November 10, 2015

Track reconstruction
Excercises from Lecture notes

Nordic detector course, November 2015

Peter Hansen

0.1 Exercises

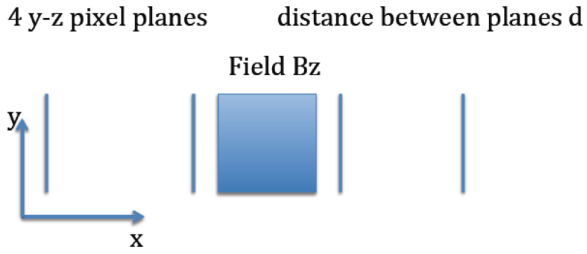

Figure 1: A simple spectrometer

1. **Design a spectrometer**

   - Copy all the files in

     `http://www.nbi.dk/~phansen/nordforsk/`

     This is C++ code using ROOT libraries that simulates a compact spectrometer with very thin (0.05mm) CMOS pixel layers (the MIMOSA chip) and a weak spectrometer magnet in the middle. You are assumed to be running some unix on your computer. Figure 1 illustrates the setup, except that we here use three planes on each side of the magnet. The goal is to measure positron production in the 50-1000 MeV range from electrons impinging on a diamond target upstream of the first layer. Such a spectrometer is used by an Aarhus group in connection with studies of a positron factory for the future CLIC accelerator.

     The program simulates mono-energetic single positrons emitted along the beam (x) axis, at a small cost of generality. The simulated positrons are traced and digitized through the spectrometer, including effects from multiple scattering, noise and detection inefficiencies.

     In this excise you implement a Kalman Filter, in a way suggested below, for the pattern recognition of the simulated data, resulting in a number of track candidates After that, a global chi-squared fit finds the best among the candidates. The pattern recognition requires hits in all layers, except for the last two where only one is required.

   - Implement the method Spectrometer::KalmanFilter in Spectrometer.C using the matrix formalism of these notes with help of the ROOT TMatrixD class. The method must propagate a track candidate from the last detector layer with a hit to layer $p1$ in the non-bending x-z plane. It must then update the track parameters in x-z using the hit *ihit* and return the chisquared contributed by the hit as well as the updated RecoTrack, tout. The first lines could be:

     ```
     float Spectrometer::KalmanFilter
     (int p1, int ihit, RecoTrack t, RecoTrack& tout) {
     ```

```
if(debug) cout << `` in Kalman Filter `` << endl;
double s2   = resolution*resolution;
double pinv = 1./beamMomentum;        //use here a fixed momentum
double t0   = multScattAngle*pinv;    //average multiple scattering angle
TMatrixD z  = t.GetPar();             //track parameters at current plane
TMatrixD C  = t.GetCov();             //their covariance matrix
//we are not sure which plane is the current, so check
vector<int> hits=t.GetHits();
int plane0  = hitdata.at(hits.back()).GetPlane(); //so this is it
double d=distBetweenPlanes*(p1-plane0);  //propagation distance
if(debug) cout << ``  propagate from plane `` << plane0
               << `` to plane `` << p1 << endl;

Then make the propagator matrix (TMatrixD) F and it's transposed.
Then make the multiple scattering smearing matrix.
Then make the total covariance matrix of the propagated track.
Then propagate the track parameters, z, to plane p1.
Then make the covariance matrix of the updated track.
Then make the parameters of the updated track.
Then, if chi-squared<Cut1, construct the updated RecoTrack, tout.
Finally return the chi-squared.

Warning: If you transpose or invert a TMatrixD into a new matrix
(by doing HT=H.T() or Cinv=C.Invert()) then H or C is also left in the
transformed state, so you need to transform them back ( just do H.T()
or C.Invert() again, or else copy the matrices before transforming).

When you have edited Spectrometer.C to your liking, then type make.
Then type ./SMgr.exe | tee spectrometer.out
Look then at the output, type root and in ROOT type
TBrowser b and click at the histogram file and the histograms
you want to see, fit, zoom or whatever.
```

- Is the spectrometer configuration optimal? Try vary the length by varying distBetweenPlanes. Can we do with fewer planes? Try 4 instead of 6. What happens if the magnet is twice as strong? What should be chosen for the adjustable noise occupancy? What is a reasonable chi-squared cut? Use the 1/p accuracy, the track reconstruction efficiency and the computing time as deciding quality factors. Try out both 0.05 GeV and 1 GeV in each case.
- The simulated efficiency is in fact a pure guess. If the simulated data had been real, how would you measure the plane efficiency (the probability for a track to produce a hit in a plane). How would you then estimate the track reconstruction efficency?
- Since we deal with positrons, we need to worry about bremsstrahlung, but this is not taken into account in the program. How much trouble do you expect from this?

