Wykład wstępny (I) *<< zrealizowany w 2006, przedrostki-nazwy zretuszowane w Acrobat >>*

z Podstaw Przetwarzania Informacji

(na danych obrazów 2D w środowisku Matlab 6.x 7.x)

W ramach serii wstępnych wykładów dotyczących prezentacji możliwości dokonywania przekształceń na danych obrazów 2D w *Image Processing Toolbox*, postanowiono przedstawić na początku uwagi dotyczące specyfiki składni poleceń wydawanych w środowisku Matlab.

```
>> a=2;

>> b=3;

>> a/b
```

```
ans =
   0.6667
>> b/a
ans =
   1.5000
```

Specyficznie działające operatory dwuargumentowe: dodawania, odejmowania, mnożenia, potęgowania, a w szczególności dzielenia lewo- i prawostronnego w połączeniu z jednolitym traktowaniem zmiennych skalarnych, wektorowych i skalarnych to możliwość o wiele bardziej zwięzłego ujęcia formuł obliczeniowych w konkretne wyrażenia, niż w przypadku składni pascalo-podobnej, czy osławionej w zastosowaniach inżynierskich, składni języka C.

Nadto, przydział pamięci, tj. alokacja pamięci pod zmienne w środowisku Matlab jest automatyczna, bez względu na typ, czy chwilowy charakter przechowywanych w niej danych. Ze względu na jednolitą w konstruowanych wyrażeniach składnię wyrażeń z udziałem zarówno elementów skalarnych, jak i wektorowo-macierzowych, czasami zachodzi potrzeba odróżnienia operacji po-elementowych w mnożeniu, potęgowaniu, dzieleniu, itp., od działań właściwych algebrze liniowej:

```
>> A=eye(5);%przekątna niezerowych elementów(jedynek)

>> B=ones(5)+eye(5);% macierz jedynek oraz dwójek na przekątnej

>> A/B          %dzielenie zgodne z rachunkiem algebry liniowej
```

```
ans =
   0.8333  -0.1667  -0.1667  -0.1667  -0.1667
  -0.1667   0.8333  -0.1667  -0.1667  -0.1667
  -0.1667  -0.1667   0.8333  -0.1667  -0.1667
  -0.1667  -0.1667  -0.1667   0.8333  -0.1667
  -0.1667  -0.1667  -0.1667  -0.1667   0.8333
```

```
>> inv(B)*A   % operacja powyżej to operacja mnożenia prawostronnego przez odwrotność B
```

```
ans =
   0.8333  -0.1667  -0.1667  -0.1667  -0.1667
  -0.1667   0.8333  -0.1667  -0.1667  -0.1667
  -0.1667  -0.1667   0.8333  -0.1667  -0.1667
  -0.1667  -0.1667  -0.1667   0.8333  -0.1667
  -0.1667  -0.1667  -0.1667  -0.1667   0.8333
```

```
>> A\B        %dzielenie B przez A zgodne z rachunkiem algebry liniowej
```

```
ans =
   2   1   1   1   1
   1   2   1   1   1
   1   1   2   1   1
   1   1   1   2   1
   1   1   1   1   2
```

```
>> inv(A)*B    %to również operacja mnożenia prawostronnego lecz macierzy B przez odwrotność A
```

```
ans =
   2   1   1   1   1
   1   2   1   1   1
   1   1   2   1   1
   1   1   1   2   1
   1   1   1   1   2
```

