Introdução à Física Nuclear
Simulação da Cascata Intranuclear e Determinação da Curva de Fermi

10 de outubro de 2012

Classes de base no código CRISP (pasta base):

- NucleusDynamics.hh
  - Z, A, nucleons (vetor)...
  - GetA(), GetMass(), ChangeNucleon(Int_t index, ParticleDynamics& newParticle)...
- FermiLevels.hh
  - _fermiMomentum, _fermiEnergy, max[30] ...
  - GetFermiEnergy(), GetMaxLevel(int i) ...

Código auxiliar (pasta helpers):

- cascade.cc
- mcef_util.cc

Scripts de usuário (pasta scripts):

- script_cascade_mod.cc
- mcef_fission_proton.cc

Em que consiste o trabalho:

- Parte 1:
  - Criar um núcleo
  - Acessar as características atribuídas ao núcleo (massa, níveis de Fermi de prótons e neutrons, energias de Fermi)
- Parte 2:
  - Executar a cascata intranuclear para obtenção da curva de Fermi do núcleo no estado excitado $n(E)$
    - Núcleos: $^{66}_{30}\mathrm{Zn}$, $^{198}_{84}\mathrm{Po}$
    - Energias: 200 MeV (fóton), 500 MeV (próton)

Arquivos necessários ao trabalho:

- base/NucleusDynamics.hh (.cc)
- base/FermiLevels.hh (.cc)
- helpers/cascade.cc
- helpers/photo_abs.cc
- scripts/script_cascade_mod.cc

script_cascade_mod.cc:

```
{
   gROOT->Reset();
   gROOT->ProcessLine(".L helpers/photo_abs.cc+");
   gROOT->ProcessLine(".L helpers/cascade.cc+");

   gRandom->SetSeed(1953341347);

   Int_t A = 208,
         Z = 82;

   TString nucleusName = "Pb208";
   TString prefix = "results";

   Double_t photon_energy = 1000.;

   Int_t number_of_times_run = 1;

   TString fname = prefix + "/" + nucleusName + ".txt";

   // Nucleus and MesonsPool creation

   NucleusDynamics*  nuc = new NucleusDynamics(A, Z);
   MesonsPool*    mpool = new MesonsPool;

   //
   // Executar a cascata : nucleo, mpool, energia do foton,
   //                      numero de vezes
   //                      nome do arquivo
   //

   TString completeFileName = Form(fname.Data(), (int)photon_energy);

   execute_cascade( nuc, mpool, photon_energy, number_of_times_run, completeFileName.Data(), "proton" );

   delete nuc;
   delete mpool;

   exit(1);
}
```

cascade.cc:

```
35 void execute_cascade( NucleusDynamics* nuc,
36                       MesonsPool* mpool,
37                       Double_t photon_energy,
38                       Int_t N,
39                       char* outFile,
40                       const char* projectile_opt = "photon"){
41
42 //Definição dos canais de interação
43    TString stropt(projectile_opt);
44    EventGen* pev = 0;
45
46    if ( stropt.CompareTo("photon") == 0 ) {
47       pev = new PhotonEventGen();
48       init_photon_evt_gen((PhotonEventGen*)pev);
49    }
50    else if ( stropt.CompareTo("proton") == 0 ) {
51       pev = new ProtonEventGen();
52       init_proton_evt_gen((ProtonEventGen*)pev);
53     }
54
55    Double_t ex = photon_energy;
56    ofstream *ofs = 0;
57
58 //Arquivo genérico de saída - a ser usado para guardar qualquer informação necessária
59    ofstream *output = new ofstream(outFile);
60
61 //Backup A, Z ...
62    Int_t    A = nuc->GetA(),
63             Z = nuc->GetZ();
64
65 //Usado quando necessário na linha 80  --- pev->Generate( ex, *nuc, *mpool, 0, out) ---
66 //para registrar os canais de interação
67    //TString channels = Form("./results/%d_%d_%.0f_%s_channels.txt", Z, A, photon_energy, projectile_opt );
68    //ofstream *out = new ofstream(channels.Data());
69
70    //*output << "#i: energy num_counts ex_energy A Z" << std::endl;
71
72    for ( int i = 0; i < N; i++ ) {
73
74       std::cout << Form ("#%d",i) << std::endl;
75
76       Cascade *c = 0;
77       Int_t idx = -1;
78       Int_t counts = 0;
```

