Parametrização da resposta fotoelétrica do Módulo de Superficie do Observatório Pierre Auger

10 de Novembro de 2011

Aluno: João Pinheiro Neto[1]
Orientador: José Augusto Chinellato[2]
Co-Orientador: Márcio Aparecido Muller[3]

.

Resumo

O presente projeto visa parametrizar o sinal fotoelétrico do detector de superfície do Observatório Pierre Auger. Consideramos aqui um novo setup do módulo de superfície, que introduz uma camada de solo ao redor do módulo. Utilizando um software ("tank0Mod") baseado no toolkit geant4, simulamos a resposta do detector para passagem de particulas individuais. Analisamos os dados resultantes utilizando uma série de scripts, obtendo curvas de resposta do detector. Implementamos essas curvas na forma de uma "função resposta", escrita em C. Esta é capaz de simular a resposta do tanque à passagem de partículas em qualquer energia e ângulo zenital dentro dos limites simulados com o tank0Mod. A função resposta permite que dados sejam adicionados e removidos do seu banco sem necessidade de mudança no código, tornando mais simples melhorias futuras na parametrização do sinal.

[1]Email: joaopn at ifi.unicamp.br
[2]Email: chinella at ifi.unicamp.br
[3]Email: mamuller at ifi.unicamp.br

1 Introdução

A Terra é constantemente bombardeada por partículas subatômicas (ou núcleos atômicos) de origem cósmica, os chamados raios cósmicos. O estudo dos raios cósmicos de altíssima energia (UHECR) tem atraído muita atenção, e tem como objetivo principal a caracterização da origem de tais partículas. Ao colidir com a atmosfera, o raio cósmico (chamado de primário) inicia uma cascata de partículas e radiação, chamado de chuveiro atmosférico extenso (EAS), que pode ter na ordem de 10^{10} partículas[1] no seu máximo. Os UHECR tem energia da ordem de 10^{20} eV, e são as partículas mais energéticas do universo conhecido.

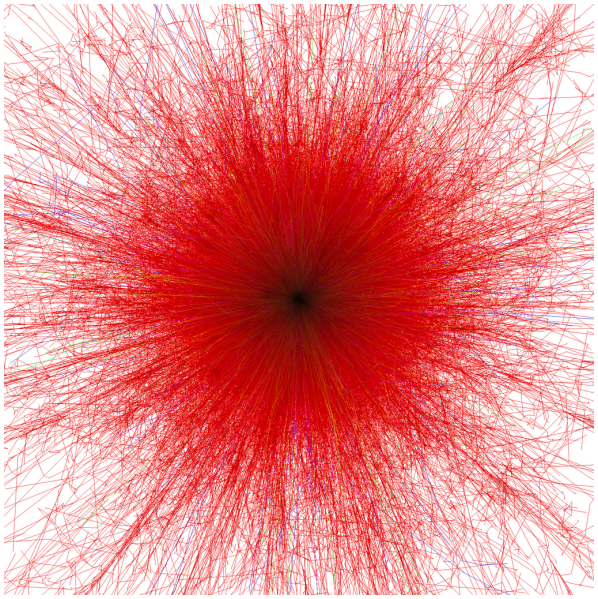


Figura 1: Simulação de chuveiro iniciado por um núcleo de Fe de 10^{13}eV, visto de baixo. Créditos: F. Schmidt, "CORSIKA Shower Images"

O estudo dos raios cósmicos de mais alta energia e chuveiros resultantes é o propósito principal do Observatório Pierre Auger (OPA), que irá operar em dois sítios (na Argentina e em local a ser determinado no hemisfério norte), de forma a contemplar todo o céu. O Observatório utiliza uma técnica híbrida de detecção: o Detector de Fluorescência e o Detector de Superfície. O Detector de Fluorescência utiliza um esquema com telescópios para observar a luz de fluorescência decorrente do chuveiro, e com isso observar o seu desenvolvimento longitudinal. O Detector de Superfície é responsável pela detecção das partículas que chegam ao solo, e é formado por tanques de água deionizada (chamados módulos de superfície), cada um contendo 3 fotomultiplicadoras, painéis solares, antenas de comunicação e eletrônica de suporte.

As partículas do chuveiros que chegam ao tanque interagem com a água no interior do tanque e produzem luz Cherenkov, que será detectada pelas fotomultiplicadoras.

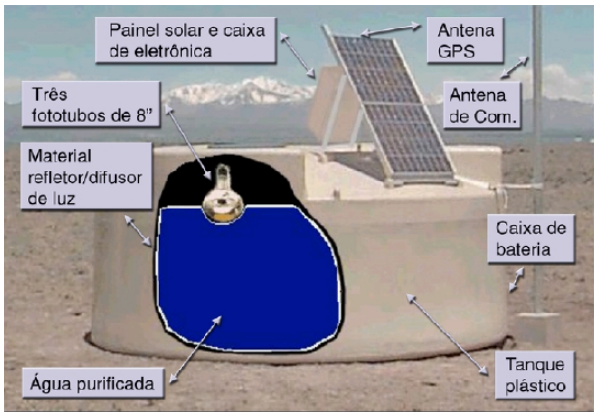


Figura 2: Esquema de um módulo de superfície do sítio Sul do OPA.

Atualmente o sítio Sul (Argentina), em pleno funcionamento, conta com 1600 detectores de superfície espalhados por uma área de $3000\,km^2$. Dada a sua escala, o OPA já acumulou uma exposição total superior a todos os outros experimentos de chuveiros atmosféricos reunidos.[2]

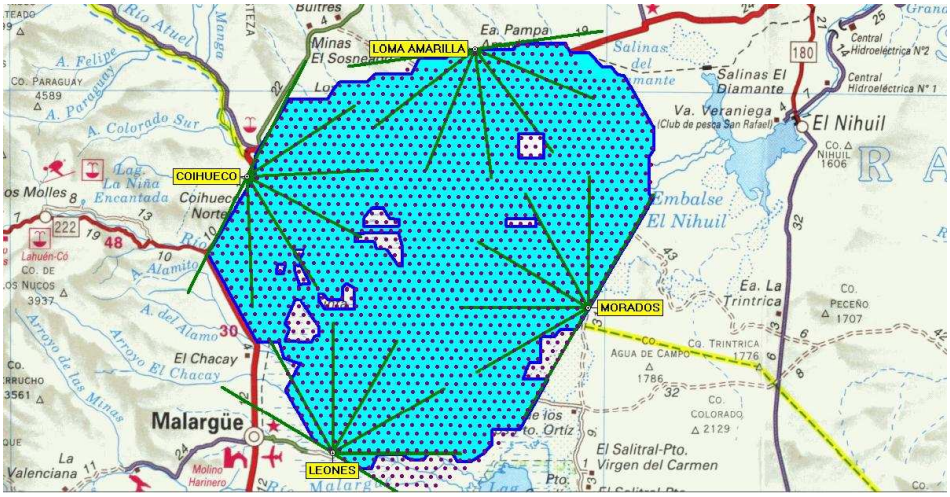


Figura 3: Esquema mostrando o sítio Sul. Nas extremidades do arranjo de tanques, temos os telescópios de fluorescência. Em azul são mostrados os tanques em funcionamento.

2 Simulação da interação de partículas com o módulo de superfície

O objetivo deste trabalho é contribuir para uma parametrização da resposta do detector de superfície a partículas individuais. Para isso, utilizamos simulações numéricas por Monte Carlo da interação de tais partículas com o detector. Este procedimento foi iniciado por M. A. Muller[3], atualmente pós-doutorando do IFGW. A motivação é redução do tempo computacional da simulação do sinal de um EAS, que pode levar meses em função da sua grande cadeia de interações.

