Kalman Filter

4

FOOT

Matteo Franchini
on behalf of the
Bologna Working Group

Outline

- Kalman Filter method
- Kal-test, inspiration code
- “State of the Art”
- Next steps

Kalman Filter

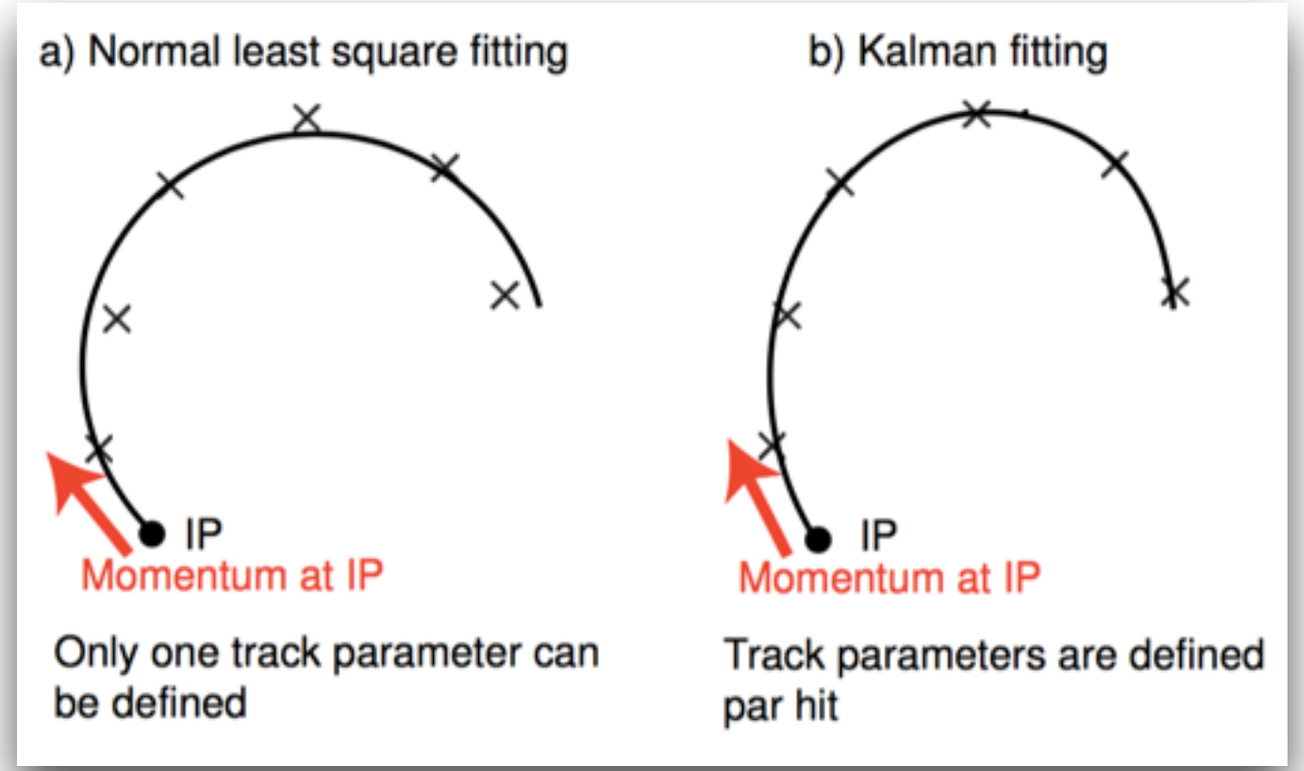
- R.E.Kalman proposed an iterative method to estimate the states of a dynamic system starting from a series of measurement points on N surfaces.

- Initially used to calculate the trajectory of ballistic missiles. Later introduced in particle physics (1984).

- Precise as a global χ^2 fitting;

- ***Fast***;

- Best ***track parameter*** found ***for each hit!!!***



"A new approach to linear filtering and prediction problems" Trans. ASME J. Basic Eng. 82 (1960), 35

Kalman in pills

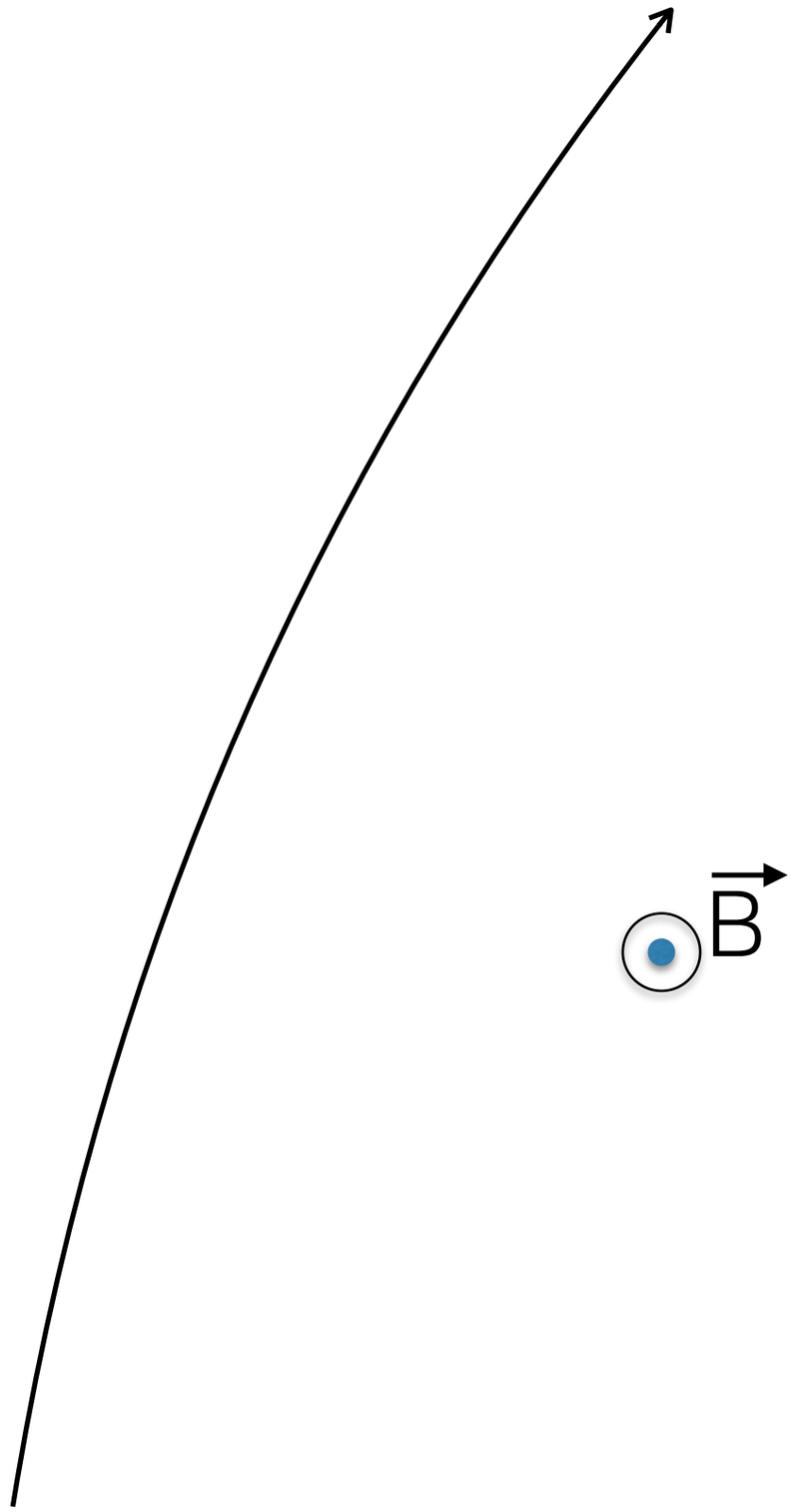
1. Take an **ideal particle** in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. Iterate 3 and 4 for the next leayers.



