

## Solutions to Physics 226: Problem Set #1

### 1. Relativistic expressions relevant for accelerator performance

- (a) The center-of-mass energy for two particles with masses  $m_1$  and  $m_2$  is

$$s = E_{cm}^2 = (P_1 + P_2)^2 = m_1^2 + m_2^2 + 2E_1E_2 - 2\vec{p}_1 \cdot \vec{p}_2$$

where the  $P_i$  are the particle 4-momenta and  $\vec{p}_i$ 's are the particle 3-momenta.

For the center-of-mass case,  $\vec{p}_2 = -\vec{p}_1$  and  $p_i = \sqrt{E_i^2 - m_i^2}$ . Thus

$$s = m_1^2 + m_2^2 + 2E_1E_2 \left( 1 + \sqrt{\left(1 - \frac{m_1^2}{E_1^2}\right)\left(1 - \frac{m_2^2}{E_2^2}\right)} \right)$$

- (b) If the masses are small compared to their energies

$$s = 4E_1E_2 + m_1^2\left(1 - \frac{E_2}{E_1}\right) + m_2^2\left(1 - \frac{E_1}{E_2}\right)$$

Neglecting masses completely

$$s = 4E_1E_2$$

$$E_{cm} = \sqrt{4E_1E_2}$$

In the lab frame, ( $M_2$  at rest)

$$\begin{aligned} s &= (E_1 + m_2)^2 - p_1^2 \\ &= m_1^2 + 2m_2E_1 + m_2^2 \end{aligned}$$

If the masses are small

$$s = 2m_2E_1$$

- (c) i. Neglecting the electron and positron masses, the boost of the center-of-mass is given by

$$\beta = \frac{p_{tot}}{E_{tot}} = \frac{E_1 - E_2}{E_1 + E_2}$$

where  $E_1$  and  $E_2$  are the lab energies of the electron and positron.  
Using

$$4E_1E_2 = M_\Upsilon^2$$

and

$$\beta\gamma = \frac{p_{tot}}{E_{cm}} = \frac{E_1 - E_2}{M_\Upsilon}$$

we can solve to get

$$E_1 = \frac{M_\Upsilon}{2} \left( \beta\gamma + \sqrt{\beta^2\gamma^2 + 1} \right) \approx 9 \text{ GeV}$$

$$E_2 = \frac{M_\Upsilon}{2} \left( -\beta\gamma + \sqrt{\beta^2\gamma^2 + 1} \right) \approx 3.1 \text{ GeV}$$

- ii. The momentum of the  $B$  in the center-of-mass frame is very close to zero. So, the momentum in the lab frame is set by the boost  $\beta\gamma = 0.065$  in the beam direction. The mean decay distance is therefore

$$\langle L \rangle = \beta\gamma c\tau_0 \approx 260 \mu\text{m}$$

- iii. In the center-of-mass of the  $B$  meson, both pions have equal and opposite momentum and  $E = m_B/2$ . The angle in the lab will depend on the decay angle with respect to the beamline

$$p_z^{lab} = \gamma_B(p_z^B + \beta_B E) \approx \gamma_B E(\beta \cos \theta + 1)$$

where we have ignored the pion mass relative to its momentum. The maximum boost is when the  $\pi$  is moving in the direction of the  $B$

$$\beta_{max} = \frac{\beta_\Upsilon + \beta_B}{1 - \beta_\Upsilon\beta_B} = 0.57$$

The minimum boost is in the opposite direction

$$\beta_{min} = \frac{\beta_\Upsilon - \beta_B}{1 + \beta_\Upsilon\beta_B} = 0.41$$

The extreme values of the momenta are:

$$p_{max} = \frac{m_B}{2} \gamma_{max} (1 + \beta_{max}) \approx 5.0 \text{ GeV}$$

$$p_{min} = \frac{m_B}{2} \gamma_{min} (1 - \beta_{min}) \approx 1.7 \text{ GeV}$$