W powyższych operacjach, skośnik i ukośnik (czytaj: slash i backslash), to zamiana w rolach dzielnej z dzielnikiem w wyrażeniach z macierzami `A` i `B`. Gdyby jednak należałoby przeskalować po-elementowo elementami macierzy A elementy macierzy B, albo chociażby podzielić po-elementowo elementy macierzy `A` przez `B` , to należałoby wykorzystać składnię:

```
>> A.*B %mnożenie po-elementowe siłą rzeczy zostawia elementy niezerowe na przekątnej
```

```
ans =
   2   0   0   0   0
   0   2   0   0   0
   0   0   2   0   0
   0   0   0   2   0
   0   0   0   0   2
```

```
>> A*B %mnożenie z algebry liniowej(bez kropki poprzedzającej), to rachunek macierzowy
```

```
ans =
   2   1   1   1   1
   1   2   1   1   1
   1   1   2   1   1
   1   1   1   2   1
   1   1   1   1   2
```

```
>> A./B %dzielenie po-elementowe dowolnej macierzy A
        %przez niezerową w elementach macierz B jest możliwe!
```

```
ans =
   0.5000        0        0        0        0
        0   0.5000        0        0        0
        0        0   0.5000        0        0
        0        0        0   0.5000        0
        0        0        0        0   0.5000
```

```
>> A.\B % dzielenie w drugą stronę macierzy B przez A nie powiedzie się!
```

```
Warning: Divide by zero.
ans =
     2   Inf   Inf   Inf   Inf
   Inf     2   Inf   Inf   Inf
   Inf   Inf     2   Inf   Inf
   Inf   Inf   Inf     2   Inf
   Inf   Inf   Inf   Inf     2
```

Jak widać z ostatniego okna rezultatów, Matlab działa na operatora wydawanych poleceń bardzo '*zmiękczająco*', mówiąc oględnie. To znaczy '*nie obraża*' się na próby dzielenia przez zero, a w miejscach wystąpień takich prób wstawia umowną stałą-wartość predefiniowaną `Inf` (*infinity*), tudzież w sytuacjach równie nieokreślonych, w nieco innym kontekście wyrażeń czasami `NaN` (*not-a-number*).

Do istotnych zalet składni środowiska Matlab należy możliwość globalnego traktowania obszarów oraz podobszarów danych wielowymiarowych, w tym w szczególności danych jedno- i dwu-wymiarowych. Z użyciem uniwersalnego operatora wyliczania ':', który jest również stosowany jako operator rozwijania szeregów liczbowych oraz określania zasięgu indeksów zmiennych w podobszarze danych. Tak więc, wyrażenia z pozoru skomplikowane zwykle można ująć z składni Matlab'a, w zaledwie kilku linijkach.

```
>> C=B(:)'%wylicz pionowo wszystkie elementy macierzy Bz jednoczesną transpozycją wektora
```

```
C =
 Columns 1 through 21
   2   1   1   1   1   1   2   1   1   1   1   1   2   1   1   1   1   1   2   1   1
 Columns 22 through 25
   1   1   1   2
```

```
>> D=[1:5] % rozwiń szereg arytmetyczny od 1 do 5 z krokiem liczby naturalnej 1
```

```
D =
   1   2   3   4   5
```

```
>> E=[D' D'+5 D'+10 D'+15 D'+20] %stwórz macierz o elementach narastających kolumnami
```

```
E =
   1   6  11  16  21
   2   7  12  17  22
   3   8  13  18  23
   4   9  14  19  24
   5  10  15  20  25
```

```
>> B  % bez średnika kończącego, to zwykłe przywołanie treści macierzy B na ekranie
```

```
B =
   2   1   1   1   1
   1   2   1   1   1
   1   1   2   1   1
   1   1   1   2   1
   1   1   1   1   2
```

```
>> F=B(2:4,2:4) % tutaj skopiowano pod zmienną F wyłącznie wewnętrzny rejon macierzy B
```

```
F =
   2   1   1
   1   2   1
   1   1   2
```

```
>> B(2:4,2:4)=0 % tutaj wyzerowano wewnętrzny rejon macierzy B
```

```
B =
   2   1   1   1   1
   1   0   0   0   1
   1   0   0   0   1
   1   0   0   0   1
   1   1   1   1   2
```

```
>> > B.^2 %obecnie, potęgowanie po-elementowe macierzy B zostawia rejon zer w jej wnętrzu
```

```
ans =
   4   1   1   1   1
   1   0   0   0   1
   1   0   0   0   1
   1   0   0   0   1
   1   1   1   1   4
```

```
>> B^2 %potęgowanie macierzy B według rachunku macierzowego
```

```
ans =
   8   3   3   3   7
   3   2   2   2   3
   3   2   2   2   3
   3   2   2   2   3
   7   3   3   3   8
```

