Uniwersytet Wrocławski

Wydział Fizyki i Astronomii

INSTYTUT FIZYKI TEORETYCZNEJ

Specjalność fizyka komputerowa

Magdalena Dukiewicz

DOSTOSOWANIE GENERATORA NUWRO DO WSPÓŁPRACY Z GENERATOREM ND280 W EKSPERYMENCIE T2K

Praca magisterska
napisana pod kierunkiem
dr Cezarego Juszczaka

Wrocław, 2008

Spis treści:

Streszczenie

„Dostosowanie generatora NuWro do generatora ND280 w eksperymencie T2K”

Głównym celem pracy było napisanie programu konwertującego plik zdarzeń tworzony przez generator NuWro, stworzony przez pracowników IFT, na format, który mógłby posłużyć jako dane wejściowe generatora ND280, używanego do symulacji bliskiego detektora w eksperymencie T2K. W części teoretycznej pracy znajduje się opis eksperymentu T2K oraz używanych w nim generatorów Monte Carlo. Omówiono również rodzaje oddziaływań neutrin oraz sposób ich implementacji w generatorze NuWro i bliźniaczym generatorze NEUT. Ze względu na temat pracy konieczne okazało się bardzo szczegółowe omówienie formatu danych wyjściowych tych generatorów. Omówiono również bibliotekę ROOT, szeroko stosowaną zarówno w NuWro jak i w ND280. Stworzony program konwertujący został zamieszczony w załączniku. Jest on również (razem z wszystkimi bibliotekami niezbędnymi do skompilowania) umieszczony na płycie CD, dołączonej do pracy. Zostało sprawdzone, że pliki przekonwertowane przez ten program są prawidłowo odczytywane przez generator ND280.

Abstract

„Conforming NuWro generator to ND280 generator in T2K experiment”

The main purpose of the thesis was to create a program converterting the output of the NuWro neutrino event generator to a format used as the input of the ND280 generator used as a simulation of the near detector in the T2K experiment. In the first part of the thesis the T2K experiment and its Monte Carlo generators are discussed. The kinds od neutrino interactions and the way they are implemented in the NuWro and Neut generators are also described. In view of the subject of the thesis it was necessary to present the details of the output data format of the Neut and NuWro geneators. The code of the converter program is listed in the appendix and (together with all the required libraries) it is recorded on the attached CD disc. It has been verified that the files converted by the program are indeed readable by the ND280 generator.

Wstęp

Fizyka neutrin jest bardzo dynamicznie rozwijającą się częścią fizyki cząstek elementarnych. Ostatni rozwój badań dotyczy głównie oscylacji neutrin. Analiza danych w doświadczeniach cząstek elementarnych opiera się na porównywaniu obserwacji z wynikami symulacji Monte Carlo. Jednym z takich eksperymentów jest nowy eksperyment T2K (Tokai To Kamioka) z długą bazą. Do przeprowadzenia symulacji potrzebne są generatory, których zadaniem jest produkcja cząstek, generacja przypadków, a także propagacja cząstek w różnych materiałach. Generatory muszą współdziałać z sobą, tak aby wyniki każdego z nich stanowiły odpowiedni zbiór danych gotowy do przetwarzania w kolejnym etapie symulacji. Czasem istnieje potrzeba stworzenia dodatkowych programów, które pozwolą na dostosowanie danych. Dlatego też, stworzenie programu, który pozwoli przetworzyć dane wyjściowe generatora przypadków NuWro, do danych, które są wymagane na wejściu generatora ND280 było głównym celem mojej pracy.

W pierwszym rozdziale tej pracy został przybliżony projekt eksperymentu T2K, a także generatory biorące udział w symulacjach Monte Carlo tego projektu. Wrocławski Generator NuWro, którego charakterystyka została także zawarta w tym rozdziale, może stać się jedną z części symulacji w eksperymencie T2K.

Rozdział drugi poświęcony jest omówieniu oddziaływań neutrin (rozpraszanie kwazielastyczne, rezonansowe, głęboko nieelastyczne, koherentne). Zawarty został opis realizacji oddziaływań w generatorach NEUT i NuWro. Oprócz klasyfikacji oddziaływań w obu generatorach, w rozdziale tym znajduje się sposób zamiany oddziaływań, który potrzebny jest do realizacji zadania.

Ponieważ zarówno NuWro jak i ND280 korzystają z programu ROOT, dlatego został on opisany w rozdziale trzecim. Przykłady dotyczące zastosowania ROOTa znalazły się w dodatku A. Dokładna specyfikacja wyjściowych danych z NuWro i wejściowych danych z ND280 została zawarta w rozdziale czwartym. Kod programu, będącego realizacją zadania wraz z omówieniem znajdują się w dodatku B.

Rozdział 1 Generatory w eksperymencie T2K

1.1 Eksperyment T2K

Projekt T2K (Tokai To Kamioka) to eksperyment neutrinowy, którego rozpoczęcie przewiduje się na rok 2009. Główną jego częścią ma być już istniejący detektor SuperKamiokande. Jest to wodny detektor o pojemności 50 000 000 l**.** wykorzystujący zjawisko Czerenkowa. Celem T2K jest użycie wiązki o energii 50 GeV, pochodzącej z akceleratora, do produkcji wiązki neutrin mionowych skierowanej w stronę Super-Kamiokande, a następnie rejestracja ich oddziaływań. Wiązka neutrin o średniej energii równej 1 GeV ma być wysyłana z JPARC (Japan Proton Accelerator Research Complex) w Tokai na wschodnim wybrzeżu Japoni. Odległość między JPARC i SuperKamiokande wynosi 295 kilometrów. W eksperymencie planowane są cztery stanowiska z detektorami: detektor na osi wiązki (INGRID), detektor pozaosiowy (ND280) w odległości 280 metrów, detektor w odległości 2 kilometrów od JPARC, w którym planuje się także budowę ciekłoargonowej komory TPC o masie 100 ton oraz detektor SuperKamiokande.

Eksperyment T2K został podzielony na dwa etapy. W pierwszym etapie planowane są pomiary kąta θ_{13} (ν_e appearance) oraz precyzyjne pomiary typu ν_μ disappearance (θ_{23} oraz Δm^2_{23}), z wykorzystaniem wiązki protonowej o mocy 0.75 MW do produkcji wiązki neutrin oraz istniejącego detektora SuperKamiokande. W drugim etapie planuje się zwiększenie mocy wiązki do 4 MW oraz powiększenie dalekiego detektora do 1 Megatony (HyperKamiokande), co ma zaowocować możliwością dokonania pomiarów łamania symetrii CP oraz sprawdzenie hierarchii mas neutrin.

Polski wkład do eksperymentu T2K to prace dla ciekłoargonowego detektora w odległości 2km (T2KLAr) oraz prace dla pozaosiowego detektora ND280, a dokładniej detektora SMRD (detekcja mionów). Zadaniem detektora pozaosiowego jest pomiar przekrojów czynnych na węglu i wodzie, pomiar strumienia wiązki i jej profilu energetycznego, a także pomiar tła.

1.2 Generatory Monte Carlo

Metoda Monte Carlo jest to metoda numerycznego rozwiązywania złożonych problemów probabilistycznych poprzez bezpośrednią symulację komputerową danego zjawiska, przy równoczesnym wykorzystaniu liczb pseudolosowych. Istotną rolę w metodzie Monte Carlo odgrywa losowanie (wybór przypadkowy) wielkości charakteryzujących proces, przy czym losowanie dotyczy rozkładów znanych wcześniej (np. z badania procesów prostszych lub niekiedy – z odpowiednio uzasadnionych lub oczywistych założeń).

Generatory Monte Carlo są narzędziami szeroko stosowanymi w fizyce cząstek elementarnych. Można je zastosować zarówno do opisu powstawania cząstek w wyniku oddziaływania jak i propagacji. Dane doświadczalne są zawsze porównywane z wynikami symulacji Monte Carlo. Konstrukcja generatorów oddziaływań cząstek składa się z dwóch etapów. Pierwszy etap to opis oddziaływania na swobodnym nukleonie, a drugi to uwzględnienie efektów jądrowych.

Zadaniem generatora oddziaływań neutrin jest symulowanie oddziaływania neutrina z materią detektora, czyli oddziaływania pierwotne, a także przejście przez jądro z uwzględnieniem kaskady oddziaływań. Wynikiem działania takiego generatora jest lista cząstek wychodzących z jądra po oddziaływaniu wraz z dodatkowymi informacjami. Powstało już kilka generatorów neutrin, m.in. NUANCE, NUGEN, NEUT, NUX+FLUKA, NuWro. Są one dostosowywane do doświadczeń, w których biorą udział, dlatego różnią się między sobą.

Metoda Monte Carlo stosowana jest do wyboru wszystkich znaczących zmiennych zgodnie z żądanym rozkładem prawdopodobieństwa.

Stosuje się generatory zdarzeń, dlatego że:

- Są czynnikiem informacyjnym na temat rodzaju zdarzeń, jakich należy oczekiwać w doświadczeniu.
- Umożliwiają poprawę jakości działania detektorów, a także planowanie nowych, tak aby ich właściwości były optymalne.
- Pomagają obmyślać strategię, jaka powinna być użyta przy analizie rzeczywistych danych, tak aby szukany sygnał był skutecznie odseparowany od tła.

- Umożliwiają ilościową ocenę, w jaki sposób nowe zjawiska fizyczne wpływają na obserwable.

1.3 Generatory w doświadczeniu T2K

Przed uruchomieniem eksperymentu trzeba opracować procedury, które będą potrzebne w trakcie jego trwania, np. rekonstrukcja przypadków, kalibracja, z tego powodu istnieje potrzeba przeprowadzenia symulacji. Dzięki symulacjom można opracować algorytmy rekonstrukcji energii, torów cząstek, określać poziom tła.

Dla ND280 wyróżniamy następujące elementy symulacji:

- symulacja wiązki (jnubeam),
- generacja przypadków (NEUT),
- propagacja przez materię (ND280),
- symulacja elektroniki (elecSim).

Efektem symulacji mogą być takie same pliki jakie będziemy dostawać w czasie działania eksperymentu.

Generator ND280 jest przeznaczony do symulowania propagacji cząstek w detektorze z uwzględnieniem efektów materiałowych i geometrycznych. Ma być na tyle szczegółowy i ogólny, żeby stać się częścią analizy fizyki T2K. Wykorzystuje bibliotekę GEANT. Generator ND280 wymaga danych wejściowych, które dostarcza mu inny generator, obecnie jest to japoński generator NEUT.

NEUT jest biblioteką programów do symulacji interakcji neutrin. Biblioteka została rozwinięta tak, aby badać atmosferyczne neutrina oraz rozpady jąder w Kamiokande i jest ciągle unowocześniana dla eksperymentu Super-Kamiokande oraz eksperymentu K2K, T2K. Zakres energii neutrin, których reakcje mogą być symulowane przez NEUT jest od 100MeV do 1TeV. Atmosferyczne neutrina i neutrina z akceleratora zachodzą w reakcje z wodą w detektorze Super-Kamiokande. Dlatego generator jest zaprojektowany by symulować interakcje z protonem i tlenem. W jądrze tlenu wytwarzane są cząstki takie jak: piony, kaony, a rozproszone nukleony zachodzą w reinterakcje przed opuszczeniem jądra. Z tego powodu wtórne reakcje wytworzonych cząstek są także symulowane.

1.4 Generator NuWro

Autorami generatora NuWro jest Wrocławska Grupa Neutrino, a w szczególności dr Cezary Juszczak, dr Jarosław Nowak, prof. Jan Sobczyk, dr Krzysztof Graczyk. Generator ten powstał w 2005 r. i jest nadal rozwijany.

Generator jest zorganizowany wokół struktury zdarzenia, które zawiera trzy zestawy wektorów dla cząstek wchodzących, tymczasowych, końcowych. Wszystkie wektory są zapisane w układzie laboratoryjnym. W zestawie cząstek wchodzących zapisane są informacje o neutrinie i nukleonie. W przypadku kiedy tarczą jest swobodny nukleon w zestawie wektorów tymczasowych zapisane są wektory cząstek pośrednich powstałych w wyniku hadronizacji, a cząstki końcowe są zapisane w zestawie wektorów końcowych. Obecnie trwają prace dotyczące kaskady oddziaływań w jądrze. W wyniku jej działania powstanie czwarty zestaw cząstek, który będzie zawierał wszystkie cząstki wychodzące z jądra. W przypadku rozpraszania na jądrach, wektory tymczasowe zawierają cząstki po hadronizacji, a zestaw wektorów końcowych zawiera opis cząstek po uwzględnieniu efektów jądrowych.

Produkcja zdarzenia w generatorze przebiega w opisanych poniżej etapach:

1) Wczytanie pliku *params.txt* – parametry do rozpoczęcia generacji zdarzeń.
2) Wybranie energii neutrina – jest to stała albo wartość wylosowana z podanego rozkładu
3) Ustalenie tarczy (jądro, nukleon swobodny). Jeśli tarcza nie jest jądrem to wybranie oddziałującego nukleonu i jego pędu z kuli Fermiego (w przypadku swobodnych cząstek pęd Fermiego równa się zero). Dokonywany jest wybór opisu tarczy (gaz Fermiego, funkcja spektralna). Trzeba też uwzględnić inne parametry tj. przybliżenie lokalnej gęstości, efektywny potencjał.
4) Wybór dynamiki: rozpraszanie kwazielastyczne, rozpraszanie elastyczne, rozpraszanie nieelastyczne. Dynamika jest wybierana według obliczanego w trakcie generacji zdarzeń próbnych stosunku przekrojów czynnych.
5) Wykonanie procesu. Obliczenie przekroju czynnego i wyznaczenie stanów końcowych.
6) Zapisanie do pliku całego zdarzenia: stany początkowe, stany końcowe, oznaczenie rodzaju dynamiki i kilku dodatkowych parametrów.

W strukturze zdarzenia zapisane są informacje o użytych parametrach. Wejściowe parametry są czytane na początku symulacji z pliku tekstowego, a zdarzenia są zbierane w pliku w formacie pakietu ROOT, w celu uproszczenia dalszej analizy i zmniejszenia wykorzystywanych zasobów systemowych. NuWro napisane jest w języku programowania C++ z wykorzystaniem ROOTA, a w szczególności pakietu PYTHIA.

