx2;
152    const float y = ( std::abs(miny) < std::abs(miny2)) ? miny:miny2;
153    return std::make_pair(x,y);
154  }
```



Issue 1 – circle parameters raised to 6th power

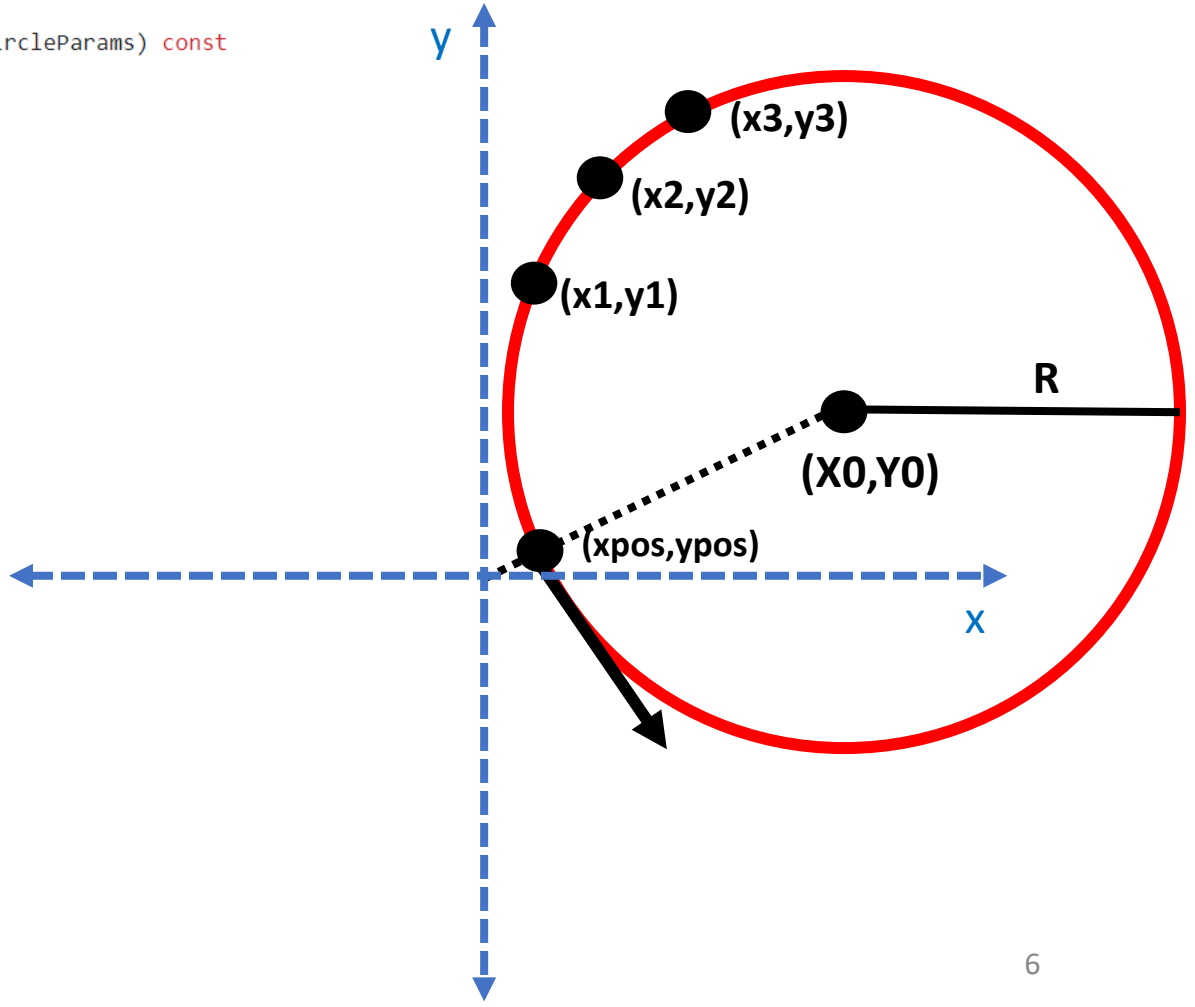
Example of a problematic seed – circle parameters are very large, cause float overflow in findRoot function



Circle parameters (R,X,Y); 1743956700 1272490600 -1192540400

Issue 2 – negative argument for square root

```
134  std::pair<float, float> eicrecon::TrackSeeding::findRoot(std::tuple<float,float,float>& circleParams) const
135  {
136    const float R = std::get<0>(circleParams);
137    const float X0 = std::get<1>(circleParams);
138    const float Y0 = std::get<2>(circleParams);
139    const double miny = (std::sqrt(square(X0) * square(R) * square(Y0) + square(R)
140                        * pow(Y0,4)) + square(X0) * Y0 + pow(Y0, 3))
141      / (square(X0) + square(Y0));
142
143    const double miny2 = (-std::sqrt(square(X0) * square(R) * square(Y0) + square(R)
144                        * pow(Y0,4)) + square(X0) * Y0 + pow(Y0, 3))
145      / (square(X0) + square(Y0));
146
147    const double minx = std::sqrt(square(R) - square(miny - Y0)) + X0;
148    const double minx2 = -std::sqrt(square(R) - square(miny2 - Y0)) + X0;
149
150    /// Figure out which of the two roots is actually closer to the origin
151    const float x = ( std::abs(minx) < std::abs(minx2)) ? minx:minx2;
152    const float y = ( std::abs(miny) < std::abs(miny2)) ? miny:miny2;
153    return std::make_pair(x,y);
154  }
```



Issue 2 – negative argument for square root

Example of a problematic seed – circle radius is reasonable, but center is close to y axis. This can cause precision issues when subtracting two somewhat large and almost equal numbers

```
Circle Parameters: R = 4557.015625, X0 = 0.979015, Y0 = -4557.020996
Output from findRoot: miny = -0.005411, minx = 1.958031, miny2 = -9114.037159, minx2 = -nan
Point-of-closest approach returned in Global coordinates (x,y,z): -nan -0.005411 0.000969