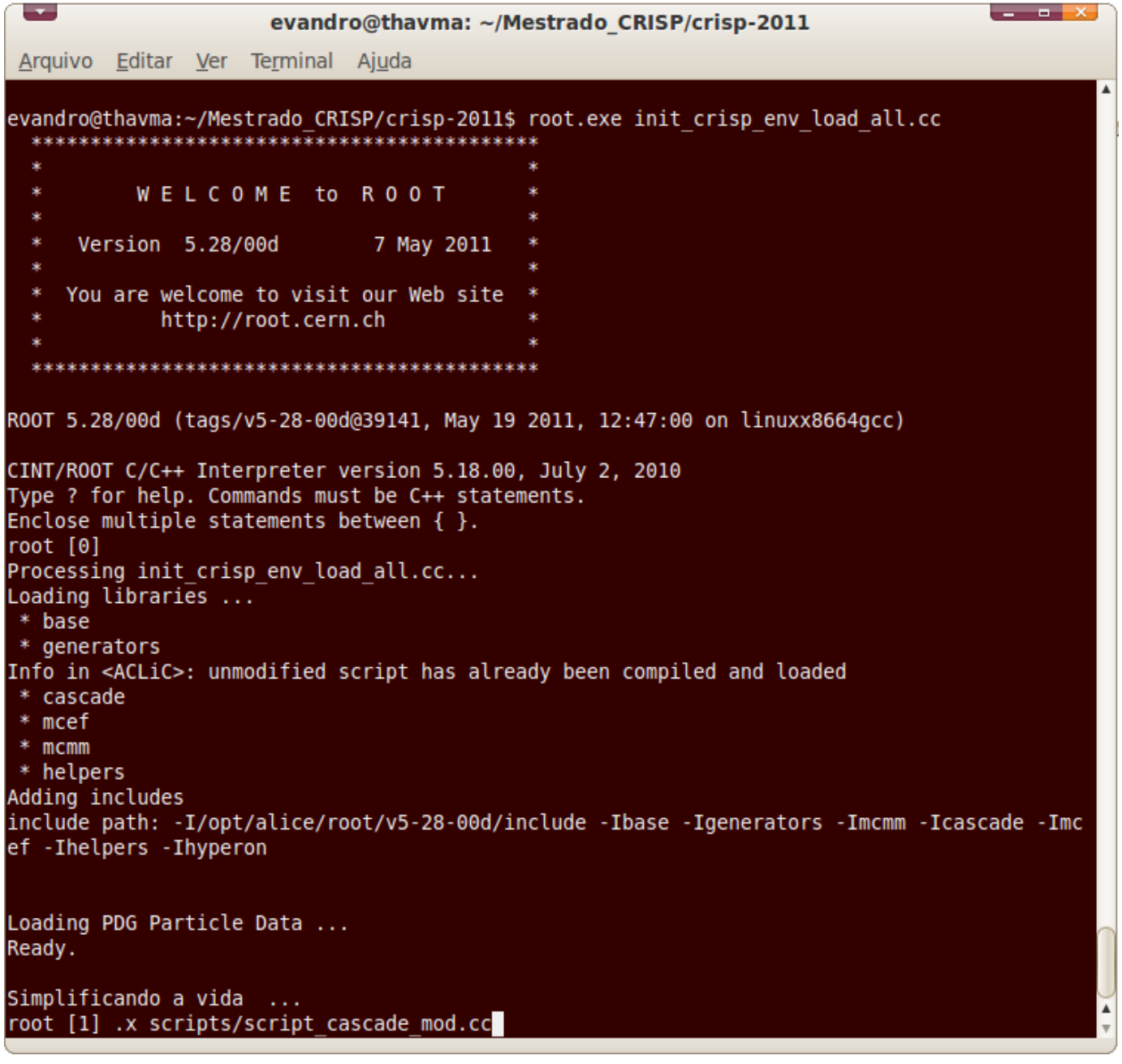
cascade.cc:

```
80      try {
81
82         Int_t a,
83               z;
84
85         Double_t ex_energy = 0.;
86
87         while( (idx = pev->Generate( ex, *nuc, *mpool ) ) < 0 ) {
88            counts++;
89            nuc->DoInitConfig(A, Z);
90         }
91
92         TString fname = Form("/tmp/logs/__nucxxx_%d.log",i);
93         ofs = new ofstream();
94         ofs->open(fname.Data(), std::ofstream::out | std::ofstream::app);
95
96         std::cout << "Initilizing intra-nuclear cascade" << std::endl;
97         *ofs << Form ("#%d",i) << std::endl;
98
99         c = new Cascade(*nuc, *mpool, ex, ofs);
100
101        ex_energy = c->Execute(*nuc, *mpool);
102
103        std::cout << Form("Total blocked %d", c->TotalBlocked()) << std::endl;
104
105        c->FinalNucleus(a, z, *nuc);
106        Double_t te = c->ElapsedTime();
107
108        std::cout << Form ("Final[%d]: Ex_energy = %f, t = %f, A = %d Z = %d", i, ex_energy, te, a, z) << std::endl;
109
110        *ofs << Form ("Final[%d]: Ex_energy = %f, t = %f, A = %d Z = %d", i, ex_energy, te, a, z) << std::endl;
111
112        //*output << Form ("%d: %f %d %f %d %d", i, photon_energy, counts + 1, ex_energy, a, z) << std::endl;
113
114        nuc->DoInitConfig(A, Z);
115        mpool->clear();
116
117        delete c;
118        delete ofs;
119
120        c = 0;
121        ofs = 0;
122     }
```

