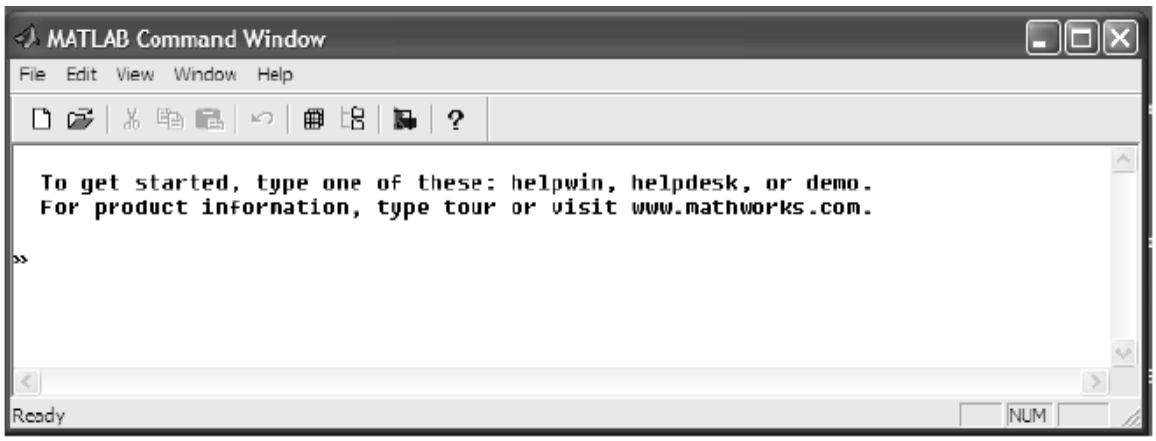
WPROWADZENIE DO MATLAB

FUNKCJE OGÓLNE

Matlab jest interaktywnym środowisdkiem , w którym użytkownik może realizować całkiem złożone zadania obliczeniowe kiloma poleceniami. Pierwotnie stworzonyu w latach siedemdziesiątych przez Cleve'a Mullera. Początkowym językiem programowania był Fortran i jakiś czas ewoluował. Ostatnie wersje są w języku C. Jeśli chodzi o programowanie numeryczne, usuwa z programowania wiele rutynowych zadań i pozwala na skocnetrowaniu się na eksperymentowaniu. Wyniki obliczeń można zobaczyć zarówno numerycznie jak również w postaci wykresów 2D i 3D, łatwo i szybko. Zawiera najnowsze narzędzia rozwiązań numerycznych, więc można być pewnym wyników. Ponadto , całkiem złożone wyliczenia mogą być wykonywane tylko za pomocą kilku poleceń. Wynika to z faktu ,że szczegóły programowe są przechowywane w oddzielnych plikach skryptowych nazywanymi plikami "m" i mogą być wywoływane bezpośrednio po nazwach. Plik m może wywołać inny plik m w razie potrzeby. W ten sposób seria plików m działa za kulisami pozwalając na łatwe wykonanie wymaganych zadań. Użytkownik może pisac swoje własne pliki m. Wszystkie takie skrypty są czytelnymi plikami tekstowymi, które można odczytać, zmodyfikować i łatwo drukować.Ta otwarta architektura Matlab pozwala programistom do tworzenia własnych obszarów określonych zbiorem plików m. Niektóre z tych zestawów napisane zostały przez ekspertów na całym świecie i dołączone do Matlab jako skrzynka narzędziowa. Tak więc w standardowej instalacji znajdziesz wiele takich skrzynek narzędziowych. Jeśli chcesz możesz ściągnąć takie skrzynki z Internetu.

Start

Kiedy klikniesz ikonę Matlab, Windows otworzy standardowe okno Matlab



Biały obszar w środku jest to obszar roboczy, w którym użytkownik wpisuje polecenia, interpretowane bezpośrednio tam, a wyniki są wyświetlane na ekranie. ">>" to znak zachęty Matlab wskazujący ,że tu można wpisywać polecenia. Wcześniejsze polecenia mogą być dostępne sprzez użycie strzałek góra dół na klawiaturze.

Proste obliczenia

Matlab używa standardowych operatorów arytmetycznych + - / * ^ dla wskazania dodawania, odejmowania, dzielenia mnożenia, i podnoszenia do potęgi, odpowiednio. Na przykład aby wyliczyć $2+3^4$, możemy napisać następująco:

```
» 2+3^4
ans =
     83
```

Pierwsza linia jest używana przez użytkownika dla wpisania polecenia, pdczas gdy druga linia jest domyślną zmienną używaną przez Matlab dla przechowywania danych wyjściowych obliczeń , a trzecia linia pokazuje wynik obliczenia. Jeśli chcesz pomnożyć ten wynik przez 2, zrób tak:

```
» ans*2
ans =
   166
```

Jak widać, wynik 83 jest przechowywany w zmiennej ans, którą mnożymy przez 2, a wynik nowego obliczenia ponownie jest przechowywany w 'ans'. W tym rzypadku, poprzednia wartość jest nadpisywana przez wartość nową. Jeśli chcesz ,możesz zdefiniować własne zmiene.Na przykład:

```
» pay=2400
pay =
        2400
```

W tym przypadku definiujesz zmienną 'pay' i przypisujesz jej wartość 2400. Matlab odzwierciedla przypisanie w drugiej i trzeciej linii. To potwierdza użytkownikowi ,że wartość 2400 została przypisana do zmiennej 'pay'. Jeśli chcesz to odbicie usunąć w Matlab, użyj średnika na końcu każdego polecenia. Na przykład:

```
» c=3*10^8;
»
```

Zmiennej 'c' została przypisana wartość $3x10^8$ a ponieważ jest średnik na końcu polecenia, dlatego nie pojawia się odzwierciedlenie.

Arytmetyczne operatory mają następujące poziomy pierwszeństwa:

1. Najpierw nawiasy. W przypadku ich zagnieżdżenia, wtedy sekwencja jest od wewnętrznych do zewnętrznych.
2. Podnoszenie do potęgi
3. Mnożenie i dzielenie. Jeśli są tam sprzeczne operatory, sekwencja jest od lewej do prawej strony
4. Dodawanie i odejmowanie. W tym przypadku również, jeśli są sprzeczne operatory, sekwencja jest od lewej do prawej strony.

Liczby i przechowywanie

Matlab wykonuje wszystkie obliczenia z podwójną precyzą i może je wykonywać z następującymi typami danych:

Liczba	**Szczegóły**
Integer (całkowita)	Liczba bez części ułakowej i punktu dziesiętnego.Np 786

Real (rzeczywista)	Liczba z częścią ułamkową Np. 3.141559
Complex (zespolona)	Liczba mająca część rzeczywistą i urojoną np 3+4i. Matlab traktuje 'i' jak i 'j' jako √-1
Inf (nieskończoność)	Nieskończoność Np. wynik dzielenia z zerem
NaN	Nie liczba Np. 0/0

Dla wyświetlenia wyników, Matlab używa polecenia Format :

Kategoria	**Szczegóły**
format short	4 miejsca dziesiętne (3.1415)
format short e	4 miejsca dziesiętne z wykładnikiem (3.1415e+00)
format long 3	normalne z wykładnikiem (3.1415926535897e+00)
format bank	2 miejsca dziesiętne (3.14)

'format' bez przyrostka, oznacza ,ze od teraz powinien być stosowany domyślny format. Poza tym, 'format compact" tłumi wszelkie puste linie w danych wyjściowych

Nazwy zmiennych

Matlab pozwala użytkownikom na definiowanie zmiennych z nazwami zawierającymi litry i cyfry, pod warunkiem ,że zaczynają się od litry. Łącznik % i inne tego typu znaki nie są dozwolone w nazwach zmiennych. Ponadto, nazwy zarezerwowane nie powinny być używane jako nazwu zmiennych. Na przykład pi, i , j , e są zarezerwowane. Podobnie nazwy funkcji i polecenia Matlab powinny być unikane jako nazwy zmiennych. Struktura poleceń Matlab jest całkiem podobna do języka C. Zmienne są czułe na wielkość liter, więc ALPHA i alpha są traktowane jako oddzielne zmienne. Ta czułość na wielkość liter stosuje się również do poleceń Matlab. Ogólna zasada jest taka ,że zazwyczaj używamy małych liter dla nazw zmiennych i poleceń.

FUNKCJE

Matlab ma mieszankę funkcji. Niektóre z tych funkcji są funkcjami standardowymi wliczającymi funkcje trygonometryczne itp. A inne są funkcjami zdefiniowanymi przez użytkownika. Wszystkie umożliwiają użytkownikowi przeprowadzenie złożonych zadań obliczeniowych łatwiej

Funkcje trygonometryczne

Obejmują funkcje sin, cos i tan. Ich argumentami powinny być radiany. W przypadku danych w stopniach , powinny być skonwertowane do radianów przez pomnożenie ich przez pi/180/. Na przykład, obliczmy wartość $\sin^2(27^o) + \cos^2(27^o)$:

```
» (sin(27*pi/180))^2+(cos(27*pi/180))^2
ans =
     1
```

Wynik tego obliczenia nie jest żadną niespodzianką.Zwróć uwagę ,że w każdym przypadku, argument funkcji trygonometrycznej został skonwertowany na radiany przez właściwe mnożenie. Funkcje odwrotne są wywoływane przez asin, acos i atan. Na przykład, $\tan^{-1}(1)$ jest wyliczana :

```
» atan(1)
ans =
    0.7854
```

Oczywiście π / 4 = 0.7854

Funkcje elementarne

Zwykle używamy powszechnych funkcji takich jak sqr, expp, log i log10. Zwróć uwagę ,że funkcja log podaje logarytm naturalny, więc

```
» x=2; sqrt(x), exp(-x), log(x), log10(x)
ans =
    1.4142
ans =

    0.1353
 ans =
    0.6931
 ans =
    0.3010
```

Tutaj,wszystkie cztery funkcje zostało przetestowane przy użyciu tego samego polecenia.

Funkcje	**Oznaczenie**
abs	Wartość absolutna
sqrt	Funkcja pierwiastka kwadratowego
sign	Funkcja signum
conj	Koniugacja liczby zespolonej
imag	Część urojona liczby zespolonej
real	Część rzeczywista liczby zespolonej
angle	Faza kąta liczby zespolonej
cos	Funkcja cosinus
sin	Funkcja sinus
tan	Funkcja tangens
exp	Funkcja wykładnicza
log	Logarytm naturalny
log10	Logarytm dziesiętny
cosh	Hiperboliczna funkcja cosinus
sinh	Hiperboliczna funkcja sinus
tanh	Hiperboliczna funkcja tangens
acos	Odwrotny cosinus
acosh	Odwrotny hiperboliczny cosinus
asin	Odwrotny sinus
asinh	Odwrotny hiperboliczny sinus
atan	Odwrotny tangens
atan2	Dwa argumenty formy odwrotnego tangensa
atanh	Odwrotny hiperboliczny tangens

round	Zaokrąglenie do najbliższej liczby całkowitej
floor	Zaokrąglenie w kierunku minus nieskończoność
fix	Zaokrąglenie w kierunku zera
ceil	Zaokrąglenie w kierunku plus nieskończoność
rem	Pozostałość po dzieleniu

W Matlab, są dwa typy wektorów : wektor wierszowy i wektor kolumnowy

Wektor wierszowy

Wektory wierszowe są podmiotami ujętymi w parę nawiasów kwadratowych z liczbami oddielonymi albo spacjami albo przecinkami. Na przykład, możemy wprowadzić da wektory U i V jako :

```
» U=[1 2 3]; V=[4,5,6]; U+V
ans =
     5     7     9
```

Te dwa wektory wierszowe najpierw są definiowane a potem ich suma U+V jest obliczana. Wynik jest pdoany jako wektor wierszowy i przechowywany jako ans. Można przeprowadzać bardzo łatwo operacje wektorowe:

```
» 3*U+5*V
ans =
    23    31    39
```

Powyższy przykład oblicza liniowe połączenie U i V. Można połączyć wektory do postaci innego wektora:

```
» W=[U, 3*V]
W =
   1   2   3  12  15  18
```

Wektor U i V , oba mają długość 3, połączone są do postaci sześcio elementowego wektora W. Komponenty wektora mogą być sortowane za pomocą funkcji sort:

```
» sort([8 4 12 3])
ans =
     3     4     8    12
```

Wektor [8 4 12 3] został posortowany

Notacja dwukropkowa

W celu sformowania wektora jako ciagu cyfr, można użyć notacji dwukropka. Zgodnie z nią, a: b : c daje sekwencje liczb zaczynających się od 'a' z możliwym zakończeniem 'c' i krokiem 'b'. Na przykład 1:0:5:2 daje następujący wektor kolumnowy:

```
» 1:0.5:2
ans =
   1.0000   1.5000   2.0000
```

Zauważ ,że w niektórych przypadkach górna granica nie może zostać osiągnięta. Na przykład, w przypadku 1:0:3:2, górna granica nie jest osiągnięta a wektor wynikiowy jest :

```
» 1:0.3:2
ans =
    1.0000    1.3000    1.6000    1.9000
```

Jeśli są podane tylko dwa "zakresy" specyfikacji wtedy automatycznie zakładany rozmiar kroku. Na przykład 1:4 ozancza:

```
» 1:4
ans =
     1     2     3     4
```

W przypadku kiedy zakres nie jest poprawny, pojawi się komunikat o błędzie:

```
» 1:-1:5
ans =
  Empty matrix: 1-by-0
```

Tutaj, zakres liczb podanych dla generowania wektora wierszowego została podany od 1 do 5 z krokiem -1. Oczywiście nie można dojść do 5 z krokiem 1 przy użyciu rozmiaru kroku -1. Dlatego, Matlab wskazuje ,że jest to błąd pustej macierzy.