Kalman in pills

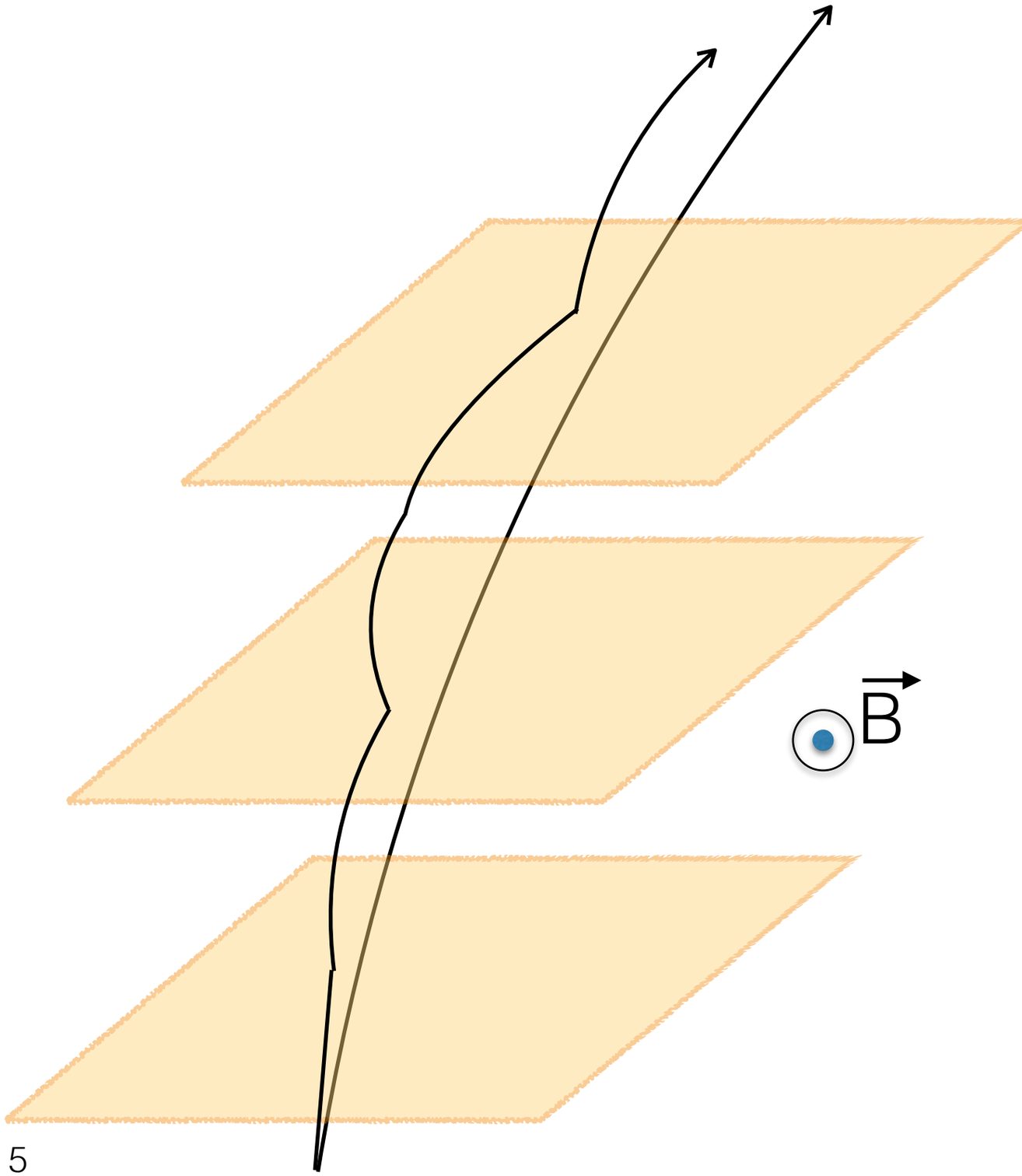
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to **M.S.** and **energy loss**.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. Iterate 3 and 4 for the next leayers.



Kalman in pills

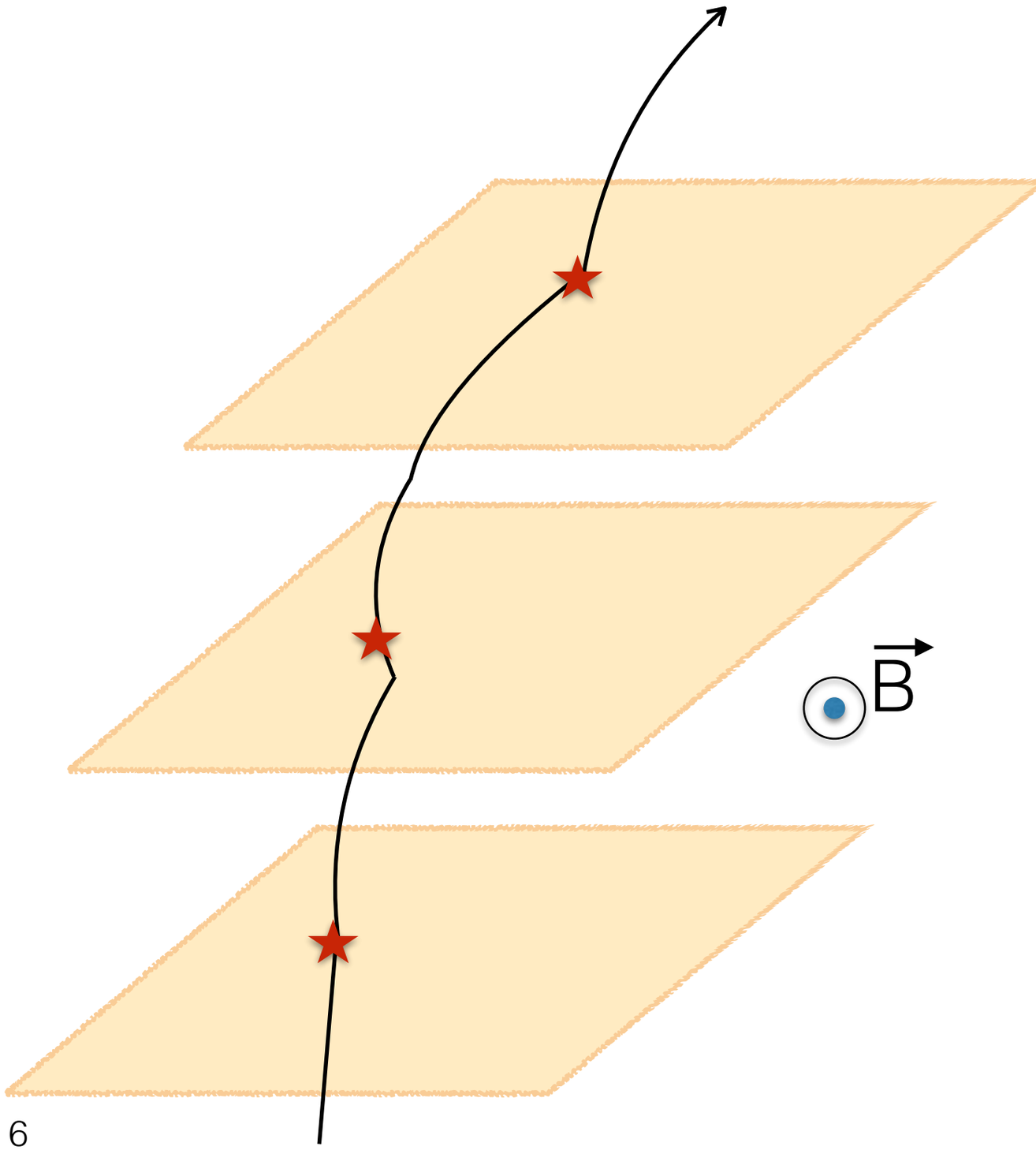
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some **measurement hits** on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. Iterate 3 and 4 for the next leayers.



Kalman in pills

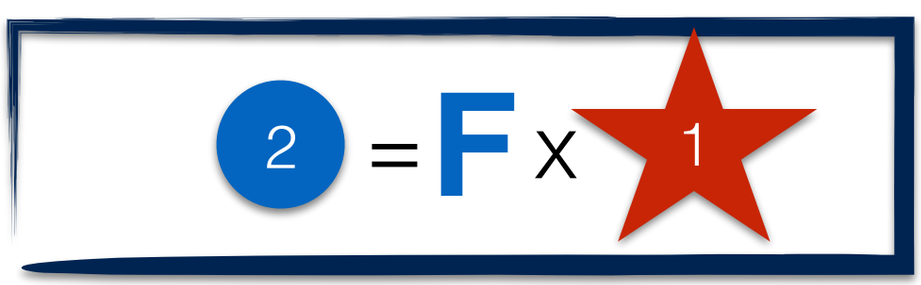
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

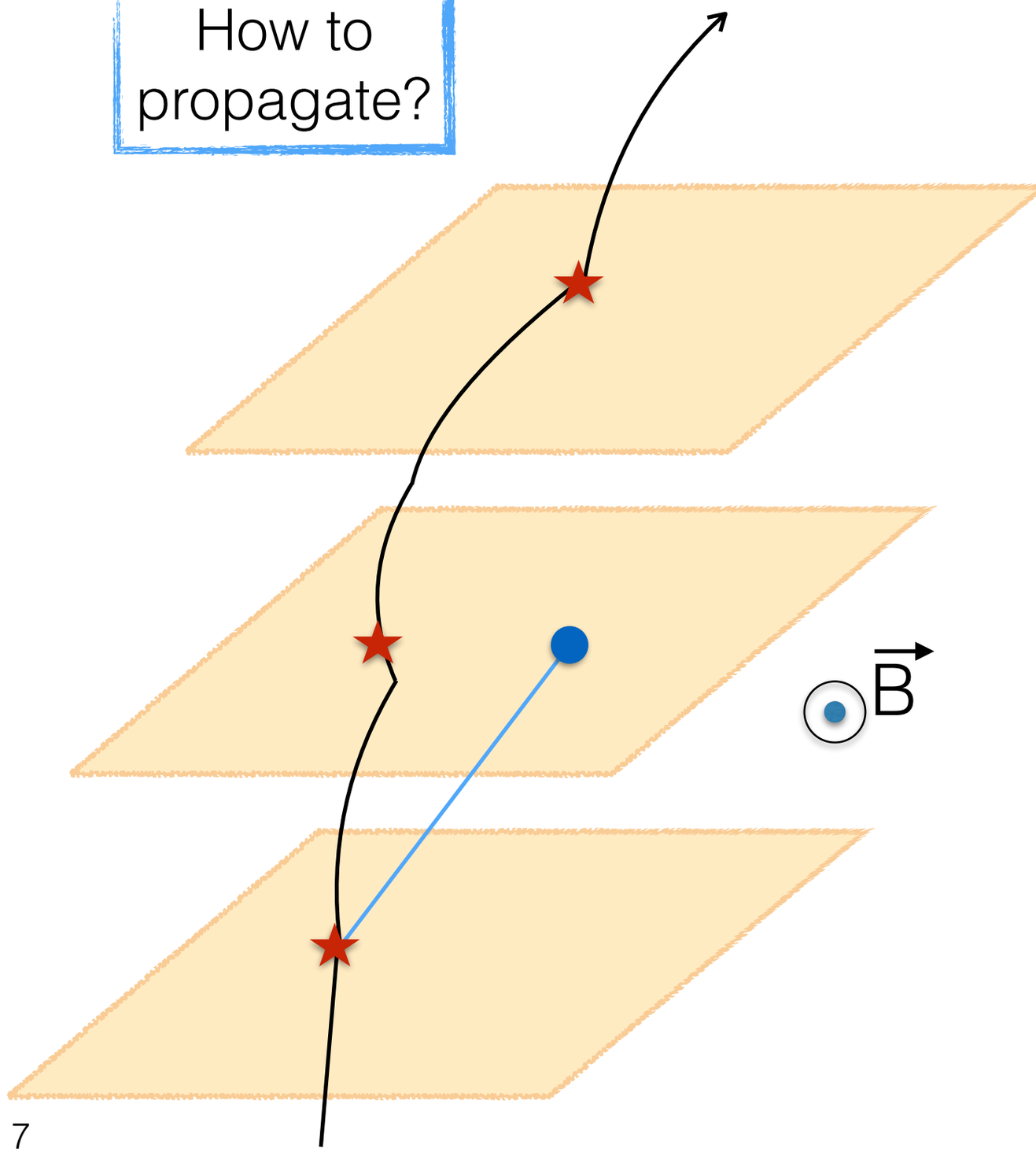
3. **Propagate** the first hit to the next layer. Propagator Matrix **F**.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. Iterate 3 and 4 for the next leayers.



How to propagate?