Rozdział 2 Oddziaływania neutrin w generatorach NEUT, NuWro

Neutrino jest to cząstka elementarna, należąca do leptonów, posiadająca zerowy ładunek elektryczny oraz niewielką masę. Istnieją trzy stany zapachowe neutrin: neutrino elektronowe, neutrino mionowe, neutrino taonowe. Podczas propagacji w przestrzeni, mogą zmieniać swój rodzaj (zapach). Zjawisko to nazywane jest oscylacją neutrin.

Powstają jako uboczny produkt reakcji przemiany wodoru w hel w gwiazdach. Te, które pochodzą ze Słońca nazywa się neutrinami słonecznymi. Neutrino, które rodzi się w środku Słońca, dociera do Ziemi już po upływie 8,3 min., podczas gdy foton światła musi przedzierać się stamtąd ku powierzchni gwiazdy ponad milion lat. Dlatego lepiej niż światło zdradzają co się dzieje aktualnie we wnętrzu gwiazdy. Ponadto unoszą ze sobą większą część energii eksplozji (tzw. Supernowe) i docierają do Ziemi niemal niezatrzymane.

2.1 Teoria oddziaływania neutrin

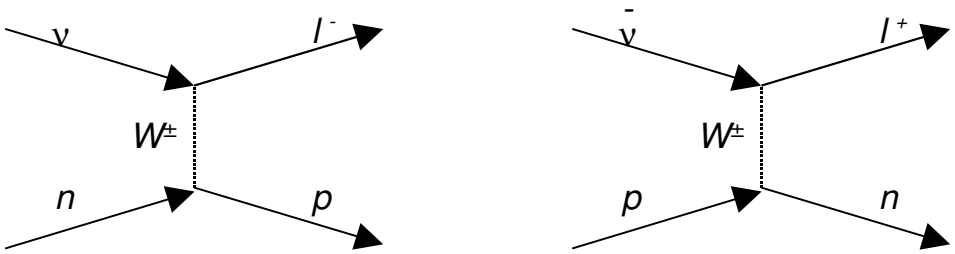
Neutrina oddziałują tylko słabo przez wymianę masywnych bozonów pośredniczących: $W^{\pm}$ i Z^0 , dlatego ich rejestracja jest zawsze pośrednia. Oddziałują przez prądy:
- naładowany CC *charge current* ($W^{\pm}$ pośredniczące bozony mają ładunek),
- neutralny NC *neutral current* (Z^0 pośredniczące bozony pozbawione są ładunku).

W opisie oddziaływania neutrin używa się kilku modeli:

- rozpraszanie kwazielastyczne (dla NC – elastycznego),
- wzbudzanie rezonansów,
- procesy rozpraszania głęboko nieelastycznego.
- rozpraszania koherentne (na całym jądrze)

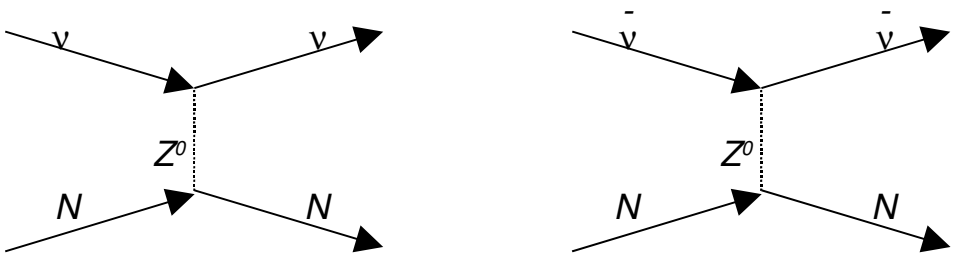
2.1.1 Rozpraszanie kwazielastyczne

Zderzenia kwazielastyczne w przypadku rozpraszania neutrina na swobodnym nukleonie, to proces, w którego wyniku neutrino przekształca się w lepton naładowany, a nukleon zmienia stan ładunkowy i nie powstają żadne dodatkowe cząstki, co przedstawia rysunek 1.



rysunek 1: *Oddziaływania z wymianą ładunku*

Ze względu na zasady zachowania protony mogą zderzać się kwazielastycznie tylko z antyneutrinami, a neutrony — tylko z neutrinami. Analogiczne oddziaływanie przez prądy neutralne określa się mianem elastycznego, patrz rysunek 2.



rysunek 2: *Oddziaływania bez wymiany ładunku*

W NEUT przekrój czynny dla kwazielastycznego oddziaływania z wymianą ładunku jest przedstawiony na zasadzie jak wymiana hadronów.

$$\langle N'|J_{had,\lambda}|N\rangle = \cos\theta_c\,\bar{u}(N')[\gamma_\lambda F_V^1(q^2) + \frac{i\sigma_{\lambda\nu}q^\nu \xi F_V^2(q^2)}{2\mathrm{M}} + \gamma_\lambda\gamma_5 F_A(q^2)]u(N)$$

gdzie F_V^1, F_V^2, są form faktorami, F_A jest aksjalnym form faktorem,

$\xi \equiv \mu_p - \mu_n = 3.71$ i θ_c jest kątem Cabibbo. Parametry form faktorów są wyznaczone z eksperymentu. Jeśli tarczą jest swobodny proton zależność q^2 przekroju czynnego jest bezpośrednio wyliczana z równania.

Do estymacji przekroju czynnego elastycznego rozpraszania bez wymiany ładunku używa się następujących relacji:

$$\begin{aligned}
\sigma(\nu p \to \nu p) &= 0.153 \times \sigma(\nu n \to e^- p) \\
\sigma(\bar{\nu} p \to \bar{\nu} p) &= 0.218 \times \sigma(\bar{\nu} n \to e^+ p) \\
\sigma(\nu n \to \nu n) &= 1.5 \times \sigma(\nu p \to \nu p) \\
\sigma(\bar{\nu} n \to \bar{\nu} n) &= 1.0 \times \sigma(\bar{\nu} p \to \bar{\nu} p)
\end{aligned}$$

W NuWro przekrój czynny obliczany jest według modelu opisanego przez C. H. Llewellyn Smith [L1].

2.1.2 Rozpraszanie rezonansowe

W wyniku oddziaływania neutrina z nukleonem w obszarze kilku GeV powstaje rezonans oraz lepton. Jest to zazwyczaj rezonans Δ , który rozpada się na nukleon i pion.

W generatorze NEUT przyjmuje się metodę Reina - Sehgala by symulować te interakcje. W tej metodzie rozpraszanie rezonansowe jest podzielone na dwie części:

$$\begin{aligned}
\nu + N &\to l + N^* \\
N^* &\to \pi + N'
\end{aligned}$$

gdzie N i N' są nukleonami, N^* jest rezonansem barionowym. Aby otrzymać przekrój czynny oblicza się amplitudę każdej produkcji rezonansowej, a później mnoży przez prawdopodobieństwo rozpadu na pion i nukleon dla każdego rezonansu. Interferencje między rezonansami są także brane pod uwagę. W obliczaniu przekroju czynnego dla tych interakcji wzięto pod uwagę 18 rezonansów poniżej 2GeV. Aby uniknąć podwójnego liczenia interakcji w symulacji, niezmiennicza masa hadronów (W), która jest masą rezonansu, przez który zachodzi reakcja, jest ograniczona do mniej niż 2GeV. W obliczeniu kątowego rozkładu pionu w końcowym stanie używa się metody Reina - Sehgala dla rezonansu $P_{33}(1232)$. Możliwym jest obliczenie przekroju czynnego dla każdego indywidualnego rezonansu w tej metodzie jeśli

zignoruje się zakłócenia.

Jak opisano model Reina - Sehgala dostarcza amplitudę produkcji rezonansu (przez neutrino). Dlatego jest także możliwe obliczenie przekroju czynnego produkcji pojedynczego K oraz η poprzez zmianę rozpadu rezonansów.

W generatorze NuWro uwzględnia się jeden rezonans Δ . Zastosowane jest gładkie przejście do DIS (rozpraszanie głęboko nieelastyczne). Sposób traktowania obszaru rezonansowego był przedmiotem doktoratu Jarosława Nowaka [N1]. Jest to połączenie oddziaływania na pierwszym rezonansie z modelem DIS oraz z modułem fragmentacji z biblioteki PYTHIA 6 ale dostosowanej do fragmentacji na poszczególnych konfiguracjach kwarków.

2.1.3 Rozpraszanie głęboko nieelastyczne

Rozpraszanie głęboko nieelastyczne można rejestrować przy dużych energiach i przekazach pędu. Ze względu na dużą energię może powstawać wiele cząstek, a stany końcowe są bardzo różnorodne.

Przekrój czynny tego oddziaływania w generatorze NEUT jest wyliczony przez scałkowanie równania

$$\frac{d^2\sigma}{dxdy} = \frac{G_F^2 M_N E_\nu}{\pi}((1-y+\frac{1}{2}y^2+C_1)F_2(x,q^2)\pm y(1-\frac{1}{2}y+C_2)[x F_3(x,q^2)])$$

$$C_1 = \frac{y M_l^2}{4\mathrm{M}_N E_\nu x}-\frac{xy M_N}{2E_\nu}-\frac{M_l^2}{4E_\nu^2}-\frac{M_l^2}{2M_N E_\nu x}, \qquad C_2 = -\frac{M_l^2}{4M_N E_\nu x}$$

dla $W > 1.3$GeV/c, gdzie W jest masą niezmienniczą układu hadronowego.

W powyższym równaniu $x = -\frac{q^2}{2\mathrm{M}(E_\nu-E_l)}$, $y = -\frac{E_\nu-E_l}{E_\nu}$ M_N jest masą nukleonu, M_l jest masą leptonu, $E_\nu i E_l$ są energiami początkowego neutrina i końcowego leptonu w układzie laboratoryjnym. Używamy GRV94 do funkcji struktur nukleonu: F_2 i xF_3. W rzeczywistych obliczeniach, aby otrzymać przekrój czynny, używamy funkcji prawdopodobieństwa wielu pionów, która jest funkcją W i daje prawdopodobieństwo

wygenerowania więcej niż 1 pionu w małym zakresie $W \leqslant 2\mathrm{GeV}$. Całkowite przekroje czynne z wymianą ładunku zawierają kwazielastyczne rozpraszanie, produkcję pojedynczych mezonów oraz głęboko nieelastyczne rozpraszanie.

Aby otrzymać przekrój czynny dla produkcji wielu pionów indukowanych bez wymiany ładunku, używamy następujących relacji:

$$\frac{\sigma(\nu N \to \nu X)}{\sigma(\nu N \to \mu^- X)} = 0.26 \quad (E_\nu \leq 3\mathrm{GeV}),$$

$$\frac{\sigma(\nu N \to \nu X)}{\sigma(\nu N \to \mu^- X)} = 0.26 + 0.04 \times \frac{E_\nu - 3}{3} \quad (3\mathrm{GeV} < E_\nu < 6\mathrm{GeV}),$$

$$\frac{\sigma(\nu N \to \nu X)}{\sigma(\nu N \to \mu^- X)} = 0.30 \quad (E_\nu \geq 6\mathrm{GeV}),$$

$$\frac{\sigma(\bar{\nu} N \to \bar{\nu} X)}{\sigma(\bar{\nu} N \to \mu^+ X)} = 0.39 \quad (E_\nu \leq 3\mathrm{GeV}),$$

$$\frac{\sigma(\bar{\nu} N \to \bar{\nu} X)}{\sigma(\bar{\nu} N \to \mu^+ X)} = 0.39 - 0.02 \times \frac{E_\nu - 3}{3} \quad (3\mathrm{GeV} < E_\nu < 6\mathrm{GeV}),$$

$$\frac{\sigma(\bar{\nu} N \to \bar{\nu} X)}{\sigma(\bar{\nu} N \to \mu^+ X)} = 0.37 \quad (E_\nu \geq 6\mathrm{GeV}).$$

Wartości są estymowane przez wyniki eksperymentów [MV1] [K1].

Ażeby wygenerować zdarzenia używa się zarówno PYTHIA/JetSet jak i kodów NEUTa, ponieważ PYTHIA/JetSet była rozwijana, aby symulować interakcje wysokich energii i nie pasuje do niższych energii. Z tego powodu używa się eksperymentalnych wyników aby wygenerować zdarzenie, którego *W* jest mniejsza niż 2GeV/c. Dla tych zdarzeń zastosowano skalowanie (KNO).

W NuWro stosuje się model partonowy R. P. Feynmana [F1] z uwzględnieniem skalowania oraz z funkcjami rozkładu partonów GRV94 i GRV98, a także modyfikacjami opisanymi w rozprawie doktorskiej J. Nowaka. Do fragmentacji oraz hadronizacji używa się PYTHIA 6.

2.1.4 Rozpraszanie koherentne

Oddziaływanie koherentne jest to oddziaływanie na całym jądrze. W wyniku oddziaływania neutrina jednocześnie z całym jądrem powstają piony. Oddziaływanie to zachodzi dla niewielkich przekazów pędów. Liczby kwantowe tj. ładunek, spin, izospin opisujące jądro atomu nie ulegają zmianie.