Sekcje wektora

Zdefiniujmy wektor używając zakresu notacji:

```
» W=[1:3, 7:9]
W =
     1     2     3     7     8     9
```

Teraz chcielibyśmy wyodrębnić dwa środkowe elementy tego wektora. Można to zrobić z zakresem notacji. Jak widać , dwa środkowe elementy są zzakresu 3:4. Dlatego też, wymaganą część wektora można uzyskać:

```
» W(3:4)
ans =
7
```

To jest rzeczywiście wymagana część. Jest wiele ciekawych rzeczy jakie można zrobić używając zakresu zapisu. Na przykład, zakres 6:-1,1 jest zakresem malejącym a kiedy używamy z częścią ekstrakcji wektora, daje :

```
» W(6:-1:1)
ans =
     9     8     7     3     2     1
```

co jest wektorem W, którego wszystkie wejścia teraz są w odwróconym porządku. Więc wektor może być łatwo odwracany. Funkcja 'size' podaje długość wektora. Dla danego wektora V, V(size(V): -1,1) będzie go odwracać. Należy pamiętać ,że przerzucanie odcinków wektora jest

również możliwe

Wektory kolumnowe

Wektory kolumnowe w Matlab są formowane przez użycie zbioru liczb w parach w nawiasach kwadratowych, oddzielonych średnikiem. Dlatego też można zdefiniować dwa wektory kolumnowe A i B i dodać je jak poniżej

```
» A=[1;2;3]; B=[4;5;6]; A+B
ans =
     5
     7
     9
```

Te dwa wektory kolumnowe zostały zdefiniowane a potem uzyskaliśmy ich sumę. W podobny sposób, wszystkie standardowe działania mogą być wykonane na wektorach kolumnowych

Transpozycje

Oczywiście, nie ma prostego sposobu tworzenia wektora wierszowego przy tworzeniu wektora kolumnowego. Ale można to zrobić tworząc najpierw wektor wierszowy przy użyciu notacji zakresu a potem transponowanie wyniku wektora wierszowego do wektora kolumnowego. Transpozycja jest uzyskiwana jak pokzano poniżej:

```
» A=[1:4]; B=A'
B =
     1
     2
     3
     4
```

Tu pierwszy wektor wierszowy [1 2 3 4] jest formowany i nazywany A. Wektor ten jest potem transponowany do postaci B – wektora kolumnowego

Uwaga: Jeśli C jest wektorem złożonym, wtedy C ' daje złożoną koniugację transpozycji wektora

```
» C=[1+i, 1-i]; D=C'
D =
   1.0000 - 1.0000i
   1.0000 + 1.0000i
```

Wketor C był złożonym wektorem a jego złożona koniugacja to wektor [1-i 1+i]. Wektor D jest wyraźnie złożoną koniugacją transpozycji wektora. Czasami , jeden nie chce być częścia złożonej koniugacji.Aby uzyskać prostą transpozycje, użyj **.'** aby uzyskać transpozycję. Na przykład:

```
» C=[1+i, 1-i]; E=C.'
```

```
E =
   1.0000 + 1.0000i
   1.0000 - 1.0000i
```

W tym przypadku, jawna transpozycja C jest przechowywan w E i nie pojawia się żadna część złożona koniugacji.

Dzienniki i sesje

W Matlab ,możemy zacząć przechowywanie całego tekstu jaki pojawia, się na ekranie w oddzielnym pliku przez użycie polecenia "diary filename" nazwa pliku powinna być poprawną nazwą pliku różniącą sie 'on' i 'off'. Zapis dziennika może być "włączane" lub "wyłączane" przez polecenia "diary on" i "diary off". Ponadto jeśli chcesz teraz przerwać sesję i zacząć od tego samego stanu kolejnym razem, można zpisać i załadować używając polecenia "save filename" i "load filename". Polecenie zapisu będzie zapisywało wszystkie zmienne użytych w tej sesji do pliku, którego nazwa podana jest w poleceniu "save filename" a odpowiednie polecenie ładowania będzie odczytywało ją podczas ostatniej sesji. Przy okazji, pełan lista wszystkich zmiennych użytych w bieżącej sesji można zaobserowować stosując polecenie 'who' :

```
» who
Your variables are:
A          D          V          c
B          E          W          pay
C          U          ans        x
```

Wartości i dalsze szczegóły są również dostępne poleceniem 'whos':

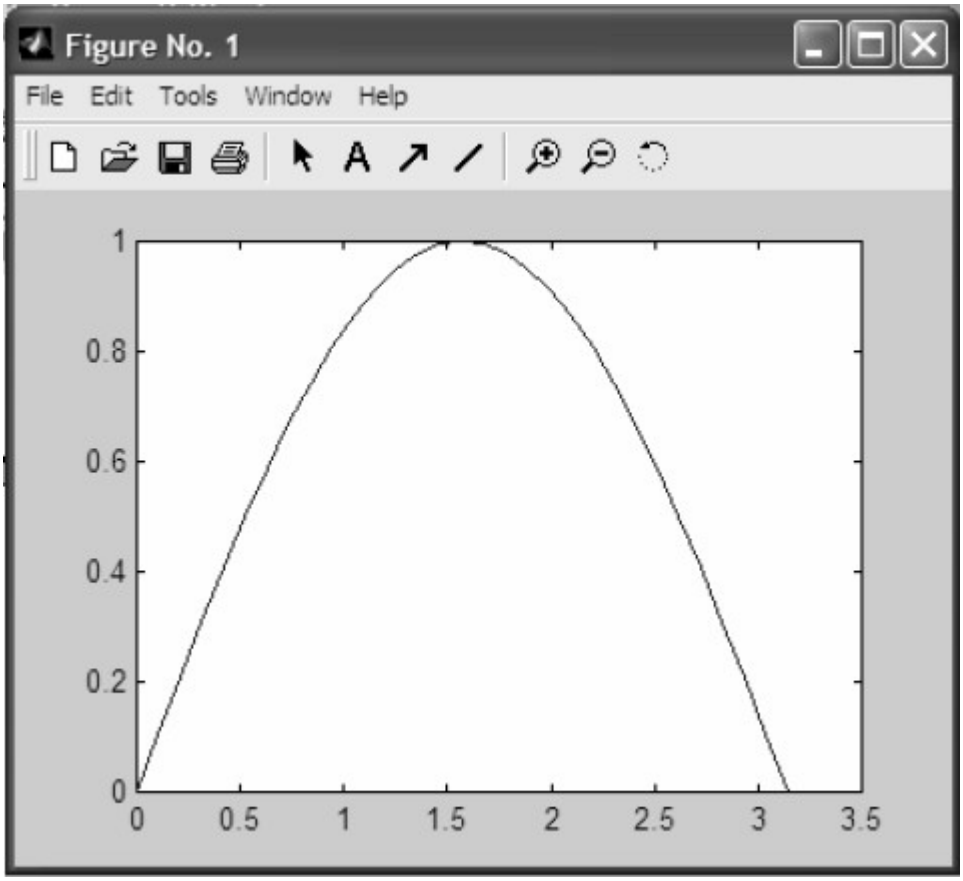
```
» whos
  Name        Size              Bytes  Class

  A           1x4                  32  double array
  B           4x1                  32  double array
  C           1x2                  32  double array (complex)
  D           2x1                  32  double array (complex)
  E           2x1                  32  double array (complex)
  U           1x3                  24  double array
  V           1x3                  24  double array
  W           1x6                  48  double array
  ans         2x1                  32  double array (complex)
  c           1x1                   8  double array
  pay         1x1                   8  double array
  x           1x1                      8  double array
Grand total is 31 elements using 312 bytes
```

Elementarne nanoszenie i wykresy

Matlab oferuje potężne narzędzia do grafiki i wizualizacji. Zacznijmy od najbardziej podstawowych funkcji graficznych Matlab. Wykres funkcji sinus od 0 do π uzyskamy w następujący sposób:

```
» N=30; h=pi/N; x=0:h:pi; y=sin(x); plot(x,y)
```

Tu w pierwszym kroku, calkowita liczba punktów pomiarowych dla tej funkcji jestz oznaczona N, i przypisano jej wartość 30. Następnie wielkość kroku "h" jest zdefiniowana i wektor wierszowy x o rozmiarze N+ jest definiowany wraz z odpowiednim wektorem wierszowym y złożonym z wartości funkcji. Polecenie "plot(x y)" wygeneruje wykres tych danych i wyświetli o oddzielnym oknie oanzaczonym jako Figure No 1:

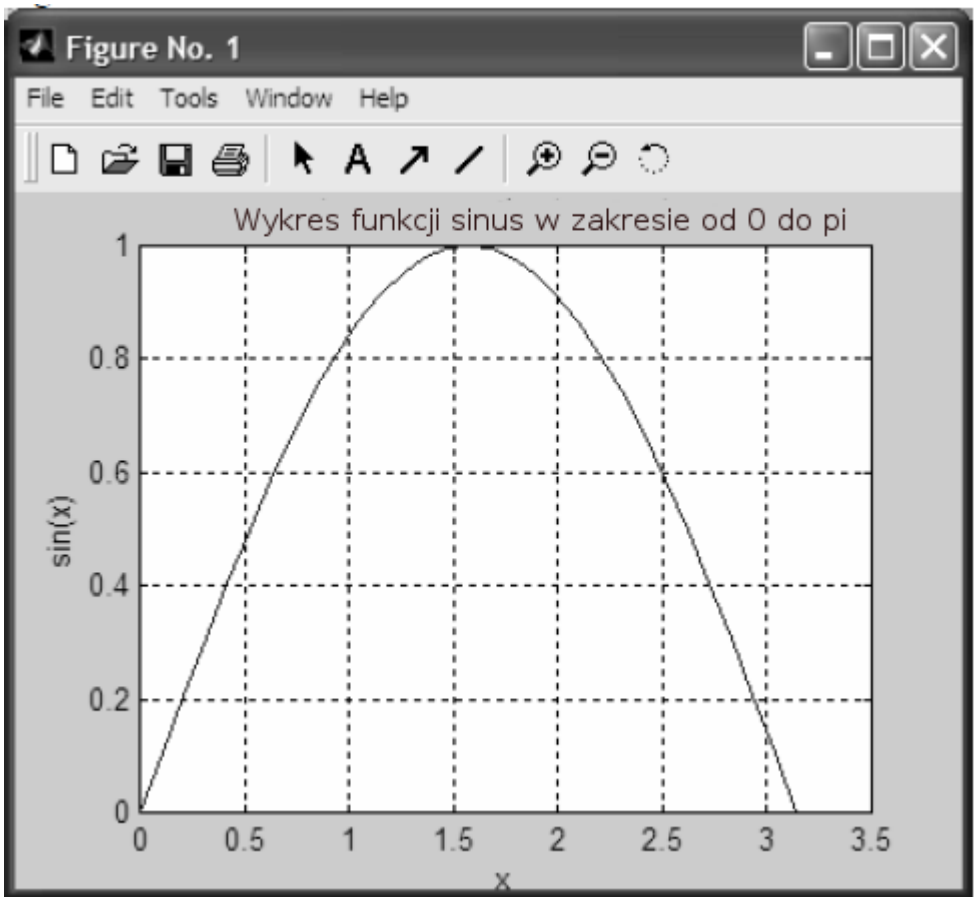


Ten wykres może być powiększany i zmniejszany. Osie x i y również mogą być przeskalowywane za pomocą myszki i użycia właściwych przycisków oraz pozycji menu. Tytył wykresu, etykiety x i y mogą być przypisywane przez następujące polecenia:

```
>> title(' Wykres funkcji sinus w zakresie od 0 do pi ')
>> xlabel('x')
>> ylabel('sin(x)')
```