Kalman in pills

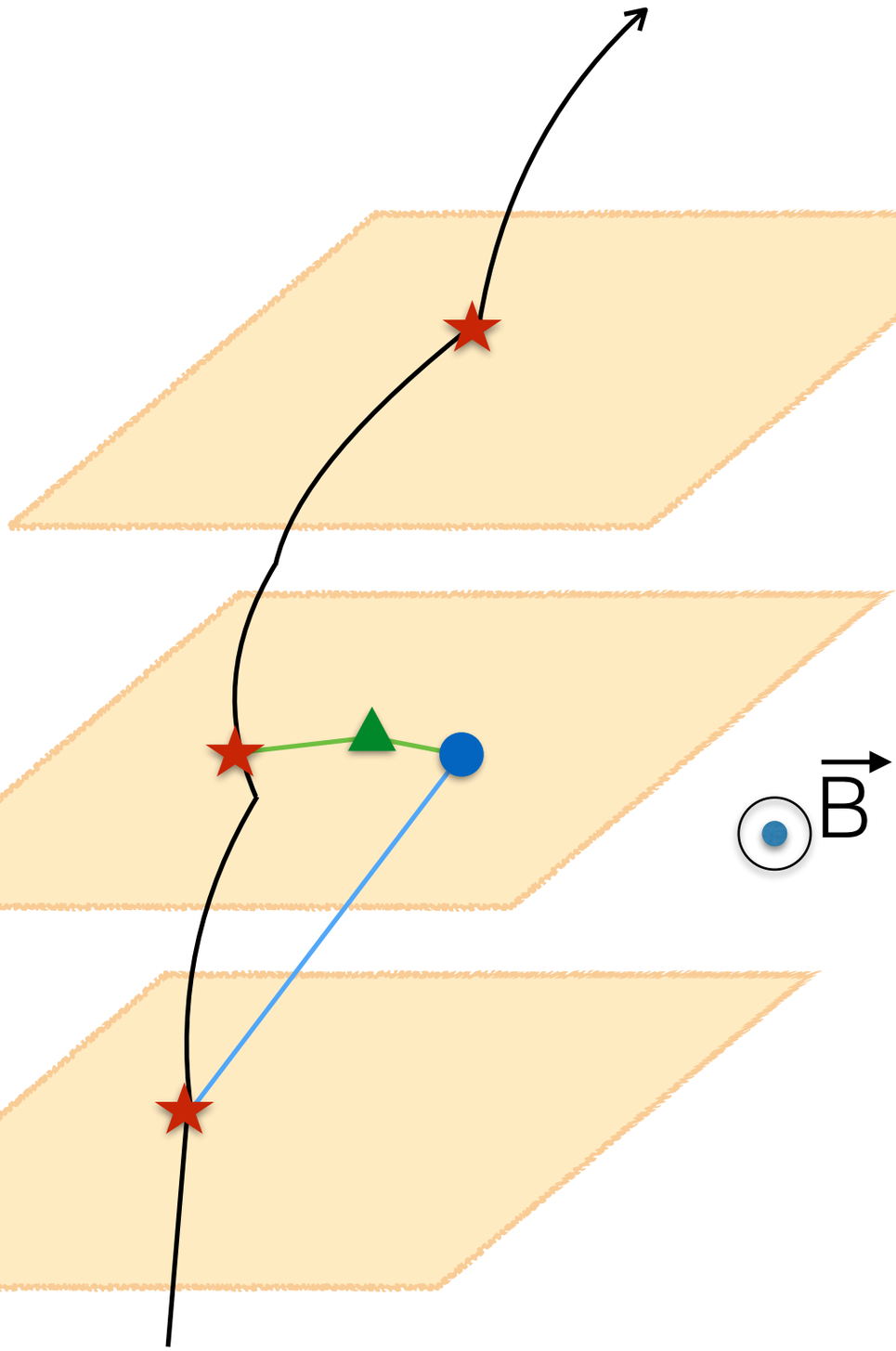
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a **Chi2** and a Projection Matrix **H**.

5. Iterate 3 and 4 for the next leayers.



$$\chi^2_{\triangle} = \chi^2\left(H \times \star_2, H \times \bullet_2\right)$$

Kalman in pills

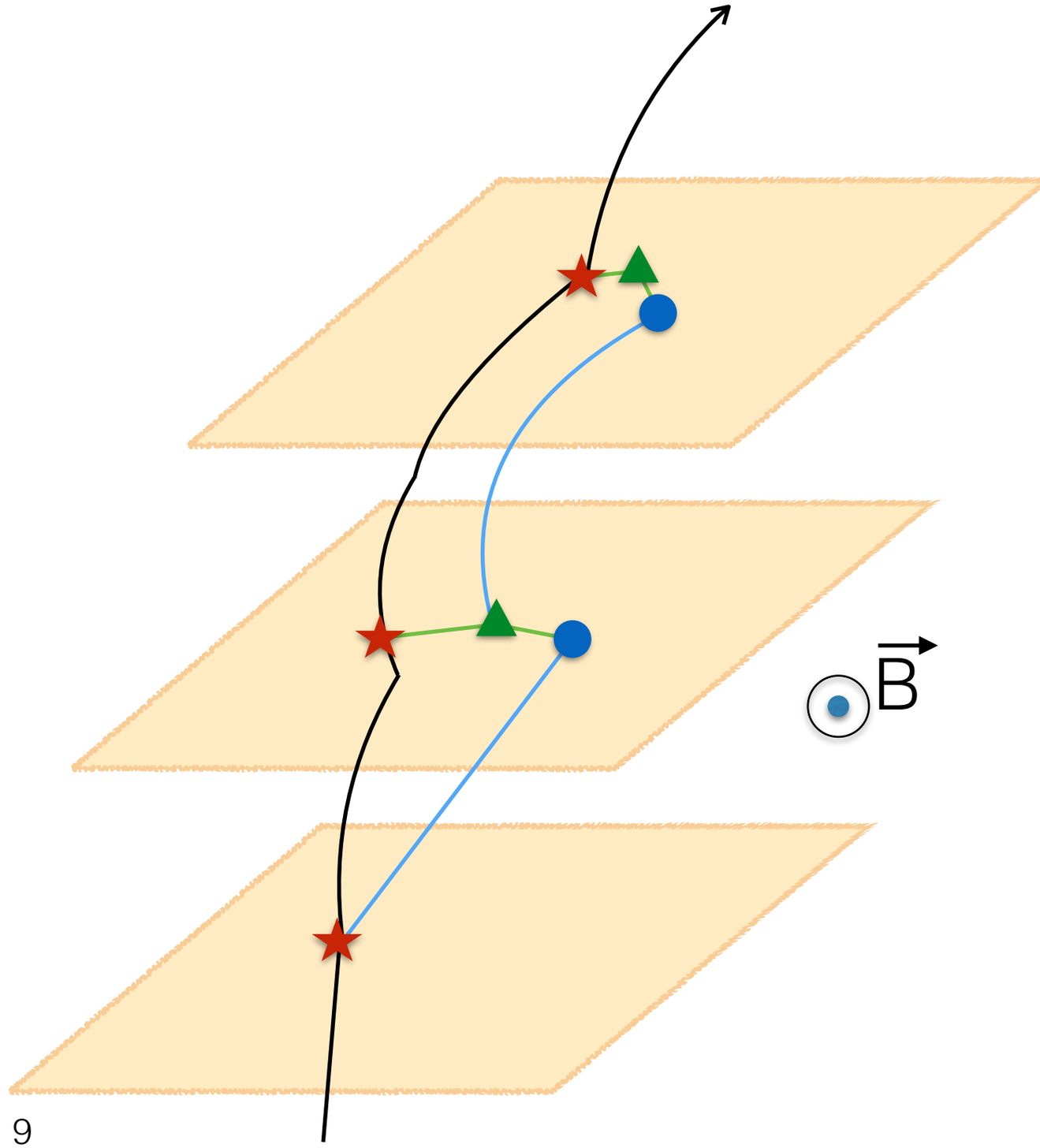
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. **Iterate** 3 and 4 for the next leayers.



