Laboratorium metod numerycznych numer 1

Dla grup:*wszystkich*

(Dated: 27 II 2013)

I. WSTĘP

Na laboratoriach z metod numerycznych będziemy posługiwali się pakietem Octave, który jest darmową alternatywą dla Matlaba (*Matrix laboratory*). Pomimo dużego podobieństwa istnieją różnice pomiędzy tymi programami dlatego na zajęciach będziemy przyjmowali, że zawsze posługujemy się Octave. Pakiet Octave można pobrać ze strony http://www.octave.org. Do dyspozycji mamy wysokopoziomowy skryptowy język programowania zgodny z Matlabem posiadający m.in. mechanizmy wejścia/wyjścia. Z poziomu środowiska do dyspozycji mamy wiele pakietów i bibliotek numerycznych (*LAPACK, BLAS i wiele innych*). Natomiast do rysowania wykresów Octave używa znanego pakietu GNUPlot.

To krótkie wprowadzenie w żaden sposób nie wyczerpuje tematu jakim jest programowanie w Octavie, albo Matlabie. Ma ono służyć jedynie jako ułatwienie pierwszego kroku w tym kierunku. Dlatego koniecznie będzie korzystanie z wielu innych źródeł i pomocy jak np. dokumentacji dostępnej na stronie http://www.octave.org. Sam Octave posiada również wbudowaną pomoc i dokumentacje do, której dostęp dają odpowiednio polecenia **help**, **doc** oraz **lookfor**:

```
> help polecenie
> doc polecenie
> lookfor słowo kluczowe.
```

II. PODSTAWY MATLAB/OCTAVE

Podstawową strukturą danych w Octave jest dwuwymiarowa tablica dynamiczna (macierz). Do dyspozycji mamy dwa podstawowe tryby pracy: pierwszy to interakcja w linii

poleceń; drugi sposób to uruchamianie gotowych skryptów zapisanych w *M-plikach*.
Wszystkie liczby są standardowo przechowywane jako **double** tzn. mamy do dyspozycji zakreś $< 10^{-308}, 10^{308} >$. Tzw. *epsilon maszynowy* czyli precyzja obliczeń równy jest $eps = 2^{-t}$, gdzie $t = 52$ dla Octave jest to około $2,22 * 10^{-16}$. Do zmiany formatu wyświetlanej liczby służy polecenie **format**.

A. Podstawy

W Octave posiada również wbudowane stałe: liczbę π, jednostka urojona i, wspomnianą już precyzję zmiennoprzecinkową **eps**, nieskończoność **Inf** oraz symbol nieoznaczony **NaN** (*Not a Number*).

Do sprawdzenia ze zmienną jakiego rodzaju mamy do czynienia służy polecenie **whos**.

B. Macierze

Macierz o rozmiarze $1x1$:

```
> a = 7
```

Wektor czyli macierz o jednym wierszu:

```
> a = [2 3 5 6 8]
```

bądź kolumnie:

```
> a = [2 3 5 6 8];
> a = [2; 3; 5; 6; 8];
```

Średnik na końcu wyrażenia włącza i wyłącza *echo*.
W przypadku małych macierzy możemy budować ją bezpośrednio:

```
> A = [1 2 3; 4 5 6; 7 8 9];
> B = [1, 2, 3; 4, 5, 6; 7, 8, 9];
```

Większe macierze możemy zbudować łącząc mniejsze jako składowe:

```
> C = [A; B]
> D = [10, 11 , 12; B]
```

Możemy też definiować po kolei jej elementy:

```
> A(1,1) = 2
> B(2,1) = 5 + A(1,1)
```

W przypadku większych macierzy używamy wbudowanych poleceń:

```
> A = zeros(N,M); %Macierz zerowa o rozmiarze N x M.
> B = eye(N,M); % Macierz jednostkowa N x M. A_ii = 1, A_ij = 0.
> C = ones(N,M);%Macierz jedynkowa rozmiaru N x M.
> D = rand(N,M);%Macierz losowa o rozkładzie jednostajnym na przedziale (0,1).
> E = randn(N,M);%Macierz losowa o rozkładzie normalnym.
```

C. Wypisywanie na ekran

Do wypisywania wartości na ekran możemy używać dwóch funkcji. Najprostszą jest **disp(coś)**:

```
> disp('test')
> a = 2
a = 2
> disp(a)
2
```

Drugą nieco bardziej zaawansowaną możliwością jest znane wszystkim z C **printf(***argumenty***)** . Zamiast opisu zapraszam do zadania **1.1**.

D. Notacja dwukropkowa

Do każdego elementu macierzy możemy odwoływać się osobno np. $x = A(2,2)$, albo używając tzw. **notacji dwukropkowej** Molera (nazwisko twórcy Matlaba). Powiedzmy, że mamy macierz $A(5,5)$. Zapis $x = A(3:4,3)$ oznacza wektor składający się z drugiego i trzeciego wiersza macierzy wybrane z trzeciej kolumny. Zapis $Y = A(2:3,4:5)$ oznacza macierz $2x2$ składającą się elementów z drugiego i trzeciego wiersza i czwartej, oraz piątej kolumny. Czas na trochę praktyki:

```
> N = 5;
> i = 1:N
i =
1 2 3 4 5
> i2 = 1:2:N
i2 =
1 3 5
```

E. Operacje na macierzach

Operacje macierzowe zaczniemy od omówienia mnożenia macierzy '*':

```
> A = [1,2;3,4]
A =
1 2
3 4
> B = [1,2;3,4]
B =
1 2
3 4
> C = A*B
C =
7 10
15 22
```

```
> D = A.*B
D =
1 4
9 16
```

Należy tu od razu podkreślić operacje macierzowe od operacji *tablicowych*. Np. mnożenie każdego elementu macierzy ' .*' - operacje taką poprzedzamy kropką.
Przyjrzyjmy się jeszcze potęgowaniu *macierzowemu* vs. potęgowaniu *tablicowemu*:

```
> C = A^ 2
C =
7 10
15 22
> C = A.^ 2
C =
1 4
9 16
```

F. Konstrukcie programistyczne

Zacznijmy od najprostszego czyli instrukcji warunkowej **if**. W Octave przyjmuje ona postać:

if *wyrażenie warunkowe*
instrukcje
else
instrukcje
end

Możliwa jest również inna wersja tej instrukcji z użyciem **elseif**:

if *wyrażenie warunkowe*
instrukcje

elseif *wyrażenie warunkowe*
instrukcje
end

Oczywiście do dyspozycji mamy standardowy zestaw pętli znanych z choćby jeżyka C. A mianowicie pętlę **for**:

```
for i = indeksy
instrukcje
end
```

Wykorzystując notację dwukropkową Molera:

```
for i = 1:2:20
> disp(i)
>end
1
3
5
7
9
```

oraz pętlę **while**:

while *wyrażenie warunkowe*
instrukcje
end

G. Wektoryzacja i kod macierzowy

Większość operacji jeśli tylko jest to możliwe należy starać się wykonywać wektorowo. Jednym z ćwiczeń jest porównanie implementacji opartej na pętli **for** mnożenia macierzy z implementacją wektorową. Powodem dla, którego powinniśmy tak robić jest fakt, że używane przez nas pakiety są w rzeczywistości interpreterami. Kod jest odczytywany wiersz

po wierszu i wykonywany krok po kroku, a nie jak w przypadku C/C++ kompilowany przed uruchomieniem. Skutkuje to dużo wolniejszym wykonaniem programu. Aby temu zaradzić Matlab, a także w pewnym stopniu Octave wykorzystuje JIT (Just-In-Time), który optymalizuje i kompiluje proste pętle. Niestety przyspieszeniu ulegają tylko najprostsze konstrukcje programistyczne. Istnieje jednak jeszcze jedna metoda przyspieszenia obliczeń, którą jest stosowanie wbudowanych funkcji. Funkcje te są najczęściej w przypadku Octave wywołaniami bardzo szybkich funkcji napisanych w Fortranie lub C.
A jak to wygląda w praktyce? Najpierw wersja przy użyciu pętli **for**:

```
> for x = 1:.01:10
> y = sin(x)
> end
y = 0.84147
y = 0.84683
y = 0.85211
y = 0.85730
y = 0.86240
...
```

