M A T L A B
Díl II. – Popis funkcí

Blanka Heringová, Petr Hora

H-S 1995

Blanka Heringová
Petr Hora

MATLAB

Obsah

Funkce Absolutní hodnota a konverze řetězce na číslo.

Syntaxe `Y=abs(X)`

Popis `abs(X)` je absolutní hodnota X.
Je-li `Z` komplexní, `abs(Z)` vrací modul komplexního čísla (amplitudu):

```
abs(Z)=sqrt(real(Z).^2+imag(Z).^2)
```

Pokud je `S` řetězec, `abs(S)` vrací numerickou hodnotu ASCII znaků v řetězci. Změní se výstupní tvar řetězce, vnitřní reprezentace je však stejná.

Příklady

```
abs(-5)=5
abs(3+4i)=5
abs('3+4i')=[51 32 43 32 52 105]
```

Viz též Strings, `setstr`, (`angle`, `sign`, `unwrap`)

Funkce Umístí osy na zadanou pozici vytvořením grafického objektu axes.

Syntaxe

```
h=axes
axes(h)
h=axes(PropertyName,PropertyValue,...)
```

Popis `axes` je grafická funkce nižší úrovně, která vytváří objekty axes. Objekty axes jsou dětmi objektů figure a rodiči objektů line, surface, patch, image a text. Objekty axes definují polohu a rozsah svých dětí uvnitř objektu figure.

`axes` – samostatně – vytváří objekt axes a vrací jeho identifikátor.

`axes(h)` nastaví objekt axes s identifikátorem `h` jako aktuální objekt axes.

`axes` je funkce, která vytváří objekt axes. Tento objekt akceptuje jako vstupní argumenty dvojice parametrů PropertyName/PropertyValue (jméno vlastnosti/hodnota vlastnosti). Tyto vlastnosti, kterými lze ovládat vzhled objektu axes, jsou popsány dále v části *Vlastnosti objektu*. Hodnoty vlastností můžeme nastavit nebo získat zpět po vytvoření objektu pomocí funkcí `set` a `get`.

Příklady Důležitou vlastností objektu axes je vlastnost `Position`. Ta nám umožňuje definovat umístění objektu axes v grafickém okně. Příkaz

```
h=axes('Position',rect)
```

vytvoří objekt axes na uvedené pozici uvnitř aktuálního grafického okna a vrátí jeho identifikátor. Umístění a velikost objektu axes určujeme prostřednictvím obdélníku definovaného čtyř-prvkovým vektorem

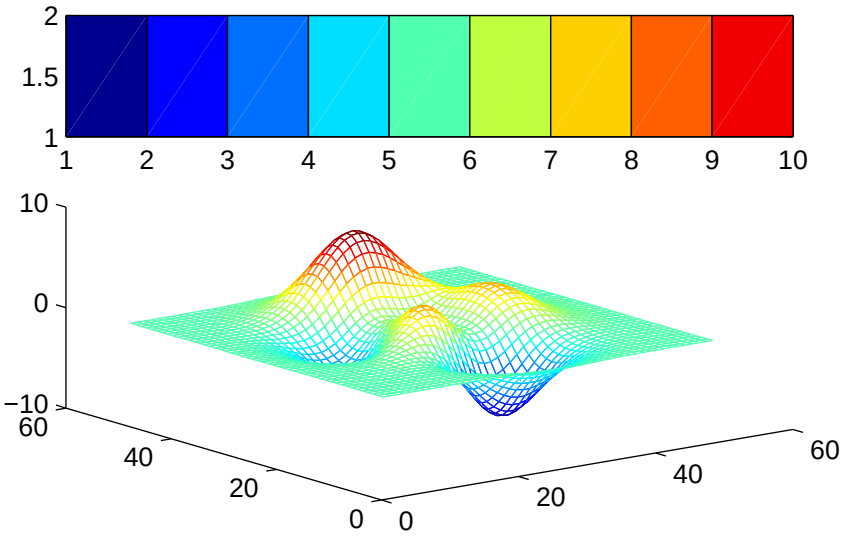
```
rect=[zleva, zespodu, šířka, výška]
```

Prvky `zleva` a `zespodu` definují vzdálenosti dolního levého rohu grafického okna od dolního levého rohu obdélníku. Prvky `šířka` a `výška` definují rozměry obdélníku. Tyto hodnoty zadáváme v jednotkách určených vlastností `Units`. MATLAB používá implicitně normalizované jednotky, kde $(0, 0)$ je levý dolní roh a $(1, 1)$ je pravý horní roh grafického okna.

V jednom grafickém okně můžeme též definovat několik objektů axes:

```
clf
axes('position',[.1 .1 .8 .6])
mesh(peaks)
axes('position',[.1 .7 .8 .2])
pcolor([1:10;1:10]);
```

V tomto příkladě zabere první objekt axes zezdola 2/3 grafického okna, druhý horní třetinu.



Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu pomocí dvojic parametrů PropertyName/PropertyValue jako vstupních argumentů ve funkci vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu prostřednictvím jeho identifikace a následným použitím funkcí `set` a `get` se získanými identifikátory.

Následující přehled obsahuje všechny vlastnosti objektu axes včetně všech jejich možných hodnot. Je-li nastavena implicitní hodnota, je tato hodnota ve složených závorkách.

AspectRatio `[axis_ratio, data_ratio]`

Poměr vzhledu 2-D objektu axes. Tato vlastnost řídí poměr vzhledu 2D objektu axes. Vektor o dvou prvcích určuje dva poměry:

- Poměr délky svislé osy k délce vodorovné osy, tj. výška dělená šířkou.
- Poměr délky datové jednotky ve směru svislé osy k poměru délky datové jednotky ve směru vodorovné osy.

Poměr definovaný v prvku `axis_ratio` užívá MATLAB pro vytvoření největšího objektu axes s tímto poměrem, který se vejde do obdélníku určeného vlastností `Position`.

Při vytváření objektu axes s požadovaným poměrem `data_ratio` mění MATLAB rozsah jedné osy a zachovává poměr délek nastavený v `axis_ratio`. Tato změna v rozsahu os nemá vliv na odpovídající vlastnost mezí (`XLim` nebo `YLim`).

Poměry mohou být libovolná čísla v rozsahu $[0, \infty]$. Implicitní pro oba poměry je hodnota `NaN`; tzn. žádný poměr. Implicitně mění MATLAB oba poměry, aby vytvořil objekt axes, který nejlépe vyplní grafické okno.

Box `on | off`

Režim rámečku objektu axes. Tato vlastnost určuje, zda má nebo nemá být grafická plocha uzavřena do rámečku (2D) nebo do krychle (3D).

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy je kurzor na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu. Všimněme si, že vlastnost `CallBack` pro objekt uimenu nahrazuje `ButtonDownFcn`, ale objekty uicontrol mají jak vlastnost `CallBack`, tak i vlastnost `ButtonDownFcn`.

Children *vektor identifikátorů*

Děti objektu axes. Tato vlastnost je vektorem identifikátorů objektů, které jsou zobrazeny v objektu axes. Dětmi objektu axes jsou objekty image, line, patch, surface a text.

Clim `[cmin,cmax]`

Rozsah barevné osy. Tato vlastnost určuje, jakým způsobem transformuje MATLAB datové hodnoty na jednotlivé položky z mapy barev. Implicitně přiřazuje MATLAB minimální datové hodnotě první položku mapy barev a maximální hodnotě dat poslední položku mapy barev. V tomto případě tedy využije MATLAB pro plný rozsah dat úplný rozsah mapy barev.

Transformaci můžeme změnit nastavením `cmin` na takovou datovou hodnotu, které chceme přiřadit první záznam z mapy barev a nastavením `cmax` na datovou hodnotu, které přiřadíme poslední položku mapy barev. Potom data, jejichž hodnoty jsou menší než `cmin` nebo větší než `cmax` nejsou zobrazena (jsou průhledná). Od verze MATLABu 4.2 jsou data, jejichž hodnoty jsou menší než `cmin`, zobrazana, jakoby měla hodnotu `cmin`; tj. jsou kreslena první barvou z mapy barev, a data, jejichž hodnota je větší než `cmax`, zobrazena, jakoby měla hodnotu `cmax`; tj. jsou kreslena poslední barvou z mapy barev. Určíme-li hodnoty `cmin` a `cmax` tak, že leží vně rozsahu našich dat, potom MATLAB použije pro zobrazení dat pouze omezenou část mapy barev.

Tato vlastnost ovlivňuje vyobrazení objektů surface a patch, ale ne objektů image, line nebo text.

CLimMode `auto | manual`

Režim rozsahu barevné osy. Je-li `CLimMode auto`, počítá MATLAB meze barevné osy podle plného rozsahu datových hodnot dětí objektu axes. Je-li `CLimMode` nastaven na `manual`, nejsou meze barevné osy automaticky měněny (viz vlastnost `CLim`). Nastavení hodnot pro `CLim` nastaví tuto vlastnost na `manual`.

Clipping {on} | off

Režim ořezávání. V objektu axes má vždy hodnotu on.

Color *ColorSpec* | none

Barva obdélníku objektu axes. Tato vlastnost určuje barvu, kterou je vyplněn obdélník z definice Position v objektu axes. Barvu můžeme zadat RGB vektorem nebo jedním z předdefinovaných jmen. Více informací o zadávání barev je uvedeno u popisu ColorSpec. Tato barva je implicitně stejná jako barva pozadí objektu figure.

ColorOrder *matice RGB hodnot typu* $(m, 3)$

Uspořádání barev. Tato vlastnost definuje RGB hodnotami m barev. Není-li funkcemi plot a plot3 definována barva čáry, je pro každé samostatné volání těchto funkcí dosazena postupně barva z tohoto seznamu (cyklicky). ColorOrder obsahuje implicitně prvních šest barev z předdefinované palety barev v následujícím pořadí:

1. žlutá
2. fialová
3. tyrkysová
4. červená
5. zelená
6. modrá

CurrentPoint *matice typu* $(2, 3)$

Tato vlastnost obsahuje souřadnice dvou bodů, které jsou definovány pozicí kurzoru. MATLAB spojitě aktualizuje tuto vlastnost. Vlastnost CurrentPoint na úrovni objektu axes je získána z vlastnosti objektu figure CurrentPoint převodem do souřadnic objektu axes.

Kurzory existují ve dvojrozměrném prostoru obrazovky, zatímco grafické objekty MATLABu ve trojrozměrném prostoru. Aby se vyrovnal tento rozdíl, vrací MATLAB čáru kolmou na rovinu obrazovky procházející tímto bodem. Poskytuje prostorové souřadnice průsečíků čáry s přední a zadní plochou objemu vymezeného rozsahem os x, y a z. Vrácená matice má tvar

$$\begin{bmatrix} x_{zadn} & y_{zadn} & z_{zadn} \\ x_{pedn} & y_{pedn} & z_{pedn} \end{bmatrix}$$

Souřadnice jsou získány z datového prostoru aktuálních os, tj. ve stejných jednotkách, v jakých jsou data v aktuálním objektu axes kreslena. Kurzor nemusí být uvnitř os, ani uvnitř grafického okna, souřadnice jsou vráceny vzhledem k požadovaným osám bez ohledu na jeho polohu.

Následující příklad ukazuje podstatu dat vrácených vlastností `CurrentPoint`. Je poučné tento příklad vyzkoušet.

Nejprve vytvoříme 2D graf sinové vlny (nebo libovolný jiný graf 2D):

```
t=0:pi/20:2*pi;
plot(sin(t))
```

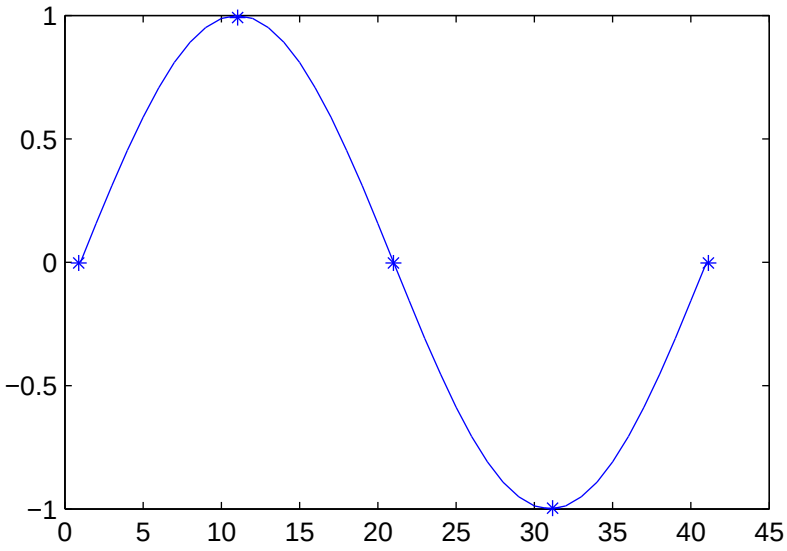
Dále nastavíme `hold on`, abychom mohli do stejných os přidat další data a použijeme příkaz `axis` ke zmrazení měřítka aktuálního rozsahu os:

```
hold on
axis(axis)
```

Nyní definujme funkci `WindowButtonDownFcn`, která obnoví a vykreslí data získaná vlastností `CurrentPoint`.

```
set(gcf,'WindowButtonDownFcn',...
    'p=get(gca,''CurrentPoint'');plot3(p(:,1),p(:,2),p(:,3),''*'');
     plot3(p(:,1),p(:,2),p(:,3),'':'')')
```

Stisknutím tlačítka myši kdekoliv na obrazovce vyvoláme funkci `WindowButtonDownFcn`. Stiskneme-li tlačítko myši, objeví se v grafu v pozici `CurrentPoint` značka `*`. (Vidíme vlastně přední koncový bod.)



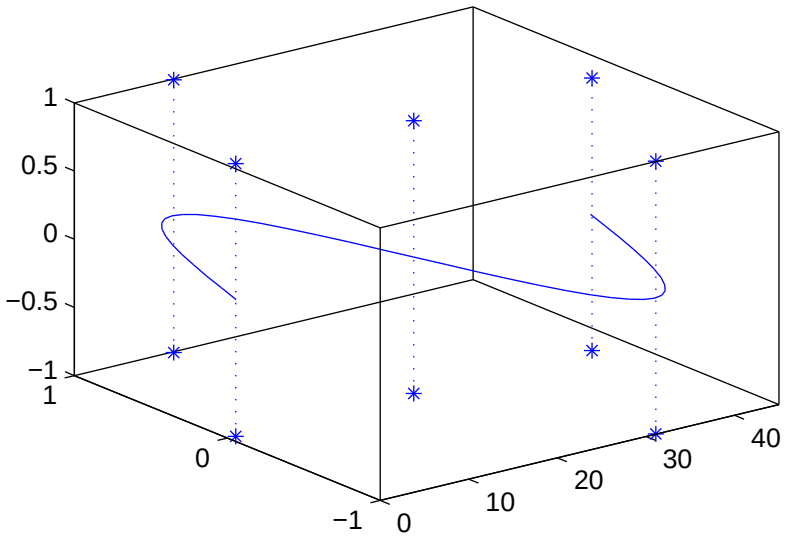
Nyní změníme pohled na 3D:

```
view(3)
```

Z jiného bodu pohledu nyní vidíme, že jsou vykresleny dva koncové body.

V tomto příkladě leží dva koncové body v rovinách $z = 1$ a $z = -1$. V obecnějších případech nemusí body získané vlastností `CurrentPoint` nutně ležet mezi osami, mohou mít libovolnou orientaci.

Abychom to ukázaly, stačí stisknout tlačítko myši v době, kdy je nastaven pohled 3D. Opět je poloha bodů `CurrentPoint` zobrazena samostatnými značkami, protože se díváme podél čáry, kterou definují. Ale nyní tato čára není rovnoběžná s žádnou osou x, y nebo z. Změníme-li opět pohled, uvidíme čáry definované koncovými body.



Této vlastnosti se využívá především tehdy, je-li pohled nastaven v libovolné orientaci a potřebujeme-li určit, který objekt čára protíná jako první . Porovnáním x-ových, y-ových a z-ových dat všech objektů v osách vidíme, zda protínají čáru definovanou koncovými body a kde ji protínají.

DrawMode `normal | fast`

Režim kreslení. Tato vlastnost umožňuje potlačit 3D třídění, které je obvykle v MATLABu prováděno. Výsledkem je rychlejší zobrazení. Je-li tato vlastnost nastavena na `fast`, kreslí MATLAB objekty v takovém pořadí, v jakém jsme je původně určili bez ohledu na to, kde jsou objekty v 3D umístěny. Je-li třídění v režimu `normal`, objekty jsou kresleny v aktuálním pohledu odzadu dopředu.

FontAngle {`normal`} | `italic` | `oblique`

Sklon písma. Tato vlastnost určuje sklon písma.

FontName *rodina fontů*

Rodina fontů. Tato vlastnost určuje rodinu fontů, např. Helvetica.

FontSize *velikost v bodech*

Velikost fontu. Tato vlastnost specifikuje velikost fontu v bodech (1 bod = 1/72 palce).

FontStrikeThrough `on | {off}`

Přeškrtnutí písma. Tato vlastnost určuje, zda je písmo přeškrtnuto.

FontUnderline `on | {off}`

Podtržení písma. Tato vlastnost určuje, zda je písmo podtrženo.

FontWeight `light | {normal} | demi | bold`

Světlost písma. Tato vlastnost určuje charakter váhy (světlost písma).

Font je kromě jména definován též dalšími charakteristikami, které se přidávají za jméno fontu. Ne všechny kombinace jsou však povoleny. MATLAB běžně podporuje čtyři rodiny fontů (Times, Helvetica, Courier a Symbol) a následujících 13 fontů:

Times-Roman	Helvetica	Courier
Times-Italic	*Helvetica-Oblique*	*Courier-Oblique*
Times-Bold	**Helvetica-Bold**	**Courier-Bold**
Times-BoldItalic	***Helvetica-BoldOblique***	***Courier-BoldOblique***
	Symbol	

Například, chceme-li používat 10-ti bodovou Helveticu (BoldOblique), musíme nastavit vlastnosti fontu následovně:

`FontName`	Helvetica
`FontSize`	10
`FontWeight`	bold
`FontAngle`	oblique

Pokud množina současně specifikovaných parametrů neodpovídá žádnému využitelnému fontu, používá MATLAB pro určení aktuálního fontu následující algoritmus:

1. MATLAB akceptuje oblique namísto italic.
2. Pokud není shoda stále nalezena, MATLAB ignoruje `FontAngle`.
3. Pokud není shoda stále nalezena, MATLAB ignoruje `FontWeght`.
4. Pokud není shoda stále nalezena, MATLAB ignoruje `FontSize`.
5. Pokud není shoda stále nalezena, MATLAB font nezmění.

Když MATLAB generuje tiskový výstup, nepokouší se před odesláním tohoto výstupu na dané tiskové zařízení zjistit, jaké fonty jsou na zařízení využitelné.

Implicitním fontem pro textové objekty a popis os je 12-ti bodová Helvetica. Pokud se používají TrueType fonty a Times i Helvetica jsou nevyužitelné, Times je nahrazen fontem NewTimesRoman, Helvetica je nahrazena fontem Arial a Courier fontem NewCourier.

GridLineStyle *symbol typu čáry*

Typ čar sítě. Tato vlastnost určuje typ čáry, kterým se vykreslí čáry sítě. Můžeme si vybrat typ čáry z následujícího seznamu stylů. Požadovaný typ čáry se určí pomocí uvedených symbolů, např.

```
set(gca,'GridLineStyle','-')
```

kde

`-`	plná čára,
`--`	čárkovaná čára,
`:`	tečkovaná čára,
`-.`	čerchovaná.

Interruptible `yes | {no}`

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv.

Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím zpětného volání pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

LineStyleOrder *matice řetězců* nebo *řetězec*

Uspořádání typů čar. Tato vlastnost definuje jaké typy čar (např. plná, čárkovaná, atd) se mají používat a v jakém pořadí se mají používat při vykreslování více průběhů. Není-li funkcemi `plot` a `plot3` definován typ čáry, je pro každé samostatné volání těchto funkcí dosazen postupně typ z tohoto seznamu (cyklicky). Např. pro použití plné, čárkované a tečkované čáry v tomto pořadí zadáme

```
set(gca,'LineStyleOrder',['- ','--',': '])
```

nebo

```
set(gca,'LineStyleOrder','-|--|:')
```

POZOR! V případě prvního způsobu musí být všechny řetězce stejně dlouhé.

`LineStyleOrder` má implicitní hodnotu `''`, což znamená, že všechna data budou vykreslena plnou čarou. K rozlišení jednotlivých průběhů se používá raději barev než různých typů čar.

LineWidth *šířka*

Šířka čáry. Tato vlastnost umožňuje určit tloušťku čar, které jsou použity k zobrazení os a vynášecích čárek. Nová šířka je určena v bodech (1 bod = 1/72 palce). Implicitní hodnota je 0.5 bodu.

NextPlot `new | add | {replace}`

Jak přidat další graf. Tato vlastnost říká vestavěným grafickým funkcím vyšší úrovně (`plot`, `plot3`, `fill`, `fill3`) a grafickým funkcím ve formě m-souborů (`mesh`, `surf`, `bar`, atd.), jaký objekt axes mají použít. Parametr `new` znamená, že bude před kreslením vytvořen nový objekt axes, parametrem `add` bude do aktuálního objektu axes

přidán nový objekt. Implicitně nastavený parametr `replace` znamená, že bude před kreslením nejprve zrušen aktuální objekt axes a na stejném místě bude vytvořen nový. Tato vlastnost je použita k provedení příkazu `hold`. Je-li `hold off`, je nastaven `NextPlot` na `replace`. Nastavení `hold` na `on` změní `NextPLot` na `add`.

Úvodní funkcí pro manipulaci s vlastností `NextPlot` je m-soubor `newplot`. Vyvoláním funkce `newplot` v grafických m-funkcích jako `mesh`, `surf`, `bar`, atd. se před kreslením grafů provedou v závislosti na nastavení vlastnosti `NextPlot` požadované akce. Při psaní vlastních grafických příkazů bychom měli volat funkci `newplot` hned na začátku. (Viz např. m-soubor `pcolor`). Viz též vlastnost `NextPlot` pro objekty figure.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu axes. Tato vlastnost je identifikátorem rodiče objektu axes. Rodičem objektu axes je objektu figure, ve kterém je objekt axes zobrazen. Identifikátor rodiče pro aktuální objekt axes dává také funkce `gcf`.

Position *4-prvkový vektor*

Umístění a velikost objektu axes. Tato vlastnost je obdélník, který specifikuje velikost a umístění objektu axes uvnitř grafického okna. Obdélník je určen vektorem

```
rect=[zleva zdola šířka výška]
```

kde `zleva` a `zdola` definují vzdálenost levého dolního rohu obdélníku od levého dolního rohu grafického okna, `šířka` a `výška` určují velikost obdélníku. U popisu vlastnosti `Units` jsou pro tuto specifikaci uvedeny informace o jednotkách.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
set(gca,'Tag','Pokusná osa')
```

označí aktuální osu visačkou `'Pokusná osa'`. Je-li potom potřeba se někdy na tuto osu odkázat, lze její identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','Pokusná osa')
```

TickLength `[2Dticklength 3Dticklength]`

Délka vynášecích čárek na osách. Tato vlastnost určuje délku vynášecích čárek v normalizovaných jednotkách vzhledem k velikosti obdélníku definovaného objektem axes. Definujeme-li délku vynášecích čárek 0.1, vykreslí se čárky délky 1/10 šířky nebo délky obdélníka. Vektor o dvou prvcích určuje délku vynášecích čárek pro 2D a 3D pohledy. Implicitní hodnoty jsou [0.010.025].

TickDir `{in} | out`

Směr vynášecích čárek. Tato vlastnost definuje, zda vynášecí čárky směřují dovnitř nebo ven z individuálních osových čar. Implicitně je nastaveno `in`, ale 3D grafické funkce nastavují směr na `out` (kromě případu, kdy je `hold on`).

Title *identifikátor textu*

Identifikátor nadpisu.. Tato vlastnost obsahuje identifikátor textového objektu, který je zobrazen jako nadpis objektu axes. Tohoto identifikátoru můžeme použít při změnách vlastností nadpisu nebo pro vytvoření nadpisu objektu axes.

Např. následující příkaz změní barvu nadpisu aktuálního objektu axes na červenou:

```
set(get(gca,'Title'),'Color','r')
```

Pro vytvoření nadpisu nastavíme vlastnost `Title` na identifikátor textu, který chceme použít jako nadpis:

```
set(gca,'Title',text('String','Naměřená data'))
```

Obecně je pro vytvoření nadpisu jednodušší použít příkaz `title`.

Type *řetězec (pouze pro čtení)*

Typ grafického objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekt axes je `Type` vždy řetězec `'axes'`.

Units `pixels | {normalized} | inches | centimeters | points`

Použité jednotky. Tato vlastnost určuje jednotky použité pro interpretaci vlastnosti `Position`. Všechny jednotky jsou počítány z levého dolního rohu grafického okna. U normalizovaných jednotek odpovídá levý dolní roh hodnotě $(0, 0)$ a horní pravý roh hodnotě $(1, 1)$. Palce, centimetry a body jsou absolutními jednotkami (1 bod = 1/72 palce).

Pokud chceme změnit hodnotu `Units`, je dobrým zvykem, vrátit ji po provedení našich výpočtů na její implicitní hodnotu, aby tato změna neovlivnila další funkce, které předpokládají implicitní nastavení `Units`.

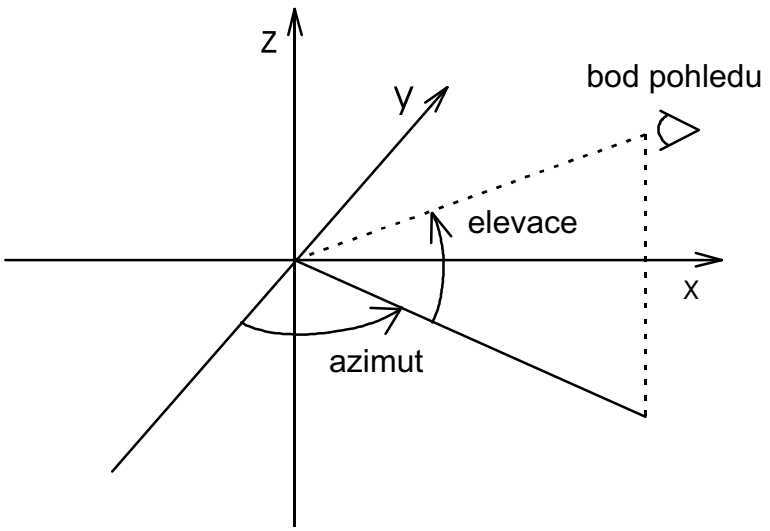
UserData *matice*

Data specifikovaná uživatelem. `UserData` může být libovolná matice, kterou chceme objektu přiřadit. Objekt tato data nepoužije, ale my je můžeme získat pomocí funkce `get`.

View `[az, el]`

Bod pohledu na objekt axes. Tato vlastnost zavádí bod pohledu, který se používá k transformaci 3D grafu na plochu obrazovky. Bod pohledu je místo pozorovatelova oka, které se dívá na 3D graf. Umístění bodu pohledu je určeno azimutem (`az`) a elevací (`el`).

Azimut obíhá kolem z-ové osy s kladnými hodnotami ve směru proti pohybu hodinových ručiček. Elevace určuje úhel nad resp. pod objektem. Kladné hodnoty elevace jsou hodnoty nad objektem, kdy se díváme na objekt shora, záporné hodnoty elevace způsobí, že je bod pohledu pod objektem. Hodnoty obou úhlů jsou ve stupních.



Změnou vlastnosti `View` se změní transformační matice u vlastnosti `XForm`.

Visible {`on`} | `off`

Viditelnost objektu axes. Tato vlastnost určuje, zda je či není objekt zobrazen na obrazovce. Je-li vlastnost `Visible` pro objekt axes `off`, pak nejsou kresleny osové čáry, vynášecí čárky a popisy os. Děti objektu axes ale nejsou vlastností `Visible` u svých rodičů ovlivněny.

XForm *matice řádu 4*

Transformační matice pohledu. Tato vlastnost je transformační matice používaná k transformaci 3D grafu na plochu obrazovky. Transformační matici můžeme určit buď přímo touto vlastností, anebo nepřímo zadáním azimutu a elevace bodu pohledu pomocí vlastnosti `View`.

Zadání nových hodnot pro vlastnost `View` změní též tuto matici. Definujeme-li bod pohledu pomocí matice `XForm`, můžeme měnit nejen azimut a elevaci, ale také perspektivu pohledu a bod, na který se v zadaném objektu díváme. Podrobněji je transformační matice popsána u funkce `viewmtx`.

Vlastnosti ovládající x-ovou osu

XColor *ColorSpec*

Barva xové osy. Tato vlastnost nastaví barvu x-ové osy, vertikálních vynášecích čárek, jejich popisů a čar sítě u x-ové osy.

XDir {`normal`} | `reverse`

Směr rostoucích hodnot na x-ové ose. Objekt axes vytváří pravotočivý souřadný systém. V implicitním pohledu rostou hodnoty x zleva doprava, nastavením vlastnosti na hodnotu `reverse` tento směr obrátíme.

XGrid `on | {off}`

Režim čar sítě x-ové osy. Je-li `XGrid on`, kreslí MATLAB v každé vynášecí čárce kolmo na x-ovou osu čáry sítě, tj. čáry s konstantní hodnotou x.

XLabel *identifikátor textu*

Popis x-ové osy. Tato vlastnost je identifikátor objektu text, který popisuje x-ovou osu. Pro popis x-ové osy musíme vytvořit textový objekt, abychom obdrželi jeho identifikátor. To provádí zároveň s určením popisu následující příkaz

```
set(gca,'xlabel',text(0,0,'axis label'))
```

Zatímco funkce `text` vyžaduje zadání pozice textového řetězce, není v tomto případě tato pozice pro umístění textu použita. MATLAB sám umístí řetězec `'axis label'` na vhodné místo pod x-ovou osu.

Textový objekt můžeme definovat také na libovolném místě a získat přímo jeho identifikátor. V tomto případě pak MATLAB přemístí textový řetězec na správnou pozici pro popis x-ové osy. Jednoduchý způsob popisu x-ové osy provádí funkce `xlabel`.

XLim `[xmin xmax]`

Meze x-ové osy. Tato vlastnost je vektor o dvou prvcích, které určují minimální a maximální hodnotu x-ové osy. Změna `XLim` má vliv jak na rozsah x-ové osy, tak na umístění popisu osy a vynášecích čárek.

XLimMode `{auto} | manual`

Režim mezí x-ové osy. Je-li `XLimMode` nastavena na `auto`, počítá MATLAB meze x-ové osy (`XLim`) podle rozpětí hodnot `XData`, která příslušejí dětem objektu axes, a výsledek zaokrouhlí. Je-li `XLimMode manual`, jsou meze x-ové osy určeny vlastností `XLim` a nezávisejí na `XData` v objektech dětí. Nastavení hodnot pro `XLim` nastaví tuto vlastnost na `manual`.

XScale `{linear} | log`

Měřítko x-ové osy. Tato vlastnost určuje buď lineární nebo logaritmickou stupnici x-ové osy.

XTick *vektor*

Odstup vynášecích čárek na x-ové ose. Tato vlastnost řídí existenci a umístění vynášecích čárek na x-ové ose. Místa na x-ové ose, ve kterých chceme umístit vynášecí čárky, zapíšeme do vektoru `XTick`.

Nechceme-li čárky zobrazit, zadáme do `XTick` prázdný vektor `[ ]`.

XTickMode `{auto} | manual`

Režim vynášecích čárek x-ové osy. Je-li `XTickMode` nastaveno na `auto`, počítá MATLAB odstup vynášecích čárek x-ové osy (`XTick`) podle rozpětí hodnot `XData`, které příslušejí dětem objektu axes. Je-li `XTickMode manual`, jsou meze x-ové osy určeny

vlastností `XTick` a nezávisejí na `XData` v objektech dětí. Nastavení hodnot pro `XTick` nastaví tuto vlastnost na `manual`.

XTickLabels *matice řetězců* nebo *řetězec* nebo *vektor čísel*

Popis vynášecích čárek. Tato vlastnost určuje text, kterým jsou označeny vynášecí čárky na x-ové ose. Tyto popisy nahrazují numerické popisy, které jsou generovány MATLABem. Není-li určen dostatek textových popisů, použije MATLAB všechny definované popisy, a zbývající vynášecí čárky značí opět od začátku již definovanými popisy (cyklicky). Následující příkaz např. označí první dvě vynášecí čárky x-ové osy popisem `'Stará data'` a `'Nová data'`.

```
set(gca,'XTickLabels',['Stará data'; 'Nová data'])
```

POZOR! Všechny znakové řetězece musí mít shodný počet znaků, protože MATLAB je ukládá do matic.

Od verze MATLABu 4.2 je možno používat jako parametru též řetězec, kde jsou jednotlivé položky odděleny svislou čárkou `'|'`, např.

```
set(gca,'XTickLabels','psi|kočky|ptáci')
```

nebo vektor čísel, např.

```
set(gca,'XTickLabels',[1:0.5:7])
```

Tato vlastnost neovlivňuje ani počet vynášecích čárek ani jejich umístění.

XTickLabelMode {`auto`} | `manual`

Režim popisu vynášecích čárek x-ové osy. Je-li tato vlastnost `auto`, počítá MATLAB popis vynášecích čárek x-ové osy (`XTickLabels`) podle rozpětí hodnot `XData`, která příslušejí dětem objektu axes. Je-li `XTickMode manual`, jsou popisy vynášecích čárek x-ové osy určeny vlastností `XTickLabels` a nezávisí na `XData` v objektech dětí. Nastavení hodnot pro `XTickLabels` nastaví tuto vlastnost na `manual`.

Vlastnosti ovládají *y*-ovou osu

YColor *ColorSpec*

Barva y-ové osy. Tato vlastnost nastaví barvu y-ové osy, vynášecích čárek, jejich popisů a čar sítě u y-ové osy.

YDir {`normal`} | `reverse`

Směr rostoucích hodnot na y-ové ose. Objekt axes vytváří pravotočivý souřadný systém. V implicitním pohledu rostou hodnoty y zdola nahoru, nastavením vlastnosti na hodnotu `reverse` tento směr obrátíme.

YGrid `on` | {`off`}

Režim čar sítě y-ové osy. Je-li `YGrid on`, kreslí MATLAB v každé vynášecí čárce kolmo na y-ovou osu čáry sítě, tj. čáry s konstantní hodnotou y.

YLabel *identifikátor textu*

Popis y-ové osy. Tato vlastnost je identifikátor objektu text, který popisuje y-ovou osu. Pro popis osy musíme vytvořit textový objekt, abychom obdrželi jeho identifikátor. To provádí zároveň s určením popisu následující příkaz

```
set(gca,'ylabel',text(0,0,'axis label'))
```

Zatímco funkce `text` vyžaduje zadání pozice textového řetězce, není v tomto případě tato pozice pro umístění textu použita. MATLAB sám umístí řetězec `'axis label'` na vhodné místo vedle y-ové osy.

Textový objekt můžeme definovat také na libovolném místě a získat přímo jeho identifikátor. V tomto případě pak MATLAB přemístí textový řetězec na správnou pozici pro popis y-ové osy. Jednoduchý způsob popisu y-ové osy provádí funkce `ylabel`.

YLim `[ymin ymax]`

Meze y-ové osy. Tato vlastnost je vektor o dvou prvcích, které určují minimální a maximální hodnotu y-ové osy. Změna `YLim` má vliv jak na rozsah y-ové osy, tak na umístění popisu osy a vynášecích čárek.

YLimMode {`auto`} | `manual`

Režim mezí y-ové osy. Je-li `YLimMode` nastavena na `auto`, počítá MATLAB meze y-ové osy (`YLim`) podle rozpětí hodnot `YData`, která příslušejí dětem objektu axes, a výsledek zaokrouhlí. Je-li `YLimMode manual`, jsou meze y-ové osy určeny vlastností `YLim` a nezávisejí na `YData` v objektech dětí. Nastavení hodnot pro `YLim` nastaví tuto vlastnost na `manual`.

YScale {`linear`} | `log`

Měřítko y-ové osy. Tato vlastnost určuje buď lineární nebo logaritmickou stupnici y-ové osy.

YTick *vektor*

Odstup vynášecích čárek na y-ové ose. Tato vlastnost řídí existenci a umístění vynášecích čárek na y-ové ose. Místa na y-ové ose, ve kterých chceme umístit vynášecí čárky, zapíšeme do vektoru `YTick`.

Nechceme-li čárky zobrazit, zadáme do `YTick` prázdný vektor `[ ]`.

YTickMode {`auto`} | `manual`

Režim vynášecích čárek y-ové osy. Je-li `YTickMode` nastaveno na `auto`, počítá MATLAB odstup vynášecích čárek y-ové osy (`YTick`) podle rozpětí hodnot `YData`, která příslušejí dětem objektu axes. Je-li `YTickMode manual`, jsou meze y-ové osy určeny vlastností `YTick` a nezávisí na `YData` v objektech dětí. Nastavení hodnot pro `YTick` nastaví tuto vlastnost na `manual`.

YTickLabels *matice řetězců* nebo *řetězec* nebo *vektor čísel*

Popis vynášecích čárek. Tato vlastnost určuje text, kterým jsou označeny vynášecí čárky na y-ové ose. Tyto popisy nahrazují numerické popisy, které jsou generovány MATLABem. Není-li určen dostatek textových popisů, použije MATLAB všechny definované popisy, a zbývající vynášecí čárky značí opět od začátku již definovanými popisy (cyklicky). Následující příkaz např. označí první dvě vynášecí čárky y-ové osy popisem `'Stará data'` a `'Nová data'`.

```
set(gca,'YTickLabels',['Stará data'; 'Nová data'])
```

POZOR! Všechny znakové řetězece musí mít shodný počet znaků, protože MATLAB je ukládá do matic.

Od verze MATLABu 4.2 je možno používat jako parametru též řetězec, kde jsou jednotlivé položky odděleny svislou čárkou `'|'`, např.

```
set(gca,'YTickLabels','psi|kočky|ptáci')
```

nebo vektor čísel, např.

```
set(gca,'YTickLabels',[1:0.5:7])
```

Tato vlastnost neovlivňuje ani počet vynášecích čárek ani jejich umístění.

YTickLabelMode {`auto`} | `manual`

Režim popisu vynášecích čárek y-ové osy. Je-li tato vlastnost `auto`, počítá MATLAB popis vynášecích čárek y-ové osy (`YTickLabels`) podle rozpětí hodnot `YData`, které příslušejí dětem objektu axes. Je-li `YTickMode manual`, jsou popisy vynášecích čárek y-ové osy určeny vlastností `YTickLabels` a nezávisí na `YData` v objektech dětí. Nastavení hodnot pro `YTickLabels` nastaví tuto vlastnost na `manual`.

Vlastnosti ovládající *z*-ovou osu

ZColor *ColorSpec*

Barva z-ové osy. Tato vlastnost nastaví barvu z-ové osy, vynášecích čárek, jejich popisů a čar sítě u z-ové osy.

ZDir {`normal`} | `reverse`

Směr rostoucích hodnot na z-ové ose. Objekt axes vytváří pravotočivý souřadný systém. V implicitním pohledu rostