Ćwiczenie 31_0

Mapy funkcji 2D

1. Cel ćwiczenia

Zajęcia dotyczą zapoznaniu się z narzędziami umożliwiającymi wykonanie wykresów funkcji dwóch zmiennych. Narzędzia te umożliwiają wykonanie wykresów konturowych na bazie izolinii, map zmienności wartości funkcji w przestrzeni 2D poprzez mapę kolorów oraz wykresów funkcji wektorowych.

2. Mapy kolorów

Mapy kolorów można wykorzystać w celu prezentacji wartości funkcji 2D. Dla każdego punktu (x, y) przestrzeni 2D, dla której funkcja $f(x, y)$ istnieje, przypisany jest kolor, proporcjonalny do wartości funkcji w tym punkcie. Mapę kolorów dopiera się tak, aby jak najlepiej oddać kontrast między punktami o najmniejszej i największej wartości funkcji $f(x, y)$. Przykładowa mapa może obejmować gradientowe przejście między kolorem niebieskim a czerwonym, gdzie niebieski kolor zwyczajowo oznacza małą wartość funkcji, a kolor czerwony dużą.

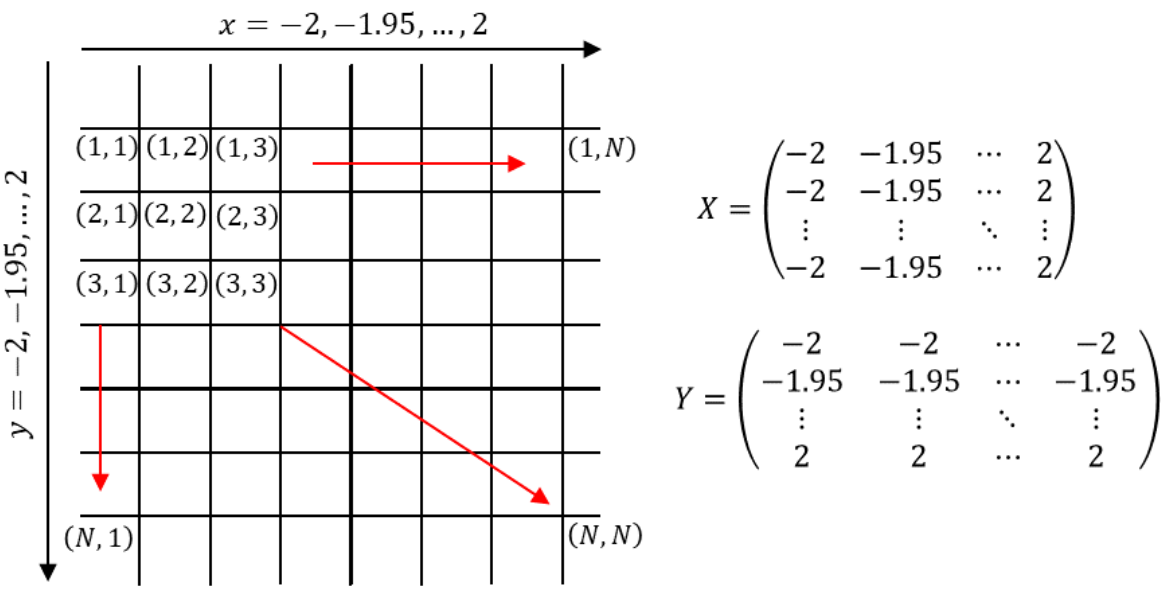
W metodach komputerowych kolor reprezentowany jest często w modelu przestrzeni RGB. Skrót ten pochodzi od pierwszych liter nazw kolorów w języku angielskim, tj. czerwony (ang. red), zielony (ang. green) oraz niebieski (ang. blue). Dowolny kolor pochodny można wyrazić jako kombinację tych trzech kolorów podstawowych. Ilość danego koloru w kombinacji oznaczony jest liczbą z zakresu [0, 1]. W tym zapisie kolor czerwony wyrażony jest przez trójkę liczb: (1, 0, 0), co oznacza, że trzy kolory należy wziąć w ilości czerwony – 1, zielony – 0, niebieski – 0. W zaprezentowanym zapisie wektorowym pierwszy element wektora to ilość koloru czerwonego, drugi to ilość koloru zielonego, a ostatni element to ilość koloru niebieskiego. Kolor biały jest reprezentowany przez wektor (1, 1, 1), a czarny przez (0, 0, 0).

