HPS Coordinates in Software...

Matt Graham
SLAC

lcsim tracking conventions

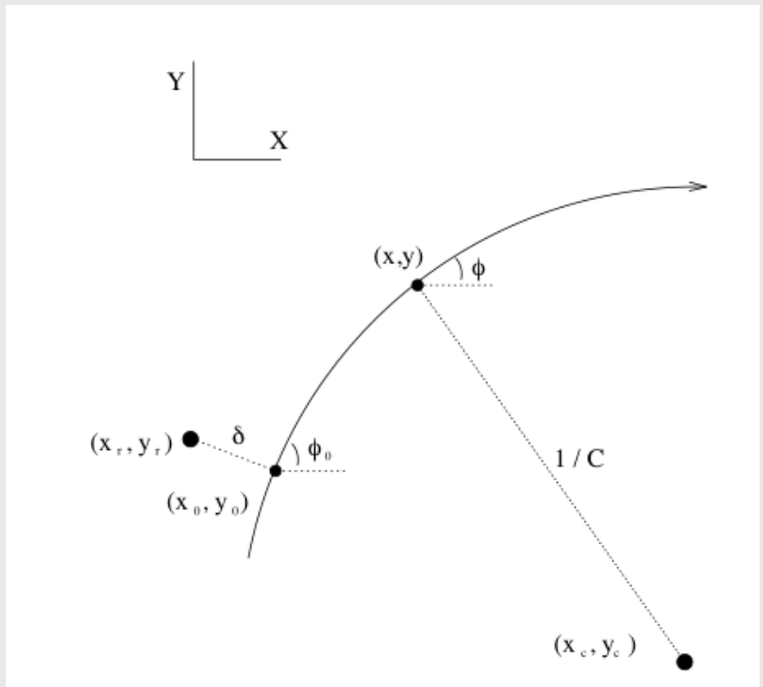
remember! In lcsim, the B-field is in the z-direction!
The beam is in x and the bend is y...

- tracks use a "perigee" parameterization similar to what was introduced by Billoir & Qian (NIM A311, 1992)

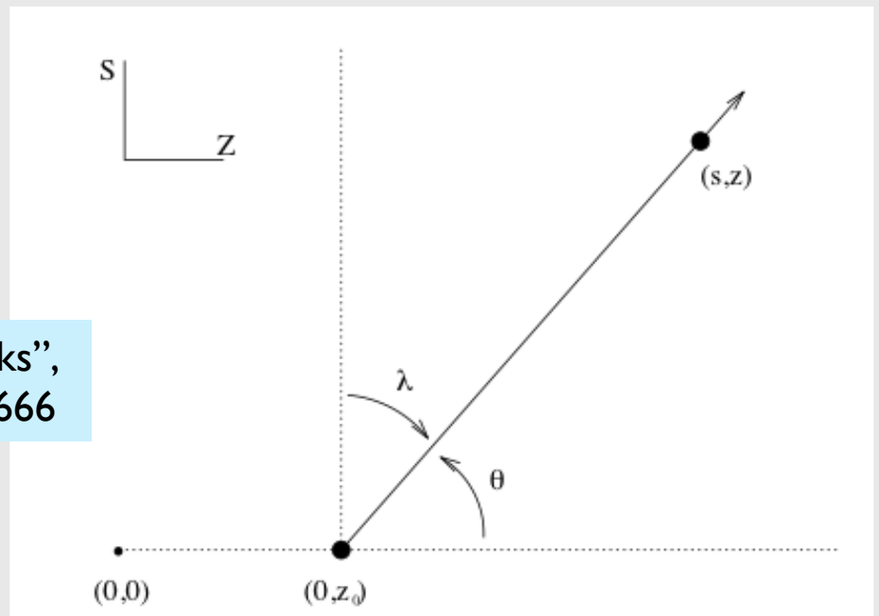
$(\varepsilon, z_0, \theta, \phi_0, \rho) \Rightarrow (\delta, z_0, \tan\lambda, \phi_0, \rho)$
$\varepsilon = -\delta;\ \theta = \pi/2 - \lambda$