Kalman in pills

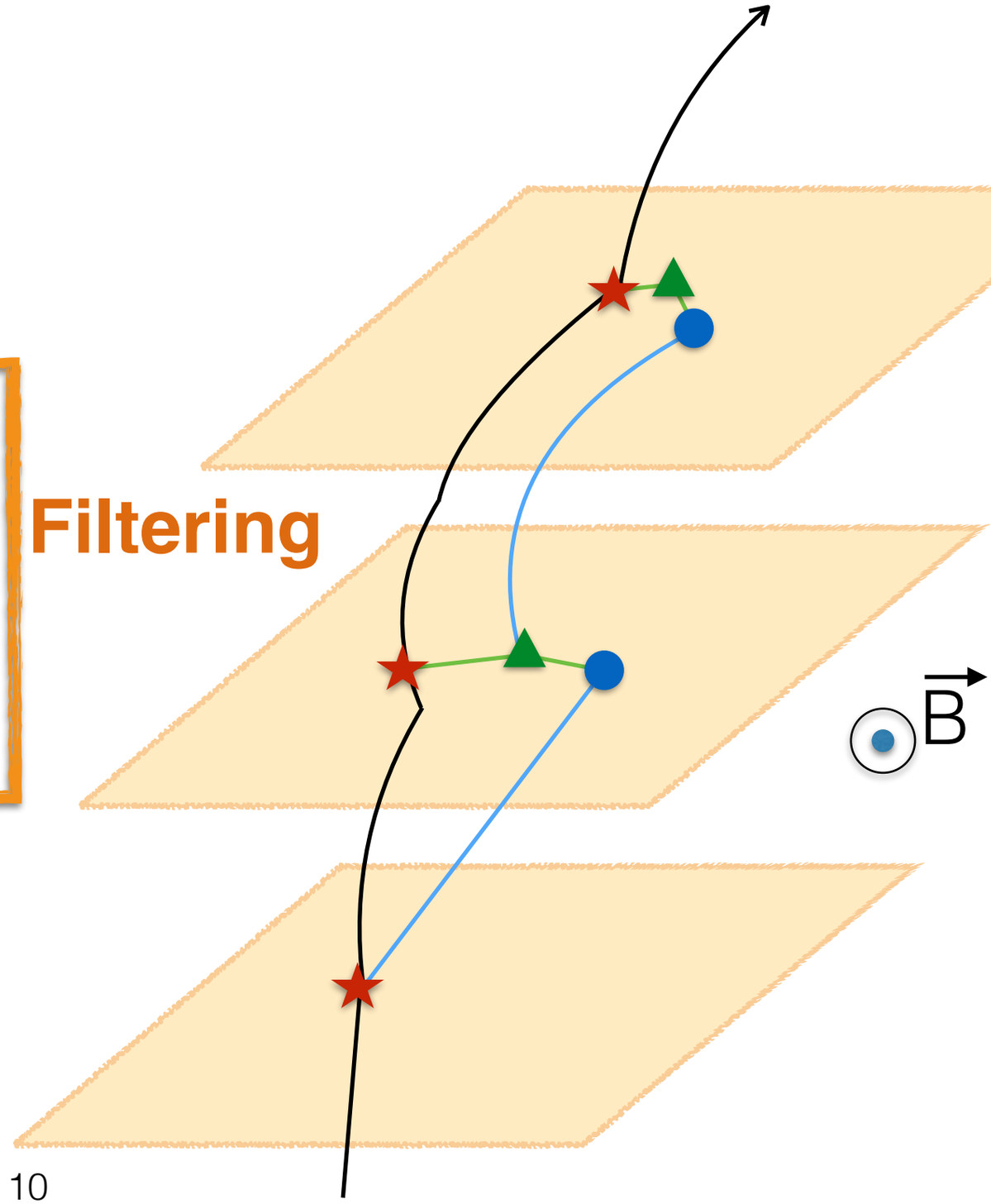
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. **Iterate** 3 and 4 for the next leayers.



Kalman in pills

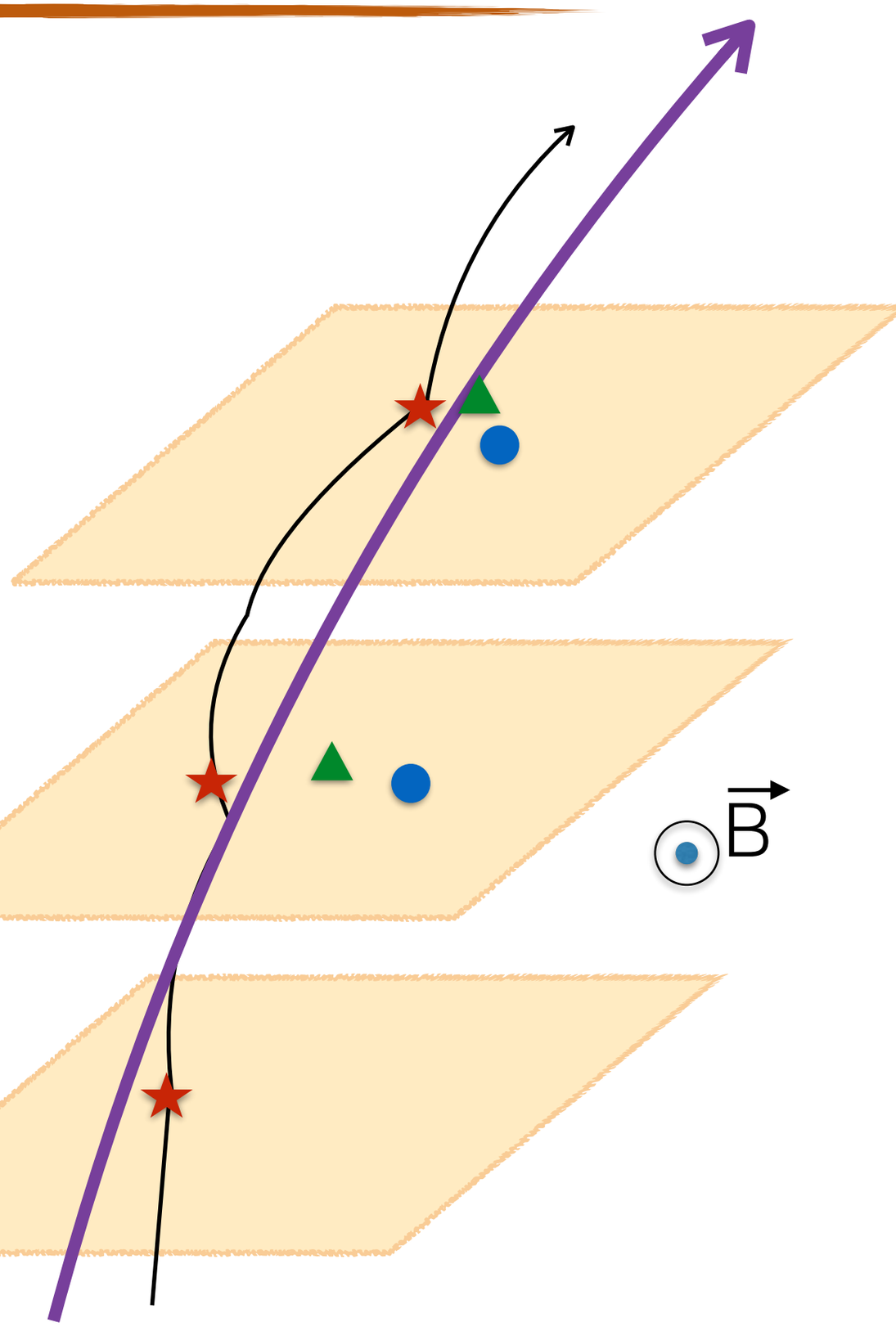
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. Iterate 3 and 4 for the next leayers.



