3D grafika

V lekci 4 jsme se seznámili s 2D grafikou (především grafy funkcí jedné proměnné). MATLAB umožňuje vizualizovat také funkce dvou proměnných. Používáme podobný postup:

1. příprava dat (nezávisle proměnné, závisle proměnná)
2. výběr grafického okna, případně podokna (viz 4. týden)
3. vykreslení 3D grafu
4. úpravy grafu (viz 4. týden; dnes navíc probereme: osvětlení, stínování, kamera)
5. uložení grafu do souboru (viz 4. týden)

Příprava dat

Pro **spojitý graf funkce dvou proměnných** potřebujeme mít **dvě matice** (např. `X`,`Y`), jejichž prvky x_{ij}, y_{ij} (respektive `X(i,j)`, `Y(i,j)`) budou obsahovat souřadnice bodů v rovině (podstava). Ke snadnému generování těchto matic (z posloupností hodnot na ose x a na ose y) slouží funkce `meshgrid`:

- **`[X,Y]=meshgrid(v1,v2)`** ... vektor `v1` určuje interval na ose x s daným krokem a vektor `v2` určuje interval na ose y s daným krokem
- **`[X,Y]=meshgrid(v)`** ... vektor `v` určuje interval na ose x i na ose y

Ukázka:

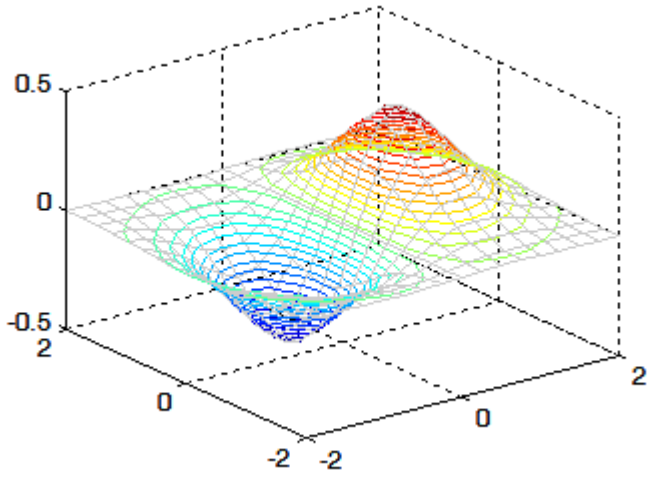
```
>> [X,Y]=meshgrid([-3:1],[-1:2:7])

X =
    -3    -2    -1     0     1
    -3    -2    -1     0     1
    -3    -2    -1     0     1
    -3    -2    -1     0     1
    -3    -2    -1     0     1

Y =
    -1    -1    -1    -1    -1
     1     1     1     1     1
     3     3     3     3     3
     5     5     5     5     5
     7     7     7     7     7

% pozice (1,1) určuje bod A=[-3;-1], kdežto např. pozice (2,4) určí bod A=[0;1]
```

Kromě nezávisle proměnných potřebujeme vytvořit také matici funkčních hodnot.
Např. pro funkci $f(x,y)=\cos(x^2)-\sin(y^3)$ vytvoříme matici `Z`:
`>> Z=cos(X.^2)-sin(Y.^3);` (o práci s maticemi jsme si říkali minule. Uvědomte si, proč je nezbytné používat operátory "s tečkou"...)

Funkce pro kreslení 3D grafů

- čáry ve 3D:
  `plot3(X,Y,Z)`, případně **`plot3(X,Y,Z,str)`**
  kreslí body nebo čáry (parametricky zadané křivky ve 3D, resp. jednotlivé sloupce matice)
  pomocí řetězce `str` můžeme nastavit barvu, značky a typ čar
  používá se podobně jako `plot`, např. lze kreslit více grafů najednou (viz 4. týden)
- 3D síťový graf:
  `mesh(X,Y,Z)` ... vykreslí 3D síťový graf ("drátěný"), body spojuje navzájem ve směru obou os
- 3D plošný graf:
  `surf(X,Y,Z)` ... vykreslí 3D plošný graf, body spojuje navzájem ve směru obou os a vzniklé plošky vyplní barvou (podle hodnot osy *z*)
- vrstevnicový graf (2D):
  `contour(Z)` ... vykreslí vrstevnice = čáry spojující body se stejnou "výškou" (`Z` je matice funkčních hodnot)
  `contour(X,Y,Z)` ... vykreslí vrstevnice + správně popíše osy *x* a *y*
  `contour(Z,n)` ... vykreslí vrstevnice v `n` rovinách, např. `contour(Z,5)`
  `contour(Z,v)` ... vykreslí vrstevnice v rovinách zadaných vektorem `v`, např. `contour(Z,[-2 0 .5 1])`
  Pozn.: chceme-li vykreslit vrstevnice v jediné rovině s hodnotou *t*, musíme použít `contour(Z,[t t])`
  `contour(...,str)` ... za výše uvedené parametry můžeme uvést řetězec `str` specifikující barvu+styl čáry (jako u `plot`, ale značka se ignoruje)
  Pozn.: `[C,h] = contour(...)` vrací matici `C` ("mapa vrstevnic", kterou lze využít pro jejich popisky ve funkci **`clabel`**) a vektor `h` grafických handlů
- vyplněný vrstevnicový graf (2D): **`contourf(Z)`** ... lze ji volat jako `contour`, ale vykresluje barevné plochy dané vrstevnicemi
- 3D vrstevnicový graf: **`contour3(Z)`** ... lze ji volat jako `contour`, ale čáry vrstevnic vykreslí prostorově
  Pozn.: výhodně se kombinuje např. s `mesh` nebo ještě lépe s low-level funkcí `surface` (při vykreslování zachová i vrstevnicový graf):

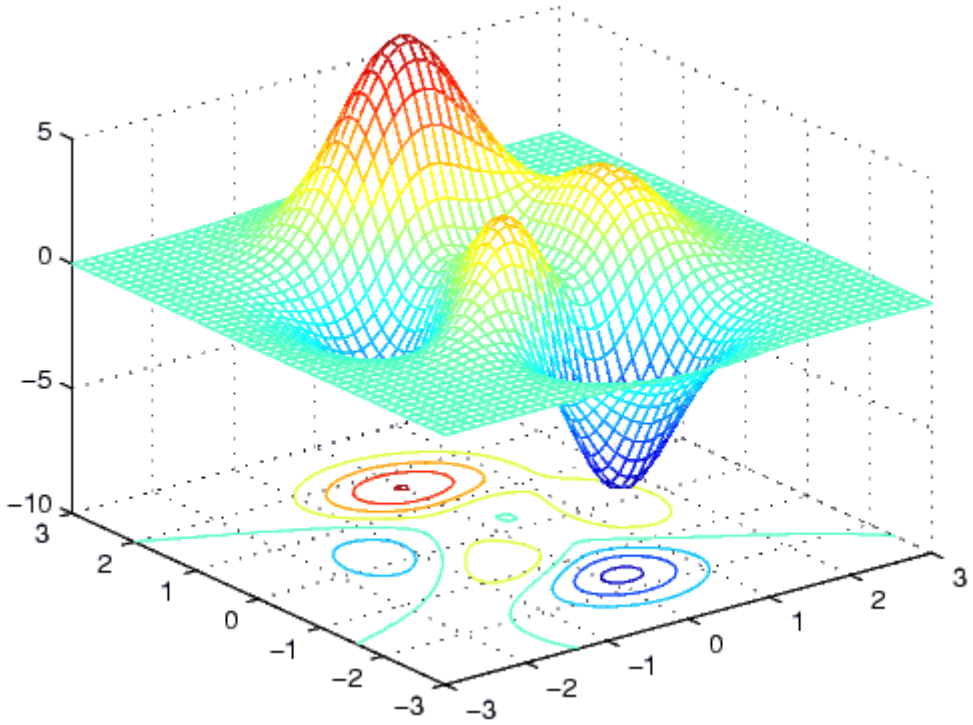
```
>> [X,Y] = meshgrid([-2:1/4:2]);
>> Z = X.*exp(-X.^2-Y.^2);
>> contour3(X,Y,Z,30)
>> surface(X,Y,Z,...        % síť
'EdgeColor',[.8 .8 .8],... % šedá
'FaceColor','none')        % bez výplně
```



- kombinované:
  `meshc(X,Y,Z)` ... vykreslí 3D síťový graf + vrstevnice
  `surfc(X,Y,Z)` ... vykreslí 3D plošný graf + vrstevnice
- jiné: např. `quiver`, `gradient`, `slice`

Všechny funkce vracejí grafický handle (pokud si jejich výstup odebereme). Při vykreslení grafů funkce dvou proměnných se nejčastěji používá síťový a plošný graf.

Příklad - graf funkce dvou proměnných (funkce `peaks` je knihovní funkce MATLABu a vrací matici vytvořenou pomocí Gaussova rozdělení):



graf3d.m

```
[X,Y] = meshgrid(-3:.125:3);
Z = peaks(X,Y);
meshc(X,Y,Z);
axis([-3 3 -3 3 -10 5])
```

Úpravy grafu

- ## Popis grafu

  Pro popis 3D grafu použijeme funkce `xlabel`, `ylabel` a `title` (viz 4. týden).
  Osu *z*, která ve 2D grafech nebyla, můžeme popsat pomocí funkce `zlabel(text)`, kde `text` je libovolný popis.

  Popis vrstevnic provede funkce `clabel(C,h)`, kde `C` a `h` jsou odebrané výstupy funkcí typu `contour` (viz výše).

Změna barevné palety

`colormap(paleta)`, kde `paleta` je předdefinovaná nebo uživatelská. Přehled předdefinovaných barevných palet naleznete v nápovědě (např. šedá `gray`, žlutočervená `bone`, duhová `hsv`,...).

Uživatelskou paletu vytvoříte pomocí matice *N*x3, kde *N* je celkový počet barev v paletě a hodnoty jsou od 0.0 do 1.0 - každý řádek představuje jednu barvu definovanou pomocí RGB (red-green-blue; např. černá barva je `[0 0 0]`, červená `[1 0 0]` a žlutá `[1 1 0]`).

Příklad šestibarevné palety (od černé přes odstíny zelené po světle zelenou):

```
>> M=[0 0 0; 0 .2 0; 0 .4 0; 0 .6 0; 0 .8 0; 0 1 0];
% úsporněji: M=zeros(6,3); M(:,2)=[0:0.2:1]';
>> colormap(M)
```

Poznámka: `colormap` si pamatuje naposledy použitý počet barev v daném grafickém okně. Jiný počet barev zajistíme změnou rozměrů palety (u předdefinovaných např. `colormap(hsv(64))`), případně se můžeme vrátit k prvotnímu nastavení pomocí `colormap('default')`

Přidání sloupce s použitou paletou

`colorbar` ... zobrazí vedle grafu sloupec s barevnou paletou
Poznámky:
- po každé změně barevné palety musíme tuto funkci použít znovu!!!
- chceme-li barvy palety zobrazit jako řádek (ne sloupec), lze použít: `colorbar('horiz')`

Nastavení rozsahu os

Již známe `xlim` a `ylim` (viz 4. týden), ve 3D přibývá `zlim`.

Možnosti funkce `axis` se rozšiřují:

```
>> axis([xmin xmax ymin ymax zmin zmax cmin cmax]).
```

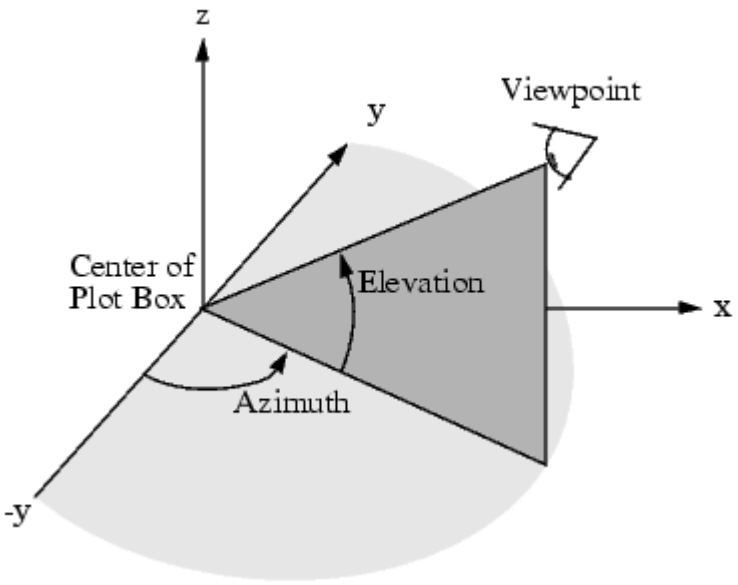
Prostudujte si také funkci `daspect`.

Definování úhlu pohledu (otočení grafu)

`view(az,el)` ... funkce nastaví úhel pohledu na graf: `az` je azimut = úhel otočení kolem osy *z* měřený od záporné části osy *y* proti směru hodinových ručiček (0 až 180) nebo po směru (-180 až 0); `el` je výška (elevace) zorného bodu, tj. úhel mezi pozorovatelem a rovinou *xy* (0 až 180: oko je nad objektem, -180 až 0: oko je pod objektem).

Standardní nastavení pohledu na 3D graf obnovíme pomocí `view(3)` (odpovídá nastavení `az=-37.5`, `el=30`).

Pomocí `[a,e] = view` *uložíme aktuální hodnoty* azimutu a výšky (elevace) do proměnných `a` a `e`.



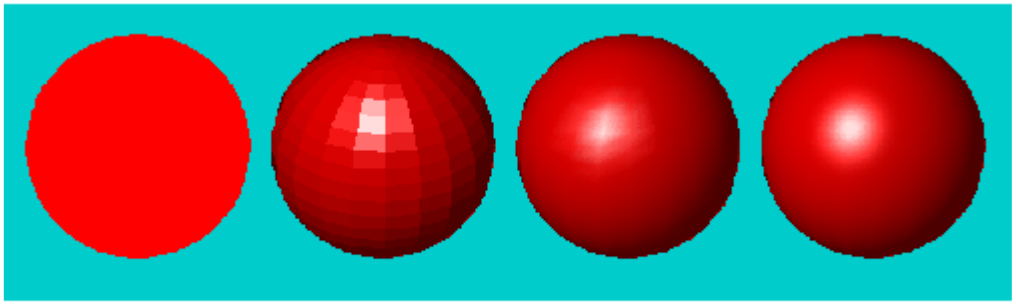
Tip: při otáčení grafu pomocí nástroje  (resp.  v Matlabu 6.5) se vlevo dole v grafickém okně objevují hodnoty azimutu a výšky (elevace).

• Osvětlení

Osvětlovací model lze aplikovat, pokud je ve scéně přidáno alespoň jedno světlo.

`camlight`	vytvoří nebo posune světlo ve scéně ◦ `camlight('headlight')` ... světlo na pozici kamery ◦ `camlight('right')` ... světlo vpravo nahoře od kamery ◦ `camlight('left')` ... světlo vlevo nahoře od kamery ◦ `camlight` ... totéž jako `camlight('right')` ◦ **`camlight(az,el)`** ... světlo na pozici dané azimutem a elevací (výškou) s ohledem na pozici kamery (ta je středem rotace); hodnoty jsou ve stupních ◦ `camlight(...,'style')` ... typ světla: `local` (defaultní) - bodové světlo vyzařující do všech směrů `infinite` - paralelní paprsky z nekonečna ◦ `h = camlight(...)` ... vrací handle vytvořeného světla (např. pro jeho budoucí posun pomocí `camlight(h,...)` Při pohybu kamery se světlo nehýbe, a proto je nutné jej posunout opět pomocí funkce `camligth`.

`light`	low-level funkce pro vytvoření světla (= grafický objekt) s nastavením např. pozice:
	`>> membrane % vykreslí logo Matlabu`
	`>> light('Position',[0 -2 1])`

`lighting`	nastaví osvětlovací model
	`>> lighting` *`alg`*, kde *`alg`* je `none`, `flat`, `gouraud` nebo `phong`
	 
	`none` `flat` `gouraud` `phong`
	Phongův model dává lepší výsledky než Gouraudův [čti gurodův], ale renderování trvá déle.

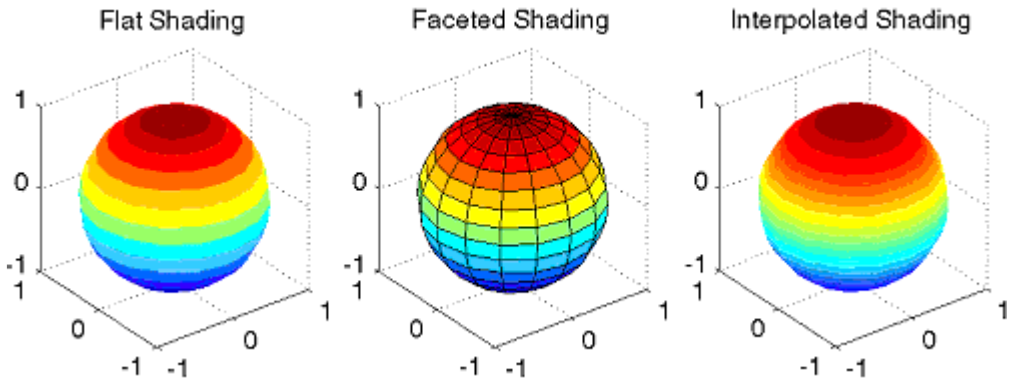
`material`	nastaví režim odrazivosti materiálu (pro plochy, tj. surfaces a patches); ve scéně musí být alespoň jedno světlo ◦ **`material shiny`** ... lesklý materiál, odráží jako zrcadlo, barva světla podle zdroje světla ◦ `material dull` ... matný materiál, odráží rozptýlené světlo, barva světla podle zdroje světla ◦ `material metal` ... kovový materiál, odráží zrcadlově a trochu i difúzní světlo, barva světla podle zdroje světla a barvy objektu

	◦ `material([ka kd ks])` ... nastaví odrazivost pro okolní (ambient), difúzní (diffuse) a zrcadlové (specular) světlo; hodnoty od 0 (min.) do 1 (max.) ◦ `material([ka kd ks n])` ... nastaví navíc i index zrcadlení (např. `dull` má `n=10` a `metal` má `n=25`) ◦ `material([ka kd ks n sc])` ... nastaví navíc i barvu zrcadlového odrazu (např. `dull` má `sc=1.0` a `metal` má `sc=0.5`) ◦ `material default` ... nastaví výchozí hodnoty

• Stínování

Barvu stínování plošných objektů (tj.surface a patch) lze nastavit pomocí funkce `shading`:

- `shading flat` ... každá ploška (a hrana) má konstantní barvu
- `shading faceted` (defaultní) ... ploché stínování s černými hranami
- `shading interp` ... interpoluje barvy plošek tak, aby plynule přecházely



• Průhlednost

`alpha`	režim průhlednosti
	Zkuste (pro plošný graf = `surf`):
	`>> alpha(0.5)`
	`>> alpha('clear') % úplně průhledný`
	`>> alpha('opaque') % neprůhledný`

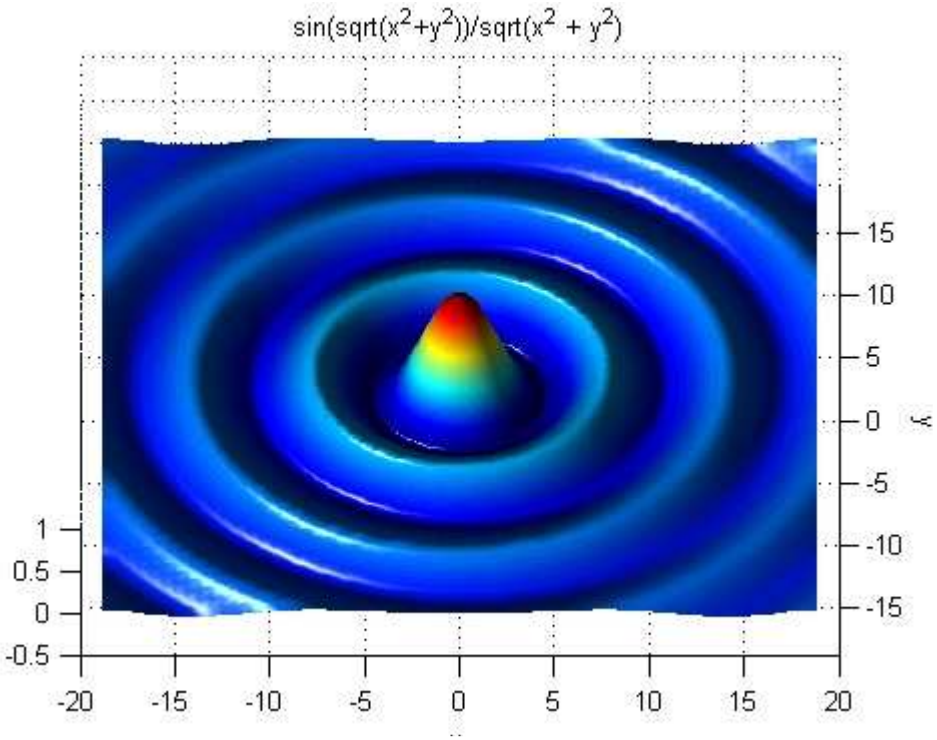
`alphamap`	nastaví tabulku průhlednosti (podobně jako `colormap` pro barvy)
`alim`	nastavení/zjištění rozsahu průhlednosti (podobně jako `xlim`)

• Úpravy grafu s využitím handlů

Platí totéž, co jsme probrali ve 4. týdnu v rámci 2D grafiky (odebrat handle a použít `get` a `set`). Samozřejmě vlastnosti ploch jsou v něčem jiné než vlastnosti čar.

Příklad - graf matematické funkce sinc($\sqrt{(x^2+y^2)}$) na intervalu [-6π,6π] (nakreslený pomocí symbolického `ezplot`):

```
>> ezsurf('sin(sqrt(x^2+y^2))/sqrt(x^2+y^2)',[-6*pi,6*pi]) % 3D graf
>> view(0,75) % nastavení pohledu
>> shading interp  % režim stínování ("plynulý")
>> lightangle(-45,30)  % nastavení sférické polohy světla
>> lighting phong  % Phongův osvětlovací model
>> set(gcf,'Renderer','zbuffer')  % metoda vyobrazení (zde: rychlejší vykreslení)
>> set(findobj(gca,'type','surface'), 'FaceLighting','phong',...
    'AmbientStrength',.3,'DiffuseStrength',.8, 'SpecularStrength',.9,...
    'SpecularExponent',25, 'BackFaceLighting','unlit')  % "zářivější" graf
```



Práce s obrázky

Maticí typu *m*×*n* můžeme kromě dat (obecně komplexních čísel) nebo hodnot funkce dvou proměnných reprezentovat také obrázek (image). K jeho nakreslení ale musíme použít jiné funkce než `mesh` nebo `surf`.

Většina dvourozměrných obrazů je ve workspace MATLABu reprezentována dvourozměrným polem (maticí), jehož každý prvek odpovídá jednomu pixelu zpracovávaného obrazu. Například obraz, který má 200 řádků a 300 sloupců různobarevných pixelů, je reprezentován maticí typu 200×300.
Některé typy obrazů, například RGB obrazy, potřebují trojrozměrné pole (matici typu *m*×*n*×3, kde první "vrstva" obsahuje intenzitu červené barvy, druhá zelené a třetí "vrstva" intenzitu modré barvy pro každý pixel).

Typy dat, které MATLAB nabízí pro uložení obrazu:

- double ... plovoucí řádová čárka, double-precision,
- uint16 ... 16bitový unsigned integer,
- uint8 ... 8bitový unsigned integer.

Nejčastěji je využíván double (všechny funkce MATLABu totiž pracují s tímto datovým typem), avšak ten zabírá nejvíce paměti. Proto se při práci s velkými obrazy (tisíce pixelů) používají ostatní dva typy - podstatně se sníží nároky na paměť.

Podporované obrazové formáty

MATLAB podporuje (tj. umí číst, zapsat a zobrazit) mimo jiné tyto obrazové formáty:

- BMP (Microsoft Windows Bitmap)
- GIF (Graphics Interchange Files)
- JPEG (Joint Photographic Experts Group)
- PNG (Portable Network Graphics)
- TIFF (Tagged Image File Format)

Podrobnosti o uložení a typech dat naleznete v nápovědě k funkci `imread`.

Načtení obrazu

MATLAB nabízí funkci `imread`, která podle typu vstupního souboru správně vytvoří dvourozměrnou (nebo trojrozměrnou) matici dat.

Možnosti volání:

- `A = imread(soubor, formát)` ... načte data ze souboru zadaného jménem (řetězec; případně i s cestou, pokud soubor není v pracovním adresáři nebo ve "vyhledávacích" cestách MATLABu), druhým parametrem je zkratka obrazového formátu - většinou shodné s příponou souboru (viz nápověda k funkci). Výstupem je matice typu *m*×*n* (obraz ve stupních šedi, resp. s indexovanou paletou) nebo *m*×*n*×3 (truecolor, RGB obraz) nebo *m*×*n*×4 (TIFF využívající barevný prostor CMYK). Datový typ matice závisí na bitové hloubce načítaného obrazu (viz nápověda);
- `[X, map] = imread(...)` ... načte data ze souboru obsahujícího indexovaný obraz, do druhého výstupu přiřadí hodnoty jednotlivých barev použité palety (automaticky převede hodnoty barev do intervalu [0,1]);
- další možnosti volání funkce najdete v nápovědě.

Přístup k pixelům obrazu

Jelikož je každý obraz matice, tak pro přístup k hodnotě intenzity každého pixelu můžeme použít kulaté závorky s uvedením řádku a sloupce (případně i jiné druhy indexace matic popsané v minulé lekci).

Hodnotu intenzity pixelu lze samozřejmě i změnit.

Informační funkce

Informace o podporovaných formátech (při načítání souborů v MATLABu) vrací funkce **`imformats`** (výsledkem je pole struktur).

Informace o konkrétním obrazu (grafickém souboru) vrací funkce **`info = imfinfo(soubor,formát)`** (výsledkem je struktura, jejíž položky závisí na konkrétním obrazovém formátu).
Např.:

```
>> info = imfinfo('moodlelogo.gif')

info =

           Filename: 'moodlelogo.gif'
        FileModDate: '25-Oct-2010 16:05:10'
           FileSize: 2617
             Format: 'GIF'
      FormatVersion: '89a'
              Width: 100
             Height: 30
           BitDepth: 8
          ColorType: 'indexed'
    FormatSignature: 'GIF89a'
    BackgroundColor: 0
        AspectRatio: 0
         ColorTable: [256x3 double]
         Interlaced: 'no'
```

Pro obrazové formáty JPEG a TIFF můžeme získat EXIF (Exchangeable Image File Format) informace pomocí funkce **`exifread`**. Funkce vrací strukturu s metadaty (EXIF značkami - viz http://www.exif.org/), ale nezpracovává je. Tato funkce bude v novějších verzích MATLABu nahrazena výše uvedenou `imfinfo`.

Zápis dat do obrazového souboru (uložení obrazu)

K zápisu obrazových dat do souboru (tj. k uložení obrazu) se používá funkce `imwrite`.

Možnosti volání:

- **`imwrite(A,soubor,formát)`** ... zapíše data obrazu (matice) `A` do souboru zadaného jménem (řetězec), použije zadaný obrazový formát (pro podporované formáty použijte `imformats`). Proměnná `A` nesmí být prázdné pole a musí obsahovat data, která lze zapsat do požadovaného formátu (např. RGB obraz nelze zapsat do GIFu);
- **`imwrite(X,mapa,soubor,formát)`** ... zapíše indexovaný obraz `X` a jeho barevnou paletu do zadaného obrazového souboru. Má-li `X` datový typ uint8 (resp. uint16), pak jsou data zapsána přímo do souboru. Má-li `X` datový typ double, funkce nejprve použije převod `uint8(X-1)`. Barevná paleta `mapa` obsahuje každou barvu jako trojici čísel z intervalu [0,1]. Pozor: většina obrazových formátů nepodporuje obrazovou paletu s více než 256 barvami!
  Při zápisu do GIFu, který obsahuje více obrázků (tzv. multiframe GIF), musí matice `X` být typu $m \times n \times 1 \times P$ (P je počet obrazů - frames);

- `imwrite(...,soubor)` ... zapíše data do souboru, přičemž obrazový formát se určí z přípony souboru;
- `imwrite(...,Param1,Val1,Param2,Val2...)` ... umožňuje zadat parametry ovlivňující výsledný grafický soubor (např. kvalitu komprese u JPG) - podrobnosti ke každému formátu najdete v nápovědě (část Format-Specific Parameters).

Kreslení (zobrazení) obrazů v MATLABu

Pro vykreslení obrázku do grafického okna můžeme použít funkce `image` nebo `imagesc`:

- Funkce `image` má dvě varianty - high-level, která volá funkci `newplot` a nastaví podle vykreslovaných dat některé vlastnosti os (XLim, YLim, Layer, YDir, View) nebo low-level, která nevolá `newplot` a jejími vstupy jsou pouze dvojice vlastnost-hodnota.
  - **`image(C)`** ... vykreslí matici `C` jako obrázek. Každý prvek matice je chápán jako barva pixelu.
  - `image(x,y,C)` ... vykreslí matici `C` jako obrázek a použije vektory `x` a `y` jako popisky os (pokud jsou hodnoty seřazené sestupně, pak obraz bude převrácený). V případě vykreslení do již existujících os s použitím `hold on` se obraz roztáhne podle potřeby.
  - `image(x,y,C,'PropertyName',PropertyValue,...)` ... kromě vykreslení matice `C` se nastaví také zadané vlastnosti (high-level funkce).
  - `image('PropertyName',PropertyValue,...)` ... low-level funkce pro nastavení grafického objektu image.

  Všechny uvedené možnosti volání mohou vracet grafický handle vytvořeného objektu: `handle = image(...)`.

  Matice `C` může obsahovat jak indexovaná data (každý prvek obsahuje index do barevné palety), tak RGB data (trojrozměrná matice, kde barvu udává trojice prvků `C(i,j,:)`).

- Funkce `imagesc` přeškáluje data na plný rozsah barevné palety a zobrazí obraz:
  - **`imagesc(C)`** ... nakreslí matici `C` jak obraz. Hodnoty prvků matice jsou chápány jako indexy do barevné palety.
  - `imagesc(x,y,C)` ... nakreslí matici `C` jako obraz a použije rozsah os zadaný vektory `x` a `y` (jsou-li řazeny sestupně, obraz je v daném směru převrácený).
  - **`imagesc(...,clims)`** ... přepočítá (normalizuje) hodnoty matice `C` do rozsahu zadaného dvouprvkovým vektorem `clims` a vykreslí obraz.

  Funkce opět může vracet grafický handle: `h = imagesc(...)`. Funkce má také low-level verzi, kde lze zadávat pouze dvojice vlastnost-hodnota.

  Poznámka: implicitně funkce `imagesc` vykresluje hodnoty na ose *y* od nejnižší po nejvyšší (odshora až dolů). Pokud si to nepřejete, nastavte `set(gca, 'YDir', 'normal')`, čímž se obrátí osa *y* i obraz.

Obrazy, které jsou součástí MATLABu

MATLAB obsahuje v podadresáři `toolbox/matlab/demos/` (pokud byl vytvořen při instalaci) několik obrazů uložených v MAT-souborech nebo v grafických souborech, např.:

jméno souboru	typ	obsah
cape	MAT	snímek Nové Anglie z družice NOAA
clown	MAT	obličej klauna

durer	MAT	obraz Melancholia od Albrechta Dürera z roku 1514
detail	MAT	detail magického čtverce z předchozího obrazu
earth	MAT	planeta Země
flujet	MAT	kolize dvou proudů od NCSA
gatlin	MAT	fotka z konference o numerické algebře (Gatlinburg, 1964)
gatlin2	MAT	výřez předchozí fotky
mandrill	MAT	"obličej" mandrila
ngc6543a.jpg	JPG	astronomický snímek (podrobnosti v souboru ngc6543a.txt)
spine	MAT	kosti

Většina MAT souborů obsahuje proměnné `X` (matice s daty), `map` (barevná paleta) a `caption` (popis obrazu). Načítáme je příkazem **`load jméno_souboru`**.
Některé z uvedených obrazů si můžete prohlédnout pomocí příkazu **`imageext`** (spustí demo, kde lze volit obraz a barevnou paletu).
Další demo pro práci s obrazy se jmenuje `imagedemo` (viz `doc imagedemo`).

JPG obrázek načteme příkazem **`X = imread('ngc6543a.jpg');`** (se středníkem!).

Další obrázky jsou uložené v toolboxu pro práci s obrazy.

Příklad práce s obrazy:

```
>> load durer % načteme obrazová data
>> image(X) % nakreslíme obraz
>> colormap(map) % nastavíme barevnou paletu
>> title(caption) % přidáme popisek obrázku
>> axis image % axis equal

>> vyrez = X(90:170, 394:480); % středník! Uděláme si výřez magického čtverce...
>> figure, image(vyrez), colormap(map), axis image % ...a vykreslíme jej

>> load detail % načteme si detail magického čtverce ve vyšším rozlišení
>> figure, image(X), colormap(map), title(caption), axis image % a vykreslíme
```

Image Processing Toolbox

MATLAB samotný nenabízí mnoho funkcí pro práci s obrazy. Pokud potřebujete provádět nějaké speciální činnosti (např. pokročilejší analýzu obrazu, vylepšování, filtraci, transformace, morfologické operace, analýzu textur nebo konverzi indexovaného obrazu na RGB), musíte zakoupit Image Processing Toolbox (nebo si příslušné funkce naprogramovat sami).

Přehled funkcí z tohoto toolboxu (je-li instalován) získáte příkazem **`help images`**.

Pokud je Image Processing Toolbox instalován, tak v podadresáři `toolbox/images/imdemos/` jsou k dispozici další grafické soubory.