} All quantities measured wrt point of closest approach to z-axis

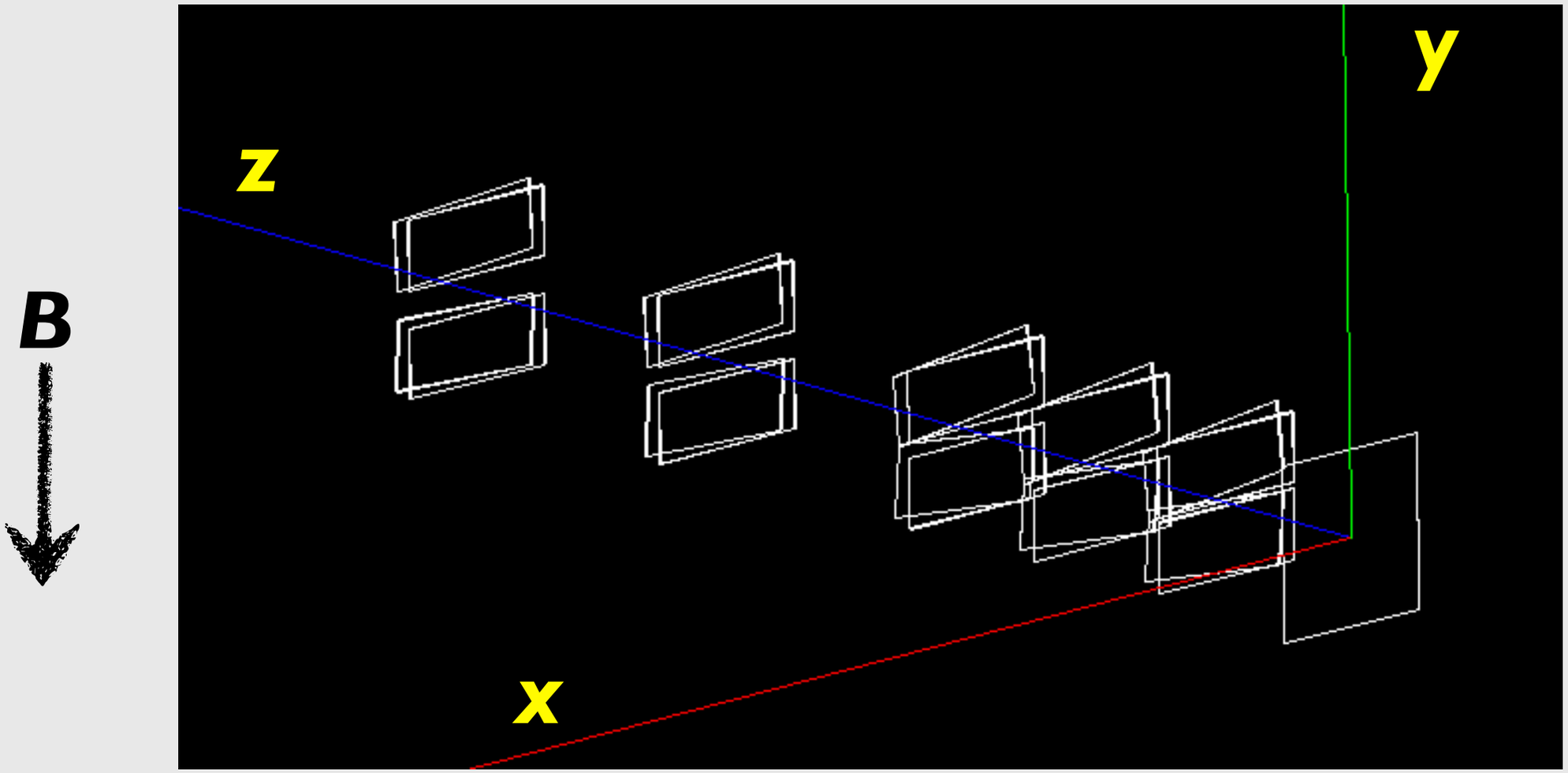
- this isn't the most natural coordinate system for us...better to have the beam in z; look into transforming (transparent for enduser)



Figures from "Helicoidal Tracks", J. Alcaraz, L3 Internal Note 1666



JLab Coordinates



Slic Frame to Lab Frame:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix}' = \begin{pmatrix} y \\ z \\ x \end{pmatrix}$$

Changes to the geometry

- New detector type: SiTrackerSpectrometer
  - same as SiTrackerFixedTarget2 but it builds the modules along the z-axis
  - this is in GeomConverter now...

```
<detector id="1" name="Tracker" type="SiTrackerSpectrometer" readout="TrackerHits" combineHits="true" reflect="true"
    <module name="Module1">
       <trd x1="modWidth/2" x2="modWidth/2" z="modLength/2" />
       <module_component    thickness="0.032*cm" material = "Silicon" sensitive="true"/><!-- X0=0.32% -->
       <module_component    thickness="0.02*cm" material = "Carbon" sensitive="false"/> <!-- X0=0.1% -->
     </module>

  <layer id="1">
    <quadrant xStart="0" nx="1" xStep="modLength" yStart="(ygap1+modWidth)/2" ny="1" yStep="modWidth" phi0="0.0" z
  </layer>
  <layer id="2">
    <quadrant xStart="0" nx="1" xStep="modLength" yStart="(ygap1+modWidth)/2" ny="1" yStep="modWidth" phi0="SA" z=
  </layer>

  <layer id="3">
    <quadrant xStart="0" nx="1" xStep="modLength" yStart="(ygap2+modWidth)/2" ny="1" yStep="modWidth" phi0="0.0" z
  </layer>
  <layer id="4">
    <quadrant xStart="0" nx="1" xStep="modLength" yStart="(ygap2+modWidth)/2" ny="1" yStep="modWidth" phi0="SA" z=
  </layer>
```

- New field type: BoxDipole
  - just a constant B-field (bx,by,bz) in a box
  - in GeomConverter and lcdd
- Haven't looked at ECal yet...

```
  <fields>
  <field type="BoxDipole" name="AnalyzingDipole"
         x="0*cm"
         y="0*cm"
         z="50*cm"
         dx="10*cm"
         dy="10*cm"
         dz="51*cm"
         bx="0.0"
         by="-0.5"
         bz="0.0">
  </field>
 </fields>
```