W pewnych praktycznie stawianych zadaniach inżynierskich, bardziej przydatna może się okazać składania polecenia `linspace` oraz `logspace`, w rozwijaniu liniowo równomiernego oraz logarytmicznie równomiernego szeregu liczb rzeczywistych:

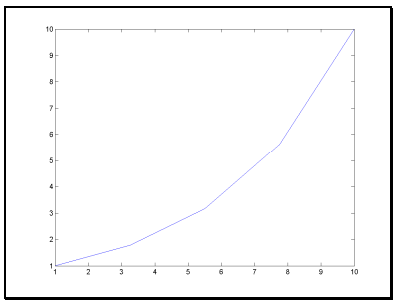
```
>> V1=linspace(1,10,5) % w przedziale od 1 do 10 włącznie rozwiń 5-elementowy szereg liczb
```

```
V1 =
   1.0000   3.2500   5.5000   7.7500  10.0000
```

```
>> V2=logspace(0,1,5) % w tym samym przedziale rozwiń logarytmicznie szereg przy podstawie 10
```

```
V2 =
   1.0000   1.7783   3.1623   5.6234  10.0000
```

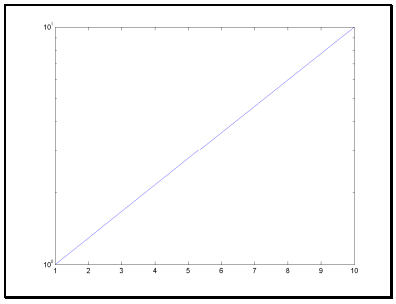
```
>> plot(V1,V2); % wyświetl przeciwdziedzinę V2 w funkcji dziedziny V1
```



Rys.1 Rezultat wywołania funkcji `plot` w zadaniu wyświetlenia przeciwdziedziny szeregu liczb rozwijanych logarytmicznie w oparciu o równomierny szereg liczb z dziedziny.

Spośród rodziny funkcji `plot` można jeszcze skorzystać w wizualizacji danych z `semilogx` albo `semilogy` albo też z funkcji `loglog`

```
>> semilogy(V1,V2);
```



Rys.2 Rezultat wywołania funkcji `semilogy` w zadaniu wyświetlenia przeciwdziedziny szeregu liczb rozwijanych logarytmicznie w oparciu o równomierny szereg liczb z dziedziny.

Aby nie trywializować, tegoż wykładu, który jest jedynie pobieżnym odświeżeniem ćwiczeń laboratoryjnych, eksperymentowanie z rodziną funkcji typu `plot`, albo `loglog`, pozostawiam studentom.

Do potężnego narzędzia w konstruowaniu wyrażeń operujących na macierzach i wektorach zmiennych należy zaliczyć operator sklejania `[]` (albo też z łaciny: *konkatenacji*). Jego przeciwieństwo funkcjonalne w stosunku do operatora indeksacji komórek w Matlabie tj. '`( )`', budzi jawny i żywy sprzeciw , w pierwszej chwili, zapaleńców języka C. Jednakże po chwili namysłu okazuje się, że symboliczny podział funkcji pomiędzy operatory `( )` oraz `[ ]`, był dobrze przemyślany i zaplanowany, jeśli składnię poleceń środowiska Matlab, traktować, jako stopień przemian i ewolucji w stosunku do standardu języka C.

```
>> G=[A(1:10); B(6:15); E(11:20)]
```

```
G =
   1   0   0   0   0   0   1   0   0   0
   1   0   0   0   1   1   0   0   0   1
  11  12  13  14  15  16  17  18  19  20
```