Nota bene

- At each layer we do not evaluate a point in space, but a **curve (helix)** that best approximate the trajectory at ▲ point;
- The curve found has an associated uncertainty that decrease layer after layer…
- …so we redo the filtering backward! (Smoothing)

Kalman in pills

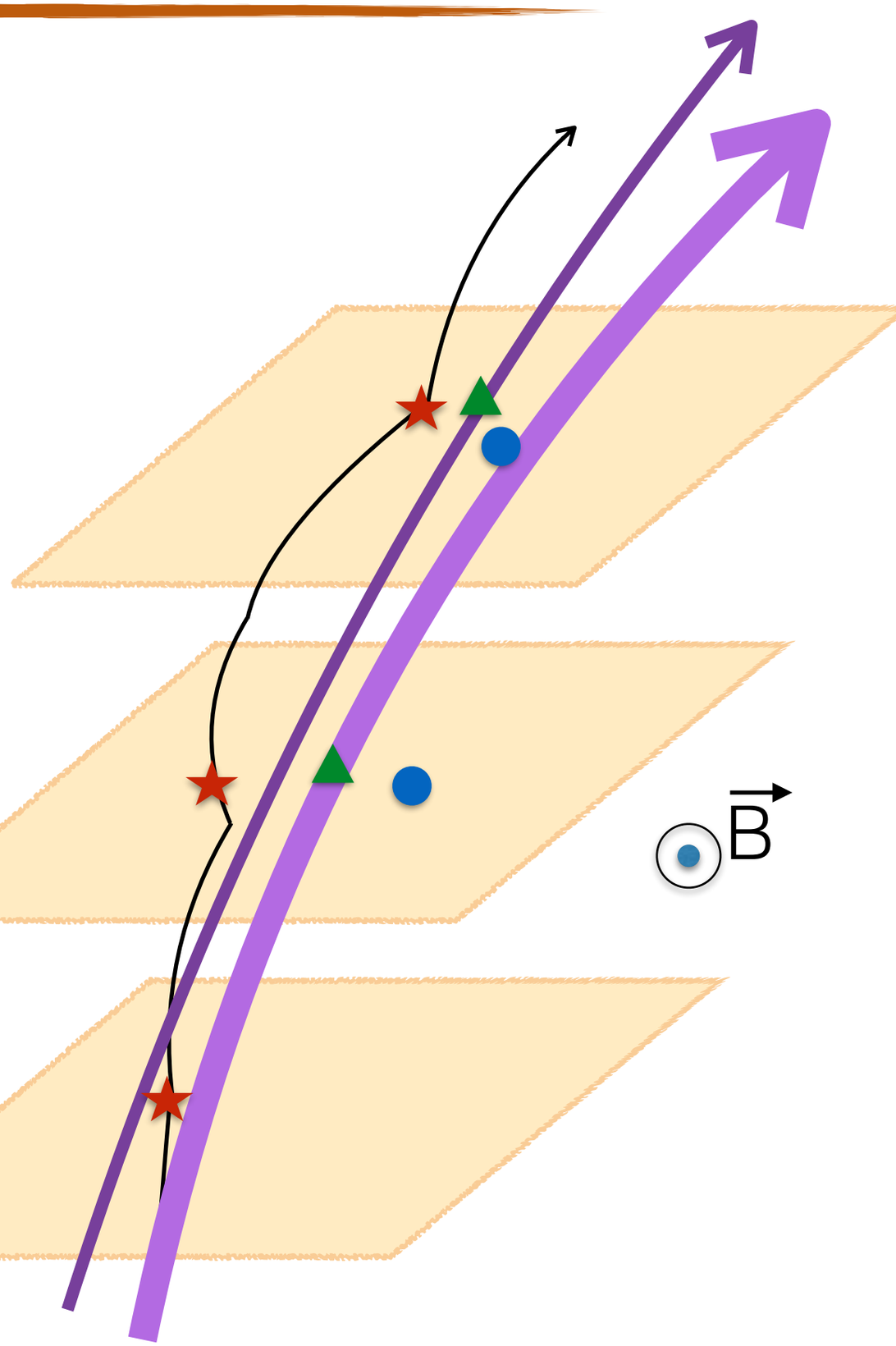
1. Take an ideal particle in vacuum. If we add air + detector layers, trajectory changes due to M.S. and energy loss.

2. We'll see some measurement hits on the detector layers (considering finite detector uncertainty).

3. Propagate the first hit to the next layer. Propagator Matrix F.

4. Find the best compromise between the propagated point and the closest hit on the 2nd layer. Use a Chi2 and a Projection Matrix H.

5. Iterate 3 and 4 for the next leayers.



Nota bene

- At each layer we do not evaluate a point in space, but a **curve (helix)** that best approximate the trajectory at ▲point;
- The curve found has an **associated uncertainty** that **decrease** layer after layer…
- …so we redo the filtering backward! (*Smoothing*)

Kal-test

- Kal-test is a standalone skeleton-code by Keisuke Fujii that performs Kalman Filtering in a general purpose approach. *Link*.

- All the ***mathematic*** and the ***basic structure*** is in place.

- ***To do:*** adapt and optimise to the specific case, alias FOOT.
  - Geometry
  - Magnetic Field —> track shape
  - I/O management.