Należy pamiętać ,że użycie takich poleceń ,daje odpowiednie odpowiedzi bezpośrednio w oknie wykresu

Linie siatki mogą być włączane lub wyłączane przy użyciu polecenia 'grid'



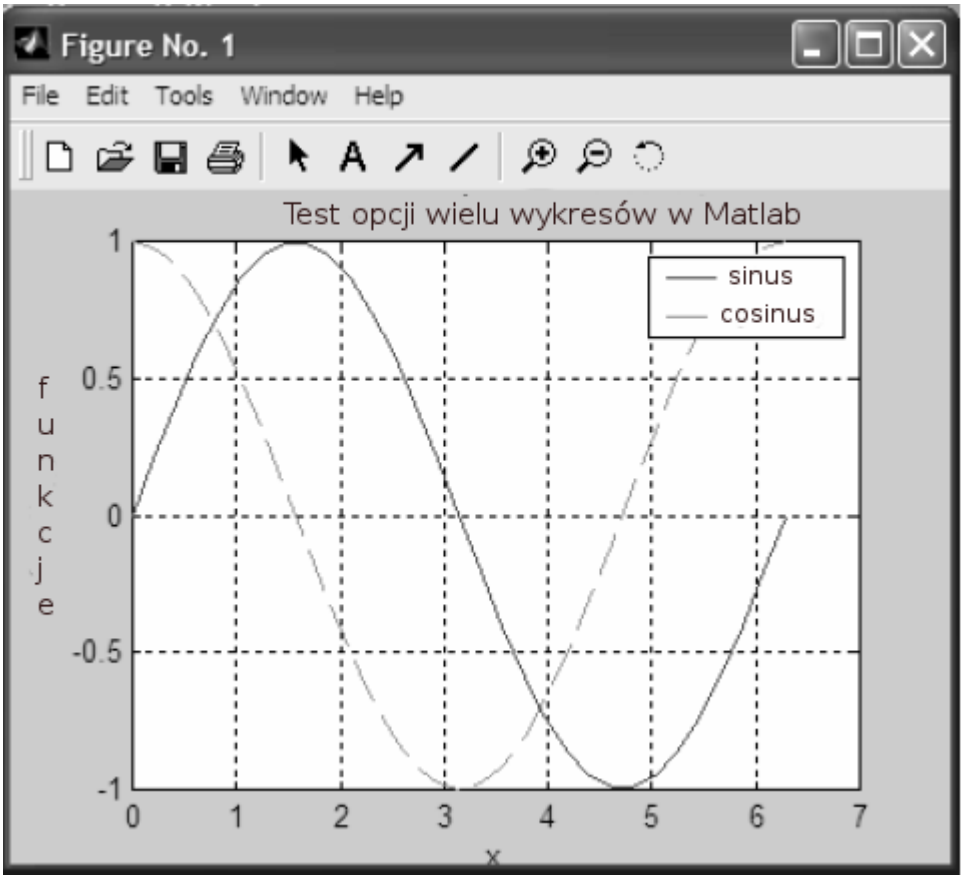
Matlab pozwala użytkownikowi na zmianę koloru jak również stylizować linie wykresu przez użycie trzeciego argumentu w poleceniu plot. Na przykład plot(x,y,'w-') będzie wykreślało dane x-y używając koloru białego (w) i linię ciągłą (-). Takie opcje są podane poniżej

Symbol koloru	**Kolor**	**Symbol linii**	**Typ linii**
y	żółty	.	Punkt
m	karmazyn	o	Okrąg
c	niebieskozielony	X	znak x
r	czerwony	+	znak plus
g	zielony	-	znak minus
b	niebieski	*	gwiazdka
w	biały	:	kropkowany
b	czarny	-.	kreska kropka
		--	linia

Teraz spróbujemy nanieść więcej niż jedną krzywą na ten sam wykres. Funkcje to sinus i cosinus. Zakres jest od 0 do 2π , w tym przypadku. Liczba punktów pomiarowych jest ustawiona na 15

```
»   N=15;|  h=pi/N;   x=0:h:2*pi;   plot(x,sin(x),'r-
',x,cos(x),'g--')
» legend('sinus ','cosinus ');
» grid
» xlabel('x');
» ylabel(' funkcje    ');
» title(' Test opcji wielu wykresów w Matlab                    ');
```

Wynik pokazano tutaj:



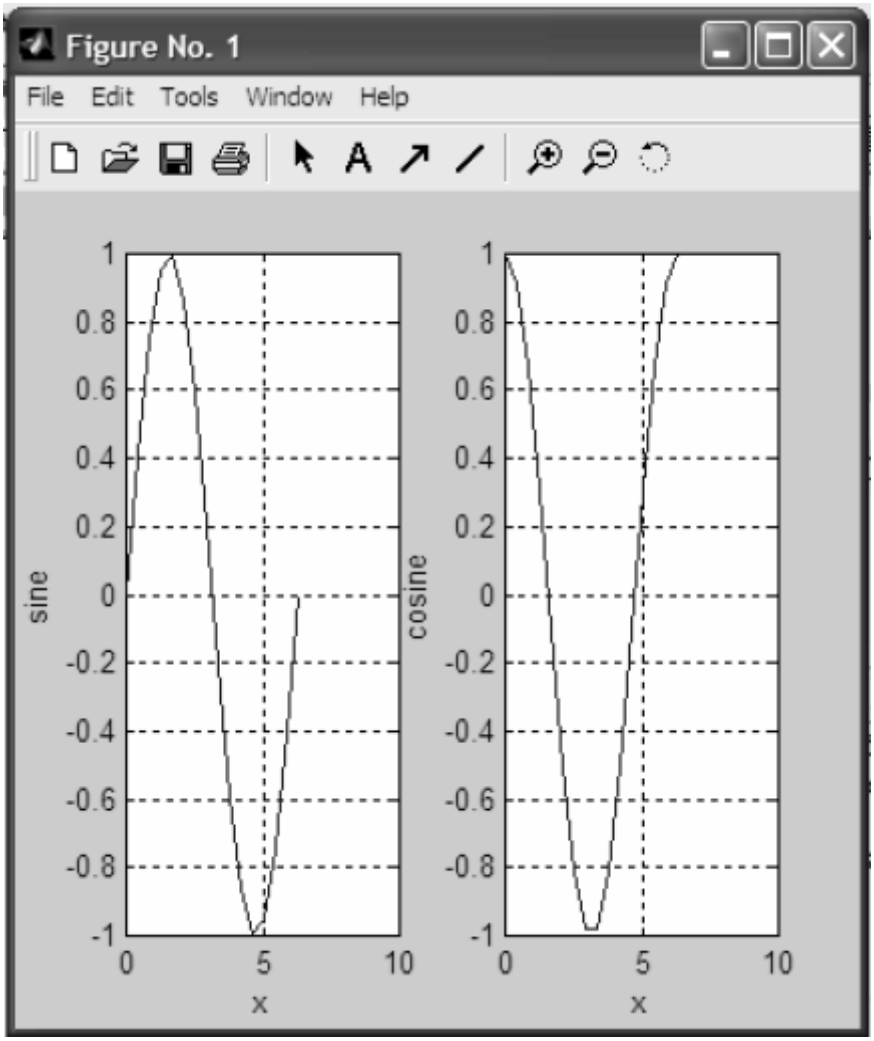
Należy pamiętać , że polecenie plot z tymi samymi trzeam opcjami powtarza się dwukrotnie generując wykres dwóch krzywych. To może być rozszerzone dla twoich potrzeb.Ponadto, polecenie legend pozwala na generowanie legendy dla tego wykresu, która może być umieszczona swobodnie przez użytkownika przez kliknięcie i przeciągnięcie na wykresie, i zwalniając przycisk myszki w pożądanym położeniu. Każde polecenie plot usuwa poprzednie okno grafiki (Figure No 1) i rysuje na nim. Jeśli chcesz możesz wysyłać wykres do tego samego okna przy pierwszym użyciu polecenia hold a pónej wysłanie wykresu do niego poleceniem plot. Polecenie hold może być wyłączone przez 'hold off' kiedy to jest pożądane.

Wykresy boczne

Rozważmy teraz inną sytuację. Chcemy wykreślić funkcje sinus i cosinus ponownie w zakresie 0 do 2π ale na oddzielnych wykresach. Jeśli użyjemy dwóch oddzielnych poleceń plot, wcześniejszy wykres zostanie wymazany. Jeśli użyjemy hold, wtedy uzyskamy wielowykresowść, a tego nie

chcemy. Chcemy dwóch wykresów umieszczonych obok siebie. Jest to uzyskiwane dzięki poleceniu subplot, które dzieli okno graficzne na tablice sekcji subwykresów. Stworzymy panel 1x2 (jeden wiersz, dwie kolumny):

```
» N=15;h=2*pi/N; x=0:h:2*pi;
» subplot(122);plot(x,cos(x));xlabel('x');
  ylabel('cosine');grid
» subplot(121);plot(x,sin(x));xlabel('x');
  ylabel('sine');grid
```



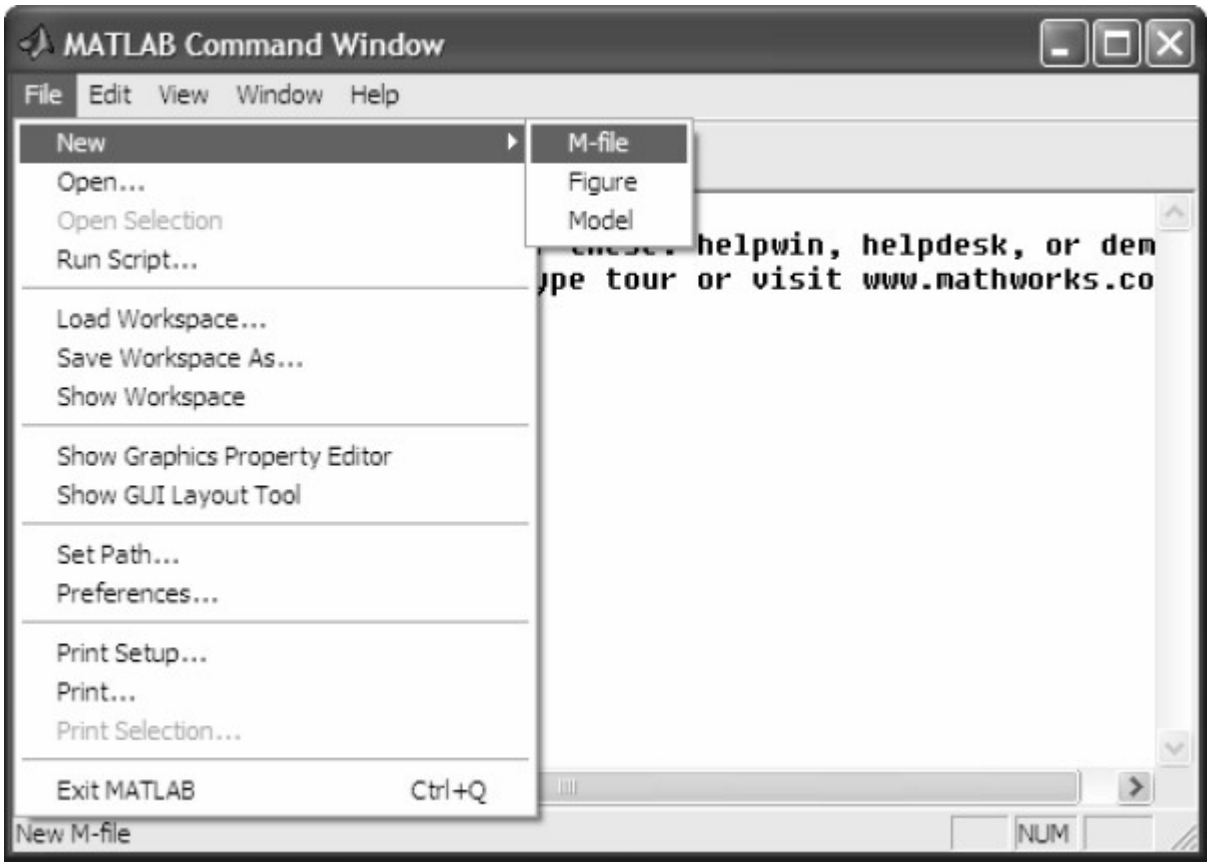
Pierwsze polecenie subplot wybiera pierwszą kolumne tego panelu i wykreśla na niej funkcję sinus. Drugie wybiera drugą kolumnę i wykreśla funkcje kosinus na niej. W ten sposób budujemy wykres.