cascade.cc:

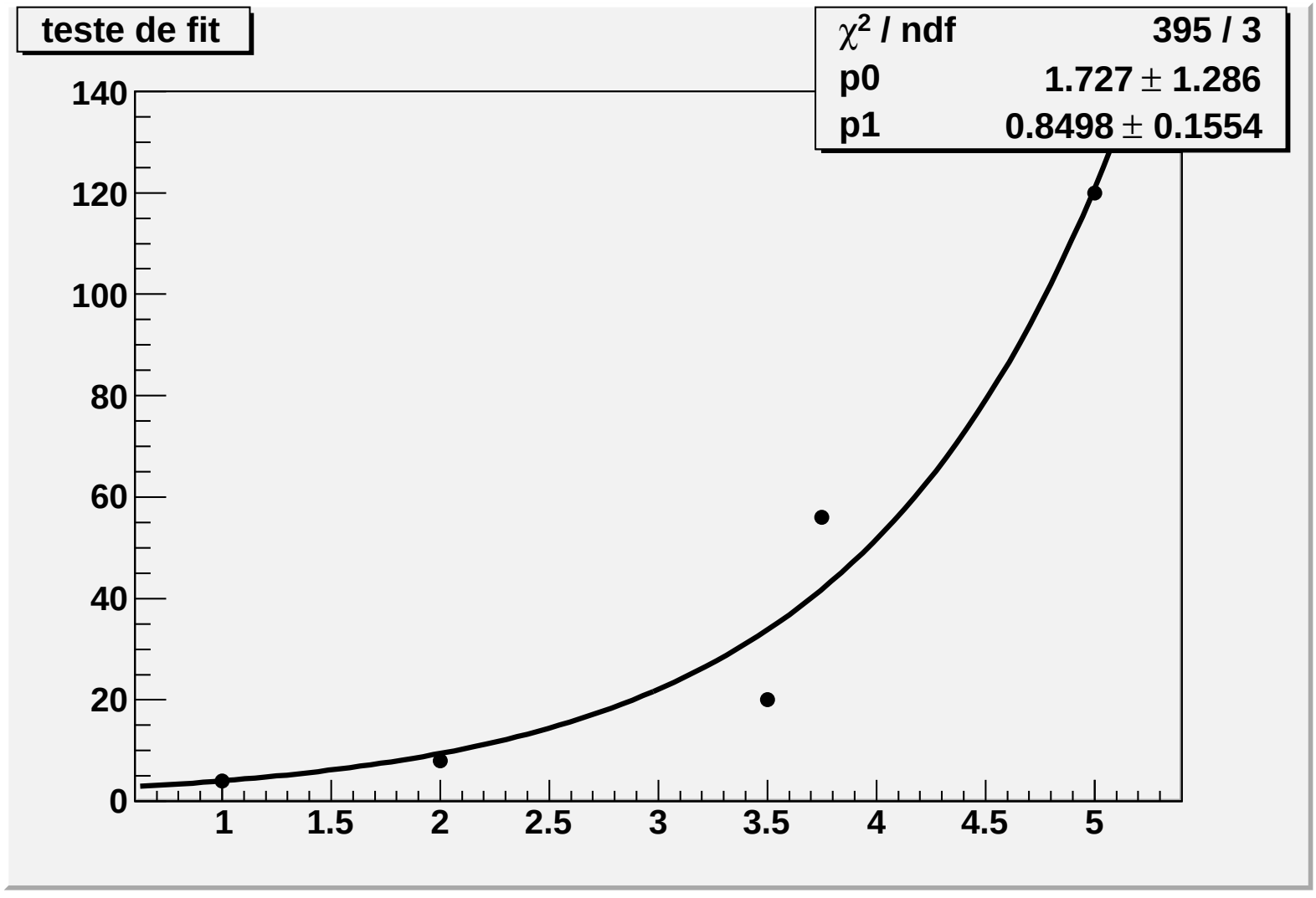
```
123       catch (BasicException e){
124
125          std::cout << "catched ..." << std::endl;
126          e.PrintOn();
127
128          nuc->DoInitConfig(A, Z);
129          mpool->clear();
130
131          std::cout << "NucleusDynamics, MesonsPool - restarted" << std::endl;
132
133          // verificar se o operador new realmente retornou um ponteiro para c.
134          if (c != 0 && !c->IsZombie() ) {
135             std::cout << "[INFO] Deleting Cascade" << std::endl;
136             delete c;
137          }
138
139          std::cout << "[INFO] Deleting ofs" << std::endl;
140          delete ofs;
141          i--;
142       }
143    }
144    delete pev;
145    delete output;
146    std::cout << "Finished." << std::endl;
147 }
```

Como executar um script do CRISP:



Fazendo ajuste diretamente de um gráfico:

```
1
2  void fit_Graph(){
3
4     const int n = 5;
5     double x[n] = {1., 2., 3.5, 3.75, 5.};
6     double y[n] = {4., 8., 20., 56, 120.};
7
8     TGraph *gr = new TGraph(n,x,y);
9     gr->SetMarkerStyle(20);
10
11    gr->SetTitle("teste de fit");
12    gr->Draw("AP");
13
14    TF1 *f = new TF1("f","[0]*exp([1]*x)",0.,10.);
15    f->SetParameters(2., 1.5);
16
17    gr->Fit(f,"R");
18
19    gStyle->SetOptFit();
20
21 }
```


teste de fit
χ² / ndf 395 / 3
p0 1.727 ± 1.286
p1 0.8498 ± 0.1554
140
120
100
80
60
40
20
0
1
1.5
2
2.5
3
3.5
4
4.5
5