Różniczkowy przekrój czynny koherentnej produkcji pionów jest wyrażony następującym wzorem [RS1].

$$\frac{d^3\sigma}{dQ^2\,dydt} = \frac{G^2 M_N}{2\pi^2} f_\pi^2 A^2 E_\nu (1-y) * \frac{1}{16\pi} [\sigma^{\pi N_{tot}}]^2 \times (1+r^2) \left(\frac{m_A^2}{m_A^2+Q^2}\right)^2 \mathrm{e}^{-b|t|} F_{\mathrm{abs}},$$

$$r = \frac{\Re f_{piN}(0)}{\Im f_{piN}(0)}$$

gdzie Q^2 jest kwadratem czteropędu transferu leptonu, t jest kwadratem czteropędu przekazanego do jądra, m_A jest masą aksjalną, $f_\pi = 0.93 \mathrm{m}_\pi, b = 80 \mathrm{GeV}^{-2}$, G jest stałą słabego oddziaływania. M_N jest masą nukleonu $y = \frac{E_\nu - E_l}{E_\nu}$ jest częścią straty energii leptonów, $E_\nu i E_l$ są energiami neutrina i leptonu, i A (=16) jest liczbą atomową tlenu. F_{abs} oznacza absorpcję pionu w jądrze, które oznacza się:

$$F_{\mathrm{abs}} = \mathrm{e}^{-\langle x \rangle / \lambda},$$
$$\lambda^{-1} = \sigma_{inel}^{\pi N} \rho,$$

gdzie $\langle x \rangle$ jest średnią drogą (swobodną) pionu w tlenie. $\rho = \left(\frac{4\pi}{3} R^3\right)^{-1}$ jest gęstością jądra gdzie R jest promieniem. $\sigma_{tot}^{\pi N}$ i $\sigma_{inel}^{\pi N}$ są uśrednionymi całkowitymi i nieelastycznymi przekrojami czynnymi pion-nukleon, które zostały otrzymane w wyniku eksperymentu i pasują one do rezultatów otrzymanych przez Reina i Sehgala.

2.1.5 Efekty jądrowe

W NEUT wszystkie oddziaływania są traktowane przy użyciu modelu kaskadowego. Wśród nich bardzo ważne są oddziaływania pionów. Rozważane są następujące oddziaływania pionów w jądrze 0^{16}: nieelastyczne rozpraszanie, wymiana ładunku i absorpcja. Rzeczywista procedura symulacji wygląda w następujący sposób. Wygenerowana pozycja pionu w jądrze jest wyznaczona według rozkładu gęstości jądra typu Woods-Saxon. Sposób oddziaływań jest wyznaczony przy użyciu obliczonej średniej drogi swobodnej każdego oddziaływania. Do wyliczenia średnich dróg swobodnych używa się modelu Oset [O1]. W obliczeniach wzięto pod uwagę: ruch Fermiego w jądrze O^{16} i efekt zakazu Pauliego. Wyliczona średnia droga swobodna zależy nie tylko od pędu pionów ale również pozycji pionów w jądrze. Jeśli pojawia się nieelastyczne rozproszenie lub wymiana ładunku, kierunek i pęd pionu są wyznaczone przez analizę zmiany fazy otrzymanych z eksperymentu rozpraszania $\pi - N$. Przy obliczaniu amplitudy rozproszenia pionu bierze się pod uwagę również efekt zakazu Pauliego przez wymóg, że pęd po oddziaływaniu musi być większy niż pęd na powierzchni Fermiego $(p_F(r))$ w punkcie oddziaływania zdefiniowanym następująco:

$$p_F(r)=[\frac{3}{2}\pi^2\rho(r)]^{\frac{-1}{3}}$$

gdzie $\rho(r)=\frac{Z}{A}\bar{\rho}(1+\exp\frac{|r|-c}{a})^{-1}$ A jest liczbą masową, Z jest liczbą atomową, $\bar{\rho}$ jest średnią gęstością jądra, a i c są parametrami gęstości jądra. Symulacja oddziaływania pionu jest testowana używając następujących trzech oddziaływań: rozpraszanie $\pi^{12}C$, rozpraszanie $\pi^{12}O$ i fotoprodukcja pionu $(\gamma+{}^{12}C\rightarrow\pi^{-}+X)$.

Oddziaływania kaonów w tlenie są przeprowadzane przy użyciu podobnej metody do symulowania interakcji pionów. Różniczkowy przekrój czynny i wyznaczenie kinematyki są zrobione przy użyciu rezultatów eksperymentów rozproszonych KN i $\bar{K}N$ [MP1].

Została również uwzględniona absorpcja η $(\eta N\rightarrow N^{*}\rightarrow\pi(\pi)N)$. Ze względu na efektywność i tło ten proces wpływa na estymację rozpadu jądra. Odpowiednie rezonanse wynoszą *N(1540)* i *N(1650)*, przekrój czynny dla $(\eta N\rightarrow\pi(\pi)N)$ jest wyliczony przy użyciu poniższej formuły,

$$\sigma = \frac{\pi}{k^2}(J+\frac{1}{2})\frac{\Gamma_{\eta N}\Gamma_{\pi(\pi)N}}{(W-M^*)^2+\Gamma_{tot}/4}$$

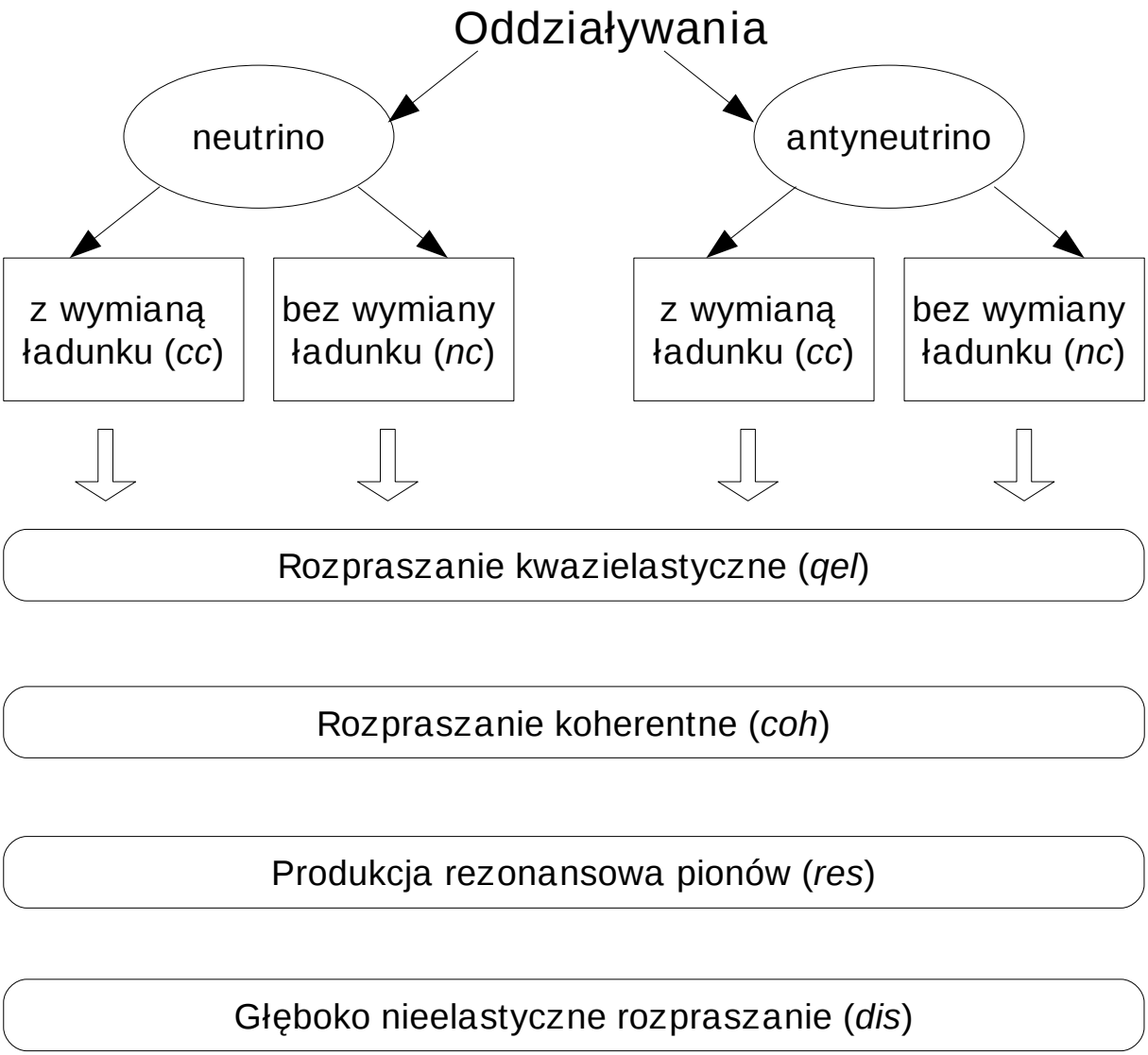
gdzie k – pęd η w środku masy układu, J spin rezonansu, G jest szerokością rezonansu, W jest niezmienniczą masą układu ηN ; M^* jest masą rezonansu. Piony z tego procesu są wyśledzone jak opisano wcześniej.

Aby symulować rozproszanie używa się modelu kaskadowego, który podobny jest do symulacji pionów. Użyte różniczkowe przekroje czynne zostały uzyskane z eksperymentu rozproszenia jądro – jądro [B1]. Rozważanymi interakcjami są: elastyczne rozpraszanie, oraz produkcja pojedynczej albo podwójnej delty. Dla tych delta produkcji używamy modelu produkcji izobar.

W modelu jądra atomowego w NuWro dominuje model gazu Fermiego, z którego wynika ruch Fermiego i zakaz Pauliego, przy czym możliwe jest uwzględnienie potencjału zależnego od pędu oraz przybliżenie lokalnej gęstości (promień Fermiego zależny od lokalnej gęstości). Efekty jądrowe w NuWro włącza się za pomocą flag. Kaskada zaimplementowana jest na podstawie pracy N. Metropolisa (1958). Aktualnie trwają prace nad uzgodnieniem efektów jądrowych obecnych w kaskadzie z poszczególnymi rodzajami dynamiki.

2.2 Klasyfikacja oddziaływań w NuWro

Sposób oddziaływania w generatorze NuWro opisany jest za pomocą wartości przyjmowanych przez odpowiednie flagi. Flaga *anty* informuje o tym, czy w danym zdarzeniu występuje antyneutrino czy neutrino. Istnieją dwa rodzaje oddziaływania ze względu na ładunek: z wymianą bądź zachowaniem ładunku. Jeśli oddziaływanie jest z wymianą ładunku, wtedy flaga *cc* przyjmuje wartość jeden, w przeciwnym wypadku flaga *nc* przyjmuje wartość jeden. Każde z tych dwóch oddziaływań można podzielić na cztery rodzaje: rozpraszanie koherentne, rozpraszanie kwazielastyczne, rozpraszanie głęboko nieelastyczne oraz produkcja rezonansowa pionów. Na rysunku 3 został zawarty omówiony podział.



rysunek 3: *Klasyfikacja oddziaływań w NuWro*

2.3 Klasyfikacja oddziaływań w NEUT

W NEUT następujące reakcje są rozważane:

Z wymianą / bez wymiany ładunku kwazielastyczne rozpraszanie $(\nu N \to l N')$

Z wymianą / bez wymiany ładunku produkcja pojedynczego π $(\nu N \to l N' \pi)$

Z wymianą / bez wymiany ładunku produkcja pojedynczego K $(\nu N \to l \Lambda K)$

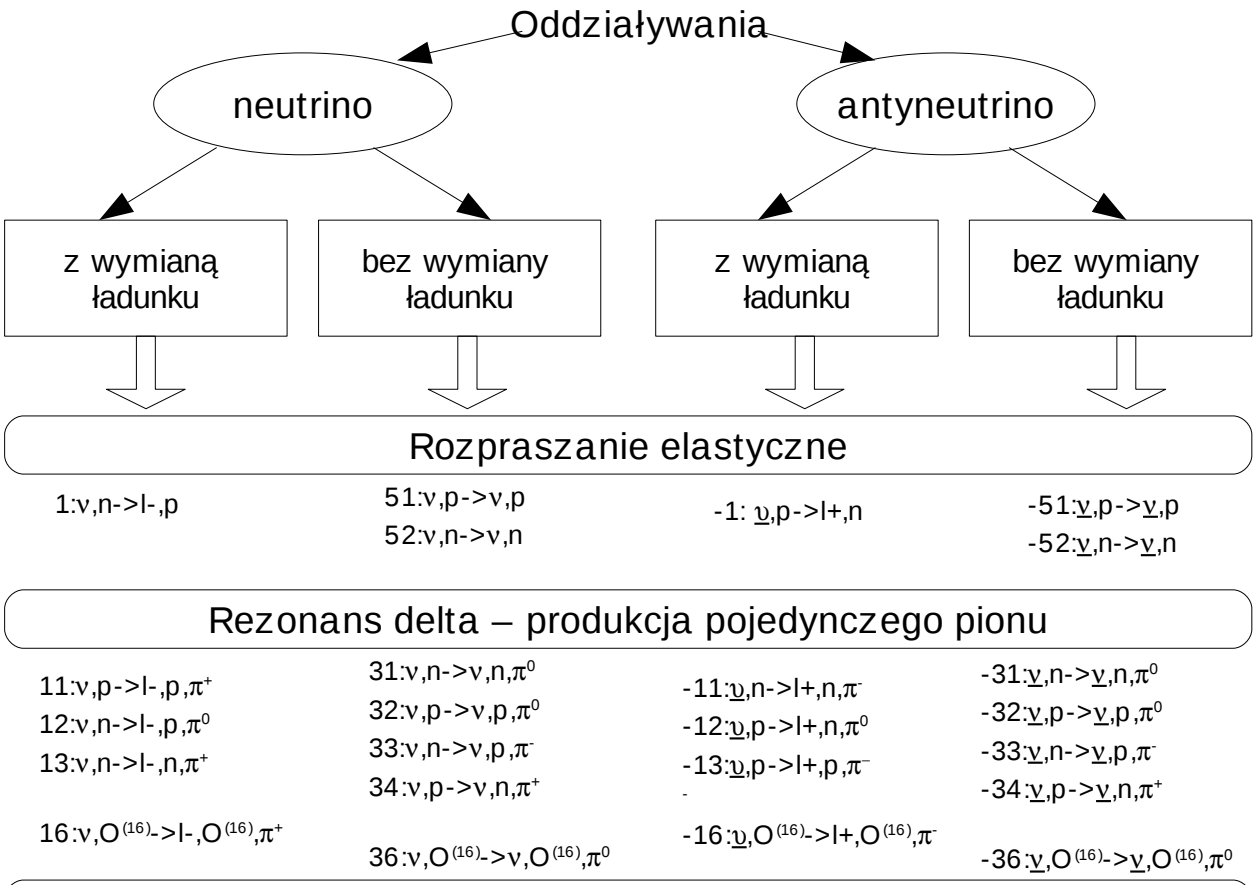
Z wymianą / bez wymiany ładunku produkcja pojedynczej η $(\nu N \to l N' \eta)$

Z wymianą / bez wymiany ładunku głęboko nieelastyczne rozpraszanie
$(\nu N \to l N' hadrony)$

Z wymianą / bez wymiany ładunku koherentna produkcja pionów $(\nu {}^{16}O \to l \pi X)$,

gdzie: N i N' są nukleonami (proton lub neutron), l jest leptonem, a X jest pozostałym nukleonem.