Kal-test

Basic Structure

Common Library

Exec

- Object oriented, inherit for ROOT classes.
- Standalone but easy to be included in a general framework.

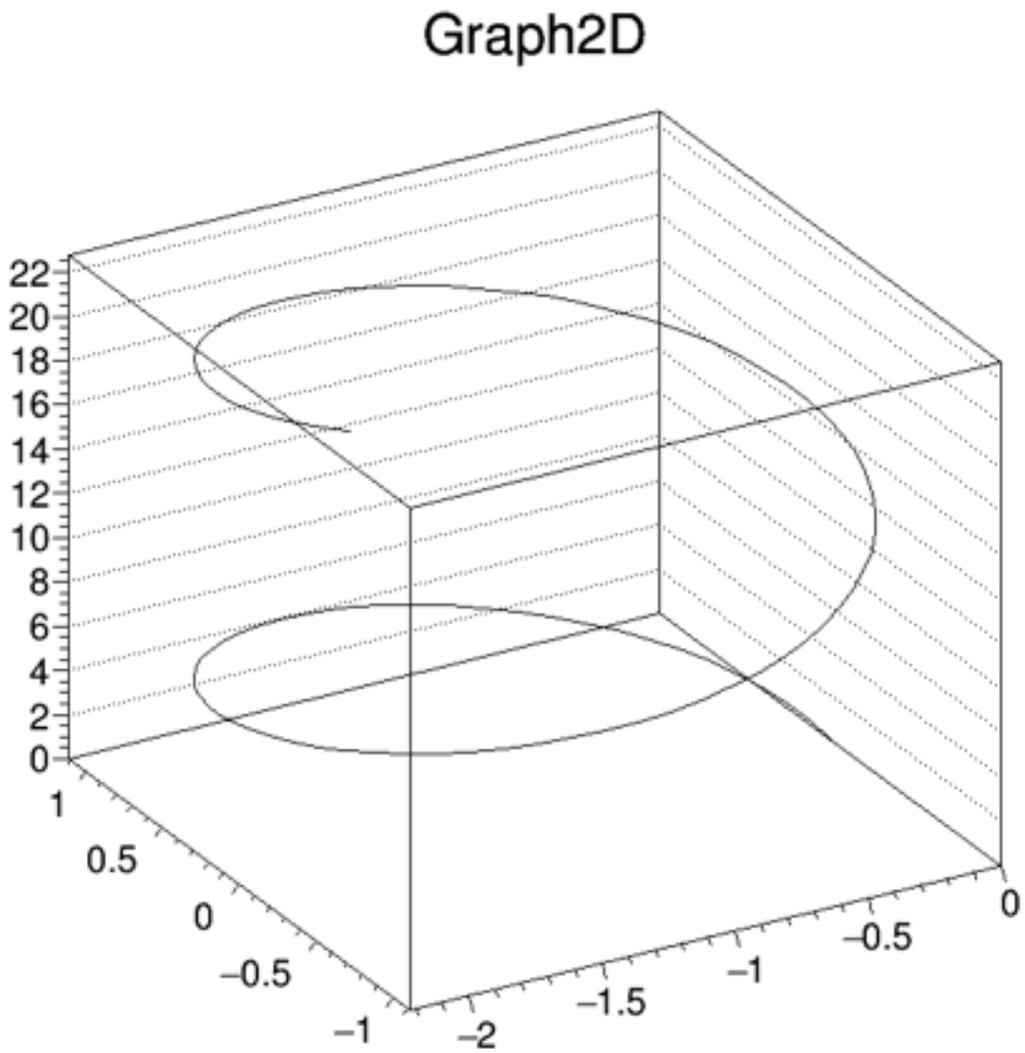
Code Testing

Fundamental to test a new code. Add print-out, plots and debugging.

Starting point

- Generate a spiral pion with
  p_T=0.01GeV, B=3T => 1,1cm

$$\begin{cases} x &= x_0 &+ d_\rho \cos\phi_0 &+ \frac{\alpha}{\kappa}(\cos\phi_0 - \cos(\phi_0+\phi)) \\ y &= y_0 &+ d_\rho \sin\phi_0 &+ \frac{\alpha}{\kappa}(\sin\phi_0 - \sin(\phi_0+\phi)) \\ z &= z_0 &+ d_z &- \frac{\alpha}{\kappa}\tan\lambda\cdot\phi, \end{cases}$$



- Built 10(11) detector measurement layers.

- Consider material layers, alternation of Silicium and Air. Consider M.S. and energy loss.