2.1 Dane

W celu pokazania narzędzi do wykonywania wykresów użyjemy przykładowych danych. Rozważmy siatkę kwadratową punktów(x, y) w przedziale $x, y \in [-2, 2]$, gdzie odległość między współrzędnymi wynosi 0.05. Funkcja skalarna niech będzie funkcją

$$f(x, y) = \sin(x^2 + y^2). \quad (1)$$

Funkcje programu Matlab umożliwiające wykonanie wykresów map kolorów wymagają, by dane były wyrażone w postaci macierzy. Konwencja do zapisu macierzowego jest następująca. Każdy punkt dyskretnej siatki obliczeniowej ma współrzędne (i, j), gdzie $i, j = 1, 2, \ldots N$, gdzie N to liczb punktów w danym wymiarze siatki. Macierz X niech będzie reprezentacją macierzową wektora współrzędnych x, a macierz Y niech będzie reprezentacją wektora współrzędnych y. Element $X(i, j)$ będzie równy współrzędnej x punktu o współrzędnych (i, j). Oznacza to, że kolumny macierzy X mają tę samą wartość. Wizualizacja konwencji zaprezentowana jest na Rysunku 1.



Rysunek 1. Konwencja zapisu macierzowego siatki punktów w programie Matlab.

Dane wygenerujemy następującymi komendami. Stwórzmy nowy skrypt programu Matlab o nazwie `plot_2D.m`. Listing kodu prezentuje Listing 1. Dwie pierwsze komendy clear oraz clc dodajemy zwyczajowo do każdego nowego skryptu. Komendy te pozwalają odpowiednio usunąć wszystkie zmienne z pamięci programu Matlab będące tam z poprzedniego wykonania skryptu oraz wyczyścić konsolę programu. Następne trzy wiersze skryptu definiują krok siatki obliczeniowej w kierunku x oraz definiują granice zmienności zmiennej. Kolejne trzy linijki definiują zakres zmienności w kierunku y. Instrukcja `x=x_min:step_x:x_max;` generuje wektor współrzędnych x w przedziale `[x_min, x_max]` z krokiem `step_x`. Analogiczna komenda występuje dla kierunku y. Funkcja `meshgrid` pozwala na wygenerowanie współrzędnych siatki obliczeniowej w postaci macierzy. Zmienne przypisujemy do nowych zmiennych zapisanych wielką literą, dla odróżnienia ich od ich wektorowych odpowiedników. Na koniec obliczamy wartość funkcji skalarnej, którą będziemy rysować. Należy pamiętać, by każdą operację mnożenia, dzielenia czy w tym wypadku potęgowania wykonać na każdym elemencie macierzy. W tym celu przed operatorem należy postawić znak '.'. Inaczej program wyrzuci błąd.

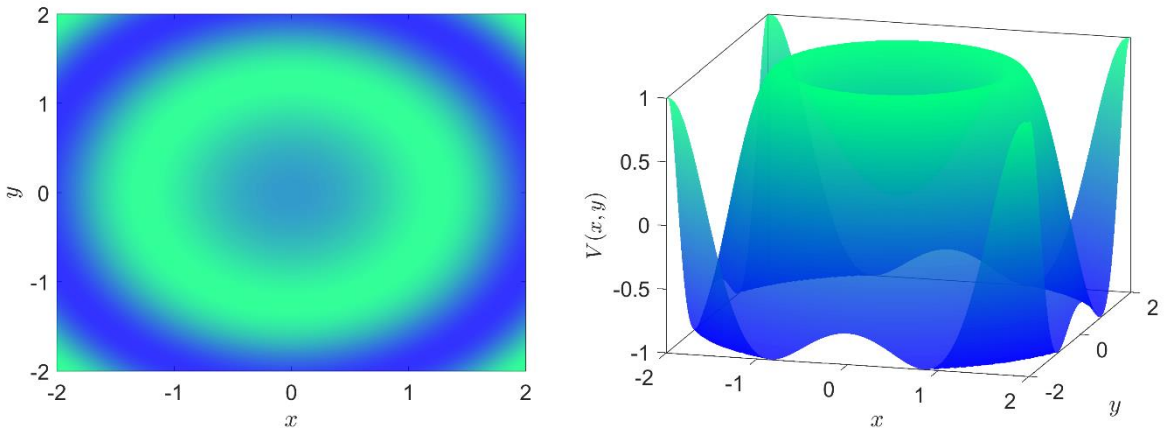
```
clear
clc
step_x=0.05;
x_min=-2;
x_max=2;
step_y=0.05;
y_min=-2;
y_max=2;
x=x_min:step_x:x_max;
y=y_min:step_y:y_max;
[X, Y]=meshgrid(x, y);
V=sin(X.^2+Y.^2);
```

Listing 1. Kod generujące dane potrzebne do wykonania wykresu.

2.2 Wykres `surf oraz mesh`

Funkcja surf pozwala na wykonanie wykresu powierzchni 2D postaci $z = f(x, y)$, gdzie współrzędna z jest trzecim kierunkiem przestrzeni. Funkcji tej można również użyć w postaci mapy kolorów, gdzie każdy poligon powierzchni ma kolor proporcjonalny do składowej z tej powierzchni. Wykres funkcji

(1) z użyciem funkcji surf programu Matlab prezentuje Rysunek 2. Kod generujący Rysunek 2. prezentuje Listing 2.



Rysunek 2. Rysunek funkcji (1), gdzie wartości funkcji prezentowane są w postaci mapy kolorów (lewy) oraz w postaci powierzchni 2D łącznie z mapą kolorów (prawy).

```
figure
hold on
surf(X, Y, V, 'FaceColor', 'interp', 'FaceAlpha', 0.8, 'FaceLighting', 'gouraud', 'EdgeColor', 'none');
xlabel('$x$', 'Interpreter', 'Latex', 'FontSize', 15);
ylabel('$y$', 'Interpreter', 'Latex', 'FontSize', 15);
zlabel('$V(x, y)$', 'Interpreter', 'Latex', 'FontSize', 15);
set(gca, 'Box', 'on', 'FontSize', 15);
view(2);
colormap winter
```

Listing 2. Kod programu rysującego funkcję (1) z użyciem funkcji `surf`.

Kod programu zaczyna się stworzeniem obiektu `figure`, w którym umieszczone są osie układu współrzędnych. Instrukcja `hold on` pozwala na wykonanie kilku instrukcji rysujących w jednym obiekcie `figure`. Jeśli chcielibyśmy wykonać naraz wykres dwóch różnych funkcji, to bez instrukcji `hold on` narysowana zostanie wyłącznie ostatnia instrukcja.

W podstawowej wersji funkcja `surf` wymaga podania jedynie macierzy wartości. Siatka elementów x, y zostałaby stworzona automatycznie. Warto jednak podać również współrzędne x, y każdego punktu w postaci macierzy X, Y. Kolejne instrukcje pojawiające się w funkcji `surf` służą do zdefiniowania atrybutów wykresu. Atrybut `'FaceColor'` z instrukcją `'interp'` koloruje poligony powierzchni interpolując wartość koloru. Atrybut `'FaceAlpa'` z wartością 0.8 informuje program, że powierzchnia powinna być przezroczysta, gdzie wartość określa współczynnik przezroczystości (80%). Atrybut `'FaceLightining'` z opcją `'gouraud'` określa sposób renderowania odbicia światła od powierzchni. Atrybut `'EdgeColor'` z instrukcją `'none'` mówi, że nie chcemy, aby węzły powierzchni były połączone linią.

Funkcja `xlabel` pozwala na opis osi x. Wielkość czcionki została ustawiona na 15. W opisie osi używamy składni Latex, stąd pojawił się atrybut `'Interpreter'`. Żeby użyć składni Latex, to opis osi jako pierwszy argument funkcji musi znajdować się między znakami $$. Użycie składni Latex pozwala na stosowanie w opisie wielu symboli, liter greckich czy nawet działań matematycznych. Opis kolejnych osi wykonujemy funkcjami `ylabel` oraz `zlabel`.

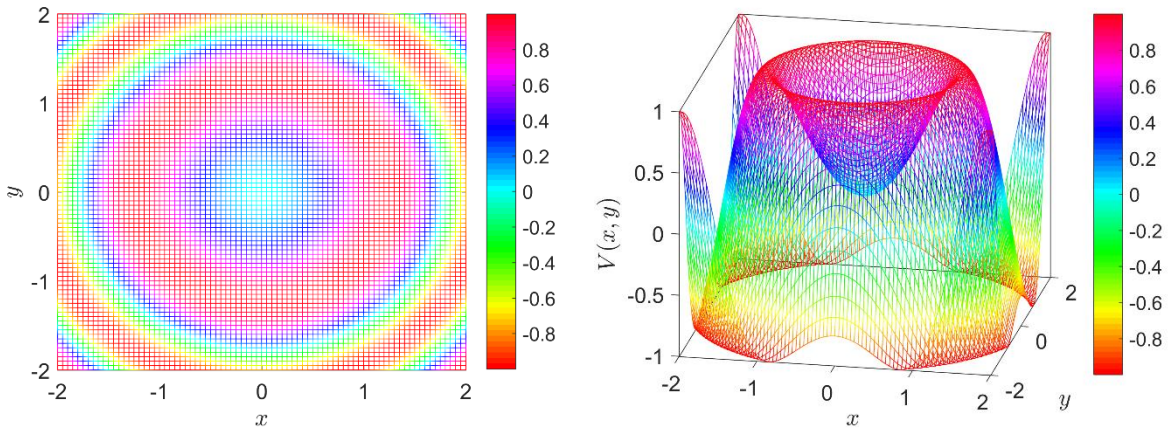
Funkcja `set` jest ogólną funkcją umożliwiającą określenie parametrów obiektu. W naszym wypadku funkcja `set` wywołana jest dla obiektu `gca`, czyli obecnie używanego układu współrzędnych (gca – ang. *get current axis*). W tej funkcji można ustawić rozmiar czcionki dla zapisu skali liczbowej

(`'FontSize'`), określi czy wykres ma znajdować się w polu otoczonym krawędziami (`'Box'` z opcją `'on'`) oraz szereg innych atrybutów. Wśród innych znajdują się np. typ skali układu współrzędnych (`'XScale'`, `'log'` pozwala ustawić skalę logarytmiczną) czy kolor osi (`'YColor'`) i wiele innych.

Opcja `view(2)` pozwala wyświetlić rysunek z punktu widzenia osi *z*. Tak powstał lewy panel Rysunku 2. Po uruchomieniu obiektu `figure` jako nowe okno istnieje opcja obrotu układu współrzędnych przyciskiem  , co pozwala np. pokazać pełny obraz rysowanej powierzchni jak na prawym panelu Rysunku 2.

Rysunek 2. powstał z użyciem palety `'winter'`, jednej z wielu do wyboru z wbudowanych palet w programie Matlab. Spis wszystkich można uzyskać w pomocy do funkcji `surf`. Jeśli żadna z wbudowanych palet nie spełnia naszych oczekiwań, to można stworzyć własną paletę.

Funkcja `mesh` działa analogicznie do funkcji `surf` i przyjmuje te same argumenty. Jedyną różnicą jest to, że funkcja `mesh` domyślnie ustawia brak rysowania poligonów powierzchni (atrybut `'FaceColor'` ustawiony jest na `'none'`), natomiast funkcja `surf` domyślnie rysuje zarówno poligony jaki i krawędzie. Rysunek 3. prezentuje wykres funkcji (1) z użyciem funkcji `mesh`. Na rysunku użyto palety `'hsv'` oraz dodatkowo pokazano pasek kolorów, dzięki którem można powiązać kolor siatki z wartością funkcji (1). Pasek kolorów uaktywnia się wpisując komendę `colorbar` w nowej linii. Kod wykonujący Rysunek 3. prezentuje Listing 4.



Rysunek 3. Rysunek funkcji (1) z użyciem funkcji `mesh`. Widok wzdłuż osi *z* (lewy) oraz widok całej powierzchi (prawy).

```
figure
hold on
mesh(X, Y, V, 'FaceColor', 'none', 'FaceAlpha', 0.8, 'FaceLighting', 'gouraud', 'EdgeColor', 'interp');
xlabel('$x$', 'Interpreter', 'Latex', 'FontSize', 15);
ylabel('$y$', 'Interpreter', 'Latex', 'FontSize', 15);
zlabel('$V(x, y)$', 'Interpreter', 'Latex', 'FontSize', 15);
set(gca, 'Box', 'on', 'FontSize', 15);
view(2);
colormap hsv
colorbar
```

Listing 4. Kod programy rysującego Rysunek 3.

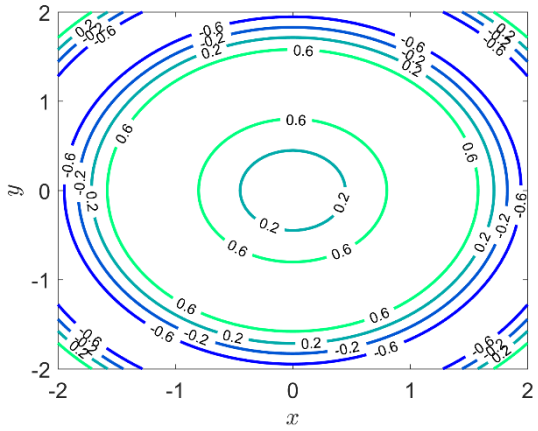
2.3. Wykres `contour` oraz `contourf`

W przypadku funkcji skalarnych może nas interesować układ punktów o tej samej wartości funkcji. Taki układ punktów dla funkcji dwóch zmiennych nazywa się izolinią. Wykresy izolinii często stosowane są na mapach topogrficznych w celu pokazania ukształtowania terenu. Wtedy zwane są

poziomicami. W fizyce izolinie to nic innego jak linie pola i pomagają określić kierunek natężenia pola, które jest w danym punkcie prostopadłe do izolinii przechodzącej przez ten punkt.

Wykres izolinii lub inaczej zwany konturowym można wykonać dzięki funkcji `contour`. W podstawowej wersji przyjmuje jedynie macierz wartości funkcji, jednak warto zdefiniować również wartości współrzędnych przestrzennych x, y oraz liczbę izolinii prezentowanych na wykresie. Przykładowo, wywołanie instrukcji `contour(X, Y, V, 10)` spowoduje narysowanie wykresu konturowego o 10 poziomach izolinii dla funkcji o wartościach przechowywanych z macierzy `V`, dla współrzędnych zdefiniowanych w macierzach `X, Y` zgodnie z konwencją omówioną w punkcie 2.1. Kolor izolinii zostanie dobrany proporcjonalnie do wartości funkcji, której izolinia odpowiada.

Dla danych wygenerowanych według instrukcji z punktu 2.1 wykres konturowy o 4 poziomach izolinii prezentuje Rysunek 4. Kod programu Matlab prezentuje Listing 5. Kod jest bardzo podobny do kodu używanego w przypadku funkcji `surf` oraz `mesh`. Wynika to z faktu, że wszystkie funkcje rysujące wykresy w programie Matlab używają wspólnej puli atrybutów, a jedyne różnice wynikają ze specyfiki danego typu wykresu. Nowością w przypadku wykresu konturowego jest atrybut '`ShowText`', który pozwala nanieść wartość funkcji na izolinię. Jeśli wartość atrybutu będzie wynosiła '`off`', to liczby pojawiające się przy izoliniach na Rysunku 4. nie zostaną wyświetlone. Atrybut '`LineWidth`' pojawia się również w innych typach wykresów i oznacza grubość rysowanej linii.



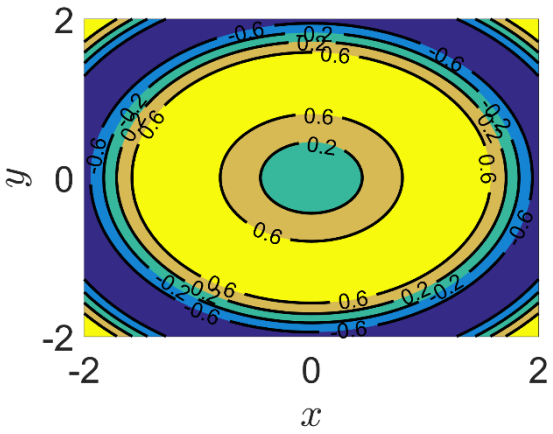
Rysunek 4. Wykres konturowy wykonany z użyciem funkcji `contour`.

```
figure
hold on
contour(X, Y, V, 4, 'ShowText', 'on', 'LineWidth', 2);
xlabel('$x$', 'Interpreter', 'Latex', 'FontSize', 15);
ylabel('$y$', 'Interpreter', 'Latex', 'FontSize', 15);
zlabel('$V(x, y)$', 'Interpreter', 'Latex', 'FontSize', 15);
set(gca, 'Box', 'on', 'FontSize', 15);
view(2);
colormap winter
```

Listing 5. Kod programu generującego wykres konturowy z użyciem funkcji `contour`.

Wariantem funkcji `countour` jest funkcja `countourf`, która rysuje izoliniie w jednym kolorze, natomiast wypełnienie przestrzeni miedzy izoliniami jest wypełnione kolorem proporcjonalnym do poziomu sąsiadującej z tym obszarem izolinii. Pozostała składnia funkcji nie ulega zmianie. Wykres wykonany funkcja `contourf` z użyciem palety kolorów '`parula`', która również jest paleta domyślną prezentuje Rysunek 5. Kod generujący prezentuje Listing 6. W kodzie pojawia się dodatkowa funkcja `clabel`, przyjmująca dwa argumenty '`C`' oraz '`h`' zwracane przez funkcję `contourf`. Te samą funkcję można użyć dla funkcji `contour`. Zmienna '`C`' przechowuje

informację o kształcie izolinii w formie macierzy o dwóch wierszach. Zmienna ‘h’ to obiekt przechowujący atrybuty izolinii. Przy użyciu funkcji `clabel` można zmienić np. czcionkę zapisu wartości na izoliniach czy ich kolor, itp.



Rysunek 5. Wykres typu contourf.

```
figure
hold on
[C, h]=contourf(X, Y, V, 4, 'ShowText', 'on', 'LineWidth', 2);
xlabel('$x$', 'Interpreter', 'Latex', 'FontSize', 15);
ylabel('$y$', 'Interpreter', 'Latex', 'FontSize', 15);
zlabel('$V(x, y)$', 'Interpreter', 'Latex', 'FontSize', 15);
set(gca, 'Box', 'on', 'FontSize', 25);
view(2);
clabel(C, h, 'FontSize', 15);
colormap parula
```

Listing 6. Kod programu generującego Rysunek 5.

3. Wykres funkcji wektorowej.

W poprzednim punkcie rozważaliśmy funkcję skalarną $f(x, y)$, czyli przyjmującą jedną wartość dla punktu przestrzeni xy. W przypadku pola wektorowego dla każdego punktu przestrzeni funkcja przyjmuje w ogólności trzy wartości, którymi są składowe wektora pola. Przykładem jest natężenie pola elektrycznego. W celu pokazania pola wektorowego wykorzystamy funkcję `quiver`.

3.1. Funkcja `griddata`

Żeby używać funkcji Matlaba należy dane przechowywać w postaci macierzowej. W rzeczywistości, bardzo często mierząc dane zapisujemy je w formie wektorów. Przykładowo wykonujemy pomiar potencjału elektrycznego na płaszczyźnie. W programie Excel zapisujemy dane pomiarowe: w pierwszej kolumnie zapisujemy współrzędną x, w drugiej kolumnie współrzędną y, a w trzeciej wartość potencjału. Żeby wykonać wykres przestrzenny potencjału musimy dokonać konwersji danych do postaci macierzowej, zgodnie z konwencją pokazaną w punkcie 2.1. Istnieje funkcja `griddata` dokonująca tego automatycznie.

Wygenerujmy dane testowe. Wylosujmy 20 losowych współrzędnych x z przedziału [-2.5, 2.5] oraz tyle samo współrzędnych y z przedziału [-2.5, 2.5]. Współrzędne przechowajmy w postaci wektora. Wyznaczmy wartości np. funkcji v=$f(x, y) = x\sin(-x^2 - y^2)$, które również będą przechowywane w postaci wektora. Generację danych i ich transformację do postaci macierzowej prezentuje Listing 7.

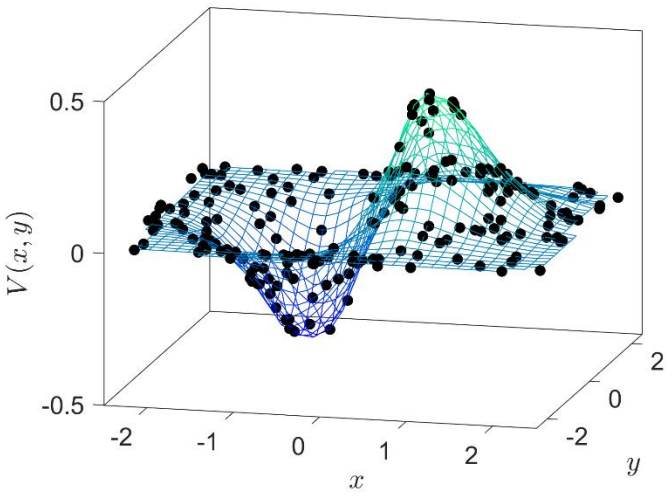
```
x = -2.5 + 5*rand([20]);
y = -2.5 + 5*rand([20]);
v = x.*exp(-x.^2-y.^2);

[Xq, Yq]=meshgrid(-2.5:0.15:2.5, -2.5:0.15:2.5);
[X, Y, V]=griddata(x, y, v, Xq, Yq, 'cubic');
```

Listing 7. Generacja danych w postaci wektorów oraz ich transformacja do postaci macierzowej.

Pierwsze, co należy zrobić, to stworzyć siatkę punktów, dla których wartości pola będą obliczane. Siatkę tą tworzy funkcja `meshgrid`. Przyjmuje ona dwa argumenty, którymi są kolejno współrzędne siatki w kierunku x oraz kierunku y. Siatka zdefiniowana jest w przedziale [-2.5 2.5] z krokiem 0.15. Krok nie musi odpowiadać rzeczywistemu krokowi poboru danych. Jeśli jest inny, to dane zostaną interpolowane. Zmienne `Xq` oraz `Yq` przechowują macierze współrzędnych zgodnie z konwencją programu Matlab.

Mając zbiór punktów siatki, możemy wygenerować macierz wartości dzięki funkcji `griddata`. Argumenty funkcji griddata to kolejno: wektor pomiarowych współrzędnych `x`, wektor pomiarowych współrzędnych `y`, wektor pomiarów wartości pola `v`, współrzędne `Xq` siatki, współrzędne `Yq` siatki, rodzaj interpolacji. W przykładzie wykorzystujemy interpolację '`cubic`', która wykonuje się dłużej niż np. inerpolacja liniowa ('`linear`'), jednak jest dokładniejsza. Macierz `V` przechowuje wartości pola nadające się do narysowania np. funkcją `surf` czy `mesh`.

Rysunek 7. prezentuje przykładowe dane wygenerowane kodem z Listingu 7. Czarne punkty do dane losowe (pomiarowe), natomiast siatka została wygenerowana funkcją `griddata` na podstawie tychże danych. Kod generujący Rysunek 7. prezentuje Listing 8. Funkcja plot3 jest uogólnieniem funkcji plot dla przestrzeni trójwymiarowej.



Rysunek 7. Przykładowe dane w formie punktów sprowadzone do postaci macierzy.

```
figure
hold on
plot3(x, y, v, 'ok', 'MarkerFaceColor', 'k');
mesh(Xq, Yq, V, 'EdgeColor', 'interp', 'FaceColor', 'none');
xlim([-2.5 2.5]);
ylim([-2.5 2.5]);
xlabel('$x$', 'Interpreter', 'Latex', 'FontSize', 15);
ylabel('$y$', 'Interpreter', 'Latex', 'FontSize', 15);
zlabel('$V(x, y)$', 'Interpreter', 'Latex', 'FontSize', 15);
set(gca, 'Box', 'on', 'FontSize', 15);
view(2);
colormap winter
```

Listing 8. Kod programu rysującego Rysunek 7.

3.2. Funkcja `quiver`

Wykres w postaci strzałek skierowanych wzdłuż kierunku pola wektorowego w danym punkcie można uzyskać dzięki funkcji `quiver`. Funkcja ta przyjmuje kolejno: współrzędne X w postaci macierzy, współrzędne Y w postaci macierzy, Składową pola w kierunku X w postaci macierzowej, składową pola w kierunku Y w postaci macierzowej, czynnik skali dla długości strzałki, dodatkowe parametry wykresu.

Aby użyć funkcji quiver musimy wygenerować składowe pola wektorowego. Załóżmy, że w punkcie 3.1 wygeneorwaliśmy wartości potencjału elektrycznego. Natężenie pola elektrycznego $\vec{\mathrm{E}}$ można obliczyć ze wzoru:

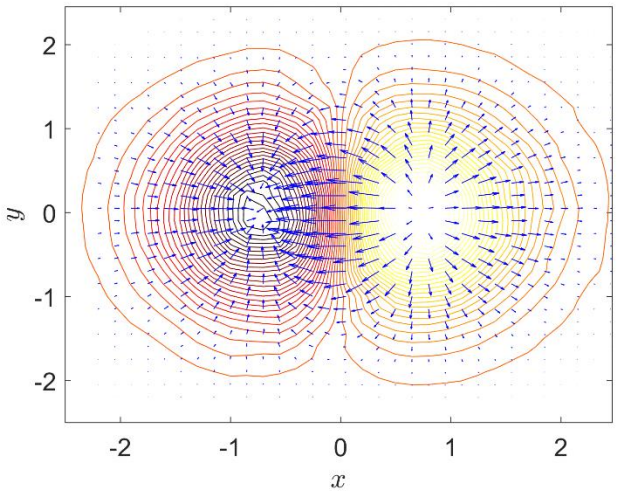
$$\vec{\mathrm{E}} = -\mathrm{grad}V. \tag{2}$$

Do obliczenia gradientu funkcji skalarnej można wykorzystać funkcję gradient. Instrukcja `[Fx, Fy]=gradient(-V);` obliczy gradient z macierzy wartości `V` oraz przypisze składowe `X` gradientu do zmiennej `Fx` oraz składowe `Y` do zmiennej `Fy`.

Listing 9. prezentuje kod obliczający gradient funkcji skalarnej oraz rysujący wekres pola wektorowego w postaci strzałek na tle wykresu konturowego. Wygenerowany w ten sposób wykres prezentuje Rysunek 8.

```
figure
hold on
contour(Xq, Yq, V, 50);
colormap hot
quiver(X, Y, Fx, Fy, 2, 'Color', 'b');
xlabel('$x$', 'Interpreter', 'Latex', 'FontSize', 15);
ylabel('$y$', 'Interpreter', 'Latex', 'FontSize', 15);
set(gca, 'Box', 'on', 'FontSize', 15);
```

Listing 9. Kod programu generującego wykres pola wektorowego obliczonego dzięki funkcji gradient.



Rysunek 8. Wykres pola wektorowego na tle wykresu konturowego. Strzałki w danym punkcie są prostopadłe do linii pola (izolinii).

4. Podsumowanie

Program Matlab ma wbudowany szereg funkcji pozwalających na wykonywanie wykresów funkcji dwóch zmiennych zarówno skalarnych jak i wektorowych. Funkcje te to w szczególności `surf`,

`mesh`, `contour`, `contourf` oraz `quiver`. W celu wykonania rysunków należy zastosować konwencję reprezentacji danych w postaci macierzowej. Dla danych przechowywanych w formie wektorów istnieje funkcja `griddata` zapewniająca konwersję oraz ewentualną interpolację danych.