2. Particle ID using a time-of-flight detector

- (a) For ultrarelativistic particles ( $m_1, m_2 \ll p$ ) the time of flight is

$$t = \frac{\ell}{v} = \frac{\ell E}{c pc}$$

where  $E$  is the energy and  $p$  is the momentum of the particle. Thus for two particles with the same momentum and different masses

$$\Delta t = t_1 - t_2 = \frac{\ell}{c} \left( \frac{\sqrt{(pc)^2 + (m_1 c)^2}}{pc} - \frac{\sqrt{(pc)^2 + (m_2 c)^2}}{pc} \right)$$

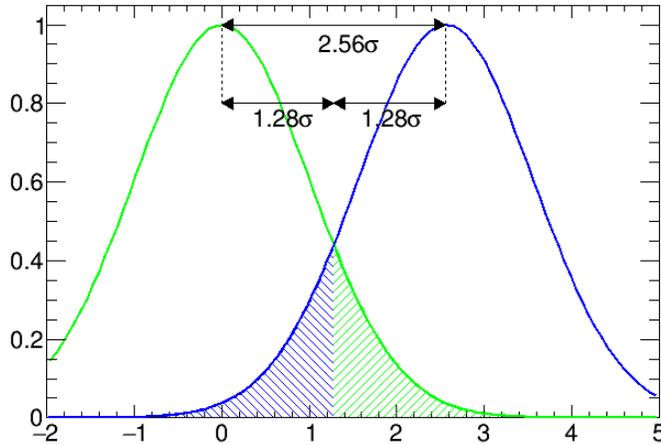
Taylor expanding  $\sqrt{(pc)^2 + (mc)^2} = pc\sqrt{1 + (mc)^2/(pc)^2} = p(1 + \frac{1}{2}(mc)^2/(pc)^2 + \dots)$  we find

$$\Delta t = \frac{\ell}{2c} \frac{(m_1 c)^2 - (m_2 c)^2}{(pc)^2}$$

setting  $c = 1$  we get:

$$\Delta t = \frac{\ell}{2} \frac{(m_1)^2 - (m_2)^2}{p^2}$$

- (b) A 90% confidence level separation corresponds to  $1.64\sigma$  for a two-sided cut. However, since we are only cutting on one side of the Gaussian, we should ask that 10% of the events be in one tail (so 20% in both tails). That corresponds to  $1.28\sigma$  from each peak, or  $2.56\sigma$  between the peaks, as shown here:



So we are looking for the momentum where the time difference  $\Delta t = 2.56 \times 100ps$  which means:

$$\begin{aligned}(pc)^2 &= \frac{\ell}{2c} \frac{(m_1c^2)^2 - (m_2c^2)^2}{\Delta t} \\(pc) &= \sqrt{\frac{1.40m}{6 \times 10^8 m/s} \frac{(0.494^2 - 0.140^2) \text{GeV}^2}{258 \times 10^{-12} s}} \\p &= 1.43 \text{GeV}/c\end{aligned}$$

(c) First, we find the separation in time of arrival of the  $\pi$  and  $K$ :

$$\Delta t = \frac{1.4}{6 \times 10^8} \frac{(0.494^2 - 0.140^2) \text{GeV}^2}{(1.5 \text{GeV}/c)^2} = 2.33 \times 10^{-10} \text{ sec} = 233 \text{ ps}$$

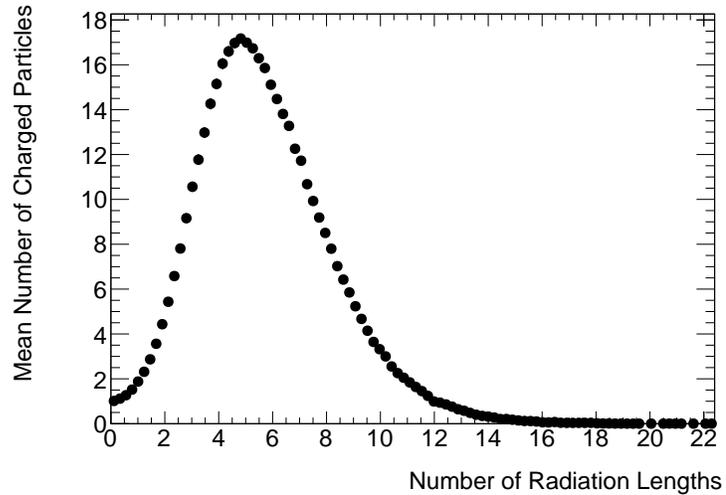
Thus, if the mean time of arrival of the  $\pi$  is -233 ps, the mean time of arrival of the  $K$  is 0 ps. In each case, the arrival time is Gaussianly distributed with a  $\sigma = 100$  ps. If we call everything a  $K$  that arrives after time  $T$ , then the purity of the sample is:

$$\begin{aligned}p(K) &= \frac{\int_T^\infty dt e^{-\frac{t^2}{2\sigma^2}}}{\int_T^\infty dt e^{-\frac{t^2}{2\sigma^2}} + 3 \int_T^\infty dt e^{-\frac{(t+233)^2}{2\sigma^2}}} \\&= \frac{\text{erfc}(T/\sqrt{2}\sigma)}{((\text{erfc}((T/\sqrt{2}\sigma) + 3(\text{erfc}((t + 233)/\sqrt{2}\sigma)))\end{aligned}$$

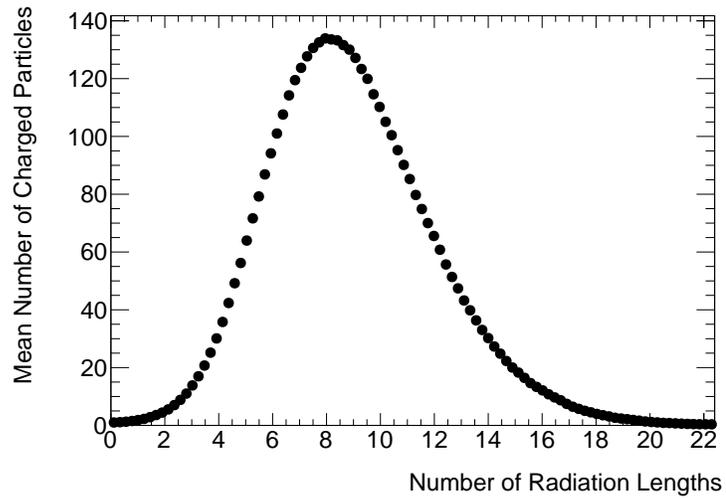
We can solve for  $T$  numerically (eg using Mathematica). The answer is 150 ps. The efficiency of this cut is 0.79.

### 3. A Monte Carlo Model of Electromagnetic Showers

Here is the distribution for 1 GeV showers



Here is the distribution for 10 GeV showers



I did this problem using Root. Here is the code I used:

```
#include <iostream>
#include <vector>
#include "TRandom.h"
#include "TProfile.h"
```

```

using std::cout;
using std::endl;
using std::vector;

class aShowerParticle{
public:
    double zBeg;
    double zEnd;
    double Einit;
    int Charge;
    aShowerParticle() {zBeg=0.0;zEnd=0.0;Einit=0.0; Charge=0;}
    aShowerParticle(double beg,double end,double e, int q)
        {zBeg=beg;zEnd=end;Einit=e; Charge=q;}
    ~aShowerParticle() {}
};

// Eincident is in GeV
void EMShower(int numEvt, double Eincident) {

    // Here we declare all the physical constants
    double Radlen=7.39/8.30; // gm/cm**2 /gm/cm**3 = cm
    double dEdxGeVperCm = 1.229*8.30*1.0e-3; //MeV cm2/gm * gm/cm*3 *10-3 GeV/M
    double length=22.4; //depth of the crystal cm

    // Make a profile histogram to store the showers
    TProfile* hProf = new TProfile("hprof","Mean number of charged particles vs
    hProf->SetXTitle("Number of Radiation Lengths");
    hProf->SetYTitle("Mean Number of Charged Particles");
    double bincenter[100];
    for(int bin=0;bin<100; bin++) {
        bincenter[bin] = (0.5+bin)*length/100;
    }
}

```

```

cout << "The requested number of events is: " << numEvt << endl;

// Instantiate the random number generator
TRandom* random = new TRandom;
vector<double> mydata;
double Zinteraction, Zstart;

// aShower is a collection of all the particles produced in a shower
// activeParticles is a collection of those still alive
// note; both these contain pointers to the same objects
// aShower owns the objects and will delete them
// failure to delete would cause a memory leak
vector<aShowerParticle*> aShower;
vector<aShowerParticle*> activeParticles;

for (int i=0; i<numEvt; i++) {
    cout << " Event " << i << endl;
    // Create our incident particle
    aShowerParticle* part = new aShowerParticle(0.0,0.0,Eincident,1);
    // and add it to the shower and the list of active particles
    activeParticles.push_back(part);
    aShower.push_back(part);
    vector<aShowerParticle*>::iterator iter;
    // We'll keep going until we run out of particles
    while(activeParticles.size()>0) {
        // Let's start by looking at the first particle in the list
        iter = activeParticles.begin();
        // Electrons and photons assumed to have exponential distribution to the
        // with argument of the exponential=Radlen
        // Photons convert with exponential distribution
        // with argument of the exponential=(9/7)Radlen
        if((*iter)->Charge!=0) {
            Zinteraction = random->Exp(Radlen);
        }
    }
}

```

```

else {
    Zinteraction = random->Exp(9.0*Radlen/7.0);
}
Zstart = (*iter)->zBeg+Zinteraction;

// For charged particles, check how much energy they loose to ionization
// to make sure that they don't stop before the next interaction
if((*iter)->Charge!=0 &&
    Zinteraction*dEdxGeVperCm > (*iter)->Einit){
// The particle stops in the crystal. Kill it at the point it stops
(*iter)->zEnd = (*iter)->zBeg+((*iter)->Einit)/dEdxGeVperCm;
    activeParticles.erase(iter);
}
else if(Zstart>=length) {
    // The next interaction is after the end of the crystal: Terminate
    (*iter)->zEnd=length;
    activeParticles.erase(iter);
}
else {
// The next interaction is in the crystal: Replace the particle with
// its interaction products
(*iter)->zEnd=Zstart;
double Eeach;
int q1,q2;
// photons produce e+e- pairs
if((*iter)->Charge==0) {
    Eeach=((*iter)->Einit)/2.0;
    q1=1;
    q2=-1;
}
// e^+ and e^- undergo brem. We will delete the original
// e^+ or e^- and replace it with a a new one plus a photon
else {
    Eeach=((*iter)->Einit-dEdxGeVperCm*Zinteraction)/2.0;

```

```

        q1>(*iter)->Charge;
        q2=0;
    }

    activeParticles.erase(iter);
    aShowerParticle* part1 = new aShowerParticle(Zstart,0.0,Eeach,q1);
    activeParticles.push_back(part1);
    aShower.push_back(part1);
    aShowerParticle* part2 = new aShowerParticle(Zstart,0.0,Eeach,q2);
    activeParticles.push_back(part2);
    aShower.push_back(part2);
    } // end of test on whether particle interacts
    } // No more particles to propagate
// Make histogram of number of particles here and then free the memory
int nCh=0;
int nChinBin[100];
for(int bin=0; bin<100; bin++) {nChinBin[bin]=0;}
for(iter=aShower.begin(); iter!=aShower.end(); iter++){
    if((*iter)->Charge!=0) {
        nCh++;
        for(int bin=0;bin<100;bin++) {
            if(((iter)->zBeg) < bincenter[bin] && ((iter)->zEnd) > bincenter[bin])
                nChinBin[bin]++;
        }
    }
}
for(iter=aShower.begin(); iter!=aShower.end(); iter++){
    delete *iter;
}
aShower.clear();
for(int bin=0; bin<100; bin++) {hProf->Fill(bincenter[bin],nChinBin[bin],1)
} // end of loop over events

cout << " All events complete" << endl;

```

```
hProf->Draw();  
}          //End of macro
```