2014년 고등과학원 겨울학교 프로젝트 해설

고려대학교 물리학과 원은일 (eunil@hep.korea.ac.kr)

2014년 1월

1 프로젝트

1. 제시문 1 에서 $f(x) = 1/\xi \exp(-x/\xi)$ 일 경우 $x(r)$ 은 어떻게 표현되는가?

답:

$$x(r) = -\xi \log(r) \tag{1}$$

이 된다. 적분을 하면 $(1 - r)$ 이지만 어차피 r 이 구간 $[0,1]$ 에서 균일하므로 위과 같이 바꾸어도 무방한다.

2. 표면뮤온의 운동량 값을 계산해 보시오.

답: 약 29 MeV/c가 나와야 한다.

3. 이 표면 뮤온 (μ^+) 은 그 스핀이 100% 분극되어 있다. 왜 그런지 생각해 보시오. 그렇다면 스핀의 방향은 μ^+ 의 운동량 방향에 대하여 어느 방향으로 분극되어 있는가?

답: 파이온이 정지한 계에서는 뮤온 (μ^+) 과 중성미자 (ν_μ) 가 약한 상호작용에 의해 붕괴됨. 중성미자는 left handed 로서 스핀과 운동방향이 반대, 따라서 뮤온 (μ^+) 도 운동방향의 반대방향으로 스핀이 정렬되어 총 스핀 값이 붕괴 전후 보존된다.

4. 이 표면 뮤온이 정지하여 있다고 가정하자. 정지한 뮤온은 평균수명값에 따라 붕괴한다. 이에 대한 시뮬레이션 코드를 작성하고 시간에 대한 분포를 그린 다음, 이를 fit 하여 평균 수명값을 한번 계산해 보자. 평균 수명값을 1%의 오차로 계산하려면 몇개의 이벤트를 생성해야 하는가? 위의 pdg 값의 정확도를 가지려면 몇개의 이벤트가 필요한가?

답:

$$\frac{1}{\sqrt{N}} = 0.01 \tag{2}$$

을 만족하는 N은 10^4 이다. 이에 대한 ROOT 프로그램은

```
//
// muon decay simulation at the rest from of muon
//
// Eunil Won (Korea Univ.) eunil@hep.korea.ac.kr
//

const Int_t ngen = 1e4;
const Double_t tau_mu = 2.197019e-6;      // muon mean life (s)

Double_t f1(Double_t *x, Double_t *par)
{
  return par[0]*TMath::Exp(-1.0*x[0]/par[1]);
}

void cmdecay_kias()
{
  TH1::SetDefaultSumw2();

  //
  // a uniform random number
  //
  gRandom->SetSeed();

  Double_t eta, theta, tdcay, weight;
  Int_t dummy;

  //
  // histograms
  //
  TCanvas* canvas = new TCanvas("canvas","eta",100,100,700,700);
  canvas->Divide(1,1);
  hdcy = new TH1F("hdcy","decay time of muon",200,0.,6*tau_mu);
  hdcy->SetFillColor(3);
```

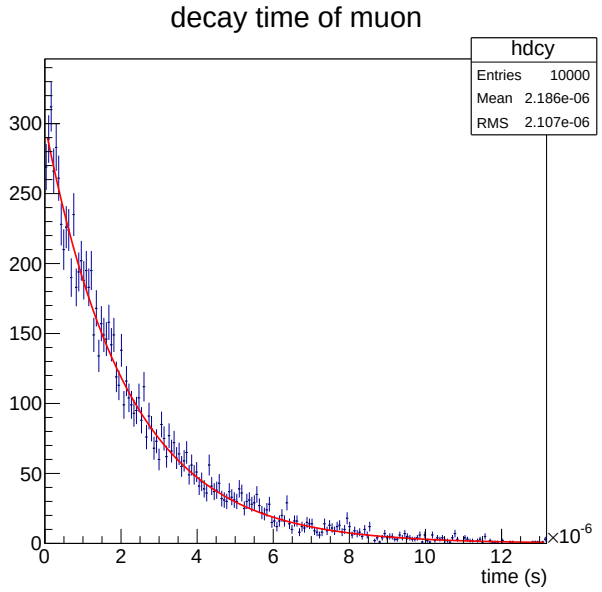

Figure 1: 파란색 점들은 양전자, 빨간색 선을 fit 결과.

```
  hdcy->SetMinimum(0.0);
  hdcy->Sumw2();

  //
  // in muon rest frame
  //
  for (Int_t i=0;i<ngen;i++) {
    tdcay = -1.0*tau_mu*TMath::Log(gRandom->Rndm(dummy));
    hdcy->Fill(tdcay);
  }
  hdcy->SetXTitle("time (s)");
  hdcy->Draw();

  TF1 *func = new TF1("func",f1,0.0,tau_mu,2);
  func->SetParameters(ngen,tau_mu);
  hdcy->Fit("func");
}
```

이고 실제로 $N = 10^4$일 경우 약 1%의 오차가 생긴다. 그림 1에서 그 결과를 볼 수 있다.