Teraz to samo przy użyciu zapisu wektorowego:

```
> x = 0:.01:100
> y = sin(x)
```

lub przy użyciu wbudowanej funkcji **linspace**:

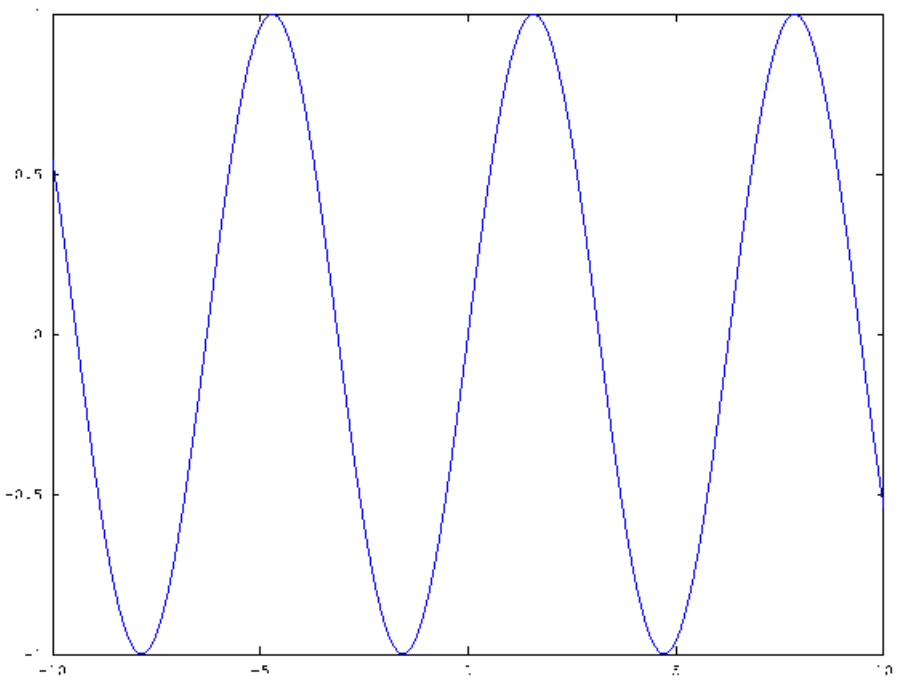
```
> x = linspace(0,10,100);
> y = sin(x)
```

Użycie operacji wektorowych staje się szczególnie istotne w przypadku macierzy. (Zadanie 1.9).

H. Prezentacja wyników

Na koniec kilka słów na temat graficznej prezentacji wyników. Octave do rysowania używa znanego dobrze GNUPlota. Podstawową instrukcją jest **plot(x,y)** tworzący wykres 2D.

```
x = -10:0.1:10; plot (x, sin (x))
```

