Laboratorium 5, 6, 7, 8 i 9 – (czas: 5 labów)
Fizyka a Informatyka – Instalacja i korzystanie z niestandardowych pakietów i programów obliczeniowych w systemach Linux na przykładzie Quantum ESPRESSO. Obliczenia równoległe za pomocą pakietu OpenMPI – podstawa.

Miłosz Martynow
Laboratoria z Administracji i Architektury Systemów Operacyjnych
Wydział Fizyki Technicznej i Matematyki Stosowanej
Zakład Fizyki Teoretycznej i Informatyki Kwantowej
Zajęcia zdalne ze względu na kwarantanne ze względu na COVID-19 na Politechnice Gdańskiej

1. Instalacja Quantum ESPRESSO (1 lab)
2. Instalacja OpenMPI (1 lab)
3. Zapoznanie się z metodą cat i zdefiniowanie projektu (1 lab)
4-5. Wykonanie projektu (2 laby)

Wstęp

Od czasu silnego rozwoju komputerów a zwłaszcza superkomputerów, kart graficznych itp; Obok Fizyki Teoretycznej i Fizyki Eksperymentalnej powstał nowy olbrzymi dział - Fizyki Obliczeniowej. Jego głównym zadaniem jest łączenie dwóch pierwszych działów. Absolwenci fizyki o specjalnościach ukierunkowanych na informatykę na całym świecie rozwijają się głównie w kierunku właśnie Fizyki Obliczeniowej, dostarczając profesjonalne rozwiązania z zakresu szeroko rozumianych Technologii Informatycznych, aby na przykład odtworzyć teoretyczne modele (Analiza numeryczna, metody numeryczne i Algorytmika), które pomagają zinterpretować wyniki eksperymentalne, pomagają przetworzyć i archiwizować duże zbiory danych (Big data) i co najważniejsze wyciągnąć z nich wnioski oraz wiele innych.

Zaraz po zapoznaniu się z Analizą Matematyczną, Algebrą i Fizyką, Studenci Fizyki muszą opanować metody numeryczne i algorytmy implementujące je w efektywny sposób – jest to jednak temat na oddzielny przedmiot. Często się jednak zdarza (a nawet bardzo często), że pakiety numeryczne dla danego problemu są dostarczane w postaci Open Source'owych paczek, które „wystarczy" skompilować, uruchomić i przetestować na swojej maszynie po czym korzystać i implementować je w różnego rodzaju własnych projektach. Programy takie najczęściej są programami wsadowymi, czyli takimi, które przyjmują plik tekstowy o odpowiednim formacie i składzie, po czym przy wywołaniu programu ze wskazaniem na plik wsadowy odpowiednio interpretują i implementują parametry w nim zawarte i wykonują obliczenia.

Open Source'owych programów dla fizyków jest całe mnóstwo i powstaje jeszcze więcej. Jeśli ktoś napisze, rozpowszechni i sprawi, że jego pakiet będzie popularny, będzie mógł liczyć na ilości cytowań idących w dziesiątki tysięcy. Takie programy lub pakiety programów są wykorzystywane chyba w każdej dziedzinie fizyki od Mechaniki Klasycznej poprzez Mechanikę Płynów i Elektrodynamikę aż do Mechaniki Kwantowej itd. Przykładem z Mechaniki Kwantowej jest pakiet Quantum ESPRESSO, z którego plikiem outputowym/wynikowych zapoznali się Państwo na poprzednich laboratoriach. Z tym pakietem będą również związane najbliższe laboratoria. Podany przykład nie jest w żaden sposób „ogólnym" sposobem na kompilacje oprogramowania naukowego, jest to tylko i aż przykład. Można powiedzieć, że co oprogramowanie to jest oddzielna historia, często ładnie opisana, ale często też opisana bardzo słabo. Dodatkowo można powiedzieć też, że co maszyna to jest inna historia...

Dodatkowo, naukowe pakiety obliczeniowe często są bardzo dobrze zrównoleglone – czyli potrafią wykonywać jednocześnie operacje numeryczne na więcej niż jednym procesorze. Jednym z popularnych pakietów umożliwiającym wykonywanie obliczeń w sposób równoległy jest OpenMPI. Quantum ESPRESSO jest ukierunkowane na ten pakiet i w czasie tego labolatorio/projektu zainstalujemy obydwa pakiety na swoich maszynach z Linuxem, wykonamy podstawowe obliczenia testowe po czym zostanie zdefiniowane zadanie, które będzie miało na celu

wyciągnięcie fizycznych wniosków za ich pomocą i utrwalenie umiejętności pracy z Bash, Python i generalnie z Administracją obliczeniowych pakietów naukowych.

Bardzo ważne jest zachowanie kolejności laboratorium. Gorąco również zachęcam wszystkich do prowadzenia własnych projektów naukowych z tego typu pakietami czy to prywatnie czy też w ramach prac dyplomowych. Dodatkowo warto wspomnieć, że Quantum ESPRESSO jest technologią obliczeniową z zakresu mechaniki kwantowej dostarczającej rozwiązania teoretyczne z frontu światowych badań. Problemy Algorytmiczne w stylu Metod Optymalizacji, Szybkich Transformacji Fouriera itp rozwiązywane przez Quantum ESPRESSO często pokrywają się w silnym stopniu z takimi działami Informatyki jak chociażby sztuczna inteligencja.

Będę korzystał z Linuxa Ubuntu 18.04 LTS – jeśli, ktoś go nie ma a zaistnieją problemy to proponuję postawić sobie wirtualną maszynę. Niestety nie mam doświadczenia z pracą z systemami od Apple, ale z tego co mi wiadomo, to powinno instalować się podobnie. Scenariusz który przedstawie, praktycznie zawsze wcielam z sukcesem w życie na starszych wersjach Ubuntu oraz Debiana.

Instalacja Quantum ESPRESSO.

Quantum ESPRESSO (QE) jest Open Source'owym pakietem wykonującym obliczenia DFT w mechanice kwantowej dla układów periodycznych jak np. ciała stałe. Programy QE są napisane w Fortranie, gdyż przez długi czas był to jeden z szybszych i najbardziej popularnych języków w nauce.

Instalacja pod Ubuntu jest możliwa teraz poprzez:

```
sudo apt-get install -y quantum-espresso
```

Ale wolałbym abyście tego nie robili tego teraz, bo zabije to wartość dydaktyczną. A dodatkowo sam nigdy nie instalowałem QE w ten sposób więc nie wiem jakie problemy tam się mogą zrodzić. Oprogramowanie zainstalujemy w sposób manualny. Poniższe instrukcje będą wykonywane z poziomu terminala.

1. Wpierw musimy zadbać o aktualność naszych pakietów systemowych itp. Jest to operacja powtarzająca się bardzo często przy najróżniejszych projektach. Pierwsza linia odświeży nam listę rzeczy do zaktualizowana, druga zaś zaktualizuje je.
   ```
   sudo apt-get update
   sudo apt-get upgrade
   ```
2. Następny krok to instalacja trzech pakietów, które nie są domyślnie zainstalowane na systemach Linux. Jeżeli Gdzieś przy którymś z waszych projektów pojawi się taka wiązka, zawsze proponuję wykonać ją w odpowiedniej kolejności – niektóre pakiety mogą wymagać tych, które są w danej „wiązce". Pierwsza linia pobierze i zainstaluje ewentualnie brakujące trzy programy – program do edycji tekstu (vim), program do pobierania plików poprzez podanie linku (wget) i program ro rozpakowywania zip'ów (unzip) – czasem nie są one domyślnie zainstalowane. Druga linia to środowisko programistyczne dla języków C i C++ zawierające takie pakiety jak na przykład make i gcc. Druga biblioteka, to chyba najstarsza i najszybsza biblioteka do obliczana N-wymiarowej dyskretnej transformacji Fouriera napisana w języku C ale dedykowana również dla C++ i Fortrana. Trzeci pakiet to kompilator kodu napisanego w języku Fortran.
   ```
   sudo apt-get install vim wget unzip
   sudo apt-get install build-essential
   sudo apt-get install fftw3-dev
   sudo apt-get install gfortran
   ```
3. Pobieramy najnowszą wersję QE 6.5 i wypakowujemy ją do folderu domowego
   ```
   cd ~
   wget https://github.com/QEF/q-e/archive/qe-6.5.zip
   unzip qe-6.5.zip
   cd q-e-qe-6.5
   ```
4. Po wejściu do folderu zawierającego kod źródłowy QE przystępujemy do jego kompilacji. Odwiedźmy czysto informacyjnie folder ls -lh PW/src/ oraz zachęcam do wybiórczego obejrzenia niektórych części kodu jak na przykład: vi PW/src/atomic_wfc.f90. W folderze głównym QE wywołujemy bashowy plik konfiguracyjny, który między innymi sprawdzi listę pakietów i skonfiguruje folder.
   ```
   ./configure
   ```
   proces ten powinien zakończyć się komentarzem:
   ```
   configure: success
   ```
   Następnie w tym samym folderze głównym QE wywołujemy plik makefile za pomocą polecenia make all. Bez wcześniejszego poprawnego wywołania pliku konfiguracyjnego nie jest możliwe uruchomienie programu make. Rozpocznie to kompilacje wszystkich (all) pakietów obliczeniowych QE, może to zabrać chwilę.
5. Zajrzyjmy jeszcze raz do odwiedzonego już przez nas folderu ls -lh PW/src/. Widzimy, że po kompilacji powstały obiekty .o oraz skompilowany kod binarny (np. pw.x), którego wywołanie w tym folderze ./pw.x uruchomi program, który będzie czekał na plik wsadowy. Aby uzyskać całą ścieżkę do folderu przechowującego pw.x wywołujemy komendę pwd.

Wracając do katalogu domowego poprzez cd ~/ widzimy, że aby wywołać wyżej wspomniany skompilowany program, musimy już podać całą ścieżkę do niego (np. /home/user/q-e-qe-6.5/PW/src/pw.x, co jest równoważne z ~/q-e-qe-6.5/PW/src/), co na dłuższe użytkowanie może być bardzo uciążliwe. Aby to ominąć proponuję umieścić/dodać nową zmienną środowiskową, na końcu pliku .bashrc (vi ~/.bashrc). Dokonamy tego poprzez dodanie po dwukropku do zmiennej globalnej PATH, części aktualizującej listę zmiennych środowiskowych, na przykład export PATH="$PATH:~/q-e-qe-6.5/PW/src/". Jak teraz spróbujemy wywołać program pw.x w dowolnym miejscu drzewa katalogu, to możecie zauważyć, że nie działa to tak jak powinno, gdyż program nadal jest niewidoczny jako zmienna środowiskowa. Aby zaktualizować plik konfiguracyjny powłoki Linuxa .bashrc, należy jeszcze w terminalu wywołać komendę source ~/.bashrc, która go zaktualizuję – często działanie komendy source, będzie widoczne dopiero po wyłączeniu i włączeniu terminala. Teraz mamy dostęp do programu pw.x z Quantum ESPRESSO z każdego miejsca w komputerze. Po dwukropku dodajemy kolejne foldery i programy, które mają być w liście zmiennych środowiskowych PATH. Można je wyświetlić poprzez echo $PATH. W razie, gdybyśmy coś „namieszali w zmiennej środowiskowej”, zawsze można ją ręcznie zresetować poprzez wpisanie w linii komend PATH="/bin:/usr/bin" (bez żadnych programów, które mogą być wtedy nie wywoływalne), co powinno przywrócić domyślną zmienną środowiskową.

6. Powyższy przykład dodania programu/folderu do zmiennej środowiskowej, był przykładem edukacyjnym. QE składa się z dużej ilości programów kwantowo mechanicznych. Po wykonaniu make all, skompilowaliśmy wszystkie, ale są one w różnych folderach – zaś dodanie wszystkich folderów do zmiennej środowiskowej, także jest uciążliwe i co najważniejsze „nieeleganckie”. Duża ilość oprogramowania i pakietów umieszcza aliasy do skompilowanych binarek w swoim folderze bin. Folder bin skompilowanego oprogramowania QE, powinien się znajdywać w jego folderze głównym (np. ~/q-e-qe-6.5/bin/). Wywołując komendę ls -lh zobaczymy wszystkie skompilowane programy pakietu QE, wraz z ich odnośnikami. Wstawiając ścieżkę do folderu bin do wcześniej opisanych zmiennych środowiskowych, mamy dostęp do wszystkich programów pakietu QE z każdego folderu komputera. Tak też poprzez dodanie export PATH="$PATH:~/q-e-qe-6.5/bin/" (przykładowa ścieżka) do końca pliku .bashrc, zaktualizowania go poprzez source ~/.bashrc oraz ewentualnym resecie terminala – otrzymujemy dostęp do wszystkich programów z pakietu Quantum ESPRESSO z każdego miejsca w komputerze. Dodatkowo na końcu warto wspomnieć, że program source, po każdym wywołaniu, może powielać listę zmiennych środowiskowych, aby tego uniknąć warto dodać przed liniami aktualizującymi $PATH itd linię eksportującą domyślne ścieżki wartości: export PATH="/bin:/usr/bin" i np. export LD_LIBRARY_PATH="".

Instalacja pakietu OpenMPI do zrównoleglenia obliczeń QE.

Obliczenia z zakresu mechaniki kwantowej układów periodycznych, wykonywane za pomocą oprogramowania pakietu Quantum ESPRESSO mogą zostać zrównoleglone na dużą ilość procesorów co znacząco przyspiesza prędkość obliczeń. Aby zainstalować program OpenMPI (MPI) należy wykonać następujące instrukcje.

Instalacja pod Ubuntu jest możliwa teraz poprzez:

```
sudo apt-get install -y openmpi-bin
```

Ale wolałbym abyście tego nie robili tego teraz, bo zabije to wartość dydaktyczną. A dodatkowo sam nigdy nie instalowałem MPI w ten sposób więc nie wiem jakie problemy tam się mogą zrodzić. Oprogramowanie zainstalujemy w sposób manualny. Poniższe instrukcje będą wykonywane z poziomu terminala.

1. Wpierw musimy zadbać o aktualność naszych pakietów systemowych itp. Jest to operacja powtarzająca się bardzo często przy najróżniejszych projektach. Pierwsza linia odświeży nam listę rzeczy do zaktualizowana, druga zaś zaktualizuje je.
   ```
   sudo apt-get update
   sudo apt-get upgrade
   ```
2. Następny krok to zainstalowanie niezbędnych pakietów dla MPI, które są łatwo dostępne w Advanced Package Tool (apt).
   ```
   sudo apt-get install vim wget tar
   sudo apt-get install libopenmpi2
   sudo apt-get install libibnetdisc-dev
   ```
3. Pobieram i rozpakowujemy OpenMPI do katalogu domowego
   ```
   wget https://download.open-mpi.org/release/open-mpi/v4.0/openmpi-4.0.3.tar.gz
   tar -xvf openmpi-4.0.3.tar.gz
   cd openmpi-4.0.3/
   ```
4. Wywołujemy konfiguracje MPI w folderze głównym programu, ale z dodatkiem prefixu na no nowy – ukryty folder. Możemy to wykonać na dwa sposoby, poprzez podanie pełnej i jawnej ścieżki do, albo poprzez użycie zmiennej globalnej $USER, która zawiera nazwę użytkownika – można ją wywołać i wyświetlić poprzez echo $USER.
   ```
   ./configure –prefix=”/home/$USER/.openmpi”
   ```
5. W tym samym folderze wywołujemy make aby zacząć kompilacje oprogramowania. Obydwa procesy mogą zabrać chwilkę, zaś zakończą się bez żadnego specjalnego komunikatu – odzyskacie dostęp do konsoli.
   ```
   sudo make
   sudo make install
   ```
6. Aby mieć dostęp do wszystkich programów pakietu OpenMPI, to należy dodać ścieżkę do folderu bin do zmiennych środowiskowych. Wraz z poprzednio dodanym Quantum ESPRESSO, do pliku .bashrc, należy dodać po dwukropku ścieżkę do folderu bin skompilowanego pakietu OpenMPI.
   ```
   export PATH="$PATH:~/q-e-qe-6.5/bin/:~/.openmpi/bin/"
   ```
   Dodatkowo należy zaktualizować zmienną globalną zawierającą listę z linkerami do odpowiednich bibliotek/liblaries. Robimy to przez dodanie dodatkowej eksportującej zmiennej:
   ```
   export LD_LIBRARY_PATH="~/.openmpi/lib/"
   ```
   Po dodaniu aktualizacji zmiennych środowiskowych i linkerów, należy zaktualizować plik .bashrc poprzez komendę source ~/.bashrc. W razie potrzeby należy zrestartować terminal/terminale.
7. Po wykonaniu powyższych etapów powinniśmy być w stanie odpalić z każdego folderu w systemie program mpirun, który zapewni nam możliwość zrównoleglania obliczeń. Teraz program można wywołać za pomocą polecenia mpirun, z dowolnego katalogu.

Instrukcja *cat* w języku Bash.

Prowadząc jakiekolwiek obliczenia i badania numeryczne, chcemy a nawet musimy unikać sentencji *„garbage in, garbage out”* (https://pl.wikipedia.org/wiki/Garbage_In,_Garbage_Out). Pakiety numeryczne w stylu Quantum ESPRESSO są najczęściej pakietami programów wsadowych – czyli takich, które jako parametr wejściowy nie przyjmuje pojedynczych parametrów a odpowiedni plik tekstowy z zestawieniem dużej ilości parametrów fizycznych i numerycznych. Nawet jeśli teoria i programowanie jest poprawne, to jeśli damy śmieciowe dane wejściowe, to otrzymamy śmieciowe dane wyjściowe. Proces odpowiedniego dobierania parametrów często jest bardziej skomplikowany niż analiza samych wyników, wystarczy spojrzeć na ilość parametrów dostępnych dla programu pw.x, którego output już analizowaliście na zeszłych laboratoriach – a to nie jest jedyny program pakietu obliczeniowego Quantum ESPRESSO (https://www.quantum-espresso.org/Doc/INPUT_PW.html). W celu automatyzacji procesu odnajdywania odpowiednich parametrów najlepiej jest skorzystać z instrukcji cat w języku Bash. Dzięki niej przykładowo możemy jeden plik wejściowy z parametrami wrzucić w pętle i przy każdej iteracji inkrementować wartość jakiegoś parametru. Najlepiej to zobrazować poprzez przykład doboru parametru Energi Kinetycznej odcięcia funkcji falowej (Epsi).

Przykład wraz z objaśnieniami znajduję się w pliku Si_scf.sh. W folderze w którym się będzie wykonywał program, musi być również zdefiniowany i obecny plik pseudopotencjału (dołączony do maila) – w tym przypadku jest to pseudopotencjał typu USPP (Ultra Soft Pseudo Potential). W skrócie, program składa się z:

1. Definicji głównych zmiennych, a w tym arraya z różnymi wartościami Ecpsi
2. Pętli, która iteruję po elementach Epsi
    1. Oblicza wartości Erho, które w przypadku pseudopotencjałów typu USPP, muszą być większe minimum osiem razy od Epsi.
    2. Wstawia do pliku wsadowego .in wyliczone wartości. Tutaj jest funkcja cat, która tworzy w locie plik tekstowy z dobranymi paramtrami
    3. Uruchamia za pomocą pakietu MPI program pw.x, tak aby przyjął jako argument stworzony cat’em plik wejściowy i zapisał wyniki do pliku wyjściowego

Przeanalizujcie dokładnie kod Si_scf.sh.

Zadanie

Do 13.05.2020 (włącznie), wykonaj następujące zadania. Wyniki zaprezentuj w formie dokumentu pdf jako raport. Punktacja od 0 do 5 punktów, bez zadań dodatkowych, z dokładnością oceniania 0.5p. W raporcie powinien znajdować się opis każdego ze skryptu oraz tego co on ma robić jak i analiza wyników oraz opis procesu instalacji niezbędnego oprogramowania QE, MPI, itd. Sam raport jest za 2p, bez raportu żadne punkty nie będą przyznane za projekt. Bez części/rozdziału opisującej dany skrypt w raporcie, nie będzie przydzielony punkt za ten skrypt. Bazując na przykładzie Si_scf.sh, wykonaj następujące analizy:

1. Skrypt 1. (1p) Wybór optymalnego Epsi (a tym samym Erho), warto wspomnieć, że wraz z większym Epsi, wyniki są stabilniejsze, ale sam proces obliczeń znacząco się wydłuża. Tak też sporządź wykres/wykresy obliczonych wyników (pliki .out) energii całkowitej w zestawieniu z czasem obliczeń. Energia całkowita zaczyna się sentencją „!”, zaś całkowity czas obliczeń na znajduję się w ostatniej linii zaczynającej się sentencją „PWSCF : ”. Odpowiedz na pytania: Jakie Epsi i Erho wydają się optymalne oraz jak zwiększanie Erho wpływa na czas obliczeń.
2. Skrypt 2. (1p) Dla wybranego Epsi (i Erho), wykonaj benchmark czasów obliczeń względem ilości użytych procesorów. Sprawdź ile masz dostępnych procesorów i z krokiem 1 procesora wyniki zaprezentuj na wykresie. Odpowiedz na pytanie: w jaki sposób zmienia się prędkość obliczeń wraz ze zwiększaniem ilości procesorów.
3. Skrypt 3. (1p) Dla wybranych Epsi, Erho, i ilości procesorów wykonaj wykres Energii całkowitych i przerw energetycznych dla 40 parametrów celldm(1) wokół wartości 10.20 (z przykładów Quantum ESPRESSO - https://github.com/maxhutch/deprecated-quantum-espresso/tree/master/PW/examples). Tak też:

   celldm(1) = 10.00
   celldm(1) = 10.01
   celldm(1) = 10.02
   …
   celldm(1) = 10.19
   celldm(1) = 10.20
   celldm(1) = 10.21
   …
   celldm(1) = 10.40

   Odpowiedz na pytania: Dla jakiej wielkości sieci krystalicznej energia całkowita jest najniższa i co to oznacza? Dla jakiego parametru sieci krystalicznej przerwa energetyczna jest najbliższa przerwie eksperymentalnej (patrz poprzedni projekt) – literaturową wartość zacytuj wraz z przypisem publikacji.

Przypominam. Każdy skrypt wywołujący obliczenia powinien być zautomatyzowany. Aby otrzymać wyniki jedyne co mam zrobić to wywołać ./nazwa_skryptu. Program można wykonać w grupach maksymalnie 3 osobowych – ale zdalnie! bez żadnych kontaktów. Jak ktoś nie może albo nie chce pracować w sposób zdalny to musi zrobić program sam. Skrypty powinny być napisane w Bash, wykresy mogą być wygenerowane w Pythonie, ale interesują mnie same wykresy. Wersje Basha, Linuxa itd. opisane są w poprzednim projekcie.