Foi utilizado o toolkit Geant4[4][10], em cima do qual foi desenvolvida por M.A. Muller a simulação tank0Mod[3], que simula o detector incluindo o solo ao seu redor. O tank0Mod injeta partículas

[4]Ele é considerado o "estado da arte" da simulação de interação de partículas, sendo muito usado tanto em física de altas energias como em física médica.

aleatoriamente em uma área de 36 m^2 centrada no delector, e na altura do topo deste. A simulação utiliza 3 variáveis de entrada: tipo de partícula, ângulo zenital de entrada e energia da partícula. Para cada partícula/energia simulamos em média3000 vezes, de forma a obter uma boa estatística. Abaixo temos um esquema do módulo utilizado:

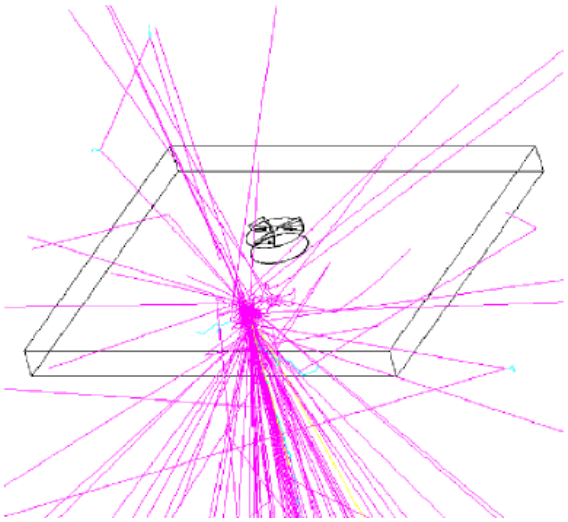
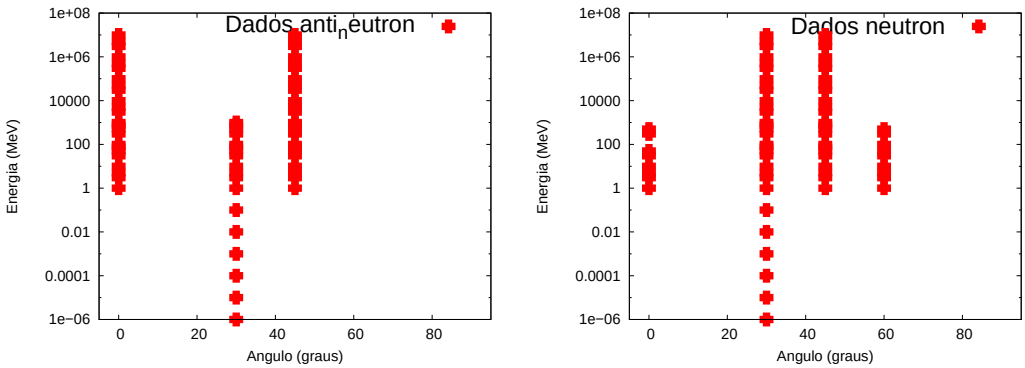


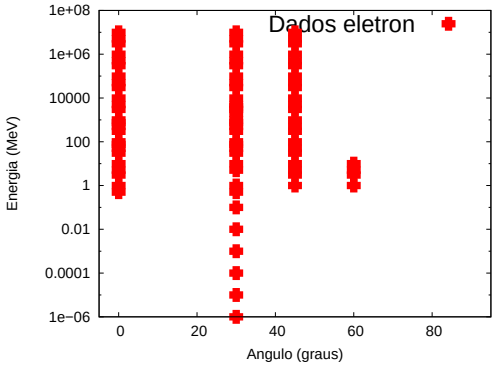
Figura 4: Simulação do módulo de superfície, com o módulo no centro.

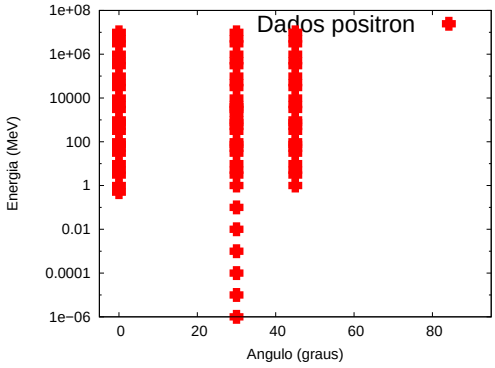
Foram feitas simulações a valores discretos de energia para os tipos de partículas mais relevantes de um EAS para obtermos uma parametrização geral do sinal que permita fazer interpolações para qualquer valor de energia. Estamos cobrindo ângulos zenitais na faixa $\theta \in [0, 90^o]$, para 12 tipos de partículas. O intervalo original das energias simuladas foi na faixa eV até 10 TeV.

Logo ficou claro que o tempo computacional de cada simulação depende fortemente da energia da partícula incidente. Devido à longa cadeia de interaçoes, a simulação de uma partícula de alguns TeV pode levar até 20 000 vezes mais tempo que a simulação da mesma partícula com energia da ordem de MeV. Sendo assim, com o objetivo de preencher ao máximo o espaço de parâmetros (energia, ângulo de entrada e tipo de partícula), resolvemos priorizar a simulação de particulas de baixa energia, que são as mais comuns ao nível do solo. Para a simulação, utilizamos o cluster SGI Altix 1350/450, administrado pelo Cenapad-SP[12]. Além das simulações recentes, temos dados obtidos anteriormente por M.A. Muller[3]. Temos abaixo mapas do conjunto de dados simulados até o momento, para cada partícula:

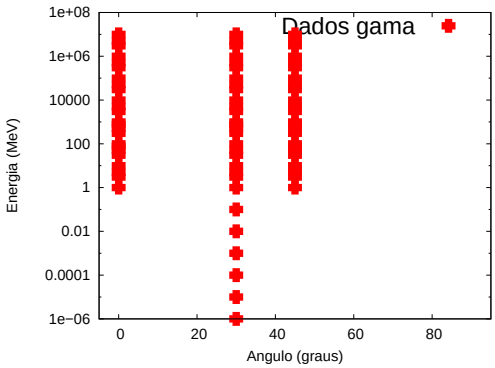


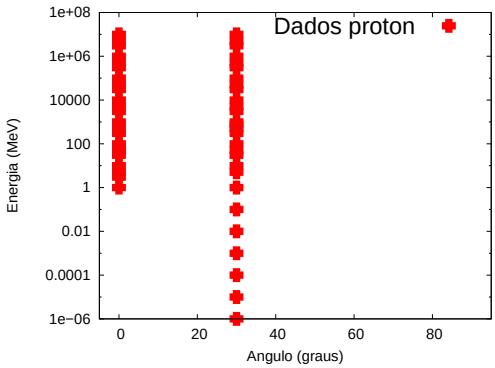
Figuras 5 e 6: Mapas de dados simulados das partículas $\bar{n}$ e n, respectivamente.



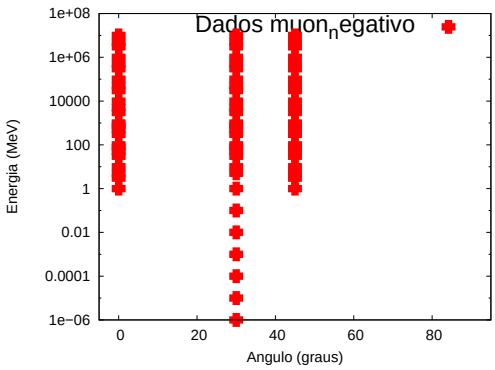


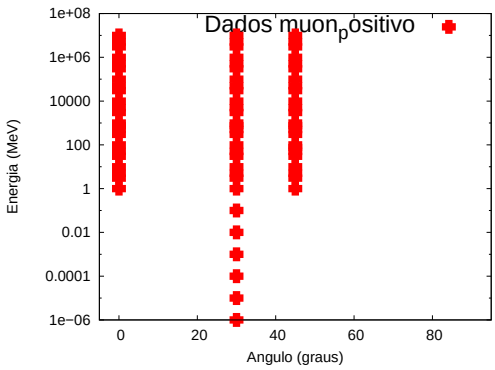
Figuras 7 e 8: Mapas de dados simulados das partículas e^- e e^+, respectivamente.



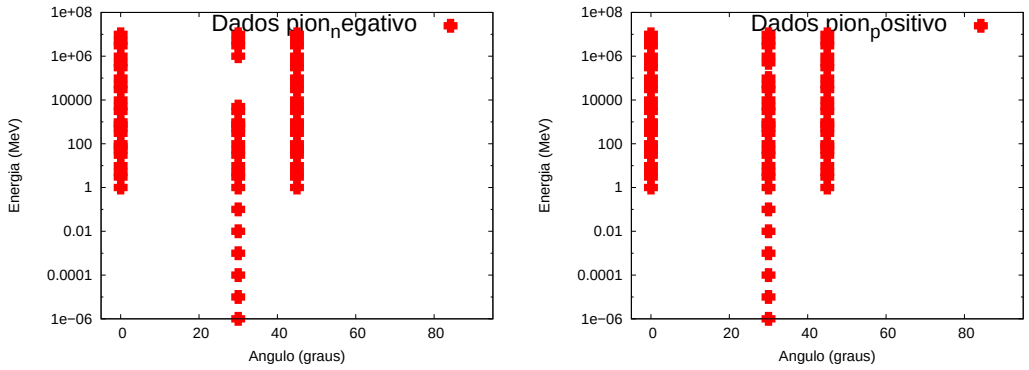


Figuras 9 e 10: Mapas de dados simulados de raios γ e prótons, respectivamente.





Figuras 11 e 12: Mapas de dados simulados das partículas μ^- e μ^+, respectivamente.



Figuras 13 e 14: Mapas de dados simulados das partículas π^- e π^+, respectivamente.

Foram escritos scripts para gerar automaticamente os mapas, bem como para facilitar o processo de submeter simulações ao cluster.

