C++ Wiederholung aus dem WS

- Zeiger Konzept
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Pointer.pdf

- Funktionen
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Funktionen.pdf

Wir wollen zwei 3er-Vektoren über die Tastatur einlesen. Es soll der Normalenvektor der Ebene, die durch die beiden Vektoren aufgespannt wird, bestimmt und ausgegeben werden.

Arbeitsvorschlag:

normalenVektor.cc

- Schreiben Sie eine Funktion, die den Normalenvektor zurück gibt.
  Wie werden die beiden Vektoren übergeben und das Resultat zurückgegeben?

- Brauchen wir weitere Hilfsfunktionen?

- Welche Rolle spielen Funktionsheader?
  Lagern Sie die header in eine eigene Datei aus.
  Welche Änderungen sind notwendig, wenn die header in ein directory `include` oberhalb ihres directories mit dem Quellcode, e.g. `src`, ausgelagert werden sollen.

- Erzeugen Sie eine Bibliothek mit `ar` die dann statisch gelinkt werden soll.
  Welche commnds auf der shell müssen wir ausführen?
  Die Bibliothek soll in einem directory mit dem Namen lib untergebracht werden.

solution.txt

Function Pointer

Pointer können nicht nur auf Variablen gesetzt werden, sondern auch auf Funktionsspeicherbereiche. Dadurch läßt sich das Programm Verhalten zur Laufzeit dynamischer gestalten. Ein modernes Konzept haben wir mit den Lambda Funktionen kennengelernt oder wird mit Funktonsobjekten (Funktoren) implementiert. Funktionszeiger werden meistens mit Type Definitionen `typedef` verwendet..

Beispiel:

```
#include <iostream>
int multiplication(int a, int b){ return a*b;}
int difference (int a, int b){return a-b;}
typedef int  (*pointerType)(int, int);
int main() {
   pointerType calcOperation = 0;
   calcOperation = &multiplication;
   std::cout << (*calcOperation)(40, 8) << std::endl;
   calcOperation = &difference;
   std::cout << (*calcOperation)(40, 8) << std::endl;
}
```

function pointer auf Funktion, die int zurück gibt und 2 int Argumente hat

Definition der Funktion

Neudefinition der Funktion

Function Pointer

Function Pointer erlauben an Funktionen mit numerischen Methoden, wie `integrate` oder `differentiate` , beliebige Funktionen zu übergeben.

Beispiel:

funcPointer.cc

```
#include <iostream>
#include <cmath>
typedef double (*FuncPtr)(double);
void printVal (FuncPtr, double);
int main() {
   FuncPtr f;
   f = & sin;
   printVal(f,1.5);
   f = & exp;
   printVal(f,2.);
   return 0;
}
void printVal (void (*FuncPtr)(double), double x){
   cout << "x = " << x << " f=" << FuncPtr(x) << endl;
}
```

Print beliebiger Funktionswerte

C++ Wiederholung aus dem WS

- Input/Output
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/FileIO.pdf

- Klassenkonzept
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Klassen.pdf

Functor / Function Object

Ein Funktionsobjekt (functor) ist eine Klasse/Struktur mit einem überladenen Operator (). Es können Objekte instanziert werden, die sich wie Funktionsaufrufe verhalten.

Beispiel:

```
#include <iostream>
struct squareVal{
      double operator()(double a) { return a*a;}
};
using namespace std;
int main( ) {
  double x = -7.;
  squareVal myObject;
  double square_x = myObject(x);
  cout << "x=" << x << " square_x=" << square_x << endl;
  return 0;}
```

myObject ist ein Objekt und keine Funktion, trotzdem können wir das Objekt durch das Überladen des () Operators wie eine Funktion aufrufen. Der Funktionsaufruf Operator ist als member Funktion deklariert.

Functor / Function Object

Das Funktionsobjekt hat alle Eigenschaften eines Objektes, es besitzt also einen Zustand und speichert Daten.

Beispiel: mit einer Klasse

```
.........
class MyAddFunctor {
      public:
      MyAddFunctor(int inp) { x = inp; }
      int operator()(int y){ return x+=y; }
      private:
      int x;
};
.........
MyAddFunctor func(5);
int v = func(10);
int u = func(25);

MyAddFunctor(5)(10) ;
```

Definition des Konstruktors

Der Operator () wird mit der gewünschten Funktionalität überladen.

Klasse wird instanziert

v = 15

u = 40

= 15

In der Standard Template Library werden häufig Funktoren benutzt.

C++ Wiederholung aus dem WS

- ROOT Installation
  https://www.physi.uni-heidelberg.de/~marks/linux_einfuehrung/Folien/RootInstall.pdf
  - Verwenden Sie die Quellcode Installation

- Datenanalyse mit ROOT Histogrammen
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Root.pdf

- Ergänzungen
  - Energieverlust von geladenen Teilchen in dünnen Materieschichten
  - Vergleich mit simulierten Daten
  - Bestimmung der Grössen einer Landauverteilung
  - Anpassung der simulierten Verteilung an die gemessene
  - Kolmogorov-Smirnov Test zum Vergleich von Verteilungen

C++ Wiederholung aus dem WS

- ROOT Installation
  https://www.physi.uni-heidelberg.de/~marks/linux_einfuehrung/Folien/RootInstall.pdf
  - Verwenden Sie die Quellcode Installation

- Datenanalyse mit ROOT Histogrammen
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Root.pdf

- Ergänzungen
  - Energieverlust von geladenen Teilchen in dünnen Materieschichten
  - Vergleich mit simulierten Daten
  - Bestimmung der Grössen einer Landauverteilung
  - Anpassung der simulierten Verteilung an die gemessene
  - Kolmogorov-Smirnov Test zum Vergleich von Verteilungen

Energieverlust geladener Teilchen

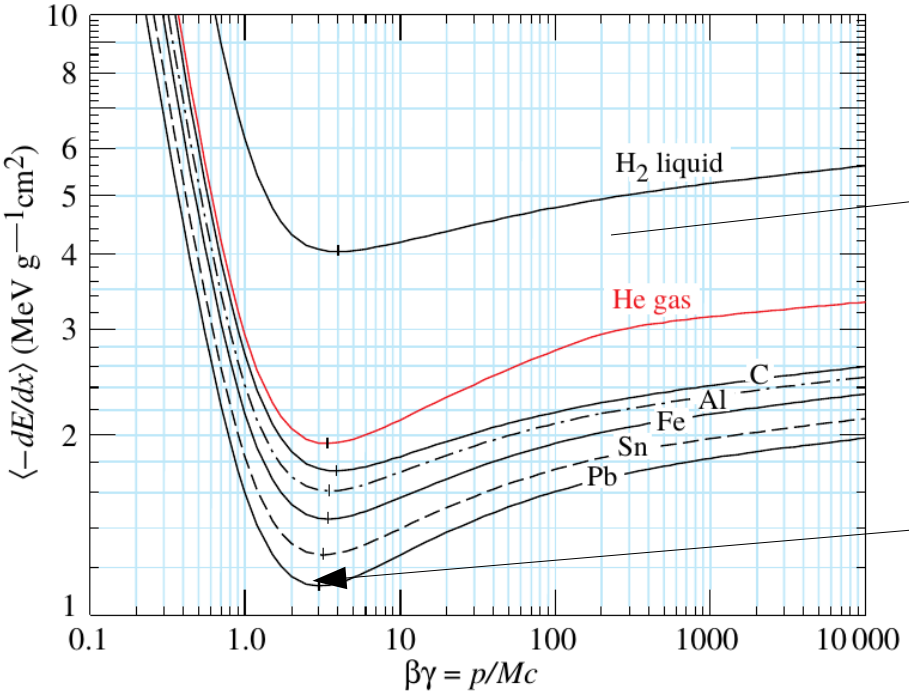
Der Energieverlust findet hauptsächlich durch inelastische Stösse mit den Elektronen der Atomhülle statt. Die quantenmechanische Behandlung führt zur Bethe-Bloch Formel.

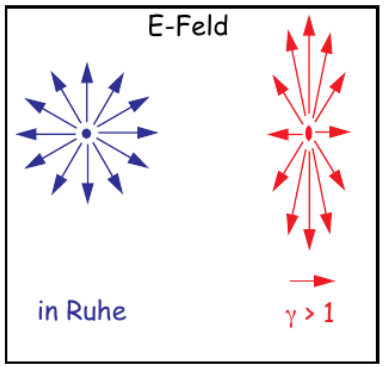
$$-\frac{dE}{dx} = 4\pi N_A r_e^2 m_e c^2 z^2 \frac{Z}{A} \cdot \frac{1}{\beta^2} \cdot \left[\frac{1}{2} \ln \frac{2 m_e c^2 \beta^2 \gamma^2 T_{max}}{I^2} - \beta^2 - \frac{\delta}{2} - \frac{C}{Z} \right] \quad \left[\frac{\text{MeV}}{(\text{g/cm}^2)} \right]$$

Anregungsenergie

Dichte Korrektur high energy

Hüllenkorrektur low energy



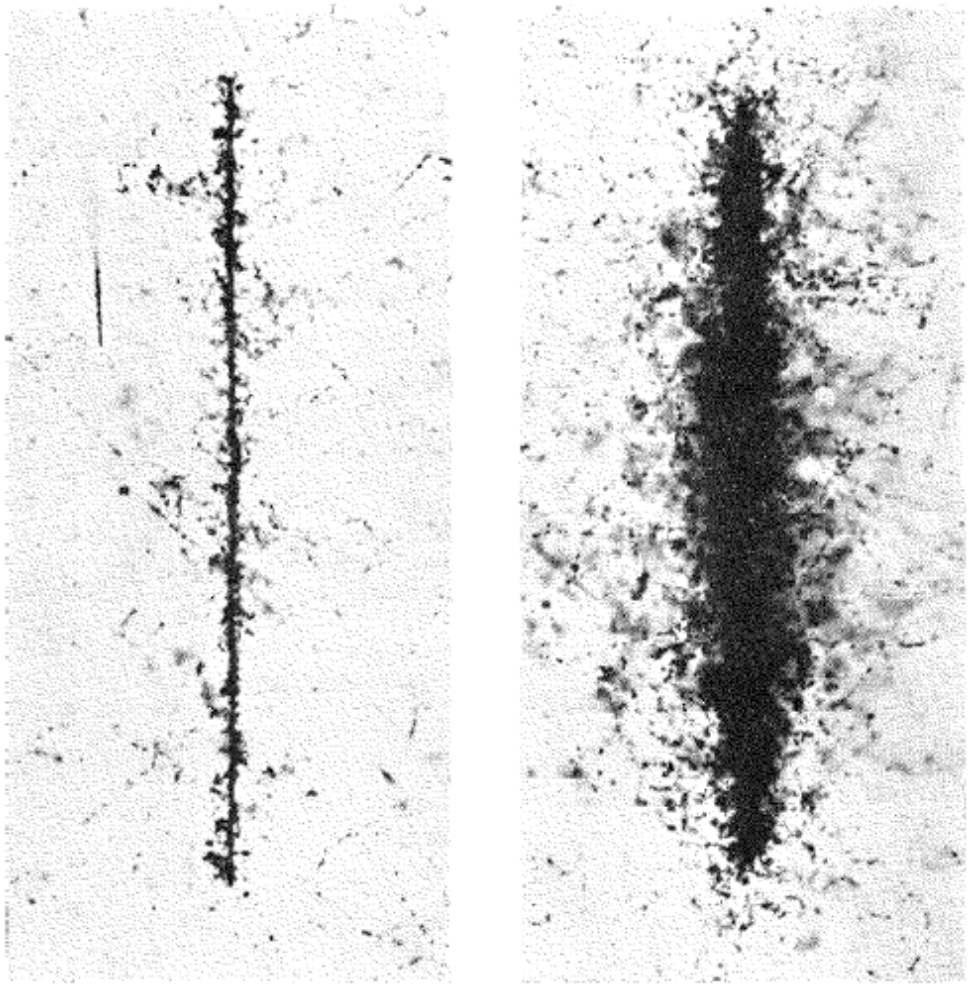


MIP = minimum ionizing particle

z-Dependence of Ionization

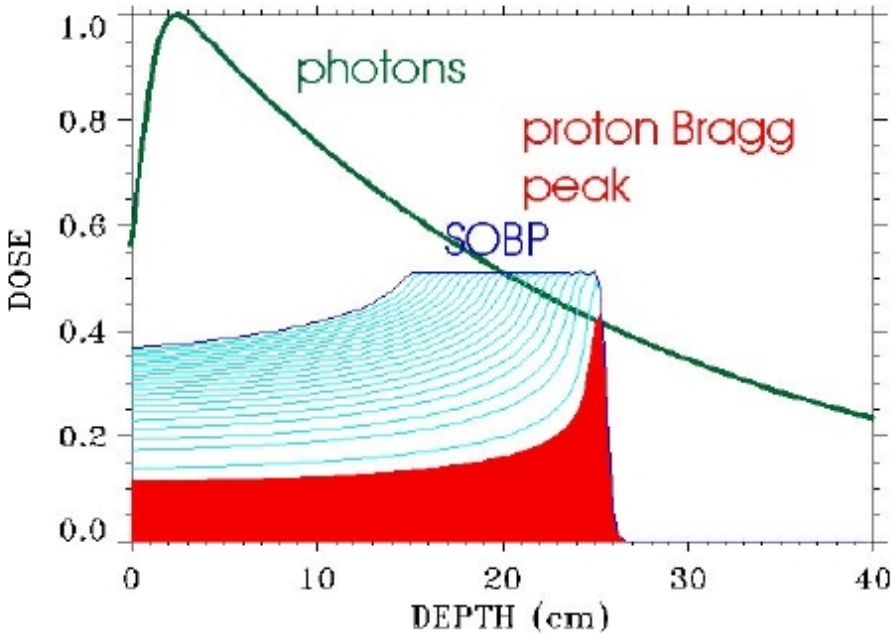
z = 26 (Fe)

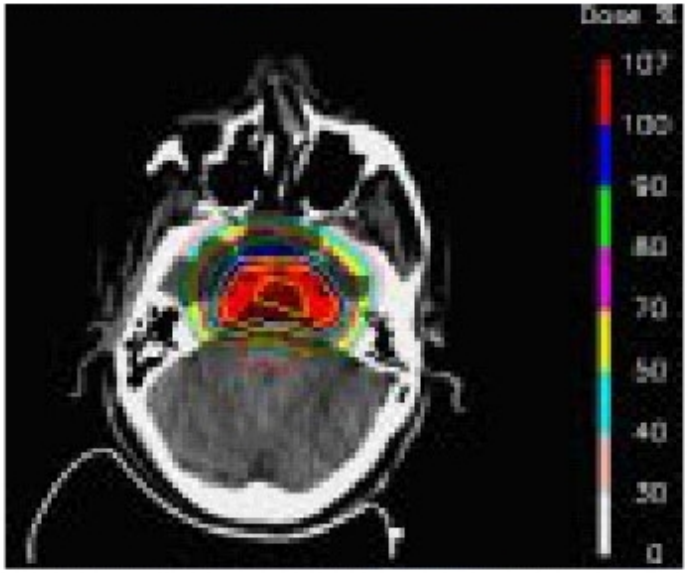
z ≈ 90 (Th)



Relativistic ions in nuclear emulsion

Example: Proton Therapy at PSI

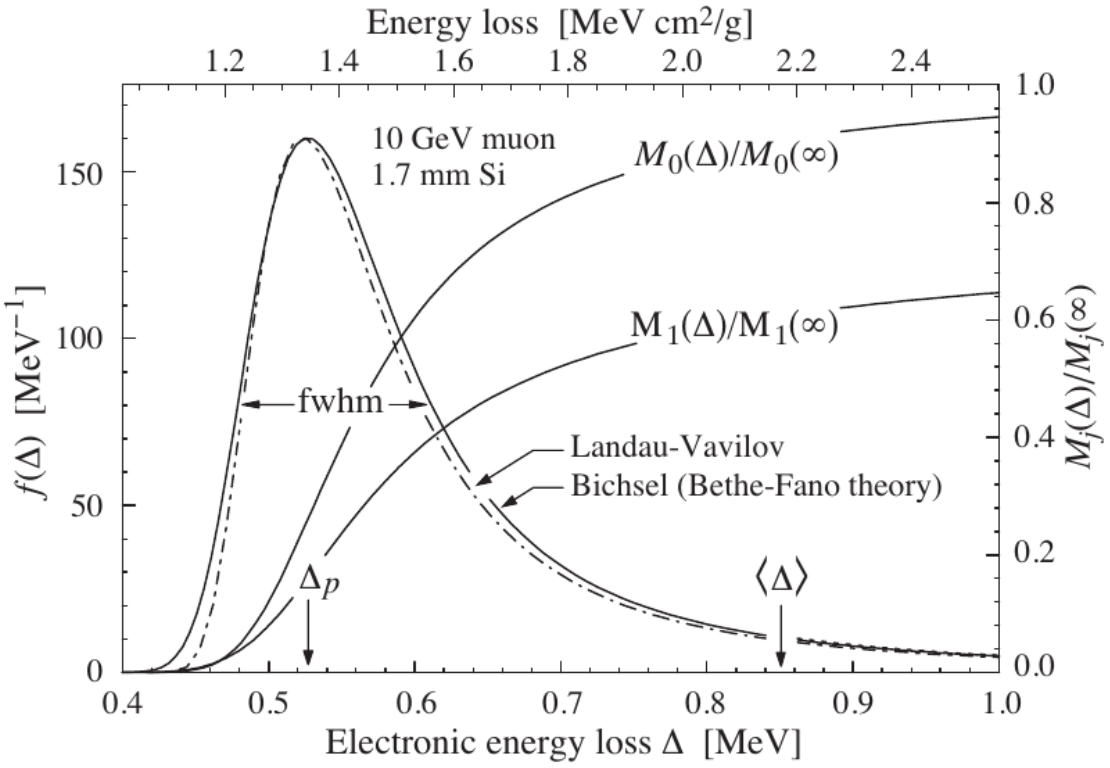




Spread out Bragg Peaks (SOBP) mit unterschiedlichen Absorbern ⇒ annähernd konstante Dosisverteilung im Bereich des Tumors

Landau Verteilung

Die Landau Verteilung beschreibt den Energieverlust von geladenen Teilchen in einer dünnen Materialschicht. Die starke Asymmetry auf der rechten Seite beruht auf der Abstrahlung von δ rays (Elektronen mit grösserer Energie). Der Mittelwert und die Varianz sind nicht als geschlossene Ausdrücke definiert Die Verteilung ist in Root definiert als



Particle Data Group, Phys Rev D98 030001 (2018)

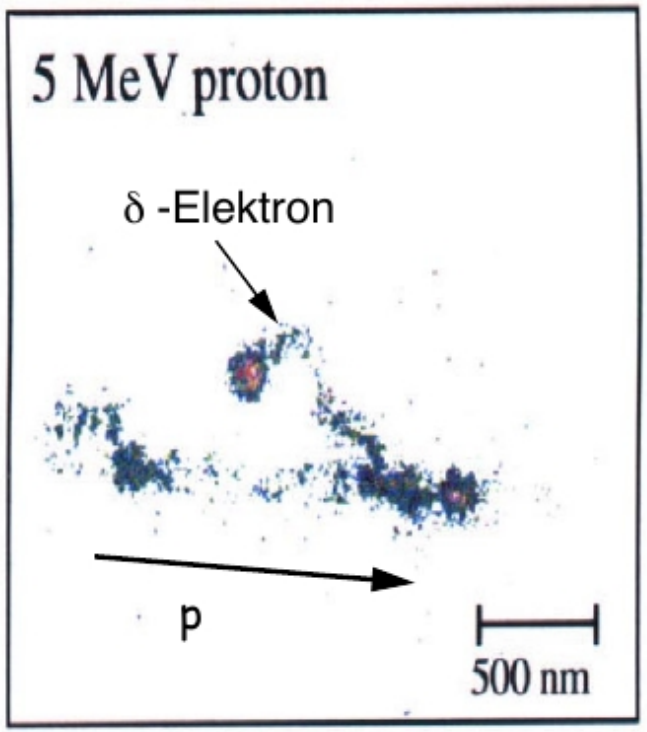
$$p(x) \approx \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}(x+e^{-x})}$$

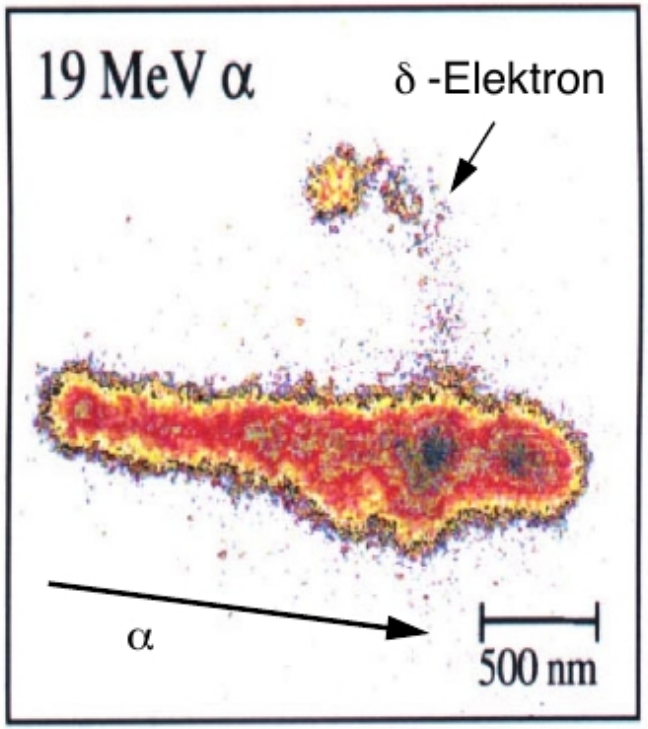
FWHM = full width half maximum
Δp = most pobable
<Δ> = mean energy loss
$M_0(\Delta)$ = cummulative 0 Moment
mean collision length
$M_1(\Delta)$ = cummulative 1 Moment
mean energy loss

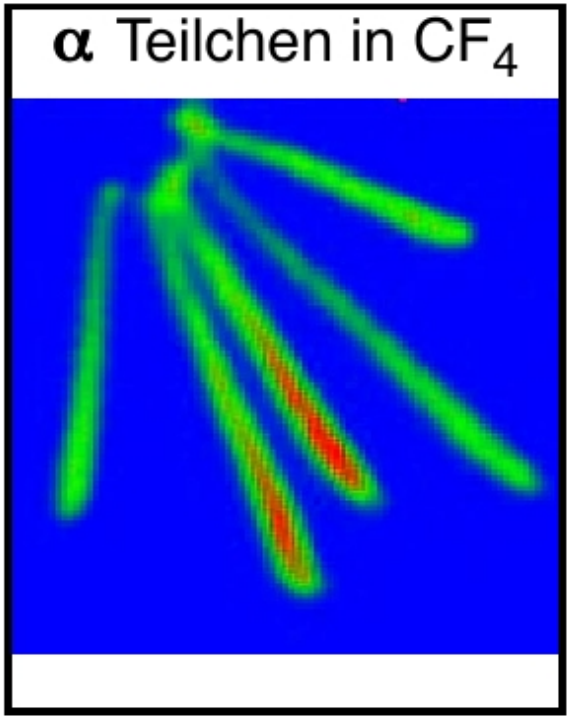
Ionization along the Track

Optical avalanche microdosimeter

Double gem microstrip gas chamber with CCD camera readout



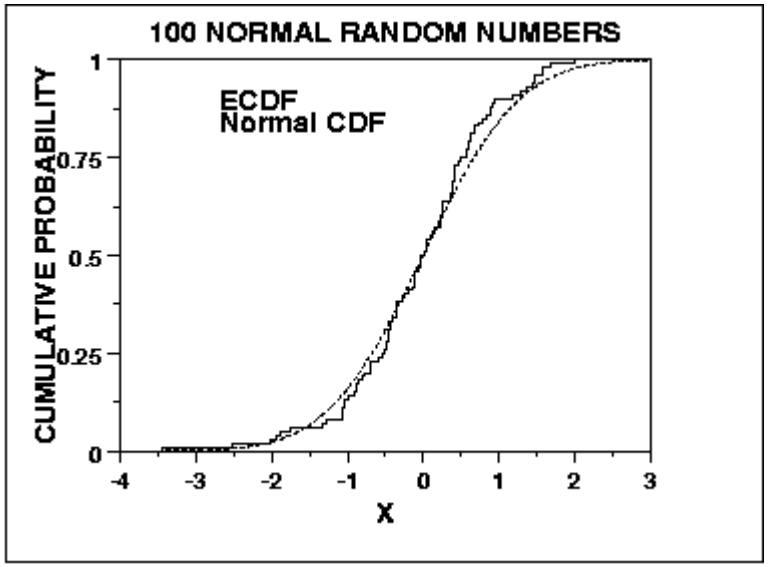




The increase of ionization at the end of the range due to the $1/\beta^2$-dependence of the energy loss is visible in all examples.

Kolgomorow Smirnov Test

Dient zur Prüfung, ob ein sample einer gegebenen Verteilung gehorcht.
Sei $x_0, x_1, x_2 ... x_i$ eine geordnete Liste von N Datenpunkten, dann ist $E_N = n(i)/N$ eine empirische Verteilung mit $n(i)$ gleich der Anzahl der Datenpunkte kleiner x_i.
Die kummulative Verteilung ist eine Treppenfunktion. Der Test bestimmt die Abweichung von 2 Verteilungen und ist gut geeignet, um die Form von Verteilungen zu vergleichen.
Details and plot: https://www.itl.nist.gov/div898/handbook/eda/section3/eda35g.htm



In ROOT gibt es die Methode TH1::KolmogorovTest() zum statistischen Test zweier Histogramme oder für unbinned data TMath::KolmogorovTest

Im File measuredLandau.txt sind die gemessenen Werte des Energieverlustes von geladenen Teilchen beim Durchgang durch flüssiges Argon gespeichert. Die Einheiten sind unkalibriert.

anaLandau.cc

Arbeitsvorschlag

- Schreiben Sie ein Programm, das die Daten liest und in einem Histogramm darstellt. Bestimmen Sie die Anzahl der Untergrund und der Signal Ereignisse.

- Wie gross ist der most Probable Value, der Median und der Mittelwert der Signalverteilung. Wie gross ist das Full Width Half Maximum?
  Wieviele Signalereignisse messen wir oberhalb von 150?

- Im File genLandau.txt sind die Werte einer simulierten Verteilung zu finden. Beschreibt diese Verteilung die Daten? Wie können wir das quantifizieren?

- Können wir die simulierten Daten so anpassen, dass die gemessenen Werte beschrieben werden?

- Das 0te Moment der Wahrscheinlichkeitsdichte Verteilung gibt die mittlere Anzahl der Kollisionen an, das 1te Moment den mittlere Energieverlust.
  Stellen Sie beide Momente der gemessenen Landau Verteilung mit TGraph dar.

C++ Wiederholung aus dem WS

- Funktionen in ROOT
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Root.pdf

- Ergänzungen
  - Detektor Antwort einer generierten p_T Verteilung

- Input / Output in ROOT
  https://www.physi.uni-heidelberg.de/~marks/root_einfuehrung/Folien/Root.pdf

- Ergänzungen
  - Schreiben beliebiger Objekte in Root files

- Vorträge: 30 min + kleines Übungsbeispiel

- STL	24.5.	S. Sonntag
- Fast Fourier Transformation (FFTW3)	17.5.	N. Holzwarth
- Wahrscheinlichkeitsverteilungen (central limit theorem)	10.5.	M. Morgenthaler
- Statistische Testverfahren	17.5	M. Griedel
- Simulationen mit GEANT	Mitte Juni	M. Piotter
- CUDA	Anfang Juli	E. Volkmann

Die p_T Verteilung von geladenen Pionen soll nach einem Modell im Bereich von 1 - 50 GeV/c mit $1/p^2_T$ verlaufen. Die relative Impulsauflösung

$$\sigma_{p_T}/p_T = \sqrt{(0.0015 \cdot p_T)^2 + (0.02)^2}$$

hat zwei Anteile (Ortsmessung und Vielfachstreuung) mit unterschiedlicher p_T Abhängigkeit.

Arbeitsvorschlag:

pTDistribution.C

- Implementieren Sie die Auflösung σ_{p_T}/p_T als Funktion von p_T einschließlich der beiden Anteile. Die Funktionen können dann graphisch dargestellt werden.

- Generieren Sie die gemessene p_T Verteilung der Pionen.

Landau Verteilung

Die Landau Verteilung beschreibt den Energieverlust von geladenen Teilchen in einer dünnen Materialschicht. Die starke Asymmetry auf der rechten Seite beruht auf der Abstrahlung von δ rays (Elektronen mit grösserer Energie). Der Mittelwert und die Varianz sind nicht als geschlossene Ausdrücke definiert Die Verteilung ist in Root definiert als

$$(\frac{\sigma_{p_T}}{p_T})_{Streuung} = \frac{0.054 \cdot sin(\theta)}{0.3LB\beta}\sqrt{\frac{L/sin(\theta)}{X_0}} \approx 0.02$$

$$(\frac{\sigma_{p_T}}{p_T})_{Ort} = \frac{p_T^2\sigma_{Ort}}{facL^2B}\sqrt{\frac{720}{N+4}} \approx 0.02$$

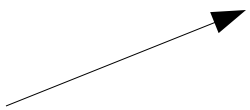
Objekte in ROOT Files

Wir können nicht nur Instanzen von `TObject` in ROOT Files schreiben/lesen, sondern auch beliebige Klassenobjekte. Dazu werden Run Time Type Information (RTTI) benötigt, d.h. die Fähigkeit einer Klasse sich selbst zu analysieren.

In computer science, reflection is the ability of a computer program to examine, introspect, and modify its own structure and behavior at runtime

In ROOT ist reflection über TClass implementiert. Hier wird Information über die Klassen, Methoden und Daten, Kommentarfelder und Parametertypen zur Verfügung gestellt. Dies erfolgt mit dem `ClassDef` macro

```
#include <TDirectory.h>
class MyClass
{
   public:
   …..
   ClassDef(MyClass,ClassVersionID);
   …..
}
```



`ClassVersionID ≥ 0`
Erlaubt eine Versionskontrolle die vom ROOT I/O benutzt werden kann

und der Erzeugung einer reflection database, einem ROOT dictionary. Für die Klasse muss ein default constructor (constructor ohne Parameter oder mit Parameter aber gesetzten default Werten) existieren.

Objekte in ROOT Files

- ROOT dictionaries

  Dictionaries werden in CLing automatisch mit ACLiC erzeugt

  ```
  root> .L MyHeader.h+
  ```

  + ruft automatisch einen Generator, der ein dictionary File, ein pcm File und eine shared library erzeugt.

  Für generelle C++ Programme werden dictionaries mit Hilfe von preprocessor link Definitionen, die in speziellen Files `LinkDef.h` untergebracht sind, und mit dem class header mit dem tool `rootcling` erzeugt.

  ```
  rootcling -f DictOut.cxx -c OPTIONS header.h … Linkdef.h
  ```

  Das File `DictOut.cxx` enthält Streamer() und ShowMembers() Methoden unserer Klasse. Dabei wird Streamer() benutzt um ein Objekt in/aus TBuffer zu streamen and ShowMembers() wird von den Dump() und Inspect() Methoden von TObject verwendet.
  Das File `LinkDef.h` definiert welche Klasse wie zum Dictionary File hinzugefügt werden sollen.

  ```
  #ifdef __CLING__
  #pragma link C++ class MyClass;
  #endif
  ```

  - :do not generate streamer method
  ! : not to generate the operator>>
  + : add byte count

Objekte in ROOT Files

Es können auch komplexe preprocessor Anweisungen verwendet werden

```
#ifdef __CLING__
#pragma link off all functions;                  Abschalten aller Funktionen
#pragma link C++ function f;                     Hinzufügen von f und g
#pragma link C++ function g(int,double);
#pragma link C++ MACRO max;                      Macro Definition
#pragma link C++ class A;                        Hinzufügen einer Klasse
#pragma link off function A::h(double);          Abschalten einer Methode
#endif
```

- Erzeugen von shared libraries

  Um die Klasse `MyClass` in einer Anwendung verwenden zu können, wird sie in Form einer shared library hinzugefügt, die auch die dictionary Information enthält

```
gcc -shared -fPIC -o libMyClass.so \
    `root-config --cflags --ldflags --glibs` \
    DictOut.cxx MyClass.cc
```

  Das File `DictOut_rdict.pcm` muss dabei im directory der shared library existieren.

- Erzeugen einer C++ Anwendung

```
g++ myApp.cc -L. -lMyClass -o myApp  \
    `root-config --ldflags --glibs --cflags `
```

Um I/O Funktionen wie das Schreiben von beliebigen Objekten in ROOT verwenden zu können, müssen wir

- das `ClassDef` macro aus `TDirectory.h` in unserer Klasse hinzufügen
- ein ROOT Dictionary mit `rootcling` erzeugen
- eine shared library bauen
- eine Anwendung erzeugen

Arbeitsvorschlag:

- Verwenden Sie unsere Klasse FourVector um das Schreiben von FourVector Objekten in ROOT Files zu testen.

- Schreiben Sie mehrere FourVector Objekte in einen ROOT Tree

- Lesen Sie den Tree wieder ein