5. (advanced) 정지한 분극되지 않은 뮤온의 붕괴에서 양전자의 에너지 분포는 어떻게 되는가? 이를 시뮬레이션 프로그램을 작성하여 검증해 볼 수 있는가?

답:

```
//
// muon decays
//
//
// Eunil Won (Korea Univ.) eunil@hep.korea.ac.kr
//
const Int_t ngen = 100000;
Int_t dummy = 1;

const Double_t tau_mu = 2.197019e-6;                    // muon mean life (s)
const Double_t m_mu = 105.658367;                       // muon mass (MeV)
const Double_t m_el = 0.511;                            // electron mass (MeV)
const Double_t m_nu = 0.0;                              // neutrino (MeV)

const Double_t Gf = 1.0;                      // set to 1 since we do not make absoulte measurement
const Double_t PI = TMath::Pi();
const Double_t psmax = m_mu/2.0;
const Double_t gamma = 3.0;                          // relativistic Lorentz factors
const Double_t beta = TMath::Sqrt(gamma*gamma-1.0)/gamma;
const Double_t plmax = gamma*(psmax+beta*psmax);

Double_t alam(Double_t a, Double_t b, Double_t c)
{
  return a*a + b*b + c*c - 2.0*a*b - 2.0*b*c - 2.0*c*a;
}

//
// based on Collider Physics by Barger and Phillips, Chapter 11.
//
void mdec(Double_t *mass, Double_t *w, Double_t *p_el, Double_t *p_nue, Double_t *p_num)
{

  Double_t mc = mass[0];
  Double_t ms = mass[1];
  Double_t me = mass[2];
  Double_t mg = mass[3];

  //
  // rest frame of muon
  //
  Double_t pmax2, p1sq, p1, e1, m23;
```

```
pmax2 = alam(mc*mc, ms*ms, (me+mg)*(me+mg))/(4.0*mc*mc);
p1sq = pmax2*gRandom->Rndm(dummy);
p1 = TMath::Sqrt(p1sq);
e1 = TMath::Sqrt(p1sq+ms*ms);
m23 = TMath::Sqrt(mc*mc + ms*ms - 2*mc*e1);

//
// now work in 2-3 rest frame
//
Double_t pesq, pe, ee, costh, sinth, phi, ex, ey, ez, ge, gx, gy, gz;

pesq = alam(m23*m23, me*me, mg*mg)/(4*m23*m23);
pe = TMath::Sqrt(pesq);
ee = (m23*m23 + me*me -mg*mg)/(2*m23);
costh = 1.0 - 2.0*gRandom->Rndm(dummy);
sinth = TMath::Sqrt(1.0 - costh*costh);
phi = 2*PI*gRandom->Rndm(dummy);

ex = pe*sinth*TMath::Sin(phi);
ey = pe*sinth*TMath::Cos(phi);
ez = pe*costh;

ge = m23 - ee;
gx = -ex;
gy = -ey;
gz = -ez;

Double_t pcsq, ce, cz, se, sz;

pcsq = alam(m23*m23, mc*mc, ms*ms)/(4.0*m23*m23);
ce = (mc*mc - ms*ms + m23*m23)/(2.0*m23);
cz = TMath::Sqrt(pcsq);
se = ce - m23;
sz = cz;

//
// matrix element squared
//
Double_t msq;
msq = 64.0*Gf*Gf*(ce*ee - cz*ez)*(se*ge-sz*gz);

*w = msq*p1*pmax2*pe/(64.0*PI*PI*PI*mc*e1*m23);

//
// boost back to c rest frame
//
Double_t g, bg, b;
g =ce/mc;
bg = -cz/mc;
```

```
  b = bg/g;
  ee = ee*g + ez*bg;
  ge = ge*g + gz*bg;
  ez = ee*b + ez/g;
  gz = ge*b + gz/g;

  //
  // randomize orientation in c-rest frame
  //
  Double_t ct, st, cp, sp, sx, sy;

  ct = 1.0 - 2.0*gRandom->Rndm(dummy);
  st = TMath::Sqrt(1-ct*ct);
  phi = 2.0 * PI * gRandom->Rndm(dummy);
  cp = TMath::Cos(phi);
  sp = TMath::Sin(phi);
  ez = ez*ct - ey*st;
  gz = gz*ct - gy*st;
  ey = (ez*st + ey)/ct;
  gy = (gz*st + gy)/ct;
  ex = ex*cp - ey*sp;
  gx = gx*cp - gy*sp;
  ey = (ex*sp + ey)/cp;
  gy = (gx*sp + gy)/cp;
  se = mc - ee - ge;
  sx = -ex - gx;
  sy = -ey - gy;
  sz = -ez - gz;

  p_el[0] = se;
  p_el[1] = sx;
  p_el[2] = sy;
  p_el[3] = sz;
  p_num[0] = ee;
  p_num[1] = ex;
  p_num[2] = ey;
  p_num[3] = ez;
  p_nue[0] = ge;
  p_nue[1] = gx;
  p_nue[2] = gy;
  p_nue[3] = gz;

}

void boost(Double_t md, Double_t *p, Double_t *p_boost)
{
  Double_t dx, dy, dz, ee, ex, ey, ez;
  dx = p[0];
  dy = p[1];
```

```
  dz = p[2];

  ee = p_boost[0];
  ex = p_boost[1];
  ey = p_boost[2];
  ez = p_boost[3];

  Double_t bg1, g1, bg2, g2, bg3, g3, e, x, y, z;

  bg1 = dx/md;
  g1 = TMath::Sqrt(1.0 + bg1*bg1);
  bg2 = dy/(md*g1);
  g2 = TMath::Sqrt(1.0 + bg2*bg2);
  bg3 = dz/(md*g1*g2);
  g3 = TMath::Sqrt(1.0 + bg3*bg3);

  e = g1*g2*g3*ee + bg1*g2*g3*ex + bg2*g3*ey + bg3*ez;
  x = g1*ex + bg1*ee;
  y = g2*ey + bg1*bg2*ex + g1*bg2*ee;
  z = g3*ez + bg2*bg3*ey + bg1*g2*bg3*ex + bg3*g2*g1*ee;

  ee = e;
  ex = x;
  ey = y;
  ez = z;

  p_boost[0] = ee;
  p_boost[1] = ex;
  p_boost[2] = ey;
  p_boost[3] = ez;
}

void gen_kias()
{

  Double_t mass[4], w, p_el[4], p_nue[4], p_num[4];
  mass[0] = m_mu; mass[1] = m_el; mass[2] = m_nu; mass[3] = m_nu;

  TCanvas* c_c = new TCanvas("c_c","positron energy",100,100,600,300);
  c_c->Divide(2,1);
  h_ps = new TH1F("h_ps","e+ p",100,0.,psmax+10.0);
  h_pl = new TH1F("h_pl","e+ p",100,0.,plmax+10.0);

  Double_t pmuon[3];  // x,y,z of muon momentum
  pmuon[0] = 300.0;   // muon momentum is 300 MeV/c in local x-axis
  pmuon[1] = 0.0;
  pmuon[2] = 0.0;

  //
```