3 Parametrização

3.1 Distribuição de fotoelétrons

Do resultado da simulação, estamos interessados no nº de fotoelétrons detectados pela eletrônica do módulo. Ao injetarmos a partícula no tanque, temos os seguintes casos para considerar:

1. Ela não gera fotoelétrons na eletrônica. Isso pode ser em função de não ter interagido com o setup[5], ou não ter gerado fotoelétrons detectáveis pela eletrônica.

2. Ela gera n fotoelétrons detectados na eletrônica. Sendo o processo estocástico, o nº segue uma certa distribuição de probabilidade.

O primeiro passo é realizar algum tipo de ajuste da distribuição de fotoelétrons detectados. Escolhemos para isso uma soma de uma curva exponencial $f_{expo}(n)$ e outra gaussiana $f_{gauss}(n)$, $f(n) = f_{expo}(n) + f_{gauss}(n)$. Elas foram definidas da seguinte forma:

$$f_{expo}(n) = \exp(\alpha + \beta n) \tag{1}$$

$$f_{gauss}(n) = \gamma \exp\left(-\frac{(n-\mu)^2}{2\sigma^2}\right) \tag{2}$$

Temos, portanto, um conjunto de 2 parâmetros da exponencial (α, β) e 3 da gaussiana (γ, μ, σ), além dos limites das curvas, que caracterizam a distribuição de fotoelétrons da partícula de tipo, energia e ângulo definidos. Assim, obtendo esses 5 parâmetros para uma partícula de determinada energia e ângulo de entrada, podemos montar sua distribuição de fotoelétrons e utilizar isso para simular a resposta do tanque várias vezes mais rápido que se utilizarmos o tank0Mod.

Utilizamos para isso uma interpolação linear dos parâmetros em função da energia e ângulo da partícula, para cada parâmetro. A interpolação é feita entre os valores superior e inferior mais próximos do parâmetro requerido. Por exemplo, para obter o valor interpolado de β(50 MeV,60º) (parâmetro β para uma partícula de 50 MeVcom ângulo zenital de 60º), fazemos

$$\beta_{interp}(50\text{ MeV},60^{\text{o}}) = (\beta_{sim}(60\text{ MeV},60^{\text{o}})\text{-}\beta_{sim}(40\text{ MeV},60^{\text{o}}))\,\frac{50\text{ MeV} - 40\text{ MeV}}{60\text{ MeV} - 40\text{ MeV}} + \beta_{sim}(40\text{ MeV},60^{\text{o}}) \tag{3}$$

onde β_{sim} são valores simulados com o tank0Mod. No exemplo acima, temos simulados β da partícula para 60 MeVe 40 MeV, para o ângulo de 60º.

[5] Comum principalmente em particulas de baixa energia.

Para simular a resposta parametrizada, levamos em conta a probabilidade da partícula interagir, tomando essa como o nº de simulações com detecção de fotoelétrons dividido pelo nº de simulações realizadas. Essa probabilidade então é interpolada entre as energias e ângulos da mesma forma que os parâmetros $(\alpha, \beta, \gamma, \mu, \sigma)$.

3.2 Área das curvas

Sendo a e b respectivamente os limites superior e inferior de $f_{gauss}(n)$ e de $f_{expo}(n)$, temos

$$A_{expo} = \int_a^b e^{\alpha} e^{\beta n} dn \tag{4}$$

$$A_{expo} = \frac{e^{\alpha}}{\beta} \left(e^{\beta b} - e^{\beta a} \right) \tag{5}$$

A área da curva gaussianaa A_{gauss} envolve o cálculo de de uma integral do tipo $\int e^{-x^2} dx$, que é imprópria. Felizmente, a biblioteca math.h possui a função **erf**, definida como

$$\text{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt \tag{6}$$

Assim, é interessante escrever A_{gauss} em função de erf. Temos então

$$A_{gauss} = \int_a^b \gamma \text{exp} - \left(\frac{n - \mu}{\sqrt{2}\sigma} \right)^2 dn \tag{7}$$

realizando a troca de variáveis $\frac{n-\mu}{\sqrt{2}\sigma} \to t$, temos

$$A_{gauss} = \gamma \sigma \sqrt{2} \int_{a'}^{b'} e^{-t^2} dt \tag{8}$$

onde $b' = \frac{b-\mu}{\sqrt{2}\sigma}$ e $a' = \frac{a-\mu}{\sqrt{2}\sigma}$. Em termos de erf, temos por fim

$$A_{gauss} = \sqrt{\frac{\pi}{2}} \gamma \sigma \left[\text{erf} \left(\frac{b - \mu}{\sqrt{2}\sigma} \right) - \text{erf} \left(\frac{a - \mu}{\sqrt{2}\sigma} \right) \right] \tag{9}$$

3.3 Simulação das distribuições

O método da inversa, também chamado de transformada de Smirnov, é utilizado para gerar números aleatorios seguindo uma distribuição de probabilidade qualquer. Ele utiliza o fato que, dada uma distribuição $f(x)$, a sua função de distribuição acumulada $F(x)$ tem distribuição uniforme U[0,1][8]. O algoritmo pode ser descrito da seguinte forma:

1. Obtenha $F(x) = u$ e a inverta para obter $x = F^{-1}(u)$.
2. Gere u da distribuição U[0,1].
3. Calcule $x = F^{-1}(u)$.

Como se pode ver, o método da inversa é bastante simples, e sempre pode ser aplicado para distribuições contínuas. Ele é especialmente útil quando possuímos uma forma explícita para $F^{-1}(u)$, sendo caso contrário computacionalmente custoso.

3.4 Simulação de $f_{expo}(n)$ pelo método da inversa

Da forma como definimos $f_{expo}(n)$, temos a sua função acumulada F_{expo}:

$$F_{expo}(n) = \frac{1}{A_{expo}} \int_a^n e^{\alpha} e^{\beta n'} dn' \tag{10}$$

$$F_{expo}(n) = \frac{e^{\alpha}}{\beta A_{expo}} \left(e^{\beta n} - e^{\beta a}\right) = u \tag{11}$$

onde a é o limite inferior de $f_{expo}(n)$, e A_{expo} garante a normalização. Invertendo a última equação, temos

$$n = \frac{1}{\beta} \ln \left(e^{\beta a} + \frac{u \beta A_{expo}}{e^{\alpha}} \right) \tag{12}$$

que é a nossa variável simulada.

3.5 Simulação de $f_{gauss}(n)$ pelo método da inversa

Da definição de $f_{gauss}(n)$, a sua função acumulada $F_{gauss}(n)$ é dada por:

$$F_{gauss}(n) = \frac{1}{A_{gauss}} \int_a^n \gamma \exp - \left(\frac{n' - \mu}{\sqrt{2}\sigma} \right)^2 dn' \tag{13}$$

Utilizando procedimento similar ao cálculo de A_{gauss}, temos

$$F_{gauss}(n) = \sqrt{\frac{\pi}{2}} \frac{\gamma \sigma}{A_{gauss}} \left[\text{erf} \left(\frac{n - \mu}{\sqrt{2}\sigma} \right) - \text{erf} \left(\frac{a - \mu}{\sqrt{2}\sigma} \right) \right] = u \tag{14}$$

Ao contrário de $\mathbf{erf}(x)$, a função $\mathbf{erf}^{-1}(x)$ não possui implementação pronta na linguagem C. Assim, utilizaremos sua expansão de Maclaurin, truncada no termo de 11ª potência:

$$\text{erf}^{-1}(x) = \frac{\sqrt{\pi}}{2} \left(x + \frac{\pi}{12} x^3 + \frac{7\pi^2}{480} x^5 + \frac{127\pi^3}{40320} x^7 + \frac{4369\pi^4}{5806080} x^9 + \frac{34807\pi^5}{182476800} x^{11} \right) \tag{15}$$

Invertendo $F_{gauss}(n)$, temos então

$$n = \mu + \sqrt{2}\sigma \text{erf}^{-1} \left(\sqrt{\frac{2}{\pi}} \frac{u A_{gauss}}{\gamma \sigma} + \text{erf} \left(\frac{a - \mu}{\sqrt{2}\sigma} \right) \right) \tag{16}$$

4 Automatização da parametrização

4.1 Scripts de análise

Em função da quantidade enorme de dados a serem analisados, escolhemos escrever um conjunto de scripts capaz de automatizar a análise de dados. A linguagem escolhida foi Shell Script (mais especificamente, com o interpretador bash), em função da sua facilidade de manipulação de arquivos de texto. Por exemplo, a substituição de uma string de texto em um arquivo pode ser feita com a linha "sed s/texto_antigo/texto_novo arquivo". Tal manipulação seria consideravelmente mais complexa se feita na linguagem C++, por exemplo.