Kontrola osi

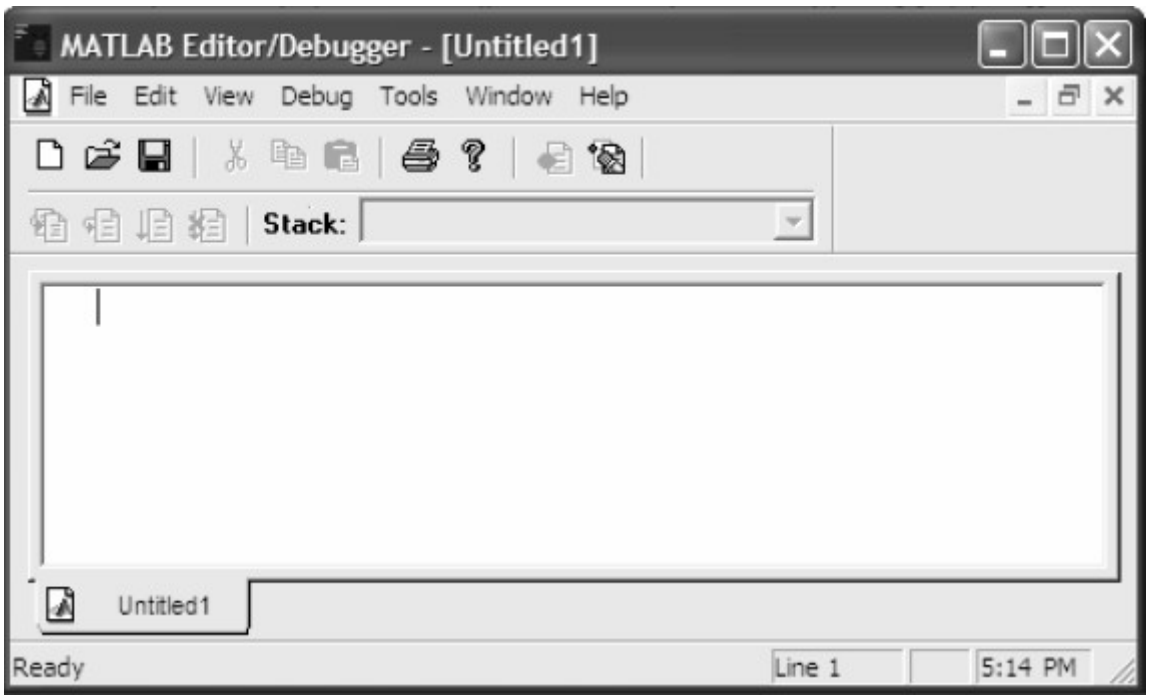
Osie mogą być kontrolowane przez użytkownika za pomocą polecenia help, które akceptuje wektro wierszowy z czterema składowymi. Pierwsze dwa z nich to minimalna i maksymalna granica osi x, a dwie ostatnie to to samo ale dla osi y. Matlab również pozwala użytkownikowi na ustawienie tych osi z opcjami 'equal' ,'auto', 'square' i 'normal'. Na przykład axis('auto') będzie skalowało wykres automatycznie. Podobnie axis9[0 10 0 100]) będzie skalowało wykres z osią x w zakresie [0 10] i zakresem osi y [0 100].

Skrypty

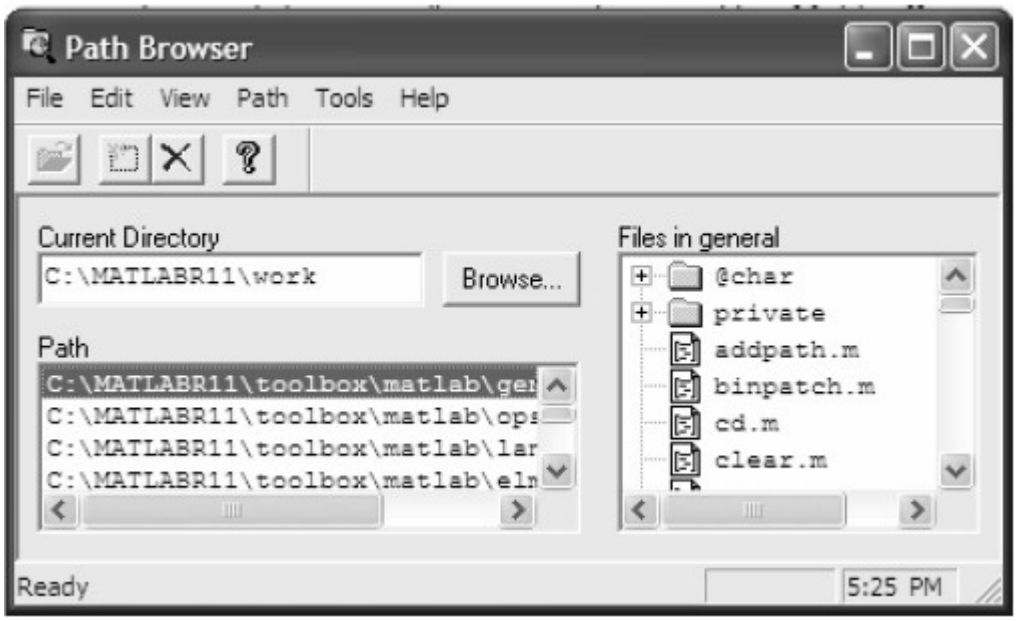
Czasami konieczne jest ponowen podanie zbioru poleceń Matlab. W takich przypadkach ,staje si,e nużącym wpisywanie wszystkieo. Matlab oferuje łatwy sposób na obsługę takiej sytuacji. Użytkownik może zapisać żądany zbiór poleceń w pliku skryptowym Matlab. Może mieć poprawną nazwę i musi mieć rozszerzenie 'm', co oznacza skrypt Matlab. Jest to standardowy plik tekstowy ASCII. Matlab ma wbudowany edtor plików, stworzony specjalnie do tego celu. Może być dostępny z menu File:



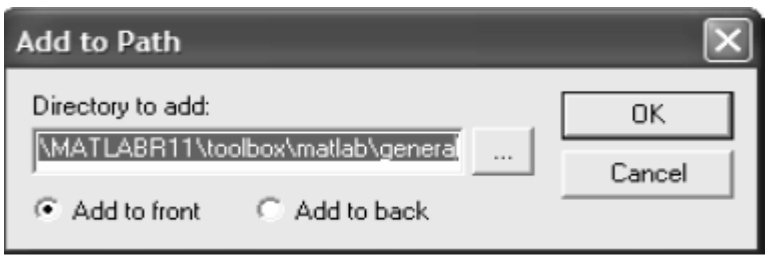
Przez kliknięcie File – New – m file, pojawi się okno edytora m-file:



Tu można wpisywać żądany zbiór poleceń i zapisać go. Domyślny katalog dla tych plików jest w ścieżce dostępu matlab. Jeśli chcesz zapisać plik do swojego własnego katalogu, nie zapomnij uczynić tego w ścieżce dostępu matlab. Może to być przez kliknięcie na pliku – wybranie pozycji menu ścieżki która otworzy ścieżkę przeglądarki:



Możesz użyć pozycji menu ścieżki – Add to path aby dodać katalog jaki wybrałeś do ścieżki Matlab :



Przez kliknięcie przycisku ... , przeglądarka katalogów może być otwarta a przez kliknięcie na żądanym katalogu, możesz wybrać katalog do dodania. Po wszystkim naciśnij OK aby dodać katalog do ścieżki. Po zapisaniu pliku sktypru w katalogu Matlab, polecenia wewnątrz będą wywoływane przez wpisane nazwy pliku (bez rozszerzenia .m)

Praca z wektorami i macierzami

Wektory mogą być manipulowane na różne sposoby. Skalar może być dodaney do elementów wektora w Matlab używając zapisu .+:

```
» A=[1 2 ];

» B=2.+A

B =

   3   4
```

Zauważ użycie 'kropki' przed znakiem '+' co oznacza zastosoanie go w elemencie. W dokładnie w ten sam sposób mogą być przeprowadzone dzielenie, mnożenie, odejmowanie i podnoszenie do potęgi. Na przykład podnieśmy do potęgi każdy element macierzy do potęgi 2 korzystając z notacji 'kropki'

```
>> B=[2 3 4; 5 4 6; 1 3 2]; B.^2
ans =
     4     9    16
    25    16    36
     1     9     4
>> B^2
ans =
    23    30    34
    36    49    56
    19    21    26
```

W pierwszym przypadku, każdy element macierzy b ma zostać podniesiony do potęgi 2. Do tego celu, została użyta notacja kropki. W drugim przypadku, ta sama macierz została podniesiona do potęgi 2 , która jest szczególnym działanie B * B. Teraz przenieśmy kropkę lub iloczyn skalarny wiersza U i kolumny wektora V:

$$U = \begin{bmatrix} 1 & 2 & 3 \end{bmatrix}; \quad V = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

```
>> U=[1 2 3]; V=[1;2;3]; U*V
ans =
    14
```

Oczywiście, wynik 1=4=9 = 14; ilość skalarna. Teraz zmieńmy porządek mnożenia. W tym

przypadku oczekiwanym wynikiem ma być macierz:

```
>> V*U
ans =
     1     2     3
     2     4     6
     3     6     9
```

Teraz obliczymy normę euklidesową wektora , który jest zdefiniowany jako :

$$\|U\| = \sqrt{\sum_{i=1}^{3} |u_i|^2}$$

Może to być uzyskane przez $\sqrt{U.U'}$; gdzie U' jest złożoną konjugacją transpozycji. Również Matlab ma wbudowaną funkcję nazwaną norm, która wykonuje tą operację dla nas:

```
>> sqrt(U*U'), norm(U)
ans =
   3.7417
ans =
   3.7417
```

Pierwsze obliczenie zwrca wartość sqrt(U*U') jako 3.7417 i dokładnie taki sam wynik jest uzyskiwany funkcją norm. Teraz obliczmy kąt między dwoma wektorami X i Y gdzie:

$$X = [7 \quad 5 \quad 9]; \qquad Y = [15 \quad 3 \quad 7]$$

W matlab, obliczymy długość tych wektorów używając funkcji norm, i podzielmy iloczyn skalarny X i Y przez tą długość, wynik będzie cosinusem kąta i w końcu, użyejmy funkcji acos i otrzymamy wynik końcowy. Matematycznie:

$$\theta = \cos^{-1}\left(\frac{X.Y}{\|X\|\|Y\|}\right)$$

```
>> X=[7 5 9]; Y=[15 3 7];
>> theta = acos(X*Y'/(norm(X)*norm(Y)))
theta =
    0.5079
```

Najpierw oba wektory są inicjalizoane. Następnie, stosujem wzór. Ważna rzecz jest taka ,że w tym przypadku był fakt ,że ponieważ oba wektory były zdefiniowane jako wektru wierszowe, musimy skonwertować wektor 'Y' na wektor kolumnowy przez użycie transpzoycji aby obliczyć iloczyn skalarny.