(Hint: make a hit-and-miss simulation of the fractional energy loss in each of the three planes before the magnet and in the line calculating the angular bend in the magnet replace beamMomentum with a reduced momentum).

2. **Reconstruct a vertex**. Let us consider the case where our spectrometer is traversed by two pions from a K0 decay somewhere upstream. Imagine that our program is now made able to return *two* fitted tracks, each with parameters (z0,z',y0,y',q/p) and their covariance matrix.

   - Write down an initial estimate of the decay vertex v0 and the derivative matrices D and E from the Billoir vertex fit.
   - How could you take advantage of external information? For example that the beam, including the K0's, propagates along the x-axis with a gaussian transverse profile of size b, or that you are sure that the two pions really come from a K0 decay.

3. **Make an alignment algorithm**. Among the downloaded files are AlignSpectrometer.C and AMgr.C. Edit the Makefile and, if needed, the AlignSpectrometor constructor. Then type make and then

   ```
   ./AMgr.exe | tee alignspectrometer.out
   ```

   Here, each plane is shifted randomly in each direction by a sigma of 0.1mm. Both the local and global method is implemented and can be chosen in the constructor. Look at the output (both the "out" file and the root file).

   - What accuracy is obtained and does that make sense? Does it help to make iterations? Is global better than local? Are there " weak modes " giving trouble? Anything you could do about those? Then assume that the true beam axis in the $x-z$ plane has been measured with beam monitoring devices to be $z_0 = 0.002 \pm 0.001 cm$ at the first plane and have a slope $dz/dx = 0.1 \pm 0.01 mrad$. Try to implement that knowledge in the simulation and take advantage of it in the alignment fit.
   - The program has only considered unwanted translations of the planes. How would you formally implement the possibility of a rotation, for example around the local z-axis of the planes?

Physical and material constants

Quantity	symbol	value
Speed of light	c	299792458 m/s
Planck's const	h	$6.6260696 \times 10^{-34}$ J s
Planck's const, reduced	$\hbar = h/2\pi$	$6.5821193 \times 10^{-22}$ MeV s
Conversion const	$\hbar c$	197.326963 MeV fm
Electron charge	e	$1.602176490\ 10^{-19}$ C
Electron mass	m_e	0.51099891 MeV/c^2
Proton mass	m_p	938.27201 MeV/c^2
Atomic mass unit	u	931.494 MeV/c^2
Atomic mass unit	u	$1.6605388 \times 10^{-27}$ kg
Vacuum permittivity	ε_0	8.854188×10^{-12} F m^{-1}
Vacuum permeability	$\mu_0 = 1/\varepsilon_0 c^2$	$4\pi \times 10^{-7}$ N A^{-2}
Fine structure const	$\alpha = e^2/4\pi\varepsilon_0\hbar c$	1/137.036
Classical electron radius	$r_e = e^2/4\pi\varepsilon_0 m_e c^2$	$2.81794029 \times 10^{-15}$ m
Bohr radius	$a_\infty = r_e\alpha^{-2}$	$0.529177209 \times 10^{-10}$ m
Rydberg energy	$hcR = m_e c^2\alpha^2/2$	13.6056919 eV
Thomson cross section	$8\pi r_e^2/3$	0.665246 barn
Bohr magneton	$\mu_B = e\hbar/2m_e$	$5.78838176 \times 10^{-11}$ MeV T^{-1}
Avogadro const	N_A	6.0221418×10^{23} mol^{-1}
Boltzmann const	k	8.617343×10^{-5} eV K^{-1}

Table 1:

Material	Z	A	Nucl. int. length g cm^{-2}	Rad. Length g cm^{-2}	$\frac{dE}{dx}$(min) MeV g^{-1} cm^2	density g cm^{-3} (g l^{-1})	Refr. index
H_2	1	1.008	52	63.04	4.104	0.071(0.084)	1.11
He	2	4.003	71.0	94.32	1.937	0.125(0.166)	1.02
Be	4	9.012	77.8	65.19	1.595	0.534	1.02
Graphite	6	12.011	85.8	42.70	1.742	2.210	
dry air	7	14	90.1	36.6	1.815	1.205	
water			83.3	36.08	1.992	1.	1.33
CO_2			88.9	36.20	1.819	1.563(1.842)	
Al	13	26.98	107.2	24.01	1.615	2.699	
Si	14	28.086	108.4	21.82	1.664	2.329	3.95
Ar	18	39.95	119.7	19.55	1.519	1.574(2.98)	1.32
Fe	26	55.85	132.1	13.84	1.451	7.874	
Cu	29	63.55	137.3	12.86	1.403	8.96	
W	74	183.84	191.9	6.76	1.145	19.3	
Pb	82	207.2	199.6	6.37	1.122	11.35	

Table 2: Gasses refer to NTP (20° 1 atm). When there are two numbers, the first is for liquid and the second for gas.