Os scrips pegam os dados de contagem de fotoelétrons do detector[6] e encontram a função de probabilidade de interação (muitas partículas atravessam o detector sem interagir) e evolução com a energia dos parâmetros das curvas ajustas. O algoritmo utilizado para os ajustes das distribuiçoes

[6] Este sera o único output da simulação, pois não estamos analisando a distribuição temporal do sinal.

de fotoelétrons é simples: manda-se o software ROOT[11] gerar e fazer o ajuste de diversos histogramas com números variados de bins e gaussianas e exponenciais de limites variados e diferentes. Os parâmetros obtidos nesses ajustes servem de parâmetros iniciais para o software realizar o ajuste da curva $f(n) = f_{expo}(n) + f_{gauss}(n)$. Posteriormente, escolhe-se entre as curvas com diversos parâmetros iniciais a com menor χ^2 como a melhor. Decidimos tornar os scripts modulares, atacando uma parte do processo de análise por vez.

Abaixo temos um esquema do funcionamento dos scrips:

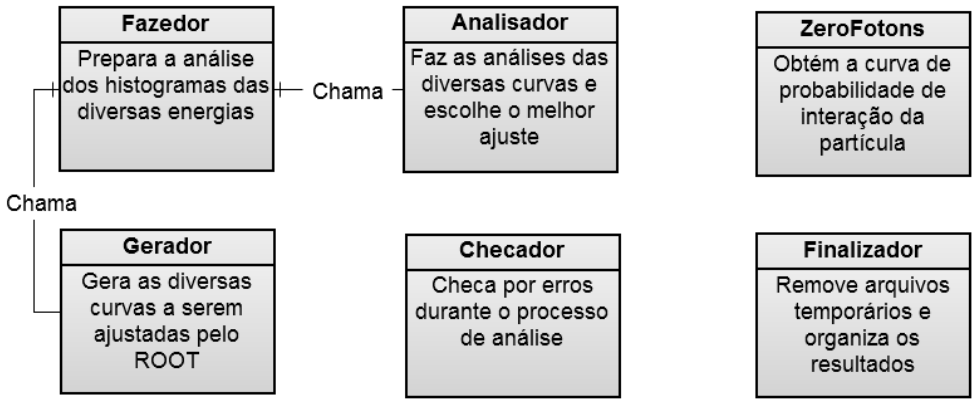


Figura 15: Esquema de funcionamentos dos scripts.

Em muitos casos a seção de choque da particula com o detector é tão pequena que não há uma real distribuição de probabilidade de fotoelétrons, e o ajuste da curva não converge. Nesses casos o ROOT pode travar de diversas formas, e o script teve que ser escrito robusto o suficiente para lidar com todas as formas com que a análise pode falhar.

Posteriormente foi escrito o script ZeroFotons, que obtém a probabilidade de interação da particula. Este ofereceu o seu conjunto de desafios, como por exemplo a inabilidade do Shell Script de trabalhar com números de ponto flutuante. O script Checador remove possiveis erros de análise dos arquivos de saída, colocando os arquivos em uma pasta separada para análise manual. Por fim o script Finalizador remove arquivos temporários e organiza os resultados da análise. Estes são armazenados em arquivos contendo os 5 parâmetros $(\alpha, \beta, \gamma, \mu, \sigma)$ das curvas, ponto final do ajuste das curvas b (o ponto inicial é fixado em $a = 1$) e a probabilidade de interação.

4.2 Resultados da análise

Utilizando o software de automatização escrito, parametrizamos a resposta do tanque nesses parâmetros. Abaixo temos exemplos de curvas de fotoelétrons geradas, onde o ajuste inicial é dado pelas curvas em preto e o final pela curva vermelha.

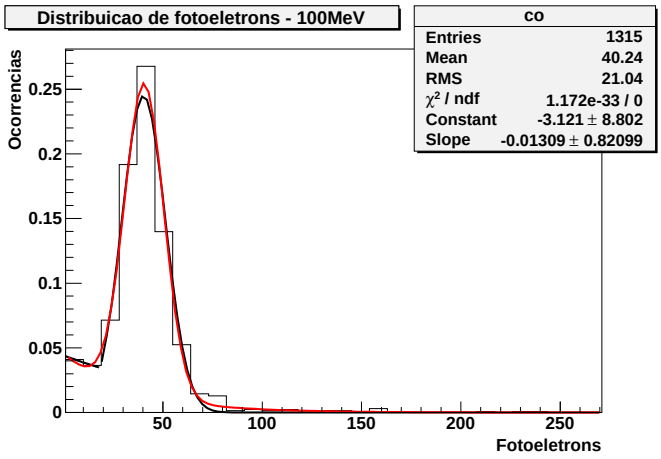


Figura 16: Distribuição de fotoelétrons do pósitron de 100 MeV com $\theta = 60^{o}$.

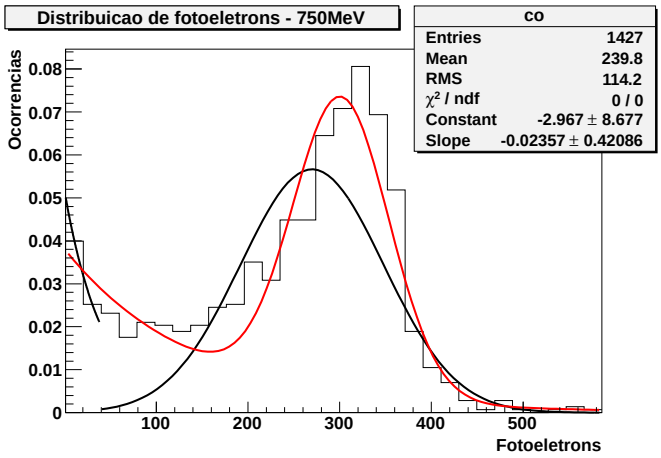


Figura 17: Distribuição de fotoelétrons do pósitron de 750 MeV com $\theta = 60^{o}$.

Abaixo temos exemplo de uma curva de probabilidade de interação. A curva ajustando os dados não é utilizada na análise.

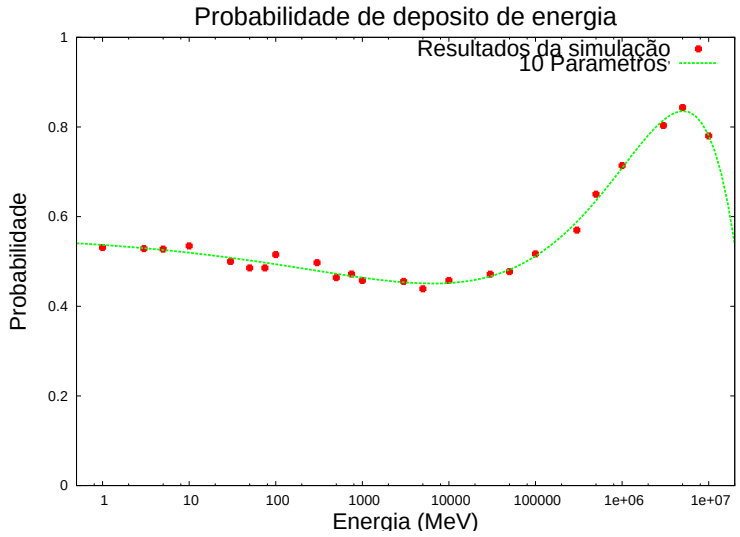


Figura 18: Probabilidade de interação do anti-neutron com $\theta = 45^{o}$.

5 Função Resposta

A análise feita através do conjunto de scripts descritos acima não é útil a menos que a parametrização possa ser implementada na forma de um software. Para esse caso uma linguagem interpretada como o Shell Script é demasiadamente lenta, removendo a utilidade da parametrização. Além disso, é de difícil integração em qualquer software mais complexo. Assim resolvemos implementar a parametrização na forma de uma função, escrita em C. Para essa implementação damos o nome de "função resposta".

No espírito dos scripts de análise, a função resposta foi escrita modularmente. Ela utiliza as técnicas descritas na sec. 3 para simular a interação do tanque com partículas individuais. Sendo θ o ângulo zenital e E a energia da particula primária, os módulos envolvidos são:

- **FResposta** (partícula, θ, E): módulo principal, devolve o n^{o} de fotoelétrons parametrizado para uma partícula individual.
- **Par_interp** (θ, E): Interpola os valores dos 5 parâmetros + probabilidade de interação + limite superior do fit entre valores de ângulo e energia simulados.
- **Get_par**(θ, E): Obtém os 5 parâmetros + probabilidade de interação + limite superior simulados com o tank0Mod para θ e E dados.
- **findEnergy**(θ, E): Encontra as energias entre as quais E está localizada, para um dado θ.
- **area_curvas** (α, β,γ, μ, σ) : Calcula a área embaixo das curvas gaussiana e exponencial.
- **sim_gaus**(γ, μ, σ) : Simula uma distribuições gaussiana seguindo o método descrito na sec. 3.5.
- **sim_expo** (α, β) : Simula uma distribuições exponencial seguindo o método descrito na sec. 3.4.

As subfunções acima são chamadas uma a uma pela FResposta de forma a obter os parâmetros simulados, interpolá-los e simular o sorteio do n^{o} de fotoelétrons utilizando os parâmetros resultantes.

A função resposta utiliza arquivos de configuração indicando, para cada particula, quais θ e E foram simulados. A interpolação e portanto a obtenção dos parâmetros (α, β,γ, μ, σ) é feita durante

a execução da função, a partir dos dados presentes da simulação do tank0Mod. Assim, é possivel complementar continuamente a parametrização sem precisar reescrever a função resposta. Basta rodar as simulações necessárias com o tank0Mod e modificar os arquivos de configuração para incluir as novas simulações.

6 Conclusões

No presente projeto desenvolvemos uma função capaz de simular a resposta do módulo de superfície do Observatório Pierre Auger a partículas individuais. Sendo uma parametrização, o trabalho nunca pode ser dito "terminado" - sempre é possivel adicionar mais dados para melhorar a simulação. Tivemos isso em mente ao escrever a função, tornando a adição de mais dados posteriores uma tarefa simples. A função resposta pode agora ser utilizada para calcular o depósito de energia de um chuveiro atmosférico, aumentando consideravelmente a velocidade desse processo. Da forma como foi escrita, ela não depende dos detalhes internos do software tank0Mod, utilizado para criar as simulações. Caso se descubra futuramente algum problema com o tank0Mod, basta remover os dados antigos e substituí-los por outros.

Comentário do orientador

> O aluno está obtendo resultados importantes para a caracterização dos módulos de superfície do Observatório Auger. Com os resultados, teremos completado um conjunto de curvas resposta para a passagem das diferentes partículas que compõem a frente de chuveiros atmosféricos extensos a energias acima de 10^{18}eV. Seu desempenho foi muito bom.

Referências

[1] Góra, D. *et al.*, *Astropart. Phys.* **16** 129 (2001)

[2] Roulet, E. "Latest results from the Pierre Auger Observatory". arXiv:1101.1825v1 [astro-ph.HE] (2011)

[3] Müller, M. A. *Estudo sobre as Interações de Hádrons nos Módulos de Superfície e Adjacências, do Observatório Pierre Auger*, Tese de Doutorado, IFGW-Unicamp (2007)

[4] Pinheiro Neto, J. *Parametrização de chuveiros atmosféricos inclinados do Observatório Pierre Auger,* Relatório de Iniciação Científica, IFGW/Unicamp (2010). Disponível em http://www.ifi.unicamp.br/~lunazzi/F530_F590_F690_F809_F895/F530_F590_F690_F895/F530_F590_F690_F895_2011_sem1/JoaoP-Chinellato_RF1_F590.pdf

[5] Burtch, K. *Linux Shell Scripting with Bash.* Sams Publishing, Indiana (2004)

[6] Kochan, S. Programming in C. Sams Publishing, Indiana (2004)

[7] Rossi, B. *High-Energy Particles.* New York: Prentice-Hall (1952)

[8] Devroye, L. *Non-Uniform Random Variate Generation*, Springer-Verlag, New York (1986)

[9] James, B. Probabilidade: um curso em nível intermediário. 2. ed, IMPA, Rio de Janeiro (2002).

[10] Disponível em http://wwwinfo.cern.ch/asd/geant4/geant4.html (acessado em 10/11/11)

[11] Disponível em http://root.cern.ch/ (acessado em 10/11/11)

[12] http://www.cenapad.unicamp.br/ (acessado em 10/11/11)

ANEXO A: Função Resposta

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <time.h>
/* Definições
par[0]: alpha
par[1]: beta
par[2]: gamma
par[3]: mu
par[4]: sigma
par[5]: lim. inferior
par[6]: prob. de interacao
lim[0]: lim. inferior
lim[1]: lim. superior
*/

/*Gera numeros aleatórios U[0,1]*/
double urand(){
double v;
v = rand();
v = v/RAND_MAX;
return v;
}

//Devolve o módulo do float
float modulo(float a){
if (a < 0){
a = - a;
}
return a;
}

//Devolve o nº de chars da string, excluindo o '\0'
int charSize(char* vec){
int i;
for(i=0;vec[i] != '\0';i++){
i++;
}
return i;
}

//Copia uma string para outra
void strcpy1(char *vf, char *vi){
int i;
for(i=0; vi[i] != '\0';i++){
vf[i] = vi[i];
}
}

/*Calcula a área das curvas
A_expo recebe a area da exponencial
A_gaus recebe a area da gaussiana
*/
void area_curvas(double par[], double lim[], double A_expo, double A_gaus){
        double lim_ginf, lim_gsup, erf1, erf2;
        const double PI = 3.14159265358979323846;

//Area da exponencial
        A_expo = (exp(par[0])/par[1])*(exp(par[1]*lim[1])-exp(par[1]*lim[0]));

//Area da gaussiana
        lim_ginf = (lim[0]-par[3])/(sqrt(2)*par[4]);
        lim_gsup = (lim[1]-par[3])/(sqrt(2)*par[4]);
        erf1 = erf(lim_ginf);
        erf2 = erf(lim_gsup);

        A_gaus = par[2]*par[4]*sqrt(PI/2)*(erf2 - erf1);
}
```

```
/*funcao para selecionar a energia dentro do angulo
E_cSUP e E_cINF recebem as strings de valor de energia para simular
path_base é o caminho até a pasta da particula
V_ANG é o ângulo escolhido
*/
void findEnergy (char *path_base, char *V_ANG, char *E_cSUP, char *E_cINF, double vEn){
int j, k;
const char cListEn[]="lista_en";
char E_cVAR[30], path[100];
FILE *arq;

//Monta o caminho para o arquivo de energias
strcpy1(path, path_base);
k = charSize(path);
path[k+1] = '/';

for (j=0; V_ANG != '\0'; j++){
path[k+j+2]=V_ANG[j];
}
k = charSize(path);
path[k+1] = '/';

for (j=0; cListEn[j] != '\0'; j++){
path[k+j+2]=cListEn[j];
}

//Obtem as energias de calculo
arq = fopen (path, "r");
double E_VAR, E_DIF, E_MIN1=10000000, E_MIN2=10000000, E_SUP, E_INF;
int isEEqual = 0;

while (!feof(arq)){
fscanf (arq, "%c", &E_cVAR);
E_VAR = atof(E_cVAR);
E_DIF = modulo(E_VAR - vEn);
if (E_DIF == 0){
isEEqual = 1;
E_INF = E_VAR;
E_SUP = E_VAR;
strcpy1(E_cSUP, E_cVAR);
strcpy1(E_cINF, E_cVAR);
}

if ((E_DIF < E_MIN1) && (E_VAR < vEn) && isEEqual == 0){
E_MIN1 = E_DIF;
E_INF = E_VAR;
strcpy1(E_cINF, E_cVAR);
}
if ((E_DIF < E_MIN2) && (E_VAR > vEn) && isEEqual == 0){
E_MIN2 = E_DIF;
E_SUP = E_VAR;
strcpy1(E_cSUP, E_cVAR);
}
}
fclose(arq);
}

/*Obtem os 7 parametros dos arquivos
path_base é o caminho até a pasta da particula
cANG e cEN são as strings do angulo e energia
par_f é o array para armazenar os parametros interpolados
*/
void Get_par (char *path_base, char *cANG, char *cEN, double par_f[]){
int j,k;
char path[100];
FILE *arq;

//Monta o caminho para o arquivo de energias
strcpy1(path, path_base);
k = charSize(path);
path[k+1] = '/';
```

```
for (j=0; cANG[j] != '\0'; j++){
path[k+j+2]=cANG[j];
}
k = charSize(path);
path[k+1] = '/';

for (j=0; cEN[j] != '\0'; j++){
path[k+j+2]=cEN[j];
}
//

arq = fopen (path, "r");
fscanf (arq, "%f", &par_f[0]);
fscanf (arq, "%f", &par_f[1]);
fscanf (arq, "%f", &par_f[2]);
fscanf (arq, "%f", &par_f[3]);
fscanf (arq, "%f", &par_f[4]);
fscanf (arq, "%f", &par_f[5]);
fscanf (arq, "%f", &par_f[6]);
fclose(arq);
}

/*Interpola os parametros entre as duas energias/dois angulos
par_inf é o array de parametros da energia inferior
par_sup é o array de parametros da energia superior
par_fin é o array que recebe os parametros interpolados
var é a energia/angulo parametrizada(o) (vEn ou vAng)
cSUP é a string da energia/angulo superior
cINF é a string da energia/angulo inferior
*/
void Par_interp(double par_inf[], double par_sup[], double par_fin[], double var, char *cSUP, char
*cINF){


int i;
double SUP, INF;

SUP = atof(cSUP);
INF = atof(cINF);

for (i=0;i<7;i++){
par_fin[i]=((par_sup[i]-par_inf[i])*(var-INF)/(SUP-INF)) + (par_inf[i]); //<-- interpolação linear
}
}