Changes to Reconstruction

- Digitization, clustering...nothing needed...
- Stereo hit making...make the normal (lab frame) HelicalTrackHits and also make set of hits that are rotated to the lcsim tracking frame
- Tracking is then performed using these rotated hits...works as exactly as before
  - Did have to modify MS code so that it knew detector was rotated
- These tracks get sent to vertexing as before (but vertex xyz are in lcsim frame! need to fix this...probably with a wrapper.)

org.lcsim.hps.event.HPSTransformation

```java
package org.lcsim.hps.event;

import hep.physics.matrix.SymmetricMatrix;
import hep.physics.vec.BasicHep3Matrix;
import hep.physics.vec.Hep3Vector;
import org.lcsim.detector.Rotation3D;
import org.lcsim.detector.Transform3D;

/**
 *  Class that contains the transformations between the JLAB and lcsim tracking coordinate systems
 * @author mgraham
 * created 6/27/2011
 */
public class HPSTransformations {

    private Transform3D _detToTrk;

    public HPSTransformations() {
        BasicHep3Matrix tmp = new BasicHep3Matrix();
        tmp.setElement(0, 2, 1);
        tmp.setElement(1, 0, 1);
        tmp.setElement(2, 1, 1);
//        _detToTrk.setRotationMatrix(tmp);
        _detToTrk = new Transform3D(new Rotation3D(tmp));
    }

    public Hep3Vector transformVectorToTracking(Hep3Vector vec) {
        return _detToTrk.transformed(vec);
    }

    public SymmetricMatrix transformCovarianceToTracking(SymmetricMatrix cov) {
        return _detToTrk.transformed(cov);
    }

    public Hep3Vector transformVectorToDetector(Hep3Vector vec) {
        return (_detToTrk.inverse()).transformed(vec);
    }

    public SymmetricMatrix transformCovarianceToDetector(SymmetricMatrix cov) {
        return (_detToTrk.inverse()).transformed(cov);
    }
    public Transform3D getTransform(){
        return _detToTrk;
    }
}
```

Simple class that defines the transformation between lcsim & jlab coordinates....probably want more later on (beam frame, ecal etc)

org.lcsim.hps.tracking.HPSTrack

```java
/**
 * Class HPSTrack:  extension of HelicalTrackFit to include HPS-specific variables
 * other useful things.
 * @author mgraham
 * created on 6/27/2011
 */
public class HPSTrack extends HelicalTrackFit {

    private BeamSpot _beam;
    //all of the variables defined below are in the jlab (detector) frame
    //this position & momentum are measured at the DOCA to the Y-axis,
    //which is where the tracking returns it's parameters by default
    private Hep3Vector _pDocaY;
    private Hep3Vector _posDocaY;
    //the position & momentum of the track at the intersection of the target (z=0)
    private Hep3Vector _pTarget;
    private Hep3Vector _posTarget;
    //the position & momentum of the track at DOCA to the beam axis (z)
    private Hep3Vector _pDocaZ;
    private Hep3Vector _posDocaZ;
    private HPSTransformations _detToTrk;
    private double bField = 0.5;  //  make this set-able

    public HPSTrack(double[] pars, SymmetricMatrix cov, double[] chisq, int[] ndf,
            Map<HelicalTrackHit, Double> smap, Map<HelicalTrackHit, MultipleScatter> msmap,
            BeamSpot beam) {
        super(pars, cov, chisq, ndf, smap, msmap);
        _beam = beam;
        _detToTrk = new HPSTransformations();
        calculateParametersAtTarget();
        calculateParametersAtDocaY();
        calculateParametersAtDocaZ();
    }

    public HPSTrack(HelicalTrackFit htf, BeamSpot beam) {
        super(htf.parameters(), htf.covariance(), htf.chisq(), htf.ndf(), htf.PathMap(), htf.ScatterMap());
        _beam = beam;
        _detToTrk = new HPSTransformations();
        calculateParametersAtTarget();
        calculateParametersAtDocaY();
        calculateParametersAtDocaZ();
    }

    private void calculateParametersAtTarget() {
        double pTot = this.p(bField);
        //currently, PathToXPlane only returns a single path (no loopers!)
        List<Double> paths = HelixUtils.PathToXPlane(this, 0.0, 100.0, 1);
        Hep3Vector posTargetTrkSystem = HelixUtils.PointOnHelix(this, paths.get(0));
        Hep3Vector dirTargetTrkSystem = HelixUtils.Direction(this, paths.get(0));
        _posTarget = _detToTrk.transformVectorToDetector(posTargetTrkSystem);
        _pTarget = VecOp.mult(pTot, _detToTrk.transformVectorToDetector(dirTargetTrkSystem));
```

wrapper extending the track fit...

- calculates momenta and track trajectory in the lab frame
- plan is to use this instead of HelicalTrackFit objects so that the lcsim↔lab transform is invisible to users