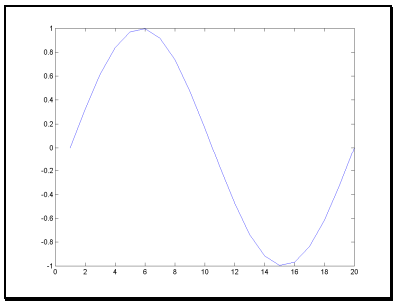
```
>> G=[A(1:10)' B(6:15)' E(11:20)']
```

```
G =
   1   1  11
   0   0  12
   0   0  13
   0   0  14
   0   1  15
   0   1  16
   1   0  17
   0   0  18
   0   0  19
   0   1  20
```

Obecnie ważniejszym i pożyteczniejszym motywem przewodnim tegoż wykładu wstępnego może okazać się prezentacja możliwości środowiska Matlab w wizualizacji danych 2D/3D z użyciem funkcji `surf`, `surfl`, `mesh`.

Wprawdzie przebieg sinusoidalny może być zaprezentowany w formie przebiegu 1D:

```
>> Y=sin(linspace(0,2*pi,20));
>> plot(Y);
```



Rys.3 Jednowymiarowy przebieg sinusoidalny.

Jednakże, dogodniej będzie, na użytek tego wykładu zaprezentować, statyczną falę sinusoidalną rozchodzącą się radialnie od jednego wybranego punktu na płaszczyźnie:

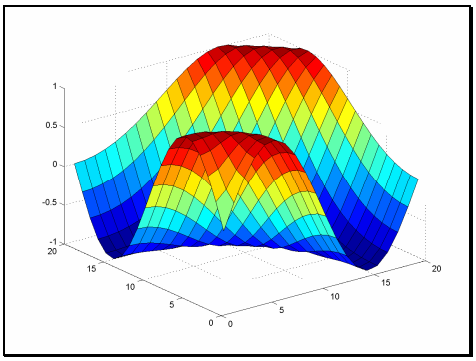
```
>> [X,Y]=meshgrid(linspace(0,2*pi,5),linspace(0,2*pi,5))% rzadka siatka współrzędnych
                                                        %5x5 punktów od 0 do 2pi
```

```
X =
    0  1.5708  3.1416  4.7124  6.2832
    0  1.5708  3.1416  4.7124  6.2832
    0  1.5708  3.1416  4.7124  6.2832
    0  1.5708  3.1416  4.7124  6.2832
    0  1.5708  3.1416  4.7124  6.2832
Y =
    0       0       0       0       0
 1.5708  1.5708  1.5708  1.5708  1.5708
 3.1416  3.1416  3.1416  3.1416  3.1416
 4.7124  4.7124  4.7124  4.7124  4.7124
```

Na początek, wywołanie funkcji `meshgrid` jest o tyle specyficzne, że zawsze pozwala ono na przygotowanie pola 2D współrzędnych zmiennej kontrolowanej(tj. *dziedziny*)

bezpośrednio pod wykorzystanie do wyświetlania danych wysokościowych Z=f(X,Y) (tj. *przeciwdziedziny*) z użyciem funkcji `surfl`, `surfl`, `mesh`. Również wykorzystanie wywołania tej funkcji pod własne '*algorytmiczne fikołki salta i koziołki*' może być bardzo pouczające. Dla początkującego operatora poleceń środowiska Matlab ważna jest informacja, że w wyniku jej wywołania powstają dwie macierze powiedzmy `X` i `Y` współrzędnych. Pierwsza macierz, to zasadniczo treść pierwszego wiersza wartości rozwiniętego szeregu liczb, powtarzanego konsekwentnie z góry na dół, w całej zawartości macierzy X. Druga macierz składa się z pionowego wektora rozwiniętego szeregu liczb, powtarzanego dla odmiany kolumnami. Na użytek dalszych rozważań macierze X i Y będą miały siatkę współrzędnych zagęszczoną do 20 punktów:

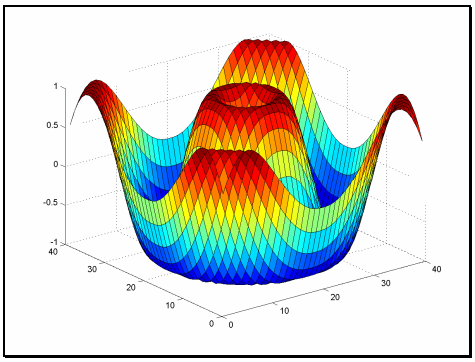
```
>> [X,Y]=meshgrid(linspace(0,2*pi,20),linspace(0,2*pi,20));
>> Z=sin(sqrt(X.^2+Y.^2));
>> surf(Z);
```



Rys.4 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie w pierwszej ćwiartce układu kartezjańskiego.

W środowisku Matlab, praktycznym zabiegiem może się okazać (w większości przypadków) skorzystanie z wywołań funkcji `flipud` lub/oraz `fliplr`, które to realizują odbicie pionowe i poziome, odpowiednio, według zasad symetrii liniowej dowolnych danych reprezentowanych w formie macierzowej:

```
>> Znew=[fliplr(Z) Z];
>> Zfinal=[flipud(Znew); Znew];
>> surf(Zfinal);
```

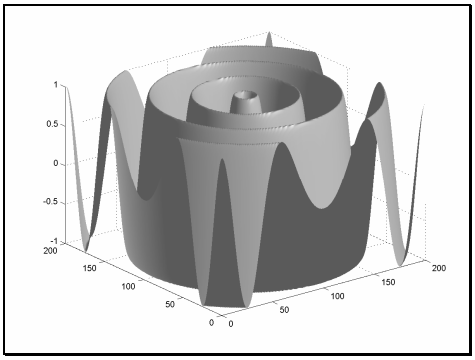


Rys.5 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie w I, II, III, IV ćwiartce układu kartezjańskiego.

Domyślna paleta kolorów, którą to środowisko Matlab 'domyślnie' obdziela każdy z wykresów 3D (*w tonacji zimnych kolorów dla punktów niżej położonych oraz cieplejszych kolorów dla punktów 3D wyżej położonych*), może być w każdej chwili podmieniona własną, zdefiniowaną indywidualnie paletą kolorów. Najlepiej na początek, aby to była paleta skali

szarości, w formie trzech kolumn liczb rozciągających się w zakresie od 0 do 1 włącznie, dla kanałów R,G i B, w podstawowej 24-bitowej reprezentacji danych kolorowych obrazów 2D.

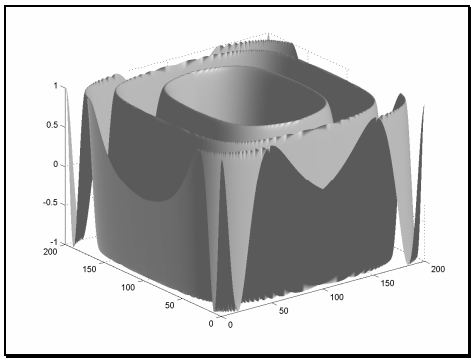
```
>> gr=linspace(0,1,190);
>> grs=[gr' gr' gr'];
>> [X,Y]=meshgrid(linspace(0,6*pi,100),linspace(0,6*pi,100));
>> Z=sin(sqrt(X.^2+Y.^2));
>> Znew=[fliplr(Z) Z];
>> Zfinal=[flipud(Znew); Znew];
>> surfl(Zfinal);shading interp;colormap(grs);
```



Rys.6 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie w I, II, III, IV ćwiartce układu kartezjańskiego w 190-stopniowej skali szarości Uwaga posłużono się wywołaniem funkcji `surfl` , która to wykorzystuje w wizualizacji mechanizm wirtualnego oświetlenia.

Na powyższym wykresie przedstawiono wykres fali radialnej, sinusoidalnej w zakresie dziedziny `X` i `Y` rozciągającej się od `-6pi` do `6pi` z rozdzielczością w zakresie tej dziedziny 200 punktów na kierunku `X`, `Y`. Z uwagi na konieczność dogłębnego zrozumienia składni wywołania polecenia `meshgrid`, dla przykładu, wprowadzono również do tego wykładu przykład siatki współrzędnych 2D w formie macierzy: `X`, `Y`, zbudowanych w oparciu o szereg logarytmicznie rozwijanych liczb od około `pi` do `6pi`:

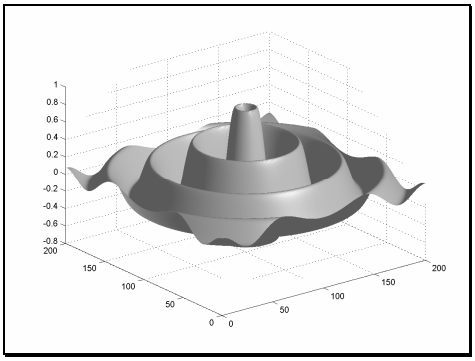
```
>> gr=linspace(0,1,190);
>> grs=[gr' gr' gr'];
>> [X,Y]=meshgrid(logspace(0.5,log10(6*pi),100),logspace(0,log10(6*pi),100));
>>  Z=sin(sqrt(X.^2+Y.^2));
>>Znew=[fliplr(Z) Z];
>>Zfinal=[flipud(Znew); Znew];
>>surfl(Zfinal);shading interp;colormap(grs);
```



Rys.7 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie w I, II, III, IV ćwiartce układu kartezjańskiego Uwaga: logarytmiczna siatka współ. 2D w zakresie od `-6pi` do `-pi` oraz od `pi` do `6pi`.

Wracając do liniowej skali współrzędnych 2D, dla fali radialnie propagowanej z radialnie malejącą amplitudą, ze współczynnikiem ekspotencjalnego wygaszenia ustawionym powiedzmy na 0.1, postać kodu i wyników może być następująca:

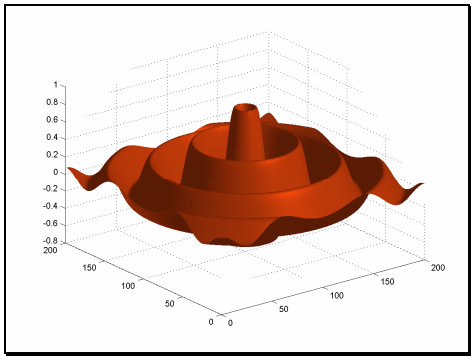
```
>> Z=exp(-0.1*sqrt(X.^2+Y.^2)).*sin(sqrt(X.^2+Y.^2));
>> Znew=[fliplr(Z) Z];
>> Zfinal=[flipud(Znew); Znew];
>> surfl(Zfinal);shading interp;colormap(grs)
```



Rys.8 Dwuwymiarowy przebieg sinusoidalny fali gasnącej radialnie w amplitudzie pobudzenia.

Czasami, w wizualizacji danych wymagane są zmiany własności odbiciowych powierzchni z zasadniczo matowego charakteru na mieszany charakter matowo-poliskliwy. Nadto, uwzględnienie innej palety kolorów, niż tej dotychczasowej, reprezentującej skalę szarości może po uwzględnieniu domieszki pewnej dozy kolorytu np. miedzi metalicznej poprawić czytelność danych na wykresie 3D:

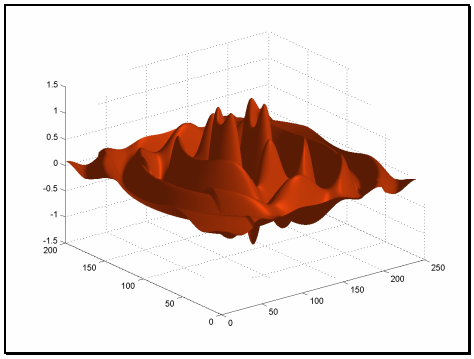
```
>> grR=linspace(0,1,190);
>> grG=linspace(0,0.3,190);
>> grB=linspace(0,0.05,190);
>> grs=[grR' grG' grB'];
>> material shiny
>> surfl(Zfinal);shading interp;colormap(grs);
```



Rys.9 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie propagowanej i gasnącej radialnie w amplitudzie pobudzenia z miedzio-podobnym kolorytem skali barw głębokości.

W następujących dalej odcinkach spisywanego wykładu, może się pojawić opis próby modelowania zjawisk interferencji fal z dwóch źródeł fal sinusoidalnych. Poprzez umiejętne zadeklarowanie pewnej zmiennej `Per`, danych peryferyjnych wykresu 3D można zsumować drgania z dwóch identycznych we własnościach źródeł drgań sinusoidalnych rozstawionych względem siebie, powiedzmy o 40 jednostek na wykresie 3D:

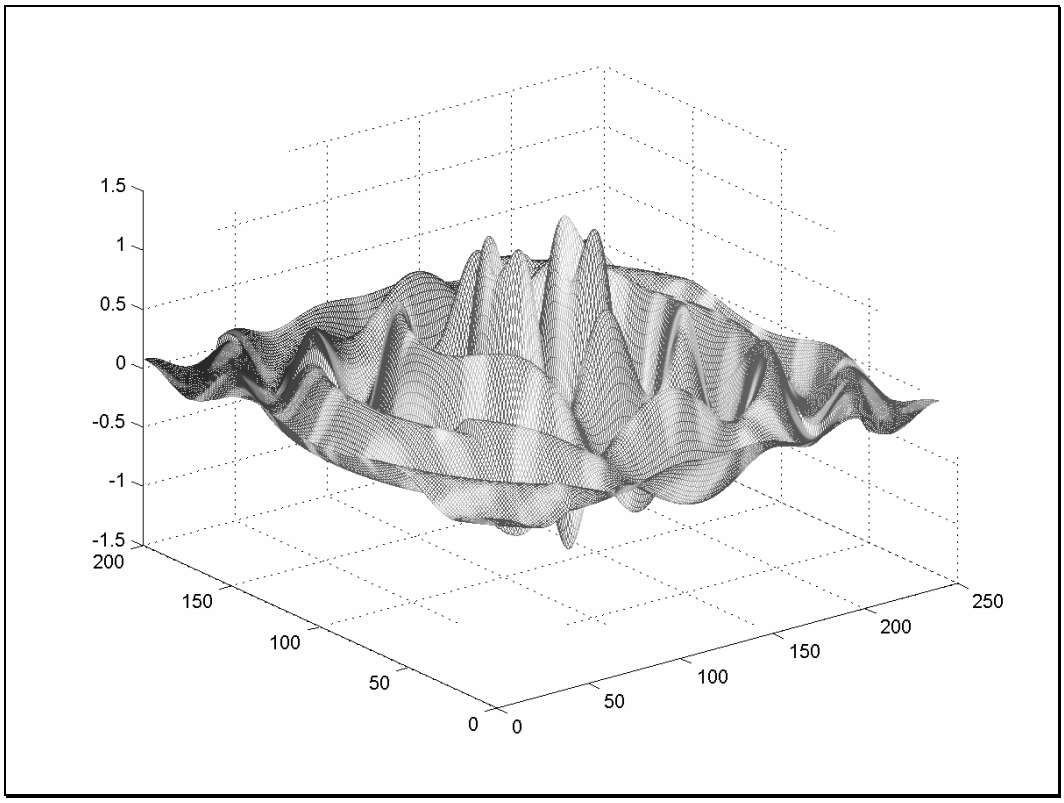
```
>> Per=zeros(200,40);
>> surfl([Per+Zfinal]+[Zfinal+Per]);shading interp;colormap(grs);
```



Rys.10 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie z dwóch źródeł.

Ostatecznie, do opcji wstępnie omawianych w tej części wykładu, należy dodać możliwość *nadteksturowywania* powierzchni wykresu 3D bitmapą, która nie wnosi żadnych informacji związanych z bieżącą, chwilową głębokością/wysokością na wykresie 3D, a jedynie dodaje do informacji wykresu 3D, swoją własną indywidualną informację przestrzenno-kolorystyczną.

```
>> im=imread('rice.tif');%Matlab 6v1
>> imr=imresize(im,[200 240]);%skalowanie do rozmiaru wykresu 3D fal interferujących
>> imRGB(:,:,1)=imr; %zamiana skali szarości na RGB
>> imRGB(:,:,2)=imr;
>> imRGB(:,:,3)=imr;
>> [imind,impal]=rgb2ind(imRGB,190); %zamiana obrazu ziaren ryżu na obraz zindeksowany
>> ZZfinal=[Per Zfinal]+[Zfinal Per]; %zmienna dwóch fal interferujacych
>> mesh(ZZfinal,double(imind));shading interp;colormap(impal);%obraz 2D jako tekstura wykr.3D
```



Rys10 Dwuwymiarowy przebieg sinusoidalny fali rozchodzącej się radialnie z dwóch źródeł, z nałożoną treścią obrazu `rice.tif` jako teksturą powierzchni 3D..

W tym przypadku, tj. nakładania treści pewnego obrazu 2D na powierzchnię 3D danych wykresu prezentowanego, należy najpierw dostosować rozmiar obrazu 2D do rozmiaru danych 3D, zindeksować dane czy to szarego, czy też lepiej kolorowego obrazu 2D, a następnie użyć wywołania funkcji `mesh`, jako że funkcje `surf` oraz `surfl` nie spełnią omawianego zadania, *nadteksturowywania* danych 3D danymi obrazu 2D.

Jeżeli jednak, w szerszym zakresie ma być wykorzystywana funkcja `surfl`, ze względu na jej dodatkowe walory związane z możliwością wirtualnego oświetlenia wykresu 3D, to należy jeszcze zapoznać się ze składnią wywołania funkcji `light`, która tworzy (czy też inicjuje) nowe wirtualne światło do naświetleń powierzchni danych 3D, oraz ze składnią wywołań funkcji `lightangle`, która umożliwia zmianę kąta elewacji i azymutu wiązki światła rzutowanego ze źródła o zadanym uchwycie. W końcu, ostatecznie funkcja `view`, w różnorakich składniach ze skrótową i rozwiniętą listą argumentów umożliwia zmianę punktu obserwatora, według zadanego kąta elewacji i kąta azymutu.

Ze zmianą kąta azymutu punktu obserwatora co 30 stopni, 12-krotnie, w nieskomplikowany sposób można pozyskać animację wykresu 3D w jego obrocie o 360 stopni:

```
M=moviein(12);      %rezerwacja pamięci w przestrzeni roboczej Matlab'a pod 12 klatek animacji
for i=1:12,
mesh(ZZfinal,double(imind));shading interp;colormap(impal);
M(:,i)=getframe;                %łap klatkę bieżącą filmu i wstawiaj ją do filmu M
view(1+i*30,45);                %zmiana cykliczna kąta azymutalnego punktu obserwatora
end;
movie(M);                       %odtworzenie filmu
```

Powyżej przedstawione sposoby tworzenia, rozwijania, manipulacji danymi w środowisku Matlab, jak również prezentacji danych 2D/3D, oczywiście nie wyczerpują większego zasobu możliwości tego pakietu. Nie mniej zadaniem nadrzędnym pierwszej części wykładu z *Image Processing Toolbox* miało być zachęcenie szerokiej rzeszy studentów do własnych wysiłków w eksperymentach na treści danych 2D/3D.