Iloczyn Hadamarda

Chociaż nie jest powszechnie używany, Hadamard jesy określany w matematyce jako iloczyn element przez element dwóch wektorów o identcyznej długości a wnik jest ponownie wektorem o

tej samej długości. Na przykład jeśli:

$$U = [u_1 \quad u_2 \quad \cdots \quad u_n]; \qquad V = [v_1 \quad v_2 \quad \cdots \quad v_n]$$

wtedy iloczyn wektorowy Hadamarda jest zdefiniownay tak :

$$U.V = [u_1 v_1 \quad u_2 v_2 \quad \cdots \quad u_n v_n]$$

W Matlab, iloczyn wektorowy Hadamarda jesy uzyskiwany operatorem **.*** Na przykład ,jeđli U = [1 3 4 7] a V = [8 3 9 2], wtdy ich iloczyn wektorowy Hadamrda jest :

```
>> U=[1 3 4 7]; V=[8 3 9 2]; U.*V
ans =
     8     9    36    14
```

Tabelaryzacja funkcji

Funkcje używane w Matlab stosują podstawę element przez element. W celu przetestowania tego, przygotujmy tabelę wartości sinus i wartości kosinus kąta z zakresu od 0 do pi w korakc pi/10. W tym celu budujemy wektor kolumnowy wartości kątów i nazwiemy fo X:

```
>> X=[0:pi/10:pi]'
X =
         0
    0.3142
    0.6283
    0.9425
    1.2566
    1.5708
    1.8850
    2.1991
    2.5133
    2.8274
    3.1416
```

Teraz użyjemy dwóch funkcji trygonometrycznych z x jako argumentem:

```
>> [X sin(X) cos(X)]
ans =
         0         0    1.0000
    0.3142    0.3090    0.9511
    0.6283    0.5878    0.8090
    0.9425    0.8090    0.5878
    1.2566    0.9511    0.3090
    1.5708    1.0000    0.0000
    1.8850    0.9511   -0.3090
    2.1991    0.8090   -0.5878
    2.5133    0.5878   -0.8090
    2.8274    0.3090   -0.9511
    3.1416    0.0000   -1.0000
```

co wyraźnie pokazuje ,że funkcje w rzeczywistości mają zastosowanie na kązdy element wektora kolumnowego. Pierwsza kolumna w powyższych danych wyyjściowych to x, druga to sin(X) a trzecia cos(X). W celu przetestowania innego rpzypadku, spróbujemy znaleźć wartość graniczną sin(y)/y, przy y dążącym do zera. Odpowiedź powinna być w Matlab 1.0, musimy najpierw zdefiniować zakres wartości y:

```
>> y=[10 1 0.1 0.01 0.001];
```

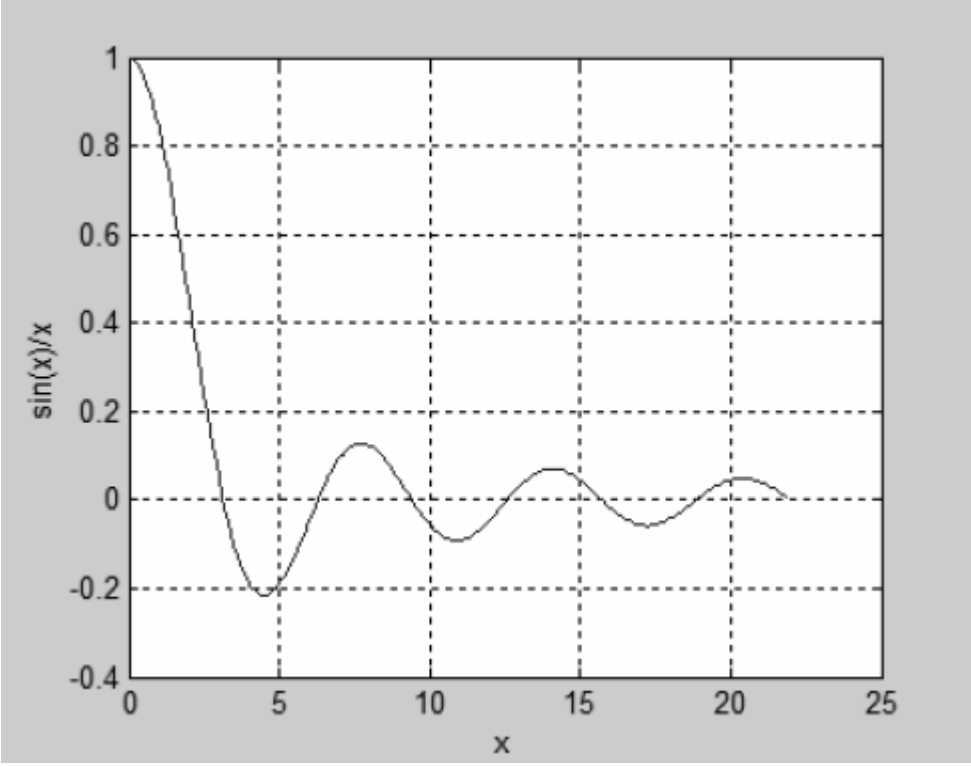
potem stosujemy wyrażenie. Zauważ ,że funkcja sinus będzie stosowała element po elemncie ale dzielenie musi być zmuszone do przeprowadzenia również element po elemencie, które może być łatwo wykonane przez użycie kropkio przed operatorem dzielenia:

```
>> sin(y)./y
ans =
   -0.0544    0.8415    0.9983    1.0000    1.0000
```

i wydaje się jasne ,że wartość graniczna zmierza do 1.0 ponieważ y staje się mniejsze i mniejsze. Aby pokazać wyniki graficznie, najpierw zdefiniujemy zakres wartości x od blizkeigo 0 do 7π w krokach π/10. Potem wykreślamy

```
>> x=[0.0001:0.1:7*pi];plot(x,sin(x)./x);
xlabel('x'); ylabel('sin(x)/x'); grid on
```

co widać tu



Funkcja wyraźnie zbliża się do 1.0 ponieważ 'x' staje się coraz mniejsze. W zakresie wartości 'x', zero musiałoby być unikane , wprzeciwnym razie Matlab dałby komunikat błędu dzielenia przez zero.

Definiowanie Macierzy

Macierz jest w istocie dwuwymiarową tablicą liczb składającą się z wierszy i kolumn. Macierz może być wproadzona w Matlab na trzy sposoby:

a) użycie klawisza powrotu karetki:

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

b) użycie średnika dla wskaznia kolejnej linii:

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

c) użycie notacji zakresu ze średnikami:

```
>> A=[1:3; 4:6; 7:9];
```

Niektóre macierze można zdefiniować po prostu za pomocą funkcji. Na przykład funkcja zer definiuje macierz ze wszystkimi wyrazami będącymi zerami, funkcja jedynek definiują macierz wypełnioną jedynkami a rand definiuje maicez z wyrazami wypełnionymi losowymi liczbami z zakresu [0,1]:

```
>> zeros(3)
ans =
     0     0     0
     0     0     0
     0     0     0
>> ones(3)
ans =
     1     1     1
     1     1     1
     1     1     1
>> rand(3)
ans =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
```

Argument jest w każdym przypadku jest wielkość macierzy. Te same funkcje mogą być również stosowane do określenia maicerzy nie kwadratowej:

```
>> rand(3,4)
ans =
    0.4447    0.9218    0.4057    0.4103
    0.6154    0.7382    0.9355    0.8936
    0.7919    0.1763    0.9169    0.0579
```

W tym przypadku musimy przedstawić dwa argumenty, pierwszy na liczbę wierszy a drugi liczbę kolumn

Rozmiary macierzy

Funkcja size zwraca rozmiar macierzy. Na przykład, zdefiniujmy macierz zerowa A o rozmiarze 135x243:

```
>> A=zeros(135,243); size(A)
ans =
   135   243
```

Funkcja size zwraca liczbę wierszy i komuny tej macierzy

Macierz jednostkowa

Macierz jednostkowa ma postać :

$$I = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & 1 \end{bmatrix}$$

może być zdefiniowana w Matlab z pomocą funkcji 'eye' z argumentem przedstawiającym rozmiar macierzy.

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1
```

Ta funkcja 'eye' generuje macierz jednostkową 3x3 keidy argumentem jest 3

Transpozycja

Jak wspomniano wcześniej, sprzężenie zespolone traspozycji macierzy można uzyskać za pomocą apostrofu. Aby uzyskać regularną transpozycję, można użyć kropka-apostrof. Na przykład:

```
>> T=[1-j 1+j
      1+j 1-j];
>> T'
ans =
   1.0000 + 1.0000i   1.0000 - 1.0000i
   1.0000 - 1.0000i   1.0000 + 1.0000i
>> T.'
ans =
   1.0000 - 1.0000i   1.0000 + 1.0000i
   1.0000 + 1.0000i   1.0000 - 1.0000i
>>
```

Najpierw definiujemy macierz złożoną T. T' jest sprzężeniem zespolonym transpozycji – sprzężenei zespolone każdego elementu zostało wykorzystane po transpozycji macierzy. T.' po prostu transponuje macierz T a sprzężenie zespolone nie zostało podjęte w tej sprawie.

Macierz diagonalna

Macierz diagonalna jest podobna do macierzy jednostkowej w tym ,ze obie mają zerowe elementy przekątne. Ayb wygenrować macierz diagonalną, pierwszy wektor wierszowy zawierający elementy diagonalne jest wymagany. Dla macierzy kwadratowej o rozmiarze n x n, przekątna będzie rozmiaru n. Ten wektor wierszowy jest używany jako argument funkcji 'diag':

```
>> d=[1 2 3 4];
>> D = diag(d)
D =
     1     0     0     0
     0     2     0     0
     0     0     3     0
     0     0     0     4
```

Zwróć uwagę ,że macierz diagonalna ma wszystkie wyrazy diagonalne pobrane z wektora "D". Ta sama funkcja zwraca wpisy przekątnej dla danej macierzy. Na przykład, weźmy macierz losową R:

```
>> R = rand(4)
R =
    0.3529    0.2028    0.1988    0.9318
    0.8132    0.1987    0.0153    0.4660
    0.0099    0.6038    0.7468    0.4186
    0.1389    0.2722    0.4451    0.8462
>> diag(R)
ans =
    0.3529
    0.1987
    0.7468
    0.8462
```

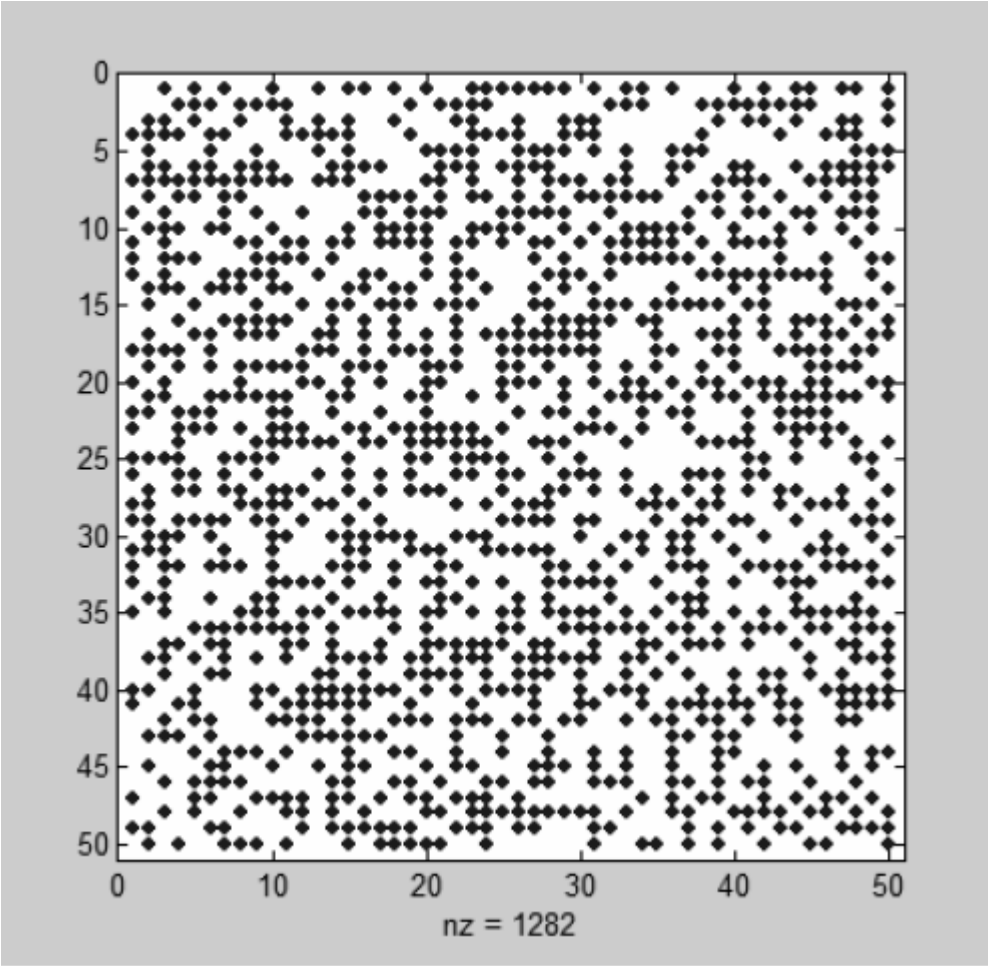
Najpierw jest definiowana macierz losowa R. Potem funkcja 'diag' z argumentem R zwraca elementy diagonalne R

Funkcja spy

Rzadki wzór macierzy pojawia się przy pomocy funkcji spy, która tworzy graficzną wizualizacje danej macierzy. Na przykład, stwórzmy losową macierz i skonwertujmy ją do całkowitych wartości dodając 0.5 do wszystkich elementów. Rozmiar macierzy to 50x50:

```
>> B=fix(0.5+rand(50,50));spy(B)
```

radn(50,50) generuje losową macierz z elementami z zakresu [0,1]. Przez dodanie 0.5 do niej, przesuwamy cały zakres macierzy do [0.5 , 1.5] Oznacza to ,że teraz, około wpisów jest poniżej 1 a połowa powyżej 1. Kiedy stosujemy funkcję fix, macierz jest konwertowana do wszystkich wejść na wartości 0, 1. Oczekuje się ,że połowa wejść będzie 1 a pozsotałe pół zerami. Polecenie spy daje wizualizację tej macierzy:



Wartość 'nz' (number of zero) to 1282, które jest prawie połową 50x50. Większe nz, skromniejsza macierz.

Sekcje macierzy

Rozważmy macierz diagonalną D z wejściami przekątnymi z zakresu 1 do 16. Chcemy wyodrębnić macierz B 4x4 z niej, która ma diagonalne wejścia zaczynające się od 9.