/*Simula a distribuição exponencial pela inversa
par são os parametros das curvas
lim são os limites das curvas
A_expo é a área da exponencial
*/
double sim_expo(double par[], double lim[], double A_expo){
double var, u;

u = urand();
var = (1/par[1])*log(exp(par[1]*lim[0])+(par[1]*u*A_expo/exp(par[0])));

return var;
}

/*Simula a distribuição gaussiana pela inversa
par são os parametros das curvas
lim são os limites das curvas
A_gaus é a área da exponencial
*/
double sim_gaus(double par[], double lim[], double A_gaus){
double var, u, a_erf, arg;
const double PI = 3.14159265358979323846;

u = rand();
arg = u*A_gaus*sqrt(2/PI)/(par[2]*par[4]) + erf((lim[0]-par[3])/(sqrt(2)*par[4]));
a_erf = (sqrt(PI)/2)*(arg+(PI/12)*pow(arg,3) + (7*pow(PI,2)/480)*pow(arg,5) + (127*pow(PI, 3)/40320)
*pow(arg,7) + (4369*pow(PI, 4)/5806080)*pow(arg,9) + (34807*pow(PI, 5)/182476800)*pow(arg,11));
```

```
var = par[3]+sqrt(2)*par[4]*a_erf;

return var;
}

/*Função Resposta*/
double FResposta(double vAng, double vEn, const char *vPar){
char vAngChar[10], vEnChar[10], path[100] =  {'D', 'a', 't', 'a', '/',}, path1[100];
const char cListAng[]="lista_ang";
int i,j,k,l,m;
int isAEqual = 0;
double par[7], lim[2], A_expo, A_gaus;
double fotons;  //N de fotos de retorno
const double PI = 3.14159265358979323846;
FILE *arq;

//Seta strings iniciais
sprintf(vAngChar, "%f", vAng);
sprintf(vEnChar, "%f", vEn);

for (m=0; path[m] != '\0';m++){
printf("%c",path[m]);
}

for (i=0; vPar[i] != '\0'; i++){
path[i+5]=vPar[i];
}
//path[i+1]='\0';

//Seleciona os angulos para buscar parametros
//A_cSUP e A_cINF - string dos angulos superior e inferior
strcpy1 (path1, path);
path1[i]='/';
for (j=0; cListAng[j] != '\0'; j++){
path1[i+j+1]=cListAng[j];
}



arq = fopen (path1, "r");
double A_VAR, A_DIF, A_MIN1=100, A_MIN2=100, A_SUP, A_INF;
char A_cVAR[30], A_cSUP[30], A_cINF[30];
while (!feof(arq)){
fscanf (arq, "%c", &A_cVAR);
A_VAR = atof(A_cVAR);
A_DIF = modulo(A_VAR - vAng);
if (A_DIF == 0){
A_INF = A_VAR;
A_SUP = A_VAR;
strcpy1(A_cSUP, A_cVAR);
strcpy1(A_cINF, A_cVAR);
}
if ((A_DIF < A_MIN1) && (A_VAR < vAng) && isAEqual == 0){
A_MIN1 = A_DIF;
A_INF = A_VAR;
strcpy1(A_cINF, A_cVAR);
}
if ((A_DIF < A_MIN2) && (A_VAR > vAng) && isAEqual == 0){
A_MIN2 = A_DIF;
A_SUP = A_VAR;
strcpy1(A_cSUP, A_cVAR);
}
}

char cE_AINF_EINF[30], cE_AINF_ESUP[30], cE_ASUP_EINF[30], cE_ASUP_ESUP[30];

findEnergy(path, A_cSUP, cE_ASUP_EINF, cE_ASUP_ESUP, vEn);
findEnergy(path, A_cINF, cE_AINF_EINF, cE_AINF_ESUP, vEn);


double par_ASUP_EINF[7], par_ASUP_ESUP[7], par_AINF_EINF[7], par_AINF_ESUP[7];
Get_par (path, A_cSUP, cE_ASUP_EINF, par_ASUP_EINF);
```

```
Get_par (path, A_cSUP, cE_ASUP_ESUP, par_ASUP_ESUP);
Get_par (path, A_cINF, cE_AINF_EINF, par_AINF_EINF);
Get_par (path, A_cINF, cE_AINF_ESUP, par_AINF_ESUP);

double par_ASUP[7], par_AINF[7];
//Interpola os parametros entre as duas energias, dentro de cada angulo
Par_interp (par_ASUP_EINF, par_ASUP_ESUP, par_ASUP, vEn, cE_ASUP_ESUP, cE_ASUP_EINF);
Par_interp (par_AINF_EINF, par_AINF_ESUP, par_AINF, vEn, cE_AINF_ESUP, cE_AINF_EINF);

//Interpola os parametros resultantes entre os angulos
char A_SUPc[10], A_INFc[10];
sprintf(A_SUPc, "%f", A_SUP);
sprintf(A_INFc, "%f", A_INF);
Par_interp (par_AINF, par_ASUP, par, vAng, A_SUPc, A_INFc);

//Define os limites da funcao
lim[0] = 1;
lim[1] = par[5];

//Gera a semente do rand()
srand ( time(NULL) );

//Sorteia se a particula interage
double p;
p = urand();
if(p < par[6]){
        fotons = 0;
}

//Escolhe a curva e simula o nº de fotons
else{
double A_gaus, A_expo, A_sum, c;
area_curvas(par, lim, A_expo, A_gaus);
A_sum = A_gaus + A_expo;
c = urand();
if(c < A_gaus/A_sum){
fotons = sim_gaus(par, lim, A_gaus);
}
else{
fotons = sim_expo(par, lim, A_expo);
}
}

return fotons;
}
```

ANEXO B - Script Fazedor

```
#!/bin/bash
#Joao PN - joaoxp-at-gmail.com

if [ ! -s nomes_histogramas ]; then
        mkdir Histogramas
        mv histo_* Histogramas
        mv saida_teste_histo_* Histogramas
        cd Histogramas
        echo histo_* > nomes_histogramas_temp
        sed -i s/histo_//g nomes_histogramas_temp
        tr ' ' '\n' < nomes_histogramas_temp >nomes_histogramas
        rm -f nomes_histogramas_temp #Remendo enquanto nao descubro como fazer o tr mexer no proprio
arquivo
        cd ..
        mv Histogramas/nomes_histogramas nomes_histogramas
        mkdir Resultados

        #Organiza e copia os arquivos de analise das energias para as respectivas pastas
        exec 3< nomes_histogramas
        while read linha <&3; do
                var_energia_nome=$linha
                mkdir Analise_$var_energia_nome
                mkdir Resultados/$var_energia_nome
                cp Analisador Analise_$var_energia_nome/
                cp Gerador Analise_$var_energia_nome/
                cp histograma_base.C Analise_$var_energia_nome/
                cp histograma_base_uma_curva.C Analise_$var_energia_nome/
                cp Histogramas/histo_$var_energia_nome Analise_$var_energia_nome/
                cp Histogramas/histo_$var_energia_nome Resultados/$var_energia_nome
        done
fi
#Faz toda a analise de todos os histogramas de todas as energias
exec 3< nomes_histogramas
while read linha <&3; do
        var_energia_nome=$linha
        cd Analise_$var_energia_nome
        declare -x energia_nome=$var_energia_nome
        declare -x limite_de_barras="31"
        ./Gerador
        ./Analisador
        cd ..

        cp Analise_$var_energia_nome/FINALHISTOGRAMA_* Resultados/$var_energia_nome
        cp Analise_$var_energia_nome/pos-analise.C Resultados/$var_energia_nome

        cd Resultados/$var_energia_nome
        root -l -b -q pos-analise.C
        cd ..
        cd ..
done

exit 0
```

ANEXO C - Script Gerador

```
#!/bin/bash
#Joao PN - joaoxp-at-gmail.com

#Espaço para código que fornece faixa de valores dos dados
#e: comprimento das barras

#Código para achar a energia máxima da plotagem
var_teste="0"
while [ $var_teste -eq 0 ]; do

        energia_final="0"
        teste_e="0"
        teste_e2="0"
        exec 3< histo_$energia_nome
        while read linha <&3; do
                if [ $energia_final -lt $linha ]; then
                        energia_final=$linha
                fi
        done

        exec 3< histo_$energia_nome
        while read linha <&3; do
                if [ $teste_e -lt $linha ] && [ $linha -ne $energia_final ]; then
                        teste_e=$linha
                fi
        done
        let "teste_e2 = teste_e + teste_e"
        if [ $energia_final -lt $teste_e2 ]; then
                var_teste="1"
        fi
        if [ $var_teste -eq 0 ]; then
                sed -i /$energia_final/d histo_$energia_nome
        fi

        if [ $energia_final -eq 0 ]; then
                var_teste="1"
        fi
done


#Faz tudo para 20, 25, ... ,limite_de_barras
numero_de_barras="20"
while [ $numero_de_barras -lt $limite_de_barras ]; do
        let "e = energia_final / numero_de_barras"
        #Gera os arquivos para análise
        numero_histo="0"
        #Copia arquivos para outra pasta, para organizacao
        while (("$numero_histo" < "$numero_de_barras")); do
                mkdir analise$numero_de_barras
                cp histograma_base_uma_curva.C analise$numero_de_barras
                cp histograma_base.C analise$numero_de_barras
                cp histo_$energia_nome analise$numero_de_barras
                cd analise$numero_de_barras
                #Gera combinações de curvas

                        #expo
                        tipo_curva="expo"
                        fitagem_final=$energia_final #Fitagem vai para todo o conjunto de dados

                        sed s/numero_histo/"$numero_histo"/g
<histograma_base_uma_curva.C>histograma"$numero_histo".C
                        sed -i s/tipo_curva/"$tipo_curva"/g histograma"$numero_histo".C
                        sed -i s/fitagem_final/"$fitagem_final"/g histograma"$numero_histo".C
                        sed -i s/energia_nome/"$energia_nome"/g histograma"$numero_histo".C
                        sed -i s/energia_final/"$energia_final"/g histograma"$numero_histo".C
                        sed -i s/numero_de_barras/"$numero_de_barras"/g
histograma"$numero_histo".C

                        #expo-gaus
                        for (( i=1 ; i<=2*numero_de_barras ; i++ )); do
```

```
                                let "l= i*e/2"
                                expo_i="1"
                                expo_f=$l
                                gaus_i=$l
                                gaus_f=$energia_final
                                fitagem_final=$energia_final #Fitagem vai para todo o conjunto de dados

                                #Substitui valores
                                let "numero_histo = numero_histo + 1"
                                sed s/numero_histo/"$numero_histo"/g
<histograma_base.C>histograma"$numero_histo".C
                                sed -i s/expo_i/"$expo_i"/g histograma"$numero_histo".C
                                sed -i s/expo_f/"$expo_f"/g histograma"$numero_histo".C
                                sed -i s/gaus_i/"$gaus_i"/g histograma"$numero_histo".C
                                sed -i s/gaus_f/"$gaus_f"/g histograma"$numero_histo".C
                                sed -i s/fitagem_final/"$fitagem_final"/g histograma"$numero_histo".C
                                sed -i s/energia_nome/"$energia_nome"/g
histograma"$numero_histo".C
                                sed -i s/energia_final/"$energia_final"/g
histograma"$numero_histo".C
                                sed -i s/numero_de_barras/"$numero_de_barras"/g
histograma"$numero_histo".C
                        done

                        #gaus-expo
                        for ((i=1; i<2*numero_de_barras ; i++))
                        do
                                let "l= i*e/2"
                                expo_i=$l
                                expo_f=$energia_final
                                gaus_i="1"
                                gaus_f=$l
                                fitagem_final=$energia_final #Fitagem vai para todo o conjunto de dados

                                #Substitui valores
                                sed s/numero_histo/"$numero_histo"/g
<histograma_base.C>histograma"$numero_histo".C
                                sed -i s/expo_i/"$expo_i"/g histograma"$numero_histo".C
                                sed -i s/expo_f/"$expo_f"/g histograma"$numero_histo".C
                                sed -i s/gaus_i/"$gaus_i"/g histograma"$numero_histo".C
                                sed -i s/gaus_f/"$gaus_f"/g histograma"$numero_histo".C
                                sed -i s/fitagem_final/"$fitagem_final"/g histograma"$numero_histo".C
                                sed -i s/energia_nome/"$energia_nome"/g
histograma"$numero_histo".C
                                sed -i s/energia_final/"$energia_final"/g
histograma"$numero_histo".C
                                sed -i s/numero_de_barras/"$numero_de_barras"/g
histograma"$numero_histo".C
                                let "numero_histo = numero_histo + 1"
                        done

                        #gaus
                        tipo_curva="gaus"
                        fitagem_final=$energia_final #Fitagem vai para todo o conjunto de dados

                        sed s/numero_histo/"$numero_histo"/g
<histograma_base_uma_curva.C>histograma"$numero_histo".C
                        sed -i s/tipo_curva/"$tipo_curva"/g histograma"$numero_histo".C
                        sed -i s/fitagem_final/"$fitagem_final"/g histograma"$numero_histo".C
                        sed -i s/energia_nome/"$energia_nome"/g histograma"$numero_histo".C
                        sed -i s/energia_final/"$energia_final"/g
histograma"$numero_histo".C
                        sed -i s/numero_de_barras/"$numero_de_barras"/g
histograma"$numero_histo".C
                done
        let "numero_de_barras = numero_de_barras + 5"
        cd ..
done
exit 0
```

ANEXO D - Script Analisador

```
#!/bin/bash
#Joao PN - joaoxp-at-gmail.com

#Espaço para código que fornece faixa de valores dos dados
#e: comprimento das barras; n:número de barras do histograma

#Código para achar a energia máxima da plotagem
energia_final="0"
exec 3< histo_$energia_nome
while read linha <&3; do
        if [ $energia_final -lt $linha ]; then
                energia_final=$linha
        fi
done

echo "{" > pos-analise.C

numero_de_barras="20"
while [ $numero_de_barras -lt $limite_de_barras ]; do
        let "ndois = 4*numero_de_barras"
        cd analise$numero_de_barras #Vai para o diretorio dos dados

        #Gera macro de análise  e analisa os histogramas
        echo "{" > analisador-root.C
        for(( i=0 ; i<ndois ; i++ )); do
                echo 'gROOT->ProcessLine(".x histograma_numero.C");' >> analisador-root.C
                sed -i s/_numero/"$i"/g analisador-root.C
        done
        echo "}" >> analisador-root.C
        root -l -b -q -n analisador-root.C

        #Faz a análise do Chi^2
        chi2_minimo="1"
        numero_chi_minimo="0"
        chi_erro="0.000000"
        for(( i=0 ; i<"$ndois" ; i++ )); do
                chi2_teste=`cat chi$i`
                if [ "$chi2_minimo" \> "$chi2_teste" ] &&  [ "$chi2_teste" != "$chi_erro" ]; then
                chi2_minimo=$chi2_teste
                numero_chi_minimo=$i
                fi
        done

        cd ..

        echo 'gROOT->ProcessLine(".x FINALHISTOGRAMA_numero_de_barras.C");' >> pos-analise.C
        sed -i s/numero_de_barras/"$numero_de_barras"/g pos-analise.C

        cp analise$numero_de_barras/histograma$numero_chi_minimo.C FINALHISTOGRAMA_$numero_de_barras.C
#       cp analise$numero_de_barras/dados$numero_chi_minimo FINALDADOS_$numero_de_barras.C
#       cp analise$numero_de_barras/chi$numero_chi_minimo FINALCHI_$numero_de_barras.C
#       cp analise$numero_de_barras/histo$numero_chi_minimo.eps FINALGRAPH_$numero_de_barras.eps
        sed -i s/dados$numero_chi_minimo/FINALDADOS_$numero_de_barras/g FINALHISTOGRAMA_
$numero_de_barras.C
        sed -i s/histo$numero_chi_minimo.eps/FINALGRAPH_$numero_de_barras/g FINALHISTOGRAMA_
$numero_de_barras.C
        sed -i s/chi$numero_chi_minimo/FINALCHI_$numero_de_barras/g FINALHISTOGRAMA_$numero_de_barras.C
#       root -l -b -q FINALHISTOGRAMA_$numero_de_barras.C
        let "numero_de_barras = numero_de_barras + 5"
done

echo "}" >> pos-analise.C

exit 0
```

ANEXO E - Script Checador

```
#!/bin/bash
#Joao PN - joaoxp-at-gmail.com

#Checa se a analise teve sucesso, e remove analise com erro
mkdir Erros
exec 3< nomes_histogramas
while read linha <&3; do
        cd Resultados/$linha

        echo Testando $linha
        if [ -s FINALCHI_20 ] && [ -s FINALCHI_25 ] && [ -s FINALCHI_30 ] && [ -s FINALDADOS_20 ] &&
[ -s FINALDADOS_25 ] && [ -s FINALDADOS_30 ] && [ -s FINALGRAPH_20 ] && [ -s FINALGRAPH_25 ] && [ -s
FINALGRAPH_30 ]; then
                echo $linha OK
                cd ..
                cd ..
        else
                cd ..
                cd ..

                mv Histogramas/histo_$linha Erros/
                rm -fR Analise_$linha
                mv Resultados/$linha Erros/
                mv Histogramas/saida_teste_histo_$linha Erros/
                sed -i /$linha/d nomes_histogramas
                echo $linha com erro
                exit
        fi


done
```

ANEXO F - Script ZeroFotons

```
#!/bin/bash

##Faz a analise para zero fotoeletrons
mkdir Resultados/Zero_fotons
#Faz uma copia dos histogramas sem os dados "0"
cp Histogramas/histo_* Resultados/Zero_fotons/
cp nomes_histogramas Resultados/Zero_fotons
cd Resultados/Zero_fotons
exec 3< nomes_histogramas
while read linha <&3; do
        sed -i s/0// histo_$linha
        grep -v "^$" histo_$linha > temp_histo_$linha
        rm -f histo_$linha
        mv temp_histo_$linha histo_$linha
done
cd ..
cd ..

