**Relazione: Esercizio 1**

“Si costruisca un generatore di numeri casuali distribuiti secondo una distribuzione di Landau e si discuta dei possibili test di casualità utilizzabili per i numeri generati”

**Introduzione**

Lo scopo dell’esercizio è costruire un generatore di numeri casuali e successivamente testarne la casualità. Di fatto testare la casualità in senso assoluto è impossibile, quando si tratta con elaboratori infatti le sequenze casuali sono generate sempre attraverso una formula matematica. Per avere dei numeri casuali puri (“truly random numbers”) bisognerebbe sfruttare dei fenomeni naturali (fenomeni fisici) per avere una sequenza del tutto irriproducibile come ad esempio un decadimento radioattivo.

In questo caso si può ambire ad avere una sequenza di numeri “pseudocasuali”, numeri generati da un algoritmo deterministico che produce una sequenza con, approssimativamente, le stesse proprietà statistiche di una sequenza di numeri generata da un processo casuale.

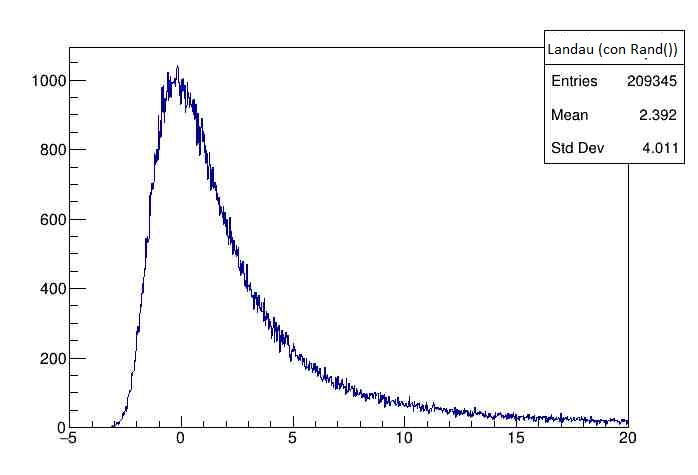
La definizione di pseudocasualità per un algoritmo implica infatti che una sequenza sia casuale solo in rapporto ad altri generatori di numeri casuali.

Per costruire un generatore di numeri casuali distribuiti secondo una distribuzione di Landau è necessario utilizzare un metodo Hit-Miss di Von Neumann. In sostanza vengono generati dei numeri casuali compresi tra gli estremi della distribuzione della PDF richiesta sì successivamente vengono generati altri numeri casuali compresi tra 0 e il massimo della funzione. Viene, quindi, effettuato un confronto tra i valori se il primo è inferiore al secondo allora questo viene accettato, in caso contrario avviene il rigetto.

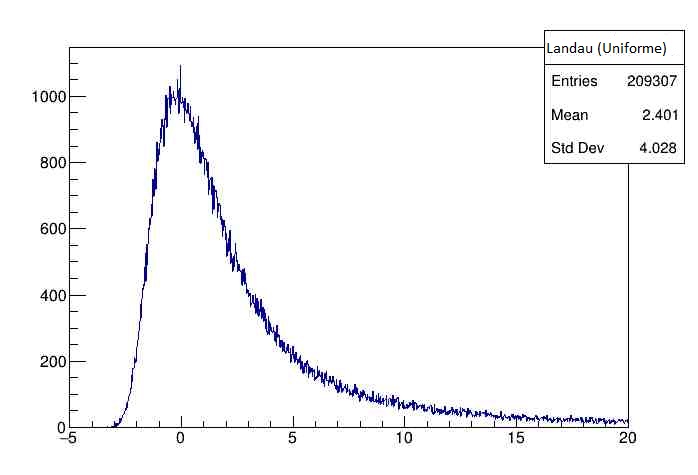
**Risultati e Discussione**

Sono stati generati due set di numeri casuali, il primo è il generatore di numeri casuali di cui voglio andare a testare la pseudocasualità mentre il secondo lo utilizzerò come riferimento per il confronto.

Il generatore che voglio testare utilizzerà la ormai sorpassata funzione rand() di C per generare i numeri casuali. I numeri vengono generati da questa funzione con un algoritmo lineare congruente (Ij+1=aIj+c(mod m) con a=1103515245, m=232, c=12345). I generatori di questo tipo non vengono più utilizzati in quanto i numeri generati se analizzati dal punto di vista tridimensionale andavano a comporre una struttura a bande problematica se utilizzati per calcoli Monte Carlo particolarmente complessi. Con un metodo Hit-Miss (descritto nell’introduzione) è stato possibile distribuirli secondo una distribuzione di Landau (si è utilizzata la versione approssimata presente nella libreria Tmath):

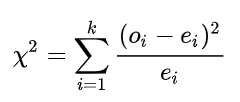


Per la seconda distribuzione è stato utilizzato invece un generatore più sofisticato, il Marsenne Twister (mt 19937) che, a differenza della funzione Rand, rispetta le proprietà della pseudocasualità. A ulteriore conferma di ciò ha superato i test statistici d Diehard (questo argomento verrà approfondito nel seminario). I numeri generati in maniera casuale ed uniforme hanno prodotto questa distribuzione (sempre con un metodo Hit-Miss):

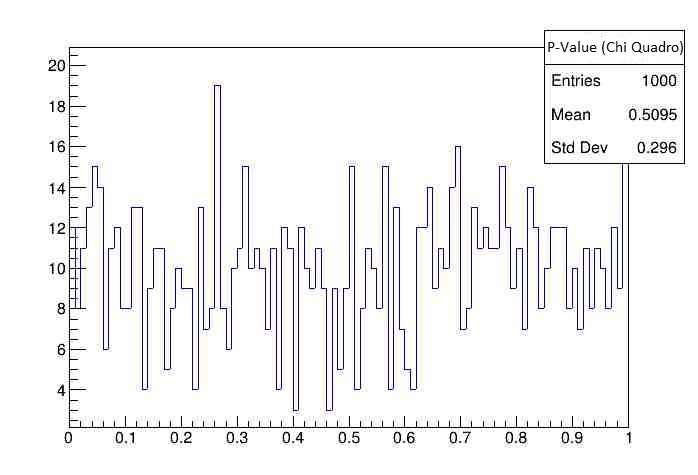


Per testare la casualità del primo generatore possiamo effettuare un test di ipotesi tenendo come riferimento di garantita casualità il secondo generatore. L’ipotesi nulla è quindi la pseudocasualità del primo generatore.

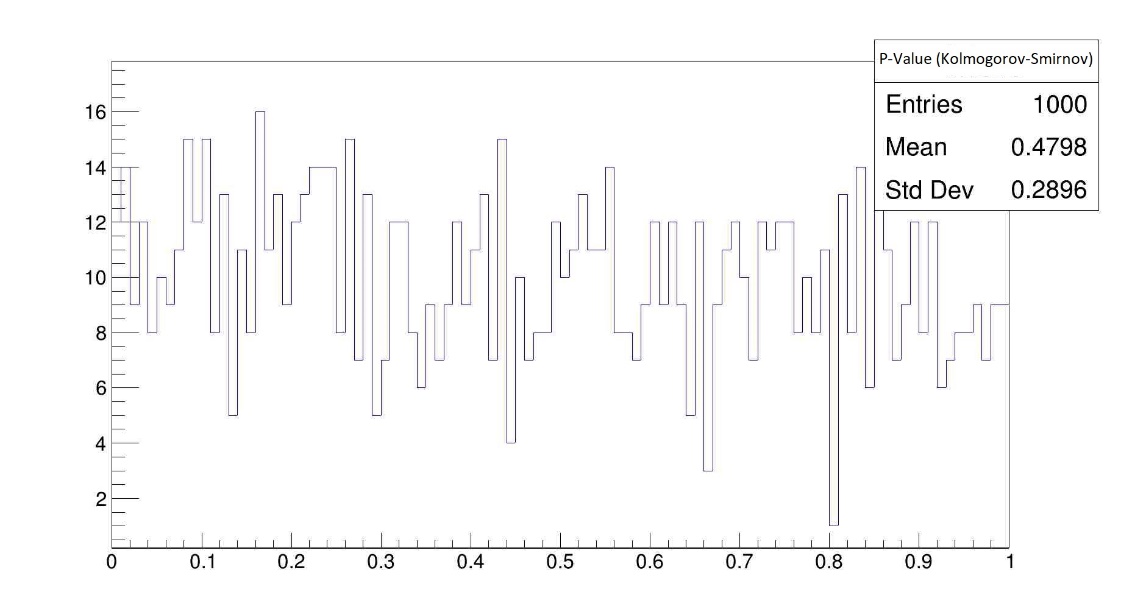
Nel programma sono stati effettuati due test, il primo è il test del chi quadro tenendo i dati binnati



Ove teniamo come valore teorico quelli del secondo generatore. Questa è la distribuzione del p-value calcolato con mille set differenti di numeri generati



Il secondo è un test di Kolmogorov-Smirnov per i dati non binnati (e quindi raccolti in vettori ordinati)



In entrambi i casi la distribuzione del p-value è uniforme, andando a confermare la pseudocasualità del generatore Rand().

Per analizzarne in maniera più raffinata la pseudocasualità era necessario un test Diehard, impossibile nei limiti della programmazione in C++. Questo verrà però analizzato nel seminario.