```
>> d=[1:16]; D=diag(d); B = D(9:13, 9:13)

B =

     9     0     0     0     0
     0    10     0     0     0
     0     0    11     0     0
     0     0     0    12     0
     0     0     0     0    13

>>
```

Jak widać, wymagana część macierzy została wyodrębniona

Iloczyn macierzy

Operator * mnoży dwie macierze jeśli są zgdone do mnożenia, podczas gdy operator .* jest ściśle dla mnożenia które jest oparte na element przez element:

```
>> A=[1 2 3; 4 5 6]; B=[1 2; 3 4]; B*A, B.*B
ans =
     9    12    15
    19    26    33
ans =
     1     4
     9    16
```

W powyższym przypadku, po pierwsze definiowane są dwie macierze. Wtedt B*A oblicza stadnardowy skalarny lub wektorowy iloczyn tych dwóch macierzy, podczas gdy działanie .* oblicza iloczyn element przez element macierzy B

PROGRAMOWANIE W MATLAB

Matlab ofruje dość prosty język programowanie, nieco podobny w wielu aspektach do języka C. Ale są pewne różnice między tymi dwoma językami. Zaczniemy od kilku podstawowych rzeczy.

Pętle FOR

W celu reazlizacji powtarzających się zadań, potrzebne są pętle. W Matlab, jest to robione przez użycie polecenia for , którego ogólna składnia :

Licznik FOR = lista poprawnych wartości

-- instrukcje, które mają być powtarzane w tej pętli

end

Jako przykład , zdefiniujemy wektor wierszowy mający 7 losowych wartości:

```
>> R=rand(1,7)
R =
0.3784 0.8600 0.8537 0.5936 0.4966 0.8998    0.8216
```

Następnie chcemy znaleźć sumę wszystkich wyrazów w R. Do tego celu, definiujemy zmienną sum i inicjalizujem jązerem. Potem budujemy pętlę, która wykonuje się dokładnie siedem razy z pomoca licznika mającego zakres wartości od 1 do 7. Wewnątrz tej pętli po prostu dodajemy różne elementy R aby po kolei je sumować

```
>> sum=0;
>> for i=1:7
     sum = sum + R(i);
   end
```

Teraz, możemy znaleźć średnią wartość elementów R dzieląc sumę przez 7:

```
>> avg = sum/7
avg =
    0.7005
>> sum
sum =
    4.9036
```

Odpowiedź to 0.7005 podczas gdy suma to 4.9036

Jako inny przykład użycia pętli wgenerujmy i znajdźy sumę pierwszych 100 liczb całkowitych

$$1 + 2 + 3 + \ldots + 100$$

Metoda jest taka sama. Używamy zmiennej sum do sumowania wewnątrz pętli:

```
>> sum = 0;
   for i=1:100
      sum=sum + i;
   end;
   sum
>>sum =
        5050
```

Wynik to 5050

Teraz użyjemy Matlab dla obliczenia ciągu Fibonnaciego:

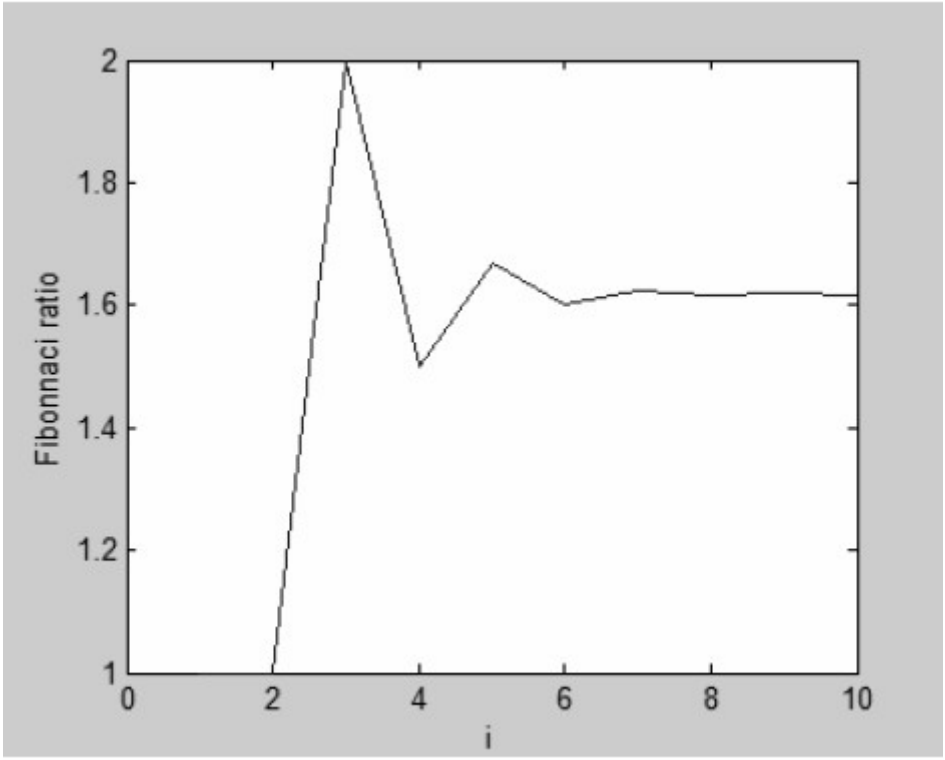
$$1, 1, 2, 3, 5, 8, 13, 21, \ldots$$

który zaczyna się od 1,1; a każdy kolejny element jest sumą dwóch poprzednich. Stosunek dwóch kolejnych elementów ciągu zbliża się do stałej liczby.

```
f(1)=1; f(2)=1;
ratio(1)=1; ratio(2)=1;
for i=3: 10
    f(i)=f(i-1)+f(i-2);
    ratio(i) = f(i)/f(i-1);
end
plot(ratio); xlabel('i');
ylabel('Fibonnaci ratio');
```

Wynikowy wykres pokazuje jasno ,że w dziesięfiu iteracjach, stosunek zbliża się do 1.61786 co

jest blisko wartości rzeczywistej 1.618



Wyrażenia logiczne

Poniższe operatory są używane w wyrażeniach logicznych różnych typów:

Operator	**Znaczenie**
<	Mniejsze niż
<=	Mniejsze lub równe
==	Równe
>	Większe niż
>=	Większe lub równe
~=	Nie równe

Wartość wynikowa to albo prawda albo fałsz. Różne logiczne wyrażenia mogą być połączone dalej za pomocą :

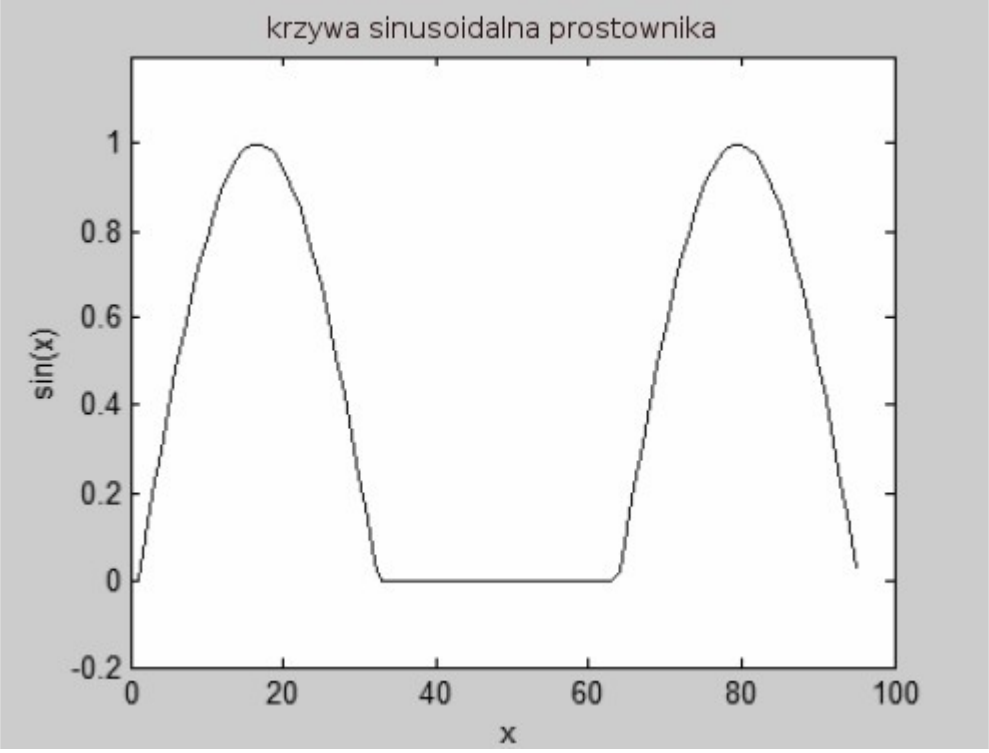
& znaczenie 'and (i)', | znaczenie or (lub) , ~ znaczenie 'not (nie)'

Zwróć uwagę ,że prawda jest oznaczona 1 a fałsz 0

Aby to zastosować, zbudujemy prostownik .Zstworzymy zakres wartości x od 0 do 3π w krokach 0.1. Następnie wyliczymy właściwe wartości funkcji sinus dla tych wartości x jako kąty. W końcu zbudujemy wektor wartości sin(x), który wybira tylko dodatnie wartości sni(x). Odpowiedni plik m-script:

```
x=[0:0.1:3*pi];
y=sin(x);
Z=(y>0).*y;
plot(Z)
xlabel('x');ylabel('sin(x)');
title('krzywa sinusoidalna prostownika');
```

a odpowiedni wykres:



Pętla While

Pętla ta jest powtarzana jeśli pewien warunek pozostaj prawdziwy. Kończy się, kiedy ten warunek jest fałszywy. Na przykład, suma liczb całkowitych od 1 do 100 może być znaleziona za pomocą takiego kodu:

```
Sum = 0;
I = 1
while I <= 100
  Sum = Sum + I;
   I = I+1;
end
```

Zwróć uwagę ,że w tym przypadku licznik pętli będzie zwiększany wewnątrz pętli

Programowanie warunkowe

Do tego celu używamy "struktury – if ". Oto jej forma ogólna

if logiczne wyrażenie -1

    instrukcje wykonywane kiedy wyrażenie -1 jest prawdziwe

elseif logiczne wyrażenie -2

    instrukcje wykonywane kiedy wyrażenie -2 jest prawdziwe

elseif logiczne wyrażenie -3

    instrukcje wykonywane kiedy wyrażenie -3 jest prawdziwe

. . . liczba części elseif powtarzana jeśli to konieczne

else

    instrukcje wykonywane kiedy żadne logiczne wyrażenie nie jest prawdziwe

end

Należy pamietać, że część elseif oraz else są opcjonalne i powinny być stosowane tylko wtedy gdy są potrzebne.Również część else powinna być ostatnią części tej struktury w razie potrzeby. Jako przykład inicjalizujemy zmienną z jakąświartością i przetestujemy czy jest parzysta czy nieparzysta. Będziemy porównywać wynik dzielenia całkowitego przez 2 a potem mnożenia przez 2, a aktualną liczbę .Dal wartości parzystych oryginalną liczbę uzyskujemy:

```
N=input(' Wpisz liczbę całkowitą          ');
if fix(N/2)*2==N
    integer_type= 'parzysta';
else
    integer_type= 'nieparzysta';
end
integer_type
```

Najpierw m-script prosi użytkownika 'Wpisz liczbę całkowitą ', co pojawia się w obszarze roboczym Matlab. Użytkownik wpisuje jakąś liczbę całkowitą, która jest przypisana do zmiennej 'N'. potem, liczba całkowita N jest dzielona przez 2. Zauważ,że jeśli nie jest to liczba parzysta, wtedy będzie część ułamkowa, która jest odrzucana przez funkcję fix. Następnie wynik jest mnożony przez 2 a następnie porównywany z 'N'. Te dwie wartości będą takie same jeśli były liczbami parzystymi, wtedy zmiennej tekstowej przypisywany jest ciąg wartości "parzyste" w przeciwnym razie "nieparzyste". W końcu wartość tej zmiennej tekstowej jest wyświetlana na ekanie. Typowy przykład:

```
Please enter an integer:786
integer_type =
parzyste
>>
```

Funkcja m-Script

W Matlab, wszystkie obliczenia są zwykle wykonwane za pomoca funkcji, które są napisane w języku skryptowym Matlab. Podstawowa struktura funkcji jest następująca:

Funkcja wartości-wyjściowe = nazwa (wartości wejściowe)

%komentarze kiedy używamy 'help nazwa'

- - - ciało funkcji - - -

end

Funckja m-script musi być zapisana w pliku z nazwą jako nazwa funkcji z rozszerzeniem .m

Jako przykład napiszmy skryp funkcji kwadratowej, która przyjmuje trzy wartości jako współczynniki równania kwadratowego a,b i c, oblicza i zwraca wartości x_1 i x_2:

Równanie : $$ax^2 + bx + c = 0$$

Pierwiastki : $$x_1, x_2 = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Zawartość pliku kwadrat.m

function [x1,x2] = quadratic(a,b,c)

%Funkcja kwadratowa

%rozwiązuje równanie kwadratowe:

% a x^2 + b x +c = 0

%używając współczynników a,b,c jako danych wejściowych

%i zwraca wartości dwóch pierwiastków

% jako x1, x2

```
d = b^2-4*a*c;
x1 = (-b+sqrt(d))/(2*a);
x2 = (-b-sqrt(d))/(2*a);
end
```

kiedy wywoływana jest pomoc w skrypcie otzymujemy:

>> help quadratic

Funkcja kwadratowa

rozwiązuje równanie kwadratowe:

 a x^2 + b x +c = 0

używając współczynników a,b,c jako danych wejściowych

i zwraca wartości dwóch pierwiastków

jako x1, x2

a kiedy testujmy ją dla następującego równania:

$$342x^2 + 127x + 8720 = 0$$

otrzymujemy

```
>> [x1,x2]=quadratic(342,127,8720); [x1, x2].'
ans =
  -0.1857 + 5.0460i
  -0.1857 - 5.0460i
```

Wywołujemy funkcję po nazwie i dostarczamy jej wymagane argumenty. Wynik funkcji jest przechowywany w wektorze wierszowym [x1 x2]. Potem ten wektor jest wyświetlany jako kolumna używając zwykłej transpozycji używając kropka-apostrof. Pierwiastki są złożone i koniugacyjne wobec siebie.

Instrukcja return

Zwykle, funkcja "zwraca" wartość , kiedy osiągnie instrukcję 'end'. Jeśli chcesz to zrobić wcześniej, instrukcja return może być użyta ia wymuszać na funkcji zwrócenie wartości w tym miejscu.

Programowanie rekurncyjne

Czasami możliwe jest wywołanie funkcji z wnętrza jej samej aby wykonać niektóre obliczenia. Jako prosty przykład, spróbujmy obliczyć wartość silni liczby całkowitej. Napiszmy skryp do tego celu i nazwijmy go 'factorial'. Wejście do tej funkcji to 'n' liczb całkowitych a wyjście to odpowiednia wartość silnii liczby całkowitej. Dla uproszczenia, załóżmy ,że użytkownik dostarcza tylko liczbę dodatnią jako argument. Teraz można obliczyć silnię w następujący sposób:

Wyrażenie rekurencyjne: Silnia(n) = n * silnia(n-1)

Będzie to powtarzane dopóki argument funkcji nie stanie się 1. Poniżej mamy skrypt Matlab :

```
function value = silnia (n)
%Funkcja silnia
%oblicza silnię liczby calkowitej
%w sposób rekurencyjny
%dane wejściowe to liczba całkowita i
%zwraca wartości silnii jako liczba całkowita
```

```
if n==1
    value = 1;
    return;
end
value = n*factorial(n-1);
end
```

Kiedy wywołujemy pomoc, otrzymujemy:

>> help silnia

Funkcja silnia

oblicza silnię liczby calkowitej

w sposób rekurencyjny

dane wejściowe to liczba całkowita i

zwraca wartości silnii jako liczba całkowita

Zwraca wartość silni szybko :

>> silnia (30

ans =

6

Matlab jest głównie zbiorem funkcji, które mogą być wywoływane w razie potrzeby. Wszystkie te funkcje są po prostu m-skryptami i mogą być modyfikowane w dowolnym życie. Jednak zaleca się aby użytkownik skopiował oryginalną funkcję do osobnego pliku u tam go edytować.modyfikować . W ten sposób praca z Matlab nie będzie zagrożona w przypadku gdy zmodyfikowana wersja nie będzie wykonywała swojego zadania.

WIZUALIZACJA FUNKCJI

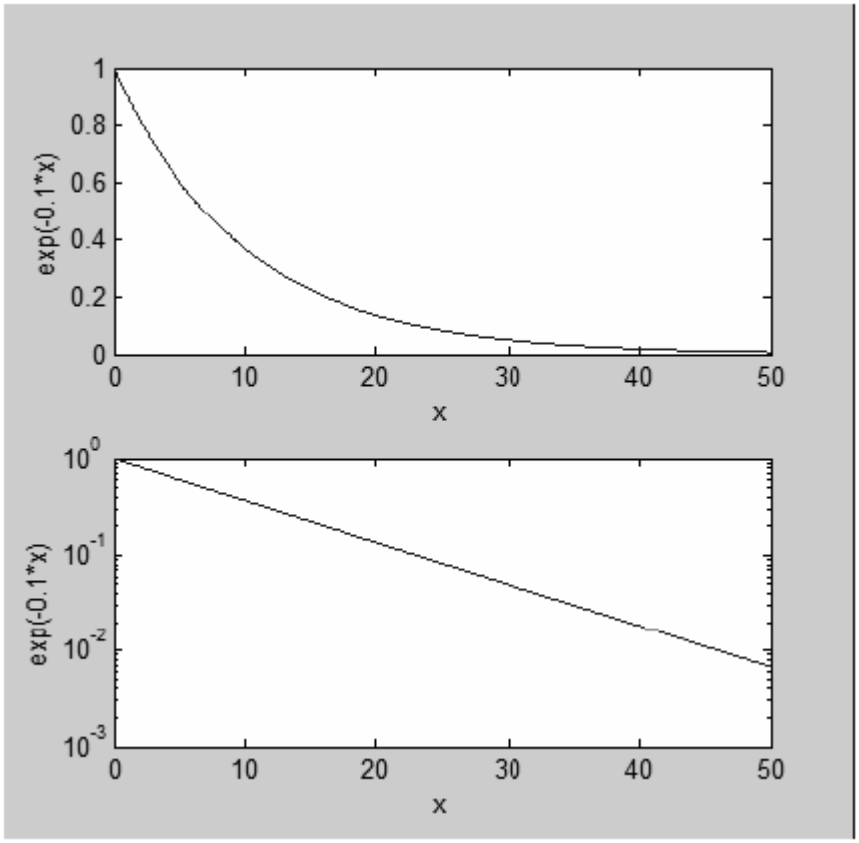
Teraz przyjrzymy się różnym sposobom w jaki można wizualizować różne funkcje. Zacznijmy od funkcji jednej zmiennej. W tym przypadku polecenie plot było już wprowadzone

Wykres liniowo-logarytmiczny

Wykreślmy gwałtowny rozpad izotopu promieniotwórczego. Można to zrobić za pomocą wykresu liniowo-liniowego. Jeśli oś y jest wybrana jako liniowo-logarytmiczna, wykres jest linią prostą.Aby to zrobić, możemy użyć funkcji 'semilogy':

```
x=[0:0.1:50];y=exp(-0.1*x);
subplot(211),plot(x,y);
    xlabel('x');ylabel('exp(-0.1*x)');
subplot(212);semilogy(x,y);
    xlabel('x');ylabel('exp(-0.1*x)');
```

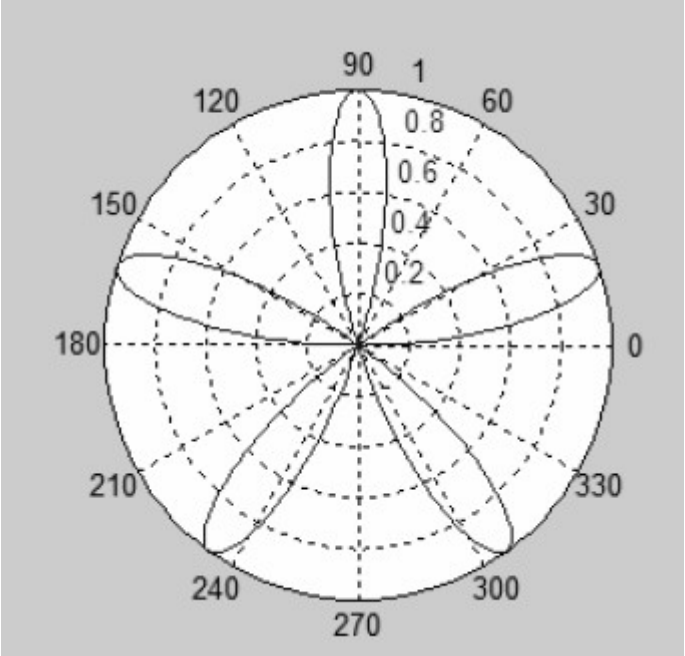
Najpierw geenrujemy równomiernie wypełnioną tablicę liczb od 0 do 50 w krokach 0.. Następnie obliczamy odpowiednie wektory wartości y, który jest odpowiadającą wartością exp(-0.1*x0. Potem, pokazujemy graficzne zastosowanie wykresu liniowo-liniowego a nastęonie używamy funkcji ;seilogy'. Na piwerszym wykreśleniu, wykres krzywej maleje wykładniczo , podczsa gdy w drugim przypadku jest to linia prosta, jak oczekiwano



Polar plot

Ta funkcja akceptuje zakres wartości kąta "theta" i odpowiednie wartości promienia "rho" i pokazuje je na wykresie biegunowym. Na przykład, wykreślmy sin(5θ) używając jej

```
>> theta=[0:0.01:pi]; rho=sin(5*theta);
   polar(theta,rho)
```

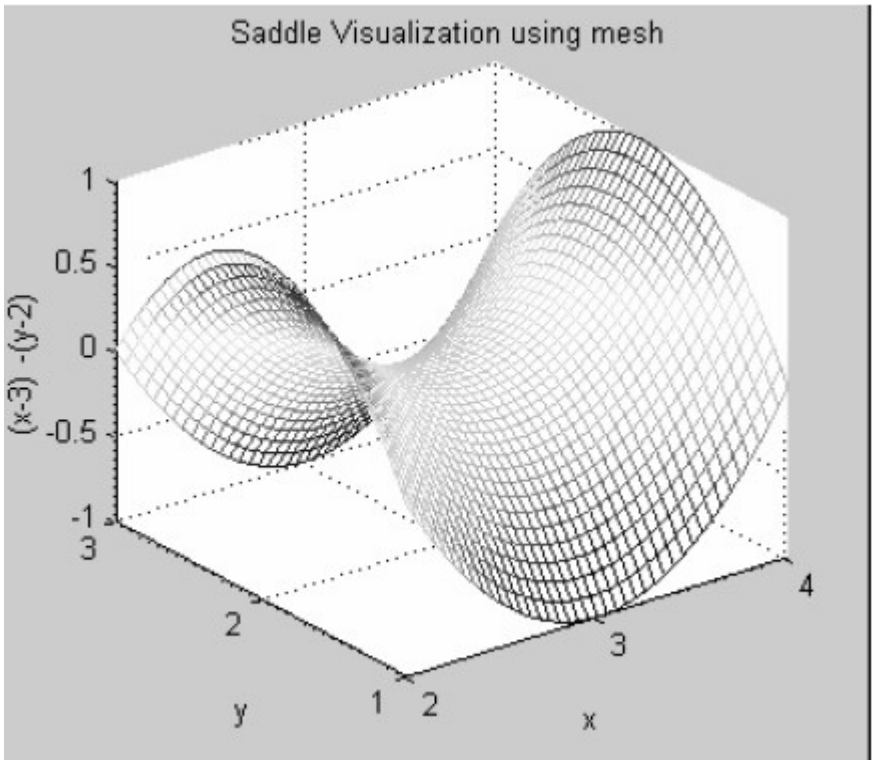


Wykreślanie oczkowe

W tym przypadku wizualizujemy funkcję dwóch zmiennych i jak poprzednio, potrzebujemy wartości wektorowych dwóch niezależnych zmiennych, dla kórych odpowiednie wartość funkcji będą wyliczane. Zwizualizujmy funkcję $Z = (x-3)^2 – (y-2)^2$ w zakresie x ε[2, 4], y ε[1,3]. Odpowiednie polecenia Matlab to:

```
[x,y]=meshgrid(2:0.05:4,1:0.05:3);
z=(x-3).^2-(y-2).^2;
mesh(x,y,z);
xlabel('x'); ylabel('y');
zlabel('(x-3)^2-(y-2)^2');
title('Saddle Visualization using mesh');
```

a odpowiednii wykres



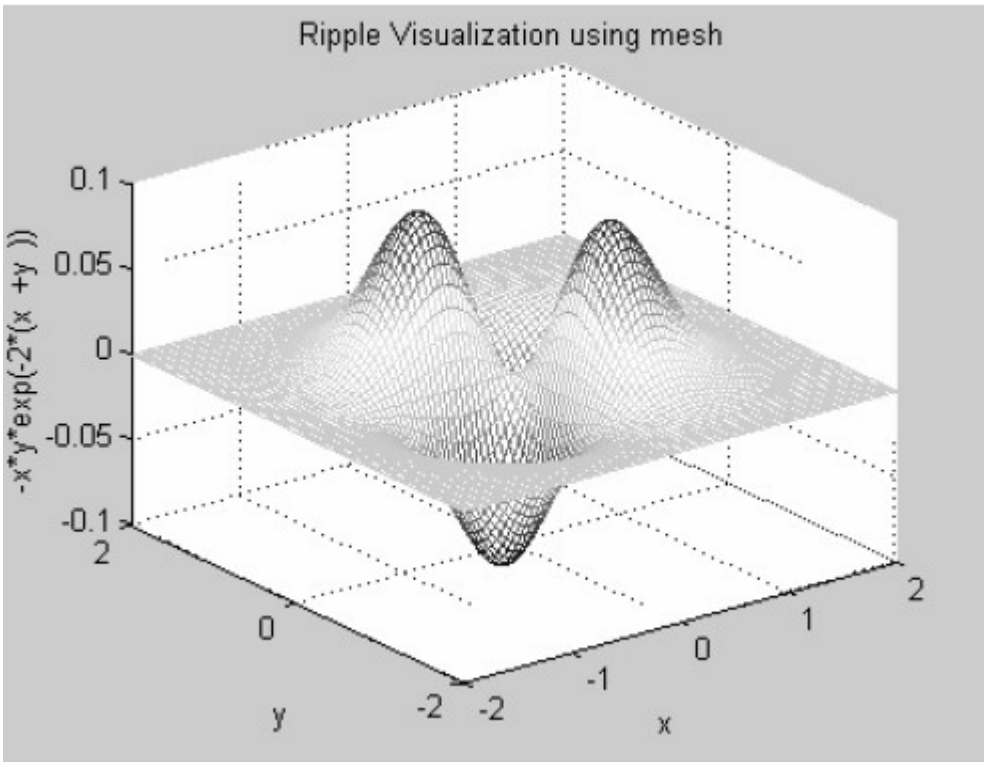
Teraz pokażemy wizualizację funkcji

$$z = -x\,y\exp\left(-2\left(x^2+y^2\right)\right)$$

Do tego celu użyjemy powierzchni i wykresu konturu różnych typów. Odpowiedni m-script:

```
[x,y]=meshgrid(-2:0.05:2,-2:0.05:2);
z=-x.*y.*exp(-2*(x.^2+y.^2));
mesh(x,y,z);
xlabel('x'); ylabel('y');
zlabel('-x*y*exp(-2*(x^2+y^2))');
title('Ripple Visualization using mesh');
```

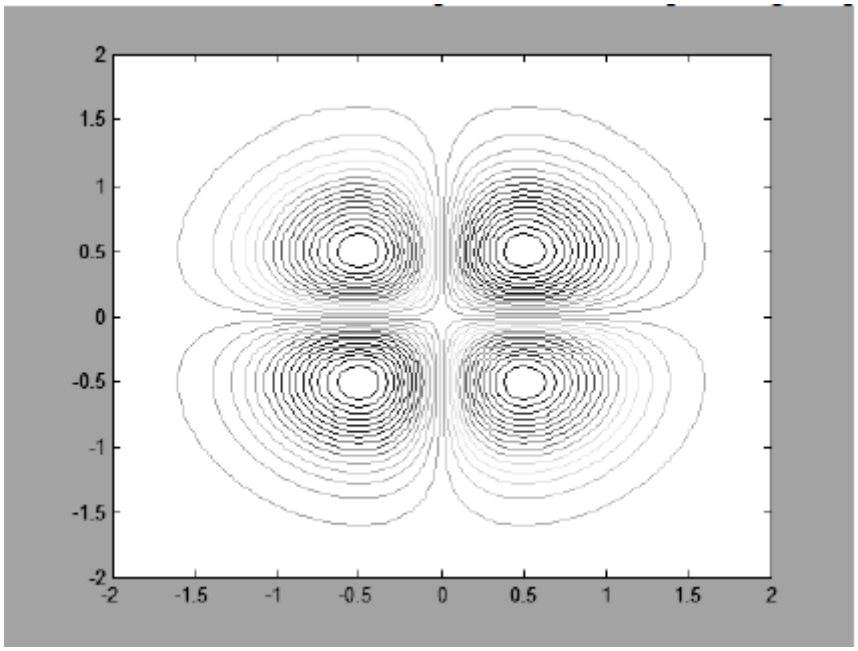
a odpowiedni wykres



Teraz narysujemy kontury dla tej funkcji używając:

Contour (x,y,x,30);

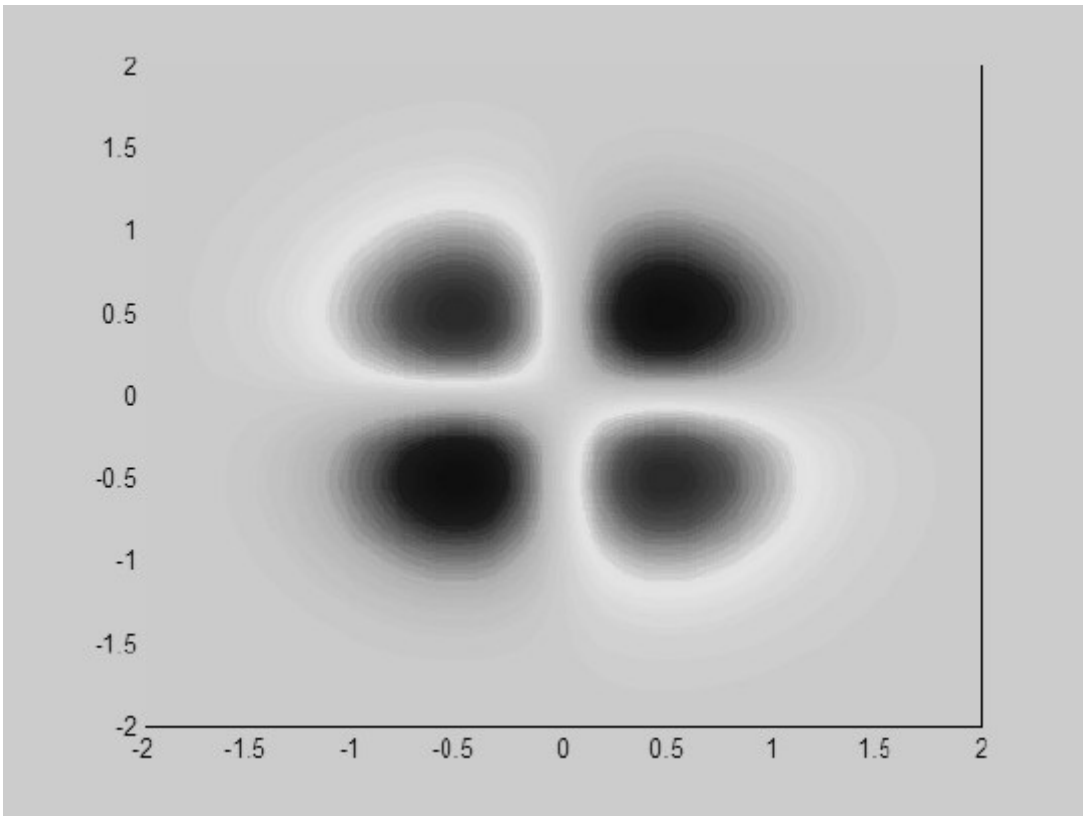
Gdzie 30 jest liczbą konturów na wykresie:



poniższe polecenie daje inną wizualizację:

contorurf(x,y,z,100); shading flat;

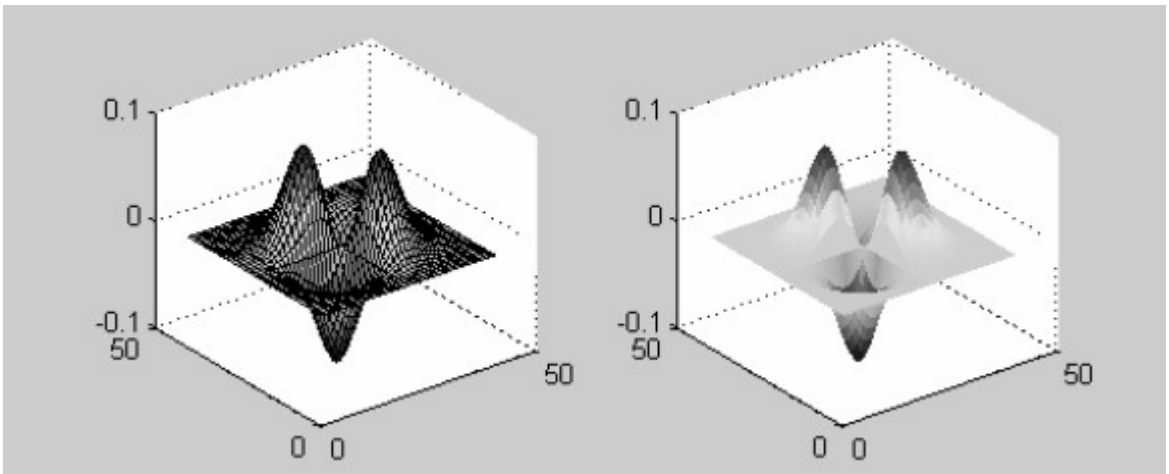
Wynik:



Z poleceniami

```
subplot(121);surf(z);
subplot(122);surf(z);shading flat,colormap('jet');
```

zwizualizujemy parę powierzchni. Lewa powierzchnia ma widoczną siatkę podczas gdy prawa powierzchnia nie.



Upływający czas

W Matlab, różne procedury mogą wymagać różnej ilości czasu obliczeniowego. Ten czas możemy uzyskać parą funkcji 'tic' i 'toc'. Pierwsza funkcja ustawia czas startowy na zero a druga zwraca czas jaki upłynąć od ostatniego uruchomienia funkcji 'tic'. Jako przykład obliczymy czas wymagany dla dodania 2000 liczb całkowitych zaczynając od 1. Odpowiedni kod Matlab został

zapisany w m-skrypcie nazwanym tictoc.m:

```
tic;
sum = 0;
for i=1:2000
    sum = sum+i;
end
sum
toc
```

Kiedy go wykonamy, uzyskujemy:

```
>> tictoc
sum =
     2001000
elapsed_time =
    0.0150
```

Ta wartość jest w sekundach. Jeśli chcesz obliczyć czas CPU, wtedy funkcja to cputime a polecenia :

```
t=cputime;  twoje działania      cputime-t
```