#Retira do saida_teste_histo a quantidade de simulacoes da particula

exec 3< nomes_histogramas
while read linha <&3; do
        count=0
        for (( i=0 ; i<=3100 ; i++ )); do
                if grep ">>> Event $i" Histogramas/saida_teste_histo_$linha; then
                        count=$i
                fi
        done
        let 'count = count + 1'
        count2=`grep -c '$' Resultados/Zero_fotons/histo_$linha`
        echo $(echo "scale=4; $count2/$count" | bc) >> Resultados/Zero_fotons/FRACAO_ZERO
done

#Adiciona, com probabilidade 0, as energias que deram erro
DIR="Erros"
if [ "$(ls -A $DIR)" ]; then
        cp nomes_histogramas Erros/ENERGIAS_tmp
        cd Erros
        echo histo_* >> ENERGIAS_tmp
        sed -i s/histo_//g ENERGIAS_tmp
        tr ' ' '\n' < ENERGIAS_tmp >ENERGIAS

        A=$(ls -1 | wc -l)
        let "B = A / 2"
        Bvar=0
        cd ..
        while [ $Bvar -lt $B ]; do
                echo .0000 >> Resultados/Zero_fotons/FRACAO_ZERO
                let "Bvar = Bvar + 1"
        done

        mv Erros/ENERGIAS Resultados/Zero_fotons/ENERGIAS

else
        cp nomes_histogramas Resultados/Zero_fotons/ENERGIAS
fi


#Lista as energias das simulacoes
cd Resultados/Zero_fotons
sed -i s/MeV//g ENERGIAS
sed -i s/GeV/000/g ENERGIAS
sed -i s/TeV/000000/g ENERGIAS

paste ENERGIAS FRACAO_ZERO > dados_zerofotons.dat
cd ..
cd ..
```

ANEXO G - Script Finalizador

```
#!/bin/bash

#Limpa arquivos de analise
rm -fR Analise_*

#Seleciona o plot com menor chi^2 entre 20, 25 e 30 barras
exec 3< nomes_histogramas
while read linha <&3; do
        cd Resultados/$linha
        numero_chi=20
        chi2=`cat FINALCHI_20`
        chi2_teste=`cat FINALCHI_25`

        if [ "$chi2" > "$chi2_teste" ]; then
                numero_chi=25
                chi2=$chi2_teste
        fi

        chi2_teste=`cat FINALCHI_30`

        if [ "$chi2" > "$chi2_teste" ]; then
                numero_chi=30
                chi2=$chi2_teste
        fi

        cd ..

        cp $linha/FINALDADOS_$numero_chi DADOS_$linha
        cp $linha/FINALGRAPH_$numero_chi GRAPH_$linha
        cp $linha/FINALCHI_$numero_chi CHI_$linha
        cp $linha/FINALHISTOGRAMA_$numero_chi.C HISTO_$linha

        rm -fR $linha

        cd ..
done
```

ANEXO H - Arquivo de entrada para o ROOT histograma_base.C

```
//[n umero_histo] - numero serial do histograma, inteiro
//[e xpo_i] - inicio da exponencial, inteiro
//[e xpo_f] - final da exponencial. inteiro
//[f itagem_final] - final da fitagem, inteiro

{
TCanvas *c1 = new TCanvas("c1", "c1",8,8,699,499);
c1->Range(0.75625,-56.875,1.19375,11.875);
c1->SetBorderSize(2);
c1->SetFrameFillColor(10);
c1->SetFillColor(10);

c1->Modified();
c1->cd();

co = new TH1F("co","Distribuicao de fotoeletrons - energia_nome",numero_de_barras,1,energia_final);

co->GetXaxis()->SetTitle("Fotoeletrons");
co->GetYaxis()->SetTitle("Ocorrencias");
co->GetYaxis()->SetTitleOffset(1.2);
co->GetXaxis()->SetTitleOffset(1.1);
//gStyle->SetOptStat("nemr");
gStyle->SetOptFit(111);

FILE *arq;
FILE *saida;
FILE *saida2;
saida=fopen("dadosnumero_histo","w");
saida2=fopen("chinumero_histo","w");
double x;
int nd;

arq=fopen("histo_energia_nome","r");
if(arq!=NULL)
{
  do
    {
      nd=fscanf(arq,"%lf",&x);
      if(nd==1)
        {
          co.Fill(x);
        }
    }
  while(nd==1);
  co.Draw();
}

Double_t norm = co->GetEntries();
if (norm) co->Scale(1/norm);

double par[5];
g1    = new TF1("g1","expo",expo_i,expo_f);
g2    = new TF1("g2","gaus",gaus_i,gaus_f);
total = new TF1("total","expo(0)+gaus(2)",1,fitagem_final);

g1->SetLineColor(1);
g2->SetLineColor(1);
total->SetLineColor(2);

co->Fit(g1,"R");
co->Fit(g2,"R+");

g1->GetParameters(&par[0]);
g2->GetParameters(&par[2]);

total->SetParameters(par);
co->Fit(total,"R+");

total->GetParameters(&par[5]);
```

```
 fprintf(saida,"%lf\n",par[0]);
 fprintf(saida,"%lf\n",par[1]);
 fprintf(saida,"%lf\n",par[2]);
 fprintf(saida,"%lf\n",par[3]);
 fprintf(saida,"%lf\n",par[4]);
 fprintf(saida, "fitagem_final",);


c1->Print("histonumero_histo.eps");


double chi2;
chi2 = total->GetChisquare();
fprintf(saida2,"%lf",chi2);

fclose(arq);
fclose(saida);
fclose(saida2);

c1->Close();
delete co;
delete g1;
delete g2;
delete total;
delete c1;
}
```

ANEXO H - Arquivo de entrada para o ROOT histograma_base_uma_curva.C

```
//[n umero_histo] - numero serial do histograma, inteiro
//[e xpo_i] - inicio da exponencial, inteiro
//[e xpo_f] - final da exponencial. inteiro
//[f inal] - final da fitagem, inteiro

{
TCanvas *c1 = new TCanvas("c1", "c1",8,8,699,499);
c1->Range(0.75625,-56.875,1.19375,11.875);
c1->SetBorderSize(2);
c1->SetFrameFillColor(10);
c1->SetFillColor(10);

c1->Modified();
c1->cd();

co = new TH1F("co","Distribuicao de fotoeletrons - energia_nome",numero_de_barras,1,energia_final);

co->GetXaxis()->SetTitle("Fotoeletrons");
co->GetYaxis()->SetTitle("Ocorrencias");
co->GetYaxis()->SetTitleOffset(1.2);
co->GetXaxis()->SetTitleOffset(1.1);
//gStyle->SetOptStat("nemr");
gStyle->SetOptFit(111);

FILE *arq;
FILE *saida;
FILE *saida2;
saida=fopen("dadosnumero_histo","w");
saida2=fopen("chinumero_histo","w");
double x;
int nd;

arq=fopen("histo_energia_nome","r");
if(arq!=NULL)
{
  do
    {
      nd=fscanf(arq,"%lf",&x);
      if(nd==1)
        {
          co.Fill(x);
        }
    }
  while(nd==1);
  co.Draw();
}

Double_t norm = co->GetEntries();
if (norm) co->Scale(1/norm);

double par[6];
g1    = new TF1("g1","tipo_curva",1,fitagem_final);
total = new TF1("total","tipo_curva(0)",1,fitagem_final);

g1->SetLineColor(1);
total->SetLineColor(2);

co->Fit(g1,"R");

g1->GetParameters(&par[0]);

total->SetParameters(par);
co->Fit(total,"R+");

total->GetParameters(&par[3]);

 fprintf(saida,"%lf\n",par[0]);
 fprintf(saida,"%lf\n",par[1]);
 fprintf(saida,"%lf\n",par[2]);
 fprintf(saida,"%lf\n",par[3]);
```

```
 fprintf(saida,"%lf\n",par[4]);
 fprintf(saida, "fitagem_final",);


c1->Print("histonumero_histo.eps");


double chi2;
chi2 = total->GetChisquare();
fprintf(saida2,"%lf",chi2);

fclose(arq);
fclose(saida);
fclose(saida2);

c1->Close();
delete co;
delete g1;
delete total;
delete c1;
}
```