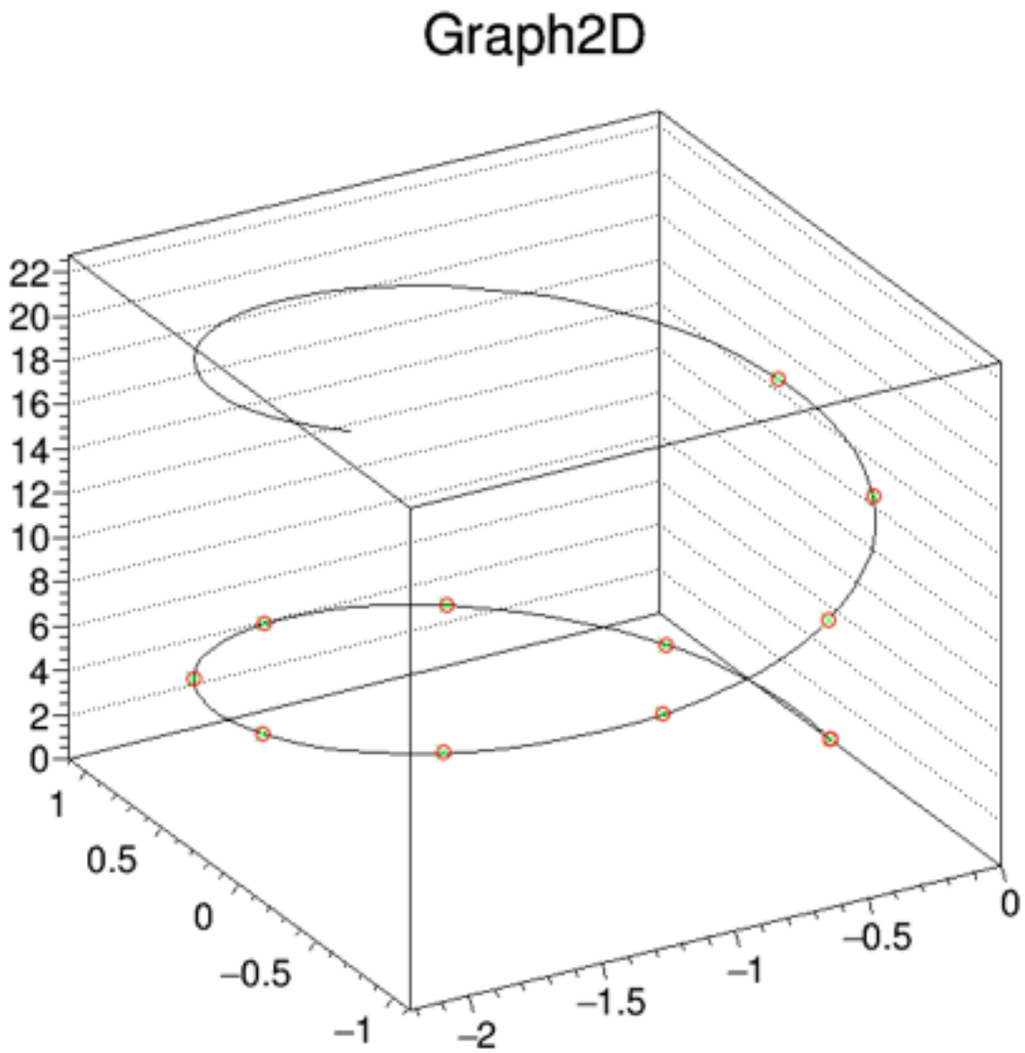
Code Testing

Fundamental to test a new code. Add print-out, plots and debugging.

Starting point

- Generate a spiral pion with
  p_T=0.01GeV, B=3T => 1,1cm

$$\begin{cases} x &= x_0 &+ d_\rho \cos\phi_0 &+ \frac{\alpha}{\kappa}(\cos\phi_0 - \cos(\phi_0+\phi)) \\ y &= y_0 &+ d_\rho \sin\phi_0 &+ \frac{\alpha}{\kappa}(\sin\phi_0 - \sin(\phi_0+\phi)) \\ z &= z_0 &+ d_z &- \frac{\alpha}{\kappa}\tan\lambda\cdot\phi, \end{cases}$$



- Built 10(11) detector measurement layers.

- Consider material layers, alternation of Silicium and Air. Consider M.S. and energy loss.

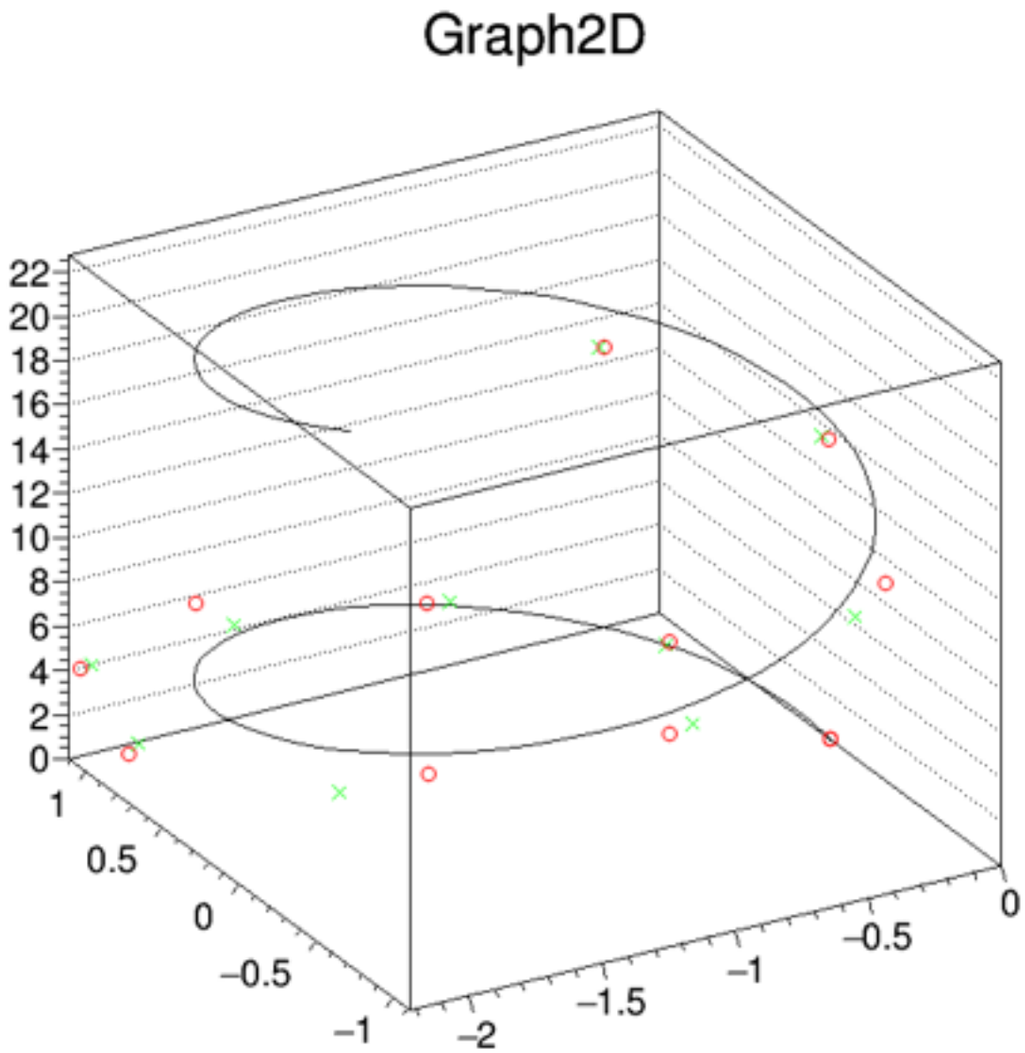
Code Testing

Fundamental to test a new code. Add print-out, plots and debugging.

Starting point

- Generate a spiral pion with
  p_T=0.01GeV, B=3T => 1,1cm

$$\begin{cases} x &= x_0 &+ d_\rho \cos\phi_0 &+ \frac{\alpha}{\kappa}(\cos\phi_0 - \cos(\phi_0+\phi)) \\ y &= y_0 &+ d_\rho \sin\phi_0 &+ \frac{\alpha}{\kappa}(\sin\phi_0 - \sin(\phi_0+\phi)) \\ z &= z_0 &+ d_z &- \frac{\alpha}{\kappa}\tan\lambda\cdot\phi, \end{cases}$$



- Built 10(11) detector measurement layers.

- Consider material layers, alternation of Silicium and Air. Consider M.S. and energy loss.

Next Steps

1 - Code adjustment

"Start with what is right rather than what is acceptable."
Franz Kafka

- Technicalities to deep understand code's doing + make it more user friendly + adjusting to our specific needs. Looking at the future!

- Class merge, code slimming, method adjustment.

Next Steps

2 - Quick standalone test

- Test full machinery in a “controlled” environment similar to FOOT.

- Change geometry to be more FOOT-like:
  - B along y,
  - correct distances between layers;

- Simulate single-track event with 1 FOOT-like particle (low pT ion).

Next Steps

3 - A more complicated test

- Hits directly coming from the FLUKA simulation.
  Adding more tracksXevents + some background hits.

- Strategies to select the points to be fitted and the initial extrapolation assumption.

- Variable B field.

Next Steps

4 - stop testing, start the real job!

- Include Kalman Filter code into the global framework.

- Continue the optimisation!!