```

Proposed update to findRoot function

$$y = \frac{\pm\sqrt{X_0^2R^2Y_0^2 + R^2Y_0^4} + X_0^2Y_0 + Y_0^3}{X_0^2 + Y_0^2} = \frac{\pm RY_0\sqrt{X_0^2 + Y_0^2} + Y_0(X_0^2 + Y_0^2)}{X_0^2 + Y_0^2} = \pm Y_0\frac{R}{R_0} + Y_0 = Y_0(1 \pm \frac{R}{R_0})$$

By symmetry and since the points need to lie on a line from the origin to the circle center, we have the two solutions:

$$[x, y] = \left[X0\left(1 - \frac{R}{R0}\right), Y0\left(1 - \frac{R}{R0}\right)\right]$$

$$[x, y] = \left[X0\left(1 + \frac{R}{R0}\right), Y0\left(1 + \frac{R}{R0}\right)\right]$$

Proposed update to findRoot function

$$y = \frac{\pm\sqrt{X_0^2R^2Y_0^2 + R^2Y_0^4} + X_0^2Y_0 + Y_0^3}{X_0^2 + Y_0^2} = \frac{\pm RY_0\sqrt{X_0^2 + Y_0^2} + Y_0(X_0^2 + Y_0^2)}{X_0^2 + Y_0^2} = \pm Y_0\frac{R}{R_0} + Y_0 = Y_0(1 \pm \frac{R}{R_0})$$

By symmetry and since the points need to lie on a line from the origin to the circle center, we have the two solutions:

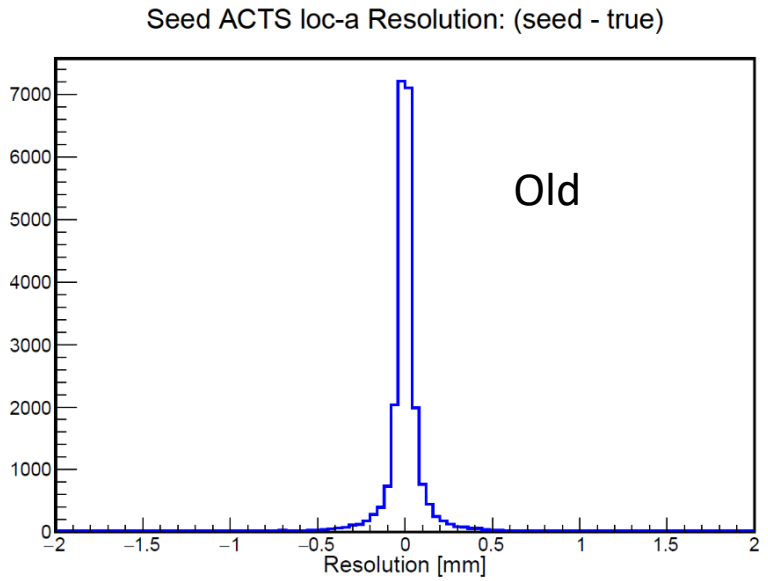
$$[x, y] = \left[X0\left(1 - \frac{R}{R0}\right), Y0\left(1 - \frac{R}{R0}\right)\right]$$

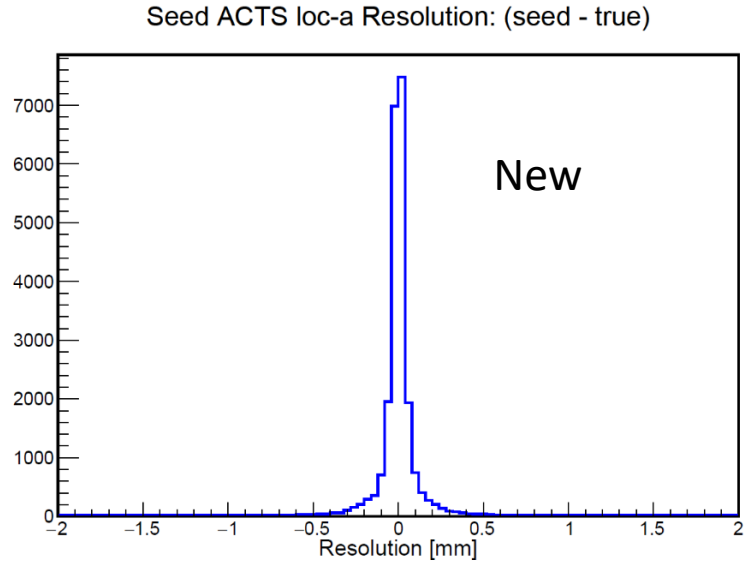
This pair will always be closer to the origin.

$$[x, y] = \left[X0\left(1 + \frac{R}{R0}\right), Y0\left(1 + \frac{R}{R0}\right)\right]$$

Results

- Testing this change on the same 10k event single-particle file, all the previously problematic seeds now return a set of finite parameters.
- The results for the seed position reconstruction are otherwise unaffected.





Summary

- We have a pull request which proposes some changes to the track seeding findRoot function. This resolves uncommon seeds where that function fails.
- We also decided to add some code to guard against overflows and the case where the circle center is right at (0,0).
- In the future, we can improve the track seeding code by separating straight line from circular tracks and/or using some ACTS functions directly.
- But for now, maybe we want to just merge in the proposed working version and make improvements later?