

LABORATORIUM 7.

METODA HOUSEHOLDERA (double precision)

Zbudujemy program wykonujący eliminację metodą Householdera¹ dla zadanego układu

$$(1) \quad A\underline{x} = \underline{b}$$

i rozwiązujący ten układ. Wykorzystamy podprogramy **th.f** i **hx.f** uruchomione na poprzednich zajęciach.

Uwaga! Proszę sprawdzić, czy w podprogramie **th.f** uwzględnili Państwo przypadek, $\underline{a} \parallel \underline{b}$ (przyjmowaliśmy zwykle, że $\underline{b} = \underline{e}_1$), który trzeba potraktować nieco inaczej.

Program powinien składać się z następujących części:

- **PROGRAM GŁÓWNY**: nazwijmy go tu **hausmain.f**. Program ten podaje jedynie wymiar zadania n (wczytywany z klawiatury) i od razu wywołuje **PODPROGRAM GŁÓWNY** (nazwijmy go tu **hmain.f**). Następnie program **hausmain.f** organizuje zakończenie działania całości naszego programu.

Taka pozornie dziwna konstrukcja umożliwi nam tworzenie tablic (również wielowymiarowych) o wymiarach zdefiniowanych przez *zmienne*, a nie przez *konkretne liczby*.

- **PODPROGRAM GŁÓWNY subroutine hmain(n)**: jedyny jego argument, to wymiar zadania n . Ten podprogram wczytuje do zadeklarowanych tablic $A(n, n)$ i $b(n)$ odpowiednio macierz A (np. kolumnami) oraz wolne wyrazy \underline{b} . Następnie **hmain.f** umieszcza to wszystko w plikach na dysku. (Pamiętajmy, że przetwarzamy dane sekwencyjnie!) Można też opcjonalnie przewidzieć generowanie A i \underline{b} , zamiast czytania z klawiatury, co będzie wygodniejsze przy dużych wymiarach. Przy czytaniu danych z klawiatury warto, dla uniknięcia błędów, sygnalizować zakończenie każdej kolumny i jej numer. Po wywołaniu kolejnego, najważniejszego podprogramu **householder.f** i zamknięciu plików na dysku - powrót do **hausmain.f**.

¹Patrz Laboratorium 6

- **PODPROGRAM HOUSEHOLDER - serce naszego programu**

subroutine householder(n): tu także jedyny argument to n . Ten podprogram będzie wykorzystywał utworzone uprzednio na dysku pliki A i b rozpoznając je *po nazwie*. Pliki te należy wczytać z dysku (sekwencyjnie!). Ponadto trzeba utworzyć na dysku pliki np. $AA(n, n)$, $bb(n)$, $WW(n, n)$ i $x(n)$. Bedziemy wpisywali:

- w AA (kolumnami) przekształconą przez **th.f** i **hx.f** macierz A (trójkątna górna, ale warto dla kontroli, wpisywać wszystkie elementy kolumn)
- w bb Przekształconą kolumnę b
- w x kolumnę zawierającą rozwiązanie układu $Ax = b$, obliczone przez rozwiązanie układu z macierzą trójkątną górną $AAx = bb$
- w WW kolejne wektory w wyliczone przez podprogram **th.f**, jako kolumny macierzy trójkątnej dolnej WW . Nie będziemy ”upychać” tych wszystkich wektorów, choć to jest możliwe, aby mieć możliwość kontroli. Wektory te mogą służyć do wygenerowania macierzy Q dla rozkładu $A = QR$.

Trzeba pamiętać o tym, żeby badać czy na diagonalu macierzy AA nie natrafiliśmy na element bardzo mały, (najlepiej użyć do tego ” małego epsilon” z którym będziemy porównywać). Przez elementy diagonalne AA trzeba dzielić przy wyliczaniu (od końca!) kolejnych współrzędnych wektora x . W przypadku trafienia na taki element, trzeba ten fakt zasygnalizować (np. wypisując na ekranie odpowiedni tekst). Rozwiązanie x jest wtedy bezwartościowe. Natomiast mamy informację o rzędzie macierzy A , oraz możemy poszukiwać bazy przestrzeni rozwiązań wykorzystując macierz trójkątną górną AA . Po zamknięciu plików na dysku, **householder.f** wraca do swojego nadrzędnego programu. **Wyniki możemy oglądać edytorem (vim, mc itp.) w plikach x , AA , bb , WW , A , b .** Z podprogramów korzystamy w trybie *include*. Przypominam: każdy moduł podaje w *include* te podprogramy, z których faktycznie korzysta.

Program jest dość trudny, trzeba dobrze zaplanować operowanie indeksami przy odczytywaniu i zapisywaniu kolumn, które zmieniają długość.