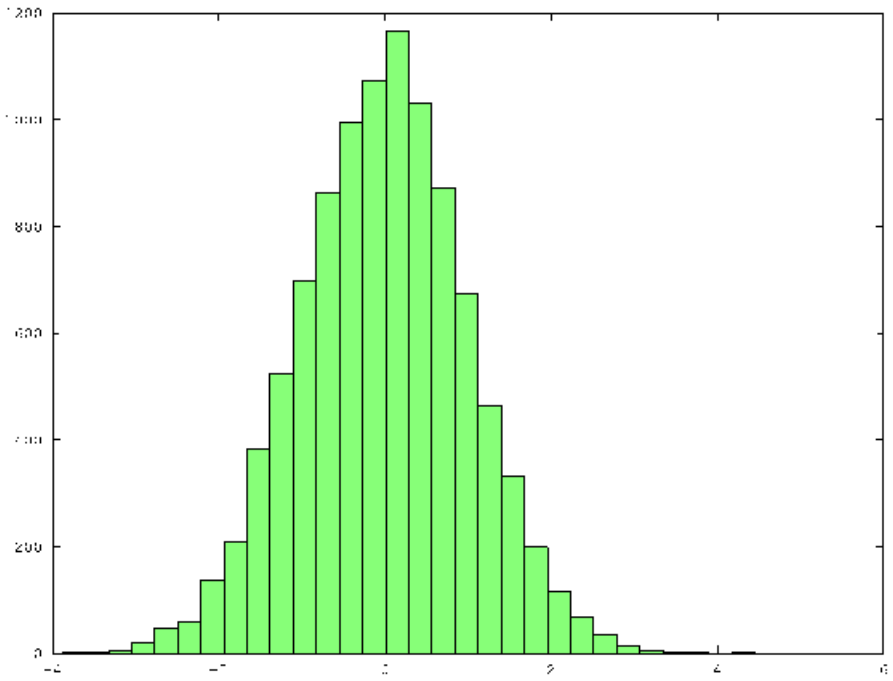


Rysunek 1: Wykres sunkcji $Sinus(x)$.

Drugą przydatną funkcją wbudowaną jest rysowanie histogramów. Służy do tego **hist**(*argumenty*):

```
  hist (randn (10000, 1), 30)
```

gdzie **randn**(*10000, 1* losuje 10000 liczb z przedziału $(0, 1)$ o rozkładzie normalnym.



Rysunek 2: Przykład histogramu.

III. ZADANIA NA DZIŚ

A. Zadanie 1.1

Korzystając z systemu pomocy wbudowanego w Octave wypisz na ekran przy użyciu instrukcji **printf** w trybie interaktywnym w jednym zdaniu odpowiedź na pytanie:

Który przedmiot jest Twoim ulubionym i dlaczego są to *metody numeryczne*?

B. Zadanie 1.2

Oblicz wartość a dla $a = \frac{1}{\exp -(15-a)/2}$. Pokaż wynik używając różnych formatów wyświetlania wyników.

C. Zadanie 1.3

Napisz program w Octave obliczający sumę liczb nieparzystych od zera do stu. Kod programu zapisz w tzw. M-pliku.

D. Zadanie 1.4

Narysuj wykres funkcji $y = sin(x^2)/2$ dla argumentu od 0 do 100.

E. Zadanie 1.5

Napisz program wykonujący poprzednie zadanie prealokując zmienne.

F. Zadanie 1.6

Napisz program rysujący otrzymaną w poprzednim zadaniu funkcję dla różnej liczby punktów np. 10 i 200.

G. Zadanie 1.7

Stwórz funkcję Octave obliczającą wartość funkcji $y = sin(x^2)/2$.

H. Zadanie 1.8

Napisz skrypt, który wykonuje po kolei następujące operacje na macierzach:
a) Utwórz macierz $10x20$ (10 wierzy na 20 kolumn), w której w pierwszych pięciu wierszach elementy przyjmują wartości 1, a w kolejnych pięciu wartość 3;
b) Zmień wartość elementów macierzy od 6 do 8 wiersza i 12 do 14 kolumny na równą 25;
c) Zmień wartość elementów macierzy od 8 do 10 wiersza i 2 do 5 kolumny na równą wartości danego elementu do kwadratu;

I. Zadanie 1.9

Zaimplementuj mnożenie macierzy przy pomocy metod iteracyjnych. Porównaj czasy wykonania dla dużych macierzy dla wersji wektorowej.

J. Zadanie 1.10

Stosując własną implementację funkcji *arcus sinus* (w postaci funkcji Octave), napisz program wyznaczający pierwsze dziesięć cyfr rozwinięcia liczby π przy użyciu wzoru Johna Machina:

$$\frac{\pi}{4} = 4\arctan(\frac{1}{5}) - \arctan(\frac{1}{239})$$

Ile składników szeregu Taylora rozwiniecia *arcusa tangensa* trzeba zsumowac, zeby obliczyc dokładnie dziesiec cyfr rozwiniecia liczby π , przy uzyciu wzoru Machina. Wynik porównaj z obliczeniami przeprowadzonymi bezposrednio przy uzyciu rozwiniecia *arcusa tangensa*, tzn.:

$$\pi = 4\arctan(1)$$

K. Zadanie 1.11

Napisz program rysujący histogram dla ruchów Browna.