W generatorze NEUT sposób oddziaływania określony jest za pomocą wartości zmiennej *Mode.* Jeśli wartość jest liczbą ujemną, wtedy w oddziaływaniu uczestniczy antyneutrino, w przeciwnym razie neutrino. Oddziaływania z wymianą ładunku oznaczone są liczbami całkowitymi zawierającymi się w przedziale od 1 do 26 dla neutrin oraz od -26 do -1 dla antyneutrin. Dla liczb z zakresu od 31 do 52 przyporządkowano oddziaływania bez wymiany ładunku dla neutrin oraz analogicznie od -52 do -31 dla antyneutrin. Rysunek 4 przedstawia wszystkie oddziaływania, które są przewidziane przez NEUT.



rysunek 4: *Klasyfikacja oddziaływań w NEUT*

2.4 Sposób zamiany oddziaływań

W celu interpretacji wyników uzyskanych przez generator NuWro w detektorze ND280, należy zanalizować i przekształcić oznaczenia w taki sposób, aby były zgodne z oznaczeniami stosowanymi w detektorze.

Jeśli zmienna *anty* przyjmuje wartość jeden (*anty* = 1), wówczas analizujemy oddziaływanie z antyneutrinem, przypisując zmiennej *Mode* minus przed wartością. W przeciwnym wypadku jest to oddziaływanie z neutrinem, a *Mode* przyjmuje wartość dodatnią.

1. Rozpraszanie kwazielastyczne (*qel* = 1)

- oddziaływanie antyneutrina z nukleonem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    Włączone flagi: *qel*, *cc*, *anty* oznaczają zdarzenie $\bar{\nu}\, p \rightarrow l^{+} n$, któremu odpowiada wartości -1 przypisanej do zmiennej *Mode.*
  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    W celu jednoznacznego określenia zdarzenia, należy dodatkowo sprawdzić jaka cząstka powstała oprócz antyneutrina. W przypadku gdy jest to proton, to miało miejsce zdarzenie: $\bar{\nu}\, p \rightarrow \bar{\nu}\, p$, które odpowiada wartości -51 (*Mode* = -51). Natomiast jeśli jest to neutron to *Mode* = -52, które odpowiada zdarzeniu: $\bar{\nu} n \rightarrow \bar{\nu} n$
- oddziaływanie neutrina z nukleonem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    Mode przypisana jest wartość 1, która oznacza zdarzenie: $\nu n \rightarrow l^{-} p$
  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    Aby przypisać wartość dla *Mode* zostaje sprawdzony wyjściowy nukleon. Jeśli sprawdzanym nukleonem jest proton, to jest to zdarzenie: $\nu\, p \rightarrow \nu\, p$ (*Mode* = 51), w przeciwnym razie *Mode* = 52, które odpowiada zdarzeniu: $\nu n \rightarrow \nu n$

2. Rozpraszanie rezonansowe (*res* = 1).

W NuWro w wyniku tego oddziaływania, powstają przede wszystkim pojedyncze piony. Inne możliwe rezultaty oddziaływania rezonansowego, to powstanie kilku pionów lub też ich brak z jednoczesnym pojawieniem się fotonów.

Analizując uzyskane wyniki z generatora NEUT można zauważyć, że oprócz pojedynczych pionów w rozpraszaniu rezonansowym, można uzyskać pojedyncze kaony wraz z cząstkami lambda, a także pojedyncze ety. Przewidziano, również powstanie wielu pionów. W każdych z tych oddziaływań mogą pojawić się fotony.

W celu przyporządkowania odpowiedniej wartości, odpowiadającej za oddziaływanie należy zliczyć piony. Jeśli w NuWro, w zbiorze cząstek wychodzących z jądra jest kilka pionów albo nie ma ich wcale, to odpowiada to produkcji rezonansowej wielu pionów.

2.1. Powstanie wielu pionów w rozpraszaniu rezonansowym

- oddziaływanie antyneutrina z nukleonem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)
    $Mode = -21 \ : \ \bar{\nu}(n|p) \rightarrow l^{+}(n|p) \sum \pi$
  - oddziaływanie bez wymiany ładunku (*nc* = 1)
    $Mode = -41 \ : \ \bar{\nu}(n|p) \rightarrow \bar{\nu}(n|p) \sum \pi$
- oddziaływanie neutrina z nukleonem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)
    $Mode = 21 \ : \ \nu(n|p) \rightarrow l^{-}(n|p) \sum \pi$
  - oddziaływanie bez wymiany ładunku (*nc* = 1)
    $Mode = 41 \ : \ \nu(n|p) \rightarrow \nu(n|p) \sum \pi$

Jeśli tylko jeden pion znajduje się w z zbiorze wyjściowych cząstek, to odpowiada to rezonansowej produkcji pojedynczego pionu. Dla jednoznacznego przyporządkowania wartości zmiennej *Mode* należy sprawdzić rodzaj uzyskanego pionu, a także rodzaj nukleonu.

2.2. Powstanie pojedynczego pionu w rozpraszaniu rezonansowym

- oddziaływanie antyneutrina z nukleonem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

$Mode = -11 \ : \ \bar{\nu} n \to l^+ n \pi^-$
$Mode = -12 \ : \ \bar{\nu} p \to l^+ n \pi^0$
$Mode = -13 \ : \ \bar{\nu} p \to l^+ p \pi^-$

- oddziaływanie bez wymiany ładunku (*nc* = 1)

$Mode = -31 \ : \ \bar{\nu} n \to \bar{\nu} n \pi^0$
$Mode = -32 \ : \ \bar{\nu} p \to \bar{\nu} p \pi^0$
$Mode = -33 \ : \ \bar{\nu} n \to \bar{\nu} p \pi^-$
$Mode = -34 \ : \ \bar{\nu} p \to \bar{\nu} n \pi^+$

- oddziaływanie neutrina z nukleonem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

$Mode = 11 \ : \ \nu p \to l^- p \pi^+$
$Mode = 12 \ : \ \nu n \to l^- p \pi^0$
$Mode = 13 \ : \ \nu n \to l^- n \pi^+$

  - oddziaływanie bez wymiany ładunku (*nc* = 1)

$Mode = 31 \ : \ \nu n \to \nu n \pi^0$
$Mode = 32 \ : \ \nu p \to \nu p \pi^0$
$Mode = 33 \ : \ \nu n \to \nu p \pi^-$
$Mode = 34 \ : \ \nu p \to \nu n \pi^+$

Obecnie w oddziaływaniu: rozpraszanie rezonansowe zastosowanym w NuWro nie występują w zbiorach wynikowych cząstki ety i kanony. Jeśli pojawiłby się, zostałby sklasyfikowane, według następującego podziału.

2.3. Powstanie pojedynczej ety.

Dla oddziaływania bez wymiany ładunku należy sprawdzić rodzaj nukleonu.

- oddziaływanie antyneutrina z nukleonem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

$Mode = -22 \ : \ \bar{\nu} p \to l^+ n \eta^0$

  - oddziaływanie bez wymiany ładunku (*nc* = 1)

$Mode = -42 \ : \ \bar{\nu} n \to \bar{\nu} n \eta^0$
$Mode = -43 \ : \ \bar{\nu} p \to \bar{\nu} p \eta^0$

- oddziaływanie neutrina z nukleonem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = 22 \ : \ \nu n \to l^- p \eta^0$
  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    $Mode = 42 \ : \ \nu n \to \nu n \eta^0$
    $Mode = 43 \ : \ \nu p \to \nu p \eta^0$

Powstanie pojedynczego kaonu oraz cząstki lambdy. Podobnie jak w produkcji pojedynczej ety, należy sprawdzić rodzaj nukleonu w oddziaływaniu bez wymiany ładunku.

2.4. Powstanie pojedynczego kaonu

- oddziaływanie antyneutrina z nukleonem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = -23 \ : \ \bar{\nu} p \to l^+ \Lambda K^0$
  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    $Mode = -44 \ : \ \bar{\nu} n \to \bar{\nu} \Lambda K^0$
    $Mode = -45 \ : \ \bar{\nu} p \to \bar{\nu} \Lambda K^+$
- oddziaływanie neutrina z nukleonem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = 23 \ : \ \nu n \to l^- \Lambda K^+$
  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    $Mode = 44 \ : \ \nu n \to \nu \Lambda K^0$
    $Mode = 45 \ : \ \nu p \to \nu \Lambda K^+$

3. Rozpraszanie koherentne (*coh* = 1). Są to oddziaływania na ^{16}O , w wyniku których powstają piony.

- oddziaływanie antyneutrina z całym jądrem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = -16 \ : \ \bar{\nu}\, ^{16}O \to l^+ \, ^{16}O \pi^-$

- oddziaływanie bez wymiany ładunku (*nc* = 1)

  $Mode = -36 \ : \ \bar{\nu}\,^{16}O \rightarrow \bar{\nu}\,^{16}O\,\pi^{0}$

- oddziaływanie neutrina z całym jądrem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = 16 \ : \ \nu\,^{16}O \rightarrow l^{-}\,^{16}O\,\pi^{+}$

  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    $Mode = 36 \ : \ \nu\,^{16}O \rightarrow \nu\,^{16}O\,\pi^{0}$

4. Rozpraszanie głęboko nieelastyczne (*dis* = 1).

- oddziaływanie antyneutrina z nukleonem (*anty* = 1)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = -26 \ : \ \bar{\nu}(n|p) \rightarrow l^{+}(n|p)\,mezony$

  - oddziaływanie bez wymiany ładunku (nc = 1)

    $Mode = -46 \ : \ \bar{\nu}(n|p) \rightarrow \bar{\nu}(n|p)\,mezony$

- oddziaływanie neutrina z nukleonem (*anty* = 0)
  - oddziaływanie z wymianą ładunku (*cc* = 1)

    $Mode = 26 \ : \ \nu(n|p) \rightarrow l^{-}(n|p)\,mezony$

  - oddziaływanie bez wymiany ładunku (*nc* = 1)

    $Mode = 46 \ : \ \nu(n|p) \rightarrow \nu(n|p)\,mezony$

2.5 Analiza przypadków

W dostarczanych przez NEUT danych zawarte są informacje o cząstce pierwotnej, z której powstało neutrino (zwanej dalej: „rodzicem neutrina"). Analizując plik z ND280MC: `ND280NeutKinematicsGenerator.cc` można zauważyć jakie cząstki mogłyby być rodzicami neutrin. Cząstki te są przedstawione za pomocą kodów Geant.

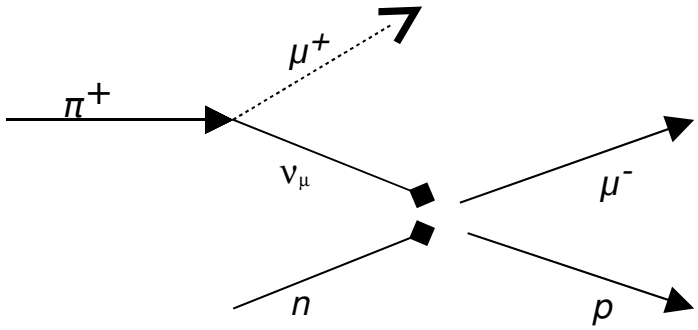
Kod Geant	Kod PDG	Cząstka
5	-13	μ^+
6	13	μ^-
8	211	π^+
9	-211	π^-
11	321	K^+
12	-321	K^-

Z analizowanych danych, które NEUT dostarcza do generatora ND280MC wynika, że ok. 87% rodziców neutrina to p^+, a pozostałych 13% to K+. Produkowane są więc wiązki neutrin mionowych (ν_μ) przez rozpady:

$$\pi^+ \rightarrow \mu^+ + \nu_\mu$$

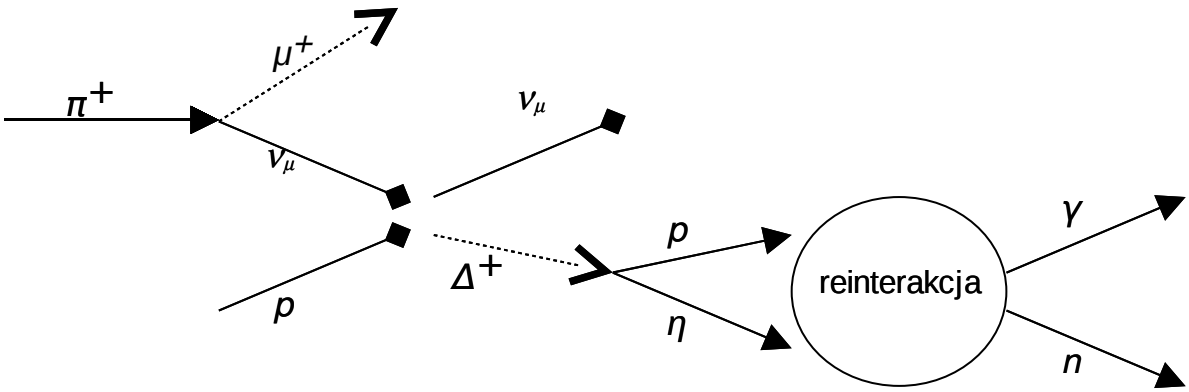
$$K^+ \rightarrow \mu^+ + \nu_\mu$$

Poniżej przedstawione są rysunki, które powstały na podstawie danych z NEUT.



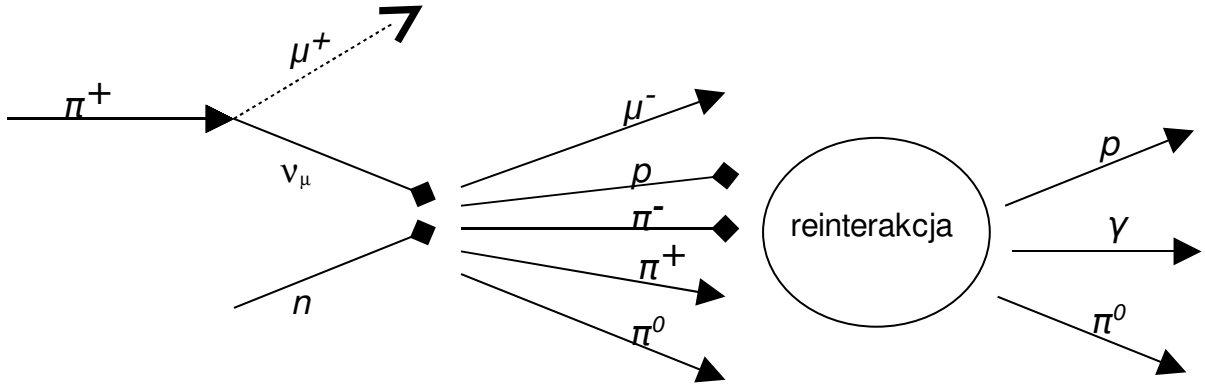
rysunek 5

Na rysunku 5 neutrino mionowe (ν_μ) powstało z rozpadu pionu plus. Antymion nie jest rejestrowany w NEUT, ale został uwzględniony na rysunku po to, żeby zobrazować pełny rozpad. Neutrino mionowe rozprasza się na neutronie. Z tego oddziaływania powstają dwie cząstki: proton i mion. Jest to rozpraszaniem kwazielastycznym z wymianą ładunku, w którym bierze udział neutrino (*Mode* = 1)



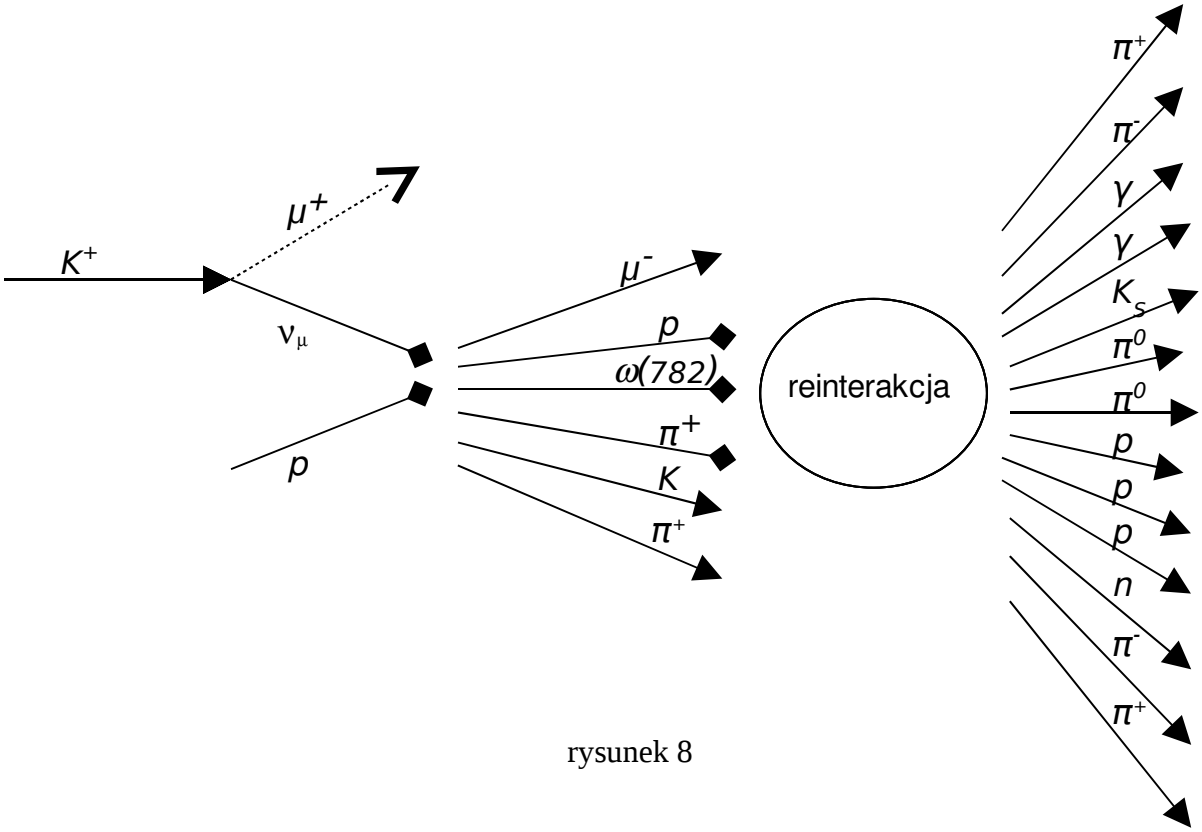
rysunek 6

Strzałkami przerywanymi zaznaczone są cząstki, które nie są rejestrowane w NEUT. Jeśli grot strzałki jest kwadratem, to oznacza to, że dana cząstka nie jest dalej śledzona. Rodzaj oddziaływania zależy od pierwszego wierzchołka, czyli co powstanie po zderzeniu neutrina z nukleonem. W tym przypadku cząstkami, które powstały są: neutrino mionowe, proton oraz eta. Z omówionych wcześniej oddziaływań z neutrinem wynika, że jest to powstanie pojedynczej ety z rozpraszania rezonansowego (Delta plus) bez wymiany ładunku – *Mode* = 43. Powstałe cząstki mogą oddziaływać z innymi nukleonami w jądrze. W wyniku reinterakcji powstają kolejne cząstki. Cząstki, które są rejestrowane po wyjściu z jądra to: proton, eta, foton, neutron.



Rysunek 7

Na rysunku 7 zobrazowane jest oddziaływanie,któremu zmiennej *Mode* przypisana jest wartość 21, czyli produkcja wielu pionów. W tym przypadku z pierwszego wierzchołka oddziaływań powstają trzy piony.



rysunek 8

Na rysunku 8 przedstawione jest głęboko nieelastyczne oddziaływanie neutrina z protonem z wymianą ładunku (Mode =26).

Rozdział 3 Zastosowanie biblioteki ROOT

3.1 ROOT – informacje ogólne

ROOT jest to obiektowy szkielet aplikacji (ang. framework) wspomagający pisanie programów do analizy danych. Powstał w 1994 roku w laboratorium CERN na potrzeby analizy danych fizyki wysokich energii i jest od tego czasu stale rozwijany.

ROOT zawiera:

- interpreter C++ (CINT)
- kompilator C++ (ACLiC)
- mechanizmy dynamicznego ładowania i kompilowania kodu oraz łączenia kodu interpretowanego i skompilowanego
- graficzny interfejs użytkownika i własną bibliotekę graficznych widżetów
- mechanizm refleksji
- mechanizm serializacji obiektów
- wbudowany system generowania dokumentacji z kodu źródłowego
- bibliotekę matematyczną
- bibliotekę statystyczną (funkcje statystyczne, dopasowywanie krzywych do danych, minimalizacja)
- bibliotekę kontenerów
- bibliotekę do tworzenia histogramów
- wejście/wyjście zoptymalizowane na potrzeby fizyki cząstek elementarnych
- bibliotekę wizualizacji trójwymiarowej, opartą na OpenGL
- klasy obudowujące funkcje systemu operacyjnego (aplikacje napisane z wykorzystaniem ROOTa mogą być całkowicie niezależne od systemu operacyjnego i dzięki temu przenośne)
- funkcje do zapisywania plików graficznych

- narzędzia do równoległej analizy danych.

Architektura ROOT umożliwia prowadzenie analizy trzema drogami:

1. Graphical User Interface (GUI) – metoda polegająca na klikaniu i otwieraniu kolejnych okien, w których wpisuje się kryteria selekcji. Okna graficzne umożliwiają: przeglądanie plików, dołączanie drzew, rysowanie rozkładów, dopasowywanie do nich funkcji (wbudowanych i własnych), zmiany kolorów, dodawanie tekstu i grafiki, zmiany skali. Nie jest konieczna znajomość C++, ale metoda raczej nużąca przy dłuższej analizie. Po uruchomieniu ROOT, komenda: *TBrowser t;* otwiera okno, w którym odszukać można plik do analizy
2. Command Line C++ Interpreter (CINT) – wykonanie analizy z linii komend. Poznać należy podstawowe struktury ROOTa, a także nie obejdzie się bez znajomości programowania obiektowego. Typy podstawowe w ROOT, to m.in. Int_t, Long_t, Float_t, Double_t, a klasy mają przedrostek T- np TFile, TH1D, itp. Praca interaktywna z linii komend jest efektywna przy krótkiej selekcji z małą liczbą zmiennych
3. Script Processor (C++) - możliwe jest wykonanie zwykłych makr, skryptów, a także kompilowanie kodu C++. Selekcja przy użyciu skryptów:

- makro - „unnamed script” - zestaw komend ujętych w nawiasy {..},
  - wszystkie zmienne są globalne,
  - nie ma deklaracji funkcji, klas i parametrów
- skrypty - „named script”
  - kompilują funkcje C++,
  - można definiować własne klasy,
  - skrypty ładujemy: .L plik.C i wykonujemy mojafunkcja(), gdy funkcja i nazwa pliku jest taka sama wystarczy .x plik.C

Podstawowe obiekty są to pliki, drzewa, funkcje, histogramy. *TObject* jest podstawowym typem ROOTa, a pozostałe od niego dziedziczą. Ma zdefiniowane protokoły IO *Write()*, obsługi błędów, sortowania, drukowania *Print()*, rysowania *Draw()*. Oprócz wbudowanych, można tworzyć własne klasy.

- TFile - plik zawierający wszystkie elementy – analizowane dane, ale również bieżące wyniki – rysunki, histogramy. Otwarty plik staje się aktualnym katalogiem gDirectory, po skończonej pracy należy zapisać wyniki, bo zostaną utracone.
- TTtree – podstawowa struktura, tupel (krotka), podobna do relacyjnej bazy danych. Przechowuje duże ilości danych bardzo skompresowane. Szybki dostęp. Można czytać tylko niektóre, zadane kryteriami lokacje (zapytania). Drzewa mają strukturę złożoną z gałęzi (branch, np event, detector) i liści (leaves, konkretne lokacje: px, n_hit, bit, itp.)
- funkcje
- histogramy

Dzięki dostarczanym narzędziom do analizy danych, ROOT stał się standardem w projektach fizycznych. Oprogramowanie wykorzystywane w ND280 również napisane jest z wykorzystaniem ROOTa, poza NEUTem. Dlatego ND280 musi konwertować uzyskane wyniki z generatora NEUT. Pliki wynikowe generatora NuWro zapisywane są w formacie ROOT.

Zarówno w ND280 jak i w NuWro szczególne miejsce zajmuje klasa TTree.

Więcej informacji o drzewach – TTree znajduje się w dodatku A.

3.2 Drzewa w NuWro i ND280MC

Drzewo w NuWro składa się z jednej gałęzi do której jest przypisany obiekt event. Sposób tworzenia drzewa jest przedstawiony za pomocą poniższego kodu:

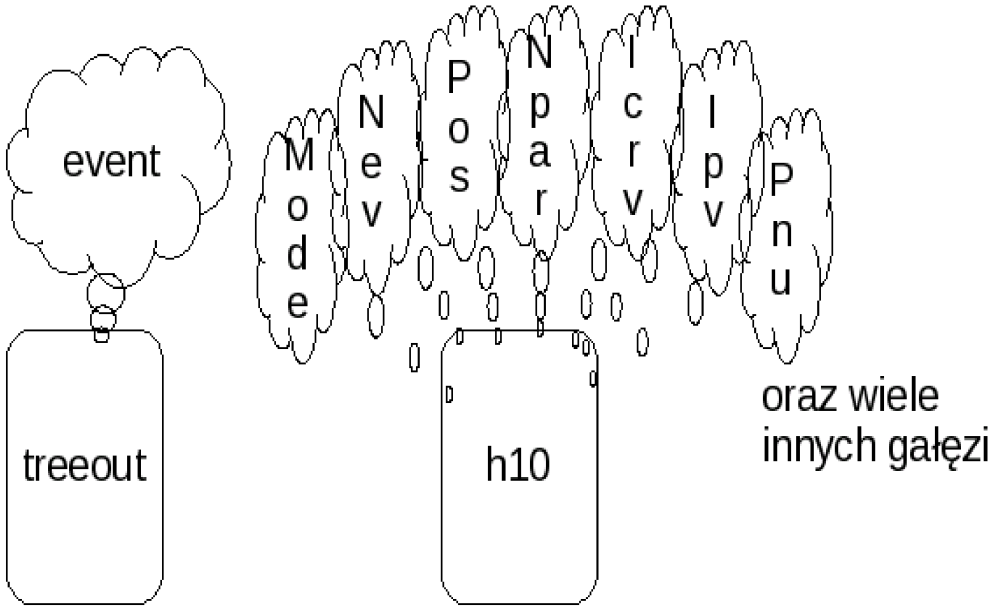
```
TTree *t = new TTree(„tree”, „Drzewo zdarzeń”);
event *e = new event;
t->Branch(„e”, „event”, &e);
```

W obiekcie event korzysta się z wektorów STL (biblioteka C++) a także z własnych zdefiniowanych klas (flags, params).

```
class event: public TObject
{
  public:
    flags flag;
    params par;
    vector < particle > in;
    vector < particle > temp;
    vector < particle > out;
    double weight;
    int Channel;
    particle nu(){return in[0];}
    particle N0(){return in[1];}
    double q2(){vect q=in[0]-out[0];return q*q;}
    double s(){vect q=in[0]+in[1];return q*q;}
    double E(){ return in[0].t;}
    int n() { return out.size();} //liczba cząstek wyjściowych
    ...
};
```

Wczytywane pliki na wejściu ND280 są w formacie ROOT, a informacje pobierane są z danych uporządkowanych za pomocą TTree. Każda zmienna stanowi osobną gałąź drzewa. Zmienne, które są wczytywane to: tablice o zmiennej długości oraz zmienne proste typu całkowitego lub zmiennoprzecinkowego.

Na rysunku 8 zostały przedstawione drzewa wraz z gałęziami wykorzystywane w NuWro (po lewej stronie) oraz ND280 (po prawej stronie).



Rysunek 8: *Drzewo w NuWro (po lewej), drzewo w Nd280 (po prawej)*

Rozdział 4 Specyfikacja danych

W wyniku działania NuWro otrzymuje się obiekt event, który składa się z następujących danych:

```
flags flag;
params par;
vector < particle > in;
vector < particle > temp;
vector < particle > out;
double weight;
int Channel;
```

Są to informacje o pojedynczym zdarzeniu. Dotyczą cząstek biorących udział w zderzeniu, sposobie ich oddziaływania. Poniżej wypisane są wszystkie zmienne wyjściowe wraz z typem i opisem znaczenia.

4.1 Struktura pliku wynikowego generatora NuWro

Flags flag; - Flagi – wartości 1- tak; 0- nie

bool coh	Czy jest to rozpraszanie koherentne (na całym jądrze)?
bool qel	Czy jest to rozpraszanie kwazielastyczne?
bool dis	Czy jest to rozpraszanie głęboko nieelastyczne?
bool res	Czy jest to produkcja rezonansowa pionów?
bool nc	Czy jest to oddziaływanie bez wymiany ładunku?
bool cc	Czy jest to oddziaływanie z wymianą ładunku?
bool anty	Czy jest to oddziaływanie z antyneutrinem ?
int target	Informacja o tarczy: 0 – Pojedynczy nukleon, 1 – Argon, 2 – Tlen;

params par;

int number_of_test_events	Liczba zdarzeń jaka będzie użyta do wyznaczenia stosunku przekrojów czynnych (zdarzenia nie są zapisywane do pliku)
int number_of_events	Liczba zdarzeń jaka po wygenerowaniu zostanie zapisana do pliku
string beam_energy	Energia neutrin. Energia może być ściśle określona lub może być wiązką energetyczną [MeV] - beam_energy = Ev – wiązka monoenergetyczna - beam_energy = Emin Emax val1 val2 ... valn Emin i Emax są wartościami minimalną i maksymakną wiązki energetycznej, a vali, i=1,2,...,n są wartościami wiązki w równych przedziałach od energii minimalnej do maksymalnej. Stosunek kolejnych energii jest obliczany automatycznie.
int beam_particle	Rodzaj oddziałującego neutrina według kodu PDG : 12 – neutrino elektronowe 14 – neutrino mionowe 16 – neutrino taonowe -12 – antyneutrino mionowe -14 - antyneutrino mionowe -16 – antyneutrino taonowe
vec par beam_direction;	Kierunek wiązki dany trzema współrzędnymi wektora równoległego do wiązki. Domyślnie beam_direction = 0 0 1
Parametry określające tarczę	
int target_p	Liczba protonów w jądrze
int target_n	Liczba neutronów w jądrze
Gęstość profilu tarczy	
double target_E_b	Energia wiązania nukleonów w jądrze wyrażona w MeV

double target_kf	Maksymalny pęd Fermiego wyrażony w MeV
bool target_local_kf	Czy używać lokalnego pędu Fermiego zależnego od lokalnej gęstości jądra?
bool target_FG	Czy używać gazu Fermiego? Jeśli fałsz to znaczy, że nukleony tarczy traktujemy jak swobodne (pęd początkowy = 0)
bool target_bodek	Rozkład pędów początkowych nukleonów w jądrze nie kończy się na pędzie Fermiego, lecz zawiera także tzw. wysokoenergetyczny „ogon”
bool target_SF	Rozkład energii i pędów w jądrze otrzymywany za pomocą funkcji spektralnej.
bool target_free	Jądrem jest swobodny nukleon
Dynamika - flagi	
bool dyn_qel_cc	Włączone rozpraszanie kwazielastyczne z wymianą ładunku
bool dyn_qel_nc	Włączone rozpraszanie kwazielastyczne bez wymiany ładunku
bool dyn_res_cc	Włączona jest produkcja rezonansowa z wymianą ładunku
bool dyn_res_nc	Włączona jest produkcja rezonansowa bez wymiany ładunku
bool dyn_dis_cc	Włączone rozpraszanie głęboko nieelastyczne z wymianą ładunku
bool dyn_dis_nc	Włączone rozpraszanie głęboko nieelastyczne bez wymiany ładunku
bool dyn_coh_cc	Włączone rozpraszanie koherentne z wymianą ładunku
bool dyn_coh_nc	Włączone rozpraszanie koherentne bez wymiany ładunku
double qel_cc_axial_mass	Wartość masy aksjalnej – parametru występującego w czynniku postaci dla oddziaływań qel cc [MeV]
double qel_nc_axial_mass	Wartość masy aksjalnej – parametru występującego w czynniku postaci dla oddziaływań qel [MeV]

double res_dis_cut	Graniczna masa niezmiennicza między rezonansową produkcją pionów a procesem głęboko nieelastycznym [MeV]
int spp_precision	
bool coh_mass_correction	Czy stosować korektę masy w modelu Rein & Sehgal dla oddziaływania: produkcja pojedynczego pionu w koherentnym rozpraszaniu z CC
bool kaskada_on	Włączony model kaskady wewnątrzjądrowej do śledzenia cząstek wychodzących z wierzchołka pierwotnego.
bool kaskada_debug	Wypisywanie dodatkowych informacji o procesach zachodzących w trakcie kaskady wewnątrzjądrowej
bool pauli_blocking	Czy włączyć zakaz Pauliego, wykluczający procesy, dla których końcowy nukleon ma pęd mniejszy od pędu Fermiego.

vector <particle> in; - wektor cząstek wejściowych

vector <particle> temp; - wektor cząstek tymczasowych

vector <particle> out; - wektor cząstek wyjściowych (opuszczających jądro)

gdzie klasa particle ma nastepujace skladowe:

<table>
<tr><td>double t;</td><td rowspan="4">Vect {double t, x, y, z;} - czterowektor</td><td>Energia cząstki</td></tr>
<tr><td>double x;</td><td rowspan="3">Składowe przestrzenne pędu</td></tr>
<tr><td>double y;</td></tr>
<tr><td>double z;</td></tr>
<tr><td colspan="2">double _mass</td><td>Masa cząstki</td></tr>
<tr><td colspan="2">vec r; {double x, y, z;}</td><td>Składowe przestrzenne położenia</td></tr>
<tr><td colspan="2">int pdg</td><td>Kod PDG cząstki</td></tr>
<tr><td colspan="2">char ks</td><td rowspan="2">Zmienne używane w procedurze fragmentacji PYTHIA6</td></tr>
<tr><td colspan="2">char orgin</td></tr>
</table>

double weight; - wielkość proporcjonalna do przekroju czynnego

int Channel; - numer użytej dynamiki w danym zderzeniu

4.2 Struktura pliku wynikowego NEUT

Informacje o cząstce – rodzicu neutrina w punkcie jej produkcji

Float_t xpi0[3];	*Pozycja produkcyjna dla cząstki - rodzica neutrina na podstawie symulacji wiązki liniowej.*	*Nie dotyczy NuWro*	*Brak zastosowania w ND280*
Float_t npi0[3];	*Kierunek produkcyjny dla cząstki - rodzica neutrina na podstawie symulacji wiązki liniowej.*	*Nie dotyczy NuWro*	*Brak zastosowania w ND280*
Float_t cospi0bm;	*Cosinus kąta między kierunkiem cząstki – rodzica neutrina a kierunkiem wiązki. Jest to kierunek cząsteczki lecącej do tarczy*	*Nie dotyczy NuWro*	*Brak zastosowania w ND280*
Float_t ppi0;	*Pęd cząstki - rodzica neutrina w produkcyjnym punkcie.*	*Nie dotyczy NuWro*	*Brak zastosowania w ND280*

Informacje o cząstce – rodzicu neutrina w punkcie jej rozpadu

Float_t xpi[3];	*Pozycja rozkładu cząstki – rodzica neutrina na podstawie symulacji wiązki liniowej. Jest to pozycja w odniesieniu do tarczy.*	*Nie dotyczy NuWro*	*Zastosowanie w ND280*
Float_t npi[3];	*Kierunek cząstki – rodzica neutrina na podstawie symulacji wiązki liniowej.*	*Nie dotyczy NuWro*	*Zastosowanie w ND280*

Float_t cospibm;	Cosinus kąta między kierunkiem cząstki - rodzica neutrina i kierunkiem wiązki	*Nie dotyczy NuWro*	*Brak zastosowania w ND280*
Float_t ppi;	Pęd cząstki – rodzica neutrina wyrażony w GeV / c	*Nie dotyczy NuWro*	*Zastosowanie w ND280*
Int_t ppid;	Kod G3 dla cząstki- rodzica neutrina.	*Nie dotyczy NuWro*	*Zastosowanie w ND280*

Wierzchołek zdarzeń

Float_t Pos[3];	Wierzchołek zdarzeń.	In[1].r	*Zastosowanie w ND280*
Int_t Mode;	Sposób oddziaływania w zdarzeniach.	ND280_Mode()	*Zastosowanie w ND280*

Cząstki uczestniczące w pierwszym wierzchołku zderzenia

Int_t Numnu;	Liczba cząstek biorących udział w pierwszym zderzeniu.	in.size() +out.size()	*Zastosowanie w ND280*
Int_t Ipnu[100]; //Numnu	Kody PDG dla cząstek w kinematycznej liście.	in[i].pdg, out[i].pdg	*Zastosowanie w ND280*
Float_t Abspnu[100]; //Numnu	Całkowita wartość pędu (GeV/c)	*0.001*vec(in[i].x,in[i].y,in[i].z).lengh()*	*Brak zastosowania w ND280*
Float_t Pnu[100][3]; //Numnu	Pęd cząsteczek (GeV/c)	*0.001*in[i].x, 0.001*in[i].y, 0.001*in[i].z*	*Zastosowanie w ND280*

		*0.001*out[i].x,* *0.001*out[i].y,* *0.001*out[i].z*	

Cząstki z uwzględnieniem kaskady

Int_t Npar;	Całkowita liczba cząstek, które powinny zostać wyśledzone przez G4.	in.size() +out.size	*Zastosowanie w ND280*
Int_t Ipv[100]; //Npar	PDG cząsteczek śledzonych przez G4.	in[i].pdg, out[i].pdg,	*Zastosowanie w ND280*
Int_t Icrnv[100]; // Npar	Flaga informująca o tym, czy dana cząstka powinna zostać wyśledzona w detektorze - śledzi, jeśli wartość jest niezerowa.	Dla in Icrnv=0 Dla in Icrnv=1 – oprócz neutrin	*Zastosowanie w ND280*
Float_t Pmomv[100][3]; // Npar	Współrzędne pędu (trójpęd) cząstek opuszczających jądro	*in[i].x, in[i].y, in[i].z* *out[i].x,* *out[i].y,* *out[i].z*	*Zastosowanie w ND280*

5 Program konwertujący

5.1 Opis programu

Opisane w poprzednich rozdziałach struktury zdarzeń generatorów NuWro oraz NEUT, oraz odpowiedniość między nimi przedstawiona w podrozdziale 4.2 znalazły zastosowanie w programie `translate.cc`, którego kod zamieszczony jest w dodatku B.

Do opisu zdarzenia generatora NuWro program korzysta z pliku nagłówkowego event1.h, który stanowi jeden z plików źródłowych generatora NuWro i zawiera definicję klasy event. Struktura zdarzenia generatora NEUT jest opisana za pomocą klasy `ND280_`event, stworzonej przeze mnie na podstawie dokumentacji generatora ND280, a konkretnie przy wykorzystaniu pliku `ND280NEUTKINEMATICSGENERATOR.CC`.

Klasa `ND280_`event oprócz zmiennych przechowujących informacje o zdarzeniach, zawiera trzy metody: `Create_Branches()`, `Translate_From_NuWro()`, `ND280_Mode()`.

Metoda `Create_Branches()` inicjuje drzewo stanowiące jedyny obiekt znajdujący się w pliku wyjściowym, zawiera ona wielokrotne wywołanie metody `TTree::Branch()`, tworzącej nową gałąź w drzewie. Znaczenie jej parametrów zostało dokładnie omówione w dodatku A.

Sercem programu jest metoda `Translate_From_NuWro()`, która wypełnia pola klasy `ND280_event` danymi obiektu klasy `event`, czyli wykonuje faktyczne odwzorowanie danych zdarzenia NuWro, w dane zdarzenia NEUT.

Metoda `ND280_Mode()` jest wykorzystywana przez metodę `Translate_From_NuWro()` do zamiany opisu dynamiki zdarzenia NuWro na kod reakcji zdarzenia NEUT, zgodnie z opisem w podrozdziale 2.4.

Oprócz definicji klasy ND280_event plik `translate.cc` zawiera również procedurę `NuWro_To_ND280()`, która dokonuje konwersję całego pliku zdarzeń, korzystając z obiektu klasy `ND280_event` i opisanych powyżej metod.

Po uruchomieniu programu translate.cc następuje wywołanie funkcji `NuWro_To_ND280`, z argumentami, którymi są podane przez użytkownika nazwy plików: wejściowego i wyjściowego. Jeśli program został wywołany bez parametrów, to wywołanie funkcji będzie miało następującą postać:

```
NuWro_To_ND280("eventsout.root","ND280.root").
```

W funkcji `NuWro_to_ND280` tworzony jest obiekt klasy `event`, a jego adres zostaje przypisany do gałęzi drzewa znajdującego się w pliku wejściowym. Następnie tworzony jest plik wyjściowy oraz drzewo o nazwie `h10` i obiekt klasy `ND280_event`. Dla obiektu `e2` klasy `ND280_event` wywołana zostaje metoda Create_Branches, która tworzy gałęzie na kształt danych z NEUTA. Z pliku wejściowego do obiektu e wczytywane są kolejne zdarzenia (metoda `GetEntry`), które funkcja `Translate_From_NuWro` przepisuje w odpowiednie pola obiektu e2 a metoda `Fill()` wypełnia nimi gałęzie drzewa `h10` zawartego w pliku wyjściowym. Na zakończenie działania programu otrzymywany jest plik `ND280.root` (ewentualnie o innej nazwie podanej przez użytkownika), gotowy do wczytania przez generator ND280.

5.2 Sposób użycia programu

Program konwertujący `translate.cc`, należy skompilować poleceniem `make`. Powstaje wtedy jego wersja wykonywalna o nazwie `translate`. Z programu korzystamy w następujący sposób:

1) wywołanie z dwoma parametrami
   polecenie: `translate nazwa_pliku_NuWro nazwa_pliku_wynikowego`
   przykład: `translate eventsout.root ND280.root`
2) wywołanie z jednym parametrem
   polecenie: `translate nazwa_pliku_NuWro`
   przykład: `translate eventsout.root`
   (plik wyjściowy będzie miał nazwę ND280.root)
3) wywołanie bez parametrów
   polecenie: `translate`
   (Dane będą czytane z pliku eventsout.root, a nazwą pliku wyjściowego będzie ND280.root.)

Zamiast programu translate.cc, można skorzystać z makra `translate.C`. W tym przypadku należy uruchomić program ROOT i wykonać następujące polecenia:

```
gSystem->Load("event1.so")
.x translate.C
```

Dane będą czytane z pliku `eventsout.root`, a nazwą pliku wyjściowego będzie `ND280.root`. Do poprawnego działania makra wymagana jest obecność w katalogu roboczym aktualnej biblioteki *event1.so*, która powstaje zawsze w trakcie kompilacji generatora NuWro.

Otrzymane pliki można przetwarzać w generatorze ND280 za pomocą skryptów `neut-event.sh` oraz `nd280mc-fg-test-neut.sh` obecnych w katalogu `nd280mc/[version]/inputs` dostarczanych z dystrybucją pakietu nd280. Dokładniejszy opis można znaleźć w rozdziale *Running Detector Simulation* na stronie *nd280.org*.

Podsumowanie

Przedmiotem pracy było napisanie programu, który pozwoli przetworzyć dane wyjściowe generatora zdarzeń NuWro, do postaci danych, które wymagane są na wejściu generatora ND280. W tym celu należało zrozumieć format danych wyjściowych generatorów NEUT oraz NuWro, a także nauczyć się korzystać z biblioteki ROOT, która umożliwia wymianę danych. Do zrozumienia, w jaki sposób odpowiadają sobie poszczególne dane z tych generatorów, konieczne okazało się porównanie i analiza konkretnych plików zdarzeń generowanych przez NEUT oraz NuWro.

Wynikiem pracy magisterskiej jest zbiór procedur, z których najważniejsza jest `NuWro_To_ND280()`, przy użyciu których został napisany program `translate.cc` realizujący konwersję plików wytwarzanych przez generator NuWro na format zgodny z generatorem NEUT, który jest obsługiwany przez generator ND280. Te same procedury znalazły zastosowanie w makrze `translate.C`, które realizuje to samo zadanie.

Program został przetestowany, tzn. zostało sprawdzone, że tworzone przez niego pliki są poprawnie wczytywane przez generator ND280.

Ważnym rezultatem niniejszej pracy jest również szczegółowy opis formatu danych generatorów NEUT oraz NuWro (rozdział 4.1 i 4.2), opis sposobów kodowania informacji o procesach fizycznych stosowanych w każdym z generatorów, oraz sposób otrzymywania kodu reakcji według NEUTA na podstawie informacji zawartych w zdarzeniu NuWro (rozdział 2).

Zawarte w dodatku A zestawienie najważniejszych informacji o stosowaniu w bibliotece ROOT obiektów typu TTree oraz kod programu stanowiący dodatek będą pomocne dla osób, które wraz z rozwojem generatora NuWro będą aktualizowały również stworzony przeze mnie program.

Alternatywnym sposobem rozwiązania problemów współpracy generatorów NuWro oraz ND280 byłoby wzbogaceniem generatora ND280 o filtr wejściowy, który akceptowałby plik zdarzeń tworzony przez NuWro. Z analizy struktury plików źródłowych generatora ND280 wynika, że konieczne byłoby stworzenie następujących plików:

`ND280NuWroKinematicsGenerator.cc`

`ND280NuWroKinematicsGenerator.hh`

`ND280NuWroKinematicsFactory.cc`

`ND280NuWroKinematicsFactory.hh`

Rozwiązanie takie wymaga nawiązania bliższej współpracy z twórcami ND280 i mogłoby funkcjonować jedynie w przyszłych wersjach tego generatora. Z tego względu, przedstawione przeze mnie rozwiązanie wydaje się najbardziej praktyczne.

Dodatki

Dodatek A: ROOT - TTree

ROOT zaprojektował specjalną klasę TTree do przechowywania dużej ilości obiektów tej samej klasy. Jest ona zoptymalizowana, w celu zmniejszenia rozmiaru przechowywanych danych, a także żeby zwiększyć szybkość dostępu do nich. Zmienne drzewa to „gałęzie” (TBranch), które z kolei zawierają zmienne „liście” (Tleaf). W TBranch mogą być przechowywane obiekty, zmienne proste, tablice obiektów, listy powiązane.

1 Tworzenie drzewa

Poniższy kod tworzy proste drzewo z jedną gałęzią, w której znajduje się zmienna typu całkowitego

```
Int_t var1;
TFile *f = new TFile („plik.root”, „RECREATE”);
TTree* tree = new TTree(„tree”,”a ROOT Tree”);
tree->Branch(„var1”, &var1, „var1/I”);
```

Na początku tworzy się zmienną prostą, tablicę lub klasę, które stanowić będą gałęzie drzewa. W powyższym przykładzie jest to zmienna całkowita:

```
Int_t var1;
```

Drzewo zapisywane jest w pliku o rozszerzeniu root, który trzeba utworzyć. Można to zrobić następująco:

```
TFile *f = new TFile („plik.root”, „RECREATE”);
```

Kolejny etap to tworzenie drzewa TTree:

```
TTree* tree = new TTree(„tree”,”a ROOT Tree”);
```

W celu stworzenia gałęzi, wykorzystuje się metodę TTree: Branch:.

```
tree->Branch(„var1”, &var1, „var1/I”);
```

Pierwszy parametr metody Branch, to nazwa gałęzi. Drugi parametr jest adresem, pod którym znajduje się pierwsza zmienna do odczytu. Trzeci parametr jest łańcuchem znaków opisujących listę liści. Każdy liść ma nazwę i typ, które oddzielone są „/”. Kolejne liście są oddzielone znakiem „:”.

```
<Variable>/<type>:<Variable>/<type>
```

Przykład:

```
tree->Branch(„E_Branch”, &event, „var1/F:var2/I:var3:var4:var5/i”);
```

W powyższym kodzie „event” jest strukturą zawierającą jedną zmienną typu zmiennoprzecinkowego (float), jedną całkowitą (integer) oraz trzy zmienne całkowite bez znaku (unsigned integer). Tę metodę możemy stosować, gdy korzystamy ze stałej długości pamięci. W przeciwnym wypadku, każdy element struktury zapisujemy jako jedną gałąź. W metodzie Branch można zapisać całą tablicę zmiennych. W celu dodania tablicy zmiennych należy w opisie liści (trzeci argument metody Branch) użyć:

```
<array[number]> / <type>.
```

Przykład:

```
Float_t f[10];
tree->Branch(„fBranch”,f,”f[10]/F”);
```

Istnieje możliwość dodawania tablic o zmiennej długości.
Przykład:

```
Int_t number;
Float_t array[100];
TTree* tree = new Ttree(„tree”,”a ROOT Tree”);
tree->Branch(„number”,&number,”number/I”);
```

```
tree->Branch(„array”,array,”array[number]/F”);
```

Poniższy przykład pokazuje w jaki sposób zapisać obiekty.

```
Event *event = new Event()
tree->Branch("EventBranch","Event", &event)
```

Na początku trzeba utworzyć wskaźnik na obiekt, który będzie odniesieniem w metodzie TTree:: Branch. Pierwszym parametrem w metodzie Branch jest nazwa gałęzi. Drugi parametr, to nazwa klasy obiektu, który ma być przechowywany. Trzeci parametr, to adres wskaźnika do obiektu.

2 Czytanie drzew

Poniższy kod odczytuje drzewo wygenerowane przez tree1w i wypełnia dwa histogramy

```
void tree1r() {
      TFile *f = new TFile("tree1.root");
      TTree *t1 = (TTree*)f->Get("t1");
      Float_t px, py, pz;
      Double_t random;
      Int_t ev;
      t1->SetBranchAddress("px",&px);
      t1->SetBranchAddress("py",&py);
      t1->SetBranchAddress("pz",&pz);
      t1->SetBranchAddress("random",&random);
      t1->SetBranchAddress("ev",&ev);
      //tworzenie histogramów
      TH1F *hpx = new TH1F("hpx","px distribution",100,-3,3);
      TH2F *hpxpy = new TH2F("hpxpy","py vs px",30,-3,3,30,-3,3);
      //wczytywanie wszystkich wpisów i wypełnianie histogramów
      Int_t nentries = (Int_t)t1->GetEntries();
      for (Int_t i=0; i<nentries; i++)
```

```
      {
            t1->GetEntry(i);
            hpx->Fill(px);
            hpxpy->Fill(px,py);
      }
}
```

Na początku tworzy się zmienne do przechowywania wartości odczytu.

```
            Float_t px, py, pz;
```

Za pomocą metody TTree:Set BranchAddress wskazuje się drzewu by w trakcie czytania danych zapisywało je do zmiennych.

```
            tree->SetBranchAddress("px",&px)
```

Pierwszym parametrem jest nazwa gałęzi, a drugim jest adres miejsca gdzie mają być umieszczone dane gałęzi. Metoda GetEntries zwraca liczbę wpisów pasujących do wyboru. Funkcja wywołana bez argumentu zwraca liczbę wpisów. Gdy zostały zapisane w pamięci adresy gałęzi, konkretny wpis można odczytać ze zmiennych za pomocą metody GetEntry.

3 Zapis drzewa

Po przypisaniu wartości zmiennym wpisuje się dane do drzewa, za pomocą metody TTree:Fill. Zapis drzewa dokonuje się przy pomocy metody Ttree:Write.

4 Metody TTree, które są przydatne do analizowania danych:

- *TTree:Show* Metoda, która wyświetla pojedynczy wpis do drzewa
  przykład: `Drzewo->Show(10)`

- *Ttree:Print* Jest to przydatne polecenie do zobaczenia struktury drzewa (liczba zdarzeń (wpisów), gałęzi, liści, rozmiar)

  przykład: `Drzewo->Print()`
- *TTree:Scan* Metoda pozwalająca wyświetlać wartości wszystkich liści z listy podanej przez użytkownika

  przykład: `Drzewo->Show(„zmienna1:zmienna2”)`

Dodatek B: Zamiana drzewa uzyskanego w NuWro na strukturę drzewa wczytywanego w ND280

Translate.cc

```
#include <iostream>
#include <stdio.h>
#include <vector>
#include "event1.h"
#include "event1dict.h"
#include "TROOT.h"
#include "TTree.h"
#include "TFile.h"


class ND280_event
{
    public:
      Int_t Nev;
      Float_t Pos[3];
      Int_t Mode;
      Int_t Numnu;
      Int_t Ipnu[100];   //[Numnu
      Float_t  Abspnu[100];   //[Numnu]
      Float_t  Pnu[100][3];   //[Numnu]
      Int_t    Npar;
      Int_t    Ipv[100];   //[Npar]
      Int_t    Icrnv[100];   //[Npar]
      Float_t  Pmomv[100][3];   //[Npar]
      Float_t  xpi[3];
      Float_t  npi[3];
      Float_t  cospibm;
      Float_t  ppi;
      Int_t    ppid;
      Float_t  xpi0[3];
```

```
      Float_t  npi0[3];
      Float_t  cospi0bm;
      Float_t  ppi0;
      Float_t Abspv[100]; //Npar
      Float_t Enu;
      Int_t Iflgv[100]; // Npar
      Int_t Iorgv[100]; // Npar
      Int_t idfd;
      Int_t modef;
      Float_t nnu[3];
      Float_t norm;
      Float_t rnu;
      Float_t xnu;
      Float_t ynu;

    void Create_Branches(TTree*);
    void Translate_From_NuWro(event *e);
    int  ND280_Mode(event *e);
};



void ND280_event::Create_Branches(TTree *tt2)
{
    tt2->Branch("Nev",&Nev,"Nev/I");
    tt2->Branch("Pos",Pos,"Pos[3]/F");
    tt2->Branch("Mode",&Mode,"Mode/I");
    tt2->Branch("Numnu",&Numnu,"Numnu/I");
    tt2->Branch("Ipnu",Ipnu,"Ipnu[Numnu]/I");
    tt2->Branch("Abspnu",Abspnu,"Abspnu[Numnu]/F");
    tt2->Branch("Pnu",Pnu,"Pnu[Numnu][3]/F");
    tt2->Branch("Npar",&Npar,"Npar/I");
    tt2->Branch("Ipv",Ipv,"Ipv[Npar]/I");
    tt2->Branch("Icrnv",Icrnv,"Icrnv[Npar]/I");
    tt2->Branch("Pmomv",Pmomv,"Pmomv[Npar][3]/F");
    tt2->Branch("xpi", xpi,"xpi[3]/F");
    tt2->Branch("npi", npi,"npi[3]/F");
```

```
    tt2->Branch("cospibm",&cospibm,"cospibm/F");
    tt2->Branch("ppi",  &ppi,"ppi/F");
    tt2->Branch("ppid", &ppid,"ppid/I");
    tt2->Branch("xpi0", xpi0,"xpi0[3]/F");
    tt2->Branch("npi0", npi0,"npi0[3]/F");
    tt2->Branch("cospi0bm", &cospi0bm,"cospi0bm/F");
    tt2->Branch("ppi0", &ppi0,"ppi0/F");

    tt2->Branch("Abspv",Abspv,"Abspv[Npar]/F");
    tt2->Branch("Enu",&Enu,"Enu/F");
    tt2->Branch("Iflgv",Iflgv,"Iflgv[Npar]/I");
    tt2->Branch("Iorgv",Iorgv,"Iorgv[Npar]/I");
    tt2->Branch("idfd",&idfd,"idfd/I");
    tt2->Branch("modef",&modef,"modef/I");
    tt2->Branch("nnu",nnu,"nnu[3]/F");
    tt2->Branch("norm",&norm,"norm/F");
    tt2->Branch("rnu",&rnu,"rnu/F");
    tt2->Branch("xnu",&xnu,"xnu/F");
    tt2->Branch("ynu",&ynu,"ynu/F");
}


int ND280_event::ND280_Mode(event *e)
{
    Int_t proton_pdg, neutron_pdg, pion_pdg, pion_plus_pdg,
pion_minus_pdg, lambda_pdg, eta_pdg, kaon_pdg, kaon_plus_pdg;
    proton_pdg = 2212;
    eta_pdg = 221;
    neutron_pdg = 2112;
    pion_pdg = 111;
    pion_plus_pdg = 211;
    pion_minus_pdg = -211;
    //O_16_pdg = 100069;   // oznacznie z Neuta
    lambda_pdg = 3122;
    kaon_pdg = 311;
    kaon_plus_pdg = 321;
```

```
    if(e->flag.qel) //kwiazielastyczne oddziaływanie
    {
      if(e->flag.anty)  //oddziaływanie z antyneutrinem
      {
          if(e->flag.cc) return -1;
          else
          {
            if(e->nof(proton_pdg)) return -51;
            else if(e->nof(neutron_pdg)) return -52;
          }
      }
      else       //oddziaływanie z neutrinem
      {
          if(e->flag.cc) return 1;
          else
          {
            if(e->nof(proton_pdg)) return 51;
            else if(e->nof( neutron_pdg)) return 52;
          }
        }
    }


    if(e->flag.res) //rezonansowa produkcja: pojedynczy pion, pojed.eta,
kaon, multipiony
    {
      Int_t liczba_pionow, liczba_kaonow;

           liczba_pionow = e->nof(pion_pdg) + e->nof(pion_plus_pdg) + e-
>nof(pion_minus_pdg);
        liczba_kaonow = e->nof(kaon_pdg) + e->nof(kaon_pdg);

        if(liczba_pionow > 1 || liczba_pionow == 0) // multipiony
        {
          if(e->flag.anty)
          {
```

```
            if(e->flag.cc) return -21;
            else return -41;
      }
        else
          {
            if(e->flag.cc) return 21;
            else return 41;
          }
        }

        if(liczba_pionow == 1)
        {
          if(e->flag.anty)   //oddziaływanie z antyneutrinem
             {
            if(e->flag.cc)
            {
                 if(e->nof(neutron_pdg) && e->nof(pion_minus_pdg)) return
-11;
                if(e->nof(neutron_pdg) && e->nof(pion_pdg)) return -12;
                 if(e->nof(proton_pdg) && e->nof(pion_minus_pdg)) return
-13;
            }
            else
            {
                if(e->nof(proton_pdg))
                {
                  if(e->nof(pion_minus_pdg)) return -33;
                  else if(e->nof(pion_pdg)) return -32;
                }
                else if(e->nof(neutron_pdg))
                {
                  if(e->nof(pion_plus_pdg)) return -34;
                  else if(e->nof(pion_pdg)) return -31;
                }
            }
          }
          else              //oddziaływanie z neutrinem
```

```
          {
            if(e->flag.cc)
            {
                  if(e->nof(proton_pdg)  &&  e->nof(pion_plus_pdg))  return
11;
                if(e->nof(proton_pdg) && e->nof(pion_pdg)) return 12;
                 if(e->nof(neutron_pdg)  &&  e->nof(pion_plus_pdg))  return
13;
            }
            else
            {
                if(e->nof(proton_pdg))
                {
                  if(e->nof(pion_minus_pdg)) return 33;
                  else if(e->nof(pion_pdg)) return 32;
                }
                else if(e->nof(neutron_pdg))
                {
                  if(e->nof(pion_plus_pdg)) return 34;
                  else if(e->nof(pion_pdg)) return 31;
                }
            }
          }

          if(e->nof(eta_pdg))   // produkcja rezonansowa ety
          {
              if(e->flag.anty)  //oddziaływanie z antyneutrinem
              {
                if(e->flag.cc) return -22;
                else
                {
                    if(e->nof(neutron_pdg)) return -42;
                    else if(e->nof(proton_pdg)) return -43;
                }
              }
              else       //oddziaływanie z neutrinem
```

```
          {
            if(e->flag.cc)  return 22;
            else
            {
                if(e->nof(neutron_pdg)) return 42;
                else if(e->nof(proton_pdg)) return 43;
            }
             }
      }

      if(e->nof(lambda_pdg) == 1 && liczba_kaonow == 1)    //produkcja
rezonansowa kaonu
      {
          if(e->flag.anty)  //oddziaływanie z antyneutrinem
          {
            if(e->flag.cc && e->nof(kaon_pdg)) return -23;
            else
            {
                if(e->nof(kaon_pdg)) return -44;
                else if(e->nof(kaon_plus_pdg)) return -45;
            }
          }
          else       // oddziaływanie z neutrinem
          {
            if(e->flag.cc && e->nof(kaon_plus_pdg))  return 23;
            else
            {
                if(e->nof(kaon_pdg)) return 44;
                else if(e->nof(kaon_plus_pdg)) return 45;
            }
             }
      }
     }

    if (e->flag.coh) //koherentne  oddziaływanie tylko na O(16)
    {
      Int_t _target;
```

```
      _target  =  e->par.target_p  +  e->par.target_n;    //liczba  masowa
O(16)

      if(_target == 16)
      {
          if(e->flag.anty)  //oddziaływanie z antyneutrinem
            {
                if(e->flag.cc && e->nof(pion_minus_pdg)) return -16;
                else if( e->nof(pion_pdg)) return -36;
            }
          else       //oddziaływanie z neutrinem
              {
                    if(e->flag.cc && e->nof(pion_plus_pdg)) return 16;
                else if(e->nof(pion_pdg)) return 36;
            }
      }
    }

    if(e->flag.dis)      // gleboko nieelastyczne rozpraszanie
    {
      if(e->flag.anty)
      {
          if(e->flag.cc) return -26;
          else return -46;
      }
      else
      {
          if(e->flag.cc) return 26;
          else return 46;
      }
    }

   return 9999;
}
```

```
void ND280_event::Translate_From_NuWro(event *e)
{
      Int_t nie_dotyczyI = -3;
      Float_t nie_dotyczyF = -3.3;
      Int_t nie_wiadomoI = -13;
      Float_t nie_wiadomoF = -13.13;

      Pos[0] = e->in[1].r.x;
      Pos[1] = e->in[1].r.y;
      Pos[2] = e->in[1].r.z;    // wartość 0 w wynikach

      Mode = ND280_Mode(e);
      Int_t Numnu_in = e->in.size();
      Int_t Numnu_out = e->out.size();
      Numnu = Numnu_in + Numnu_out;

      for(Int_t j=0; j < Numnu_in; j++)
      {
          Ipnu[j] = e->in[j].pdg;
          Abspnu[j] = 0.001*vec(e->in[j].x,e->in[j].y,e-
>in[j].z).length();
           Pnu[j][0] = 0.001*e->in[j].x;
          Pnu[j][1] = 0.001*e->in[j].y;
          Pnu[j][2] = 0.001*e->in[j].z;

        }

        for(Int_t j=0; j < Numnu_out; j++)
      {
          Ipnu[j+Numnu_in] = e->out[j].pdg;
          Abspnu[j+Numnu_in] = 0.001*vec(e->out[j].x,e->out[j].y,e-
>out[j].z).length();
           Pnu[j+Numnu_in][0] = 0.001*e->out[j].x;
          Pnu[j+Numnu_in][1] = 0.001*e->out[j].y;
          Pnu[j+Numnu_in][2] = 0.001*e->out[j].z;
```

```
        }

      Int_t Npar_in = e->in.size();
      Int_t Npar_out = e->out.size();
      Npar = Npar_in + Npar_out;

      for(Int_t k = 0; k < Npar_in; k++)
      {
           Ipv[k] = e->out[k].pdg;
           Icrnv[k] = 0;
           Pmomv[k][0] = e->out[k].x;
           Pmomv[k][1] = e->out[k].y;
           Pmomv[k][2] = e->out[k].z;
           Abspv[k] = vec(e->out[k].x,e->out[k].y,e->out[k].z).length();
           Iflgv[k] = nie_wiadomoI; // przyjmowane wartości -1,0,2,3,4,7
            Iorgv[k] = nie_wiadomoI; //przyjmowane wartości 0,1,2,4,7
      }

      for(Int_t k = 0; k < Npar_out; k++)
      {
           Ipv[k+Npar_in] = e->out[k].pdg;
           if(e->out[k].pdg == 14) Icrnv[k+Npar_in]=0;
           else Icrnv[k+Npar_in] = 1;
           Pmomv[k+Npar_in][0] = e->out[k].x;
           Pmomv[k+Npar_in][1] = e->out[k].y;
           Pmomv[k+Npar_in][2] = e->out[k].z;
           Abspv[k+Npar_in] = vec(e->out[k].x,e->out[k].y,e-
>out[k].z).length();
           Iflgv[k+Npar_in] = nie_wiadomoI; //przyjmowane wartości
-1,0,2,3,4,7
              Iorgv[k+Npar_in] = nie_wiadomoI; //przyjmowane wartości
0,1,2,4,7
      }

      xpi[0] = nie_dotyczyF;
      xpi[1] = nie_dotyczyF;
      xpi[2] = nie_dotyczyF;
```

```
      npi[0] = nie_dotyczyF;  // nie dotyczy Nuwro
      npi[1] = nie_dotyczyF;  // nie dotyczy NuWro
      npi[2] = nie_dotyczyF;  // nie dotyczy NuWro

      cospibm = nie_dotyczyF;
      ppi = nie_dotyczyF;
      ppid = nie_dotyczyI;

      xpi0[0] = nie_dotyczyF;
      xpi0[1] = nie_dotyczyF;
      xpi0[2] = nie_dotyczyF;

      npi0[0] = nie_dotyczyF;  // nie dotyczy NuWro
      npi0[1] = nie_dotyczyF;  // nie dotyczy NuWro
      npi0[2] = nie_dotyczyF;  // nie dotyczy NuWro
      cospi0bm = nie_dotyczyF;
      ppi0 = nie_dotyczyF;

      Enu = nie_wiadomoF;            // nie wiadomo
      idfd = nie_wiadomoI;           // przypisana wartosc 5
      modef = nie_wiadomoI;          // przypisana wartosc 11 ,12

      nnu[0] = nie_wiadomoF;
      nnu[1] = nie_wiadomoF;
      nnu[2] = nie_wiadomoF;

      norm = nie_wiadomoF;
      rnu = nie_wiadomoF;

      xnu = nie_wiadomoF;
      ynu = nie_wiadomoF;


}


void NuWro_To_ND280(char * plik1,char* plik2)
```

```
{
  event *e=new event;
  TFile * ff1 = new TFile(plik1);
  TTree * tt1 = (TTree*)ff1->Get("treeout");
  tt1->SetBranchAddress("e", &e);
  int n = tt1->GetEntries();
  TFile *ff2 = new TFile(plik2,"recreate");
  TTree *tt2 = new TTree("h10","NEUT like event tree");
  ND280_event e2;
  e2.Create_Branches(tt2);

  for(int i=0;i<n;i++)
  {
    tt1->GetEntry(i);
    e2.Translate_From_NuWro(e);
    e2.Nev=i;
    tt2->Fill();
  }

  ff2->Write();
  ff2->Close();
  ff1->Close();
  delete ff2;
  delete ff1;
  }


int main (int argc, char* argv[])
{
    if(argc == 1)
    {
      NuWro_To_ND280("eventsout.root","ND280.root");
      cout<<"Plik wynikowy: ND280.root " <<endl;
    }
     else if(argc == 2)
    {
      NuWro_To_ND280(argv[1],"ND280.root");
```

```
      cout<<"Plik wynikowy: ND280.root "<<endl;
    }
    else if(argc == 3)
    {
      NuWro_To_ND280(argv[1],argv[2]);
      cout<<"Plik wynikowy: " << argv[2] << endl;
    }
}
```

Bibliografia

1) [L1] C. H. Llewellyn Smith, Phys. Rept. 3 (1972) 261
2) [N1] J. Nowak, Konstruowanie generatora oddziaływań neutrin, praca doktorska, UWR Wrocław 2006r.
3) [F1] R. P. Feynman, Phys. Rev. Lett. 23 (1969) 1415.
4) [H1] Y. Hayato, NEUT: The Neutrino Interaction imulation Used in SuperK and K2K, Nucl. Phys. Proc. Suppl. 112, 171 (2002).
5) [MP1] B.R Martin, M.K.Pidcock, Nuc. Phys. B126(1997), 266,
   B.R Martin, M.K.Pidcock, Nuc. Phys. B126(1997), 285,
   J.S.Hyslop et al., Phys. Rev. D46(1992), 961
6) [B1] H.W.Bertini, Phys. Rev. C6(1972), 631
7) [MV1] Paul Musset, Jean-Pierre Vialle, Phys. Rep. C39(1978), 1.
8) [K1] J. E. Kim et al., Rev. Mod. Phys. 53(1981), 211
9) [RS1] D. Rein, L. M. sehgal, Nucl. Phys. B223(1983), 29.
10) [BEM1]B.H. Bransden, D.Evans, J.V. Major, Czastki elementarne, Warszawa 1981r.
11) [P1] P. Przewłocki,Tokai2Kamioka – pierwszy eksperyment nowej generacji w fizyce oscylacji neutrin, Warszawa, 7-12-2007
12) http://root.cern.ch/
13) http://nngroup.physics.sunysb.edu/~mcgrew/t2k/nd280mc/v4r6/dox/index.html