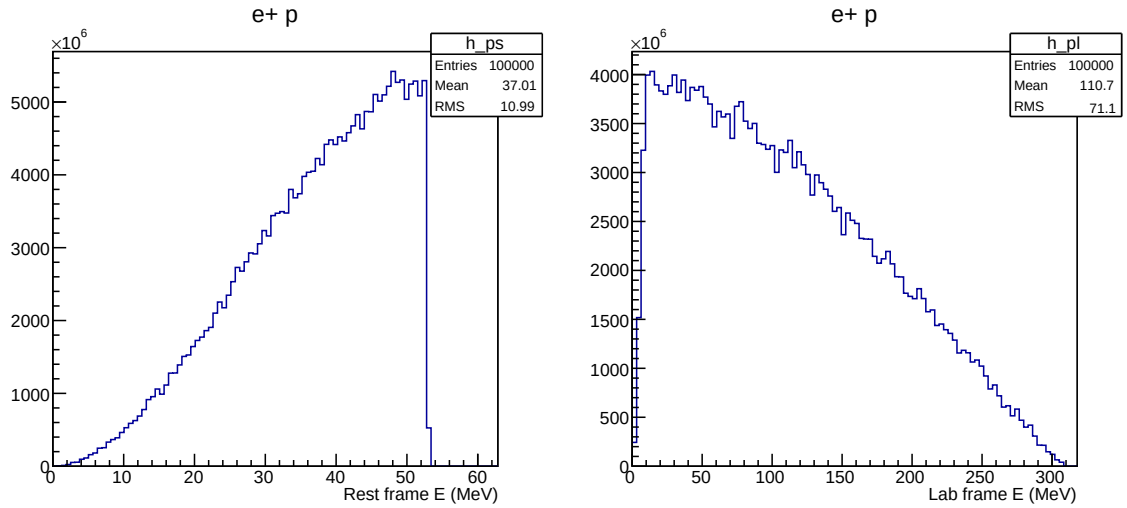

Figure 2: 왼쪽은 정지계, 오른쪽은 뮤온이 300 MeV/c로 움직이는 실험실계에서의 양전자의 에너지 분포를 보여준다.

```
  // loop over events
  //
  for (Int_t i=0;i<ngen;i++)
  {
    mdec(mass, &w, p_el, p_nue, p_num);
    h_ps->Fill(p_el[0],w);

    //
    // boost to lab frame
    //
    boost(m_mu,pmuon, p_el);
    h_pl->Fill(p_el[0],w);

  }
  c_c->cd(1); h_ps->Draw();
  h_ps->SetXTitle("Rest frame E (MeV)");
  c_c->cd(2); h_pl->Draw();
  h_pl->SetXTitle("Lab frame E (MeV)");
}
```

으로 만들 수 있다. 위의 프로그램은 뮤온이 300 MeV/c 의 운동량으로 움직이는 실험실계에서 양전자의 에너지 분포도 계산해 주고 있다. 이 300 MeV/c^2는 일본 J-PARC g-2/EDM 실험에서 사용 예정인 뮤온 운동량 값이다. 그림 2에서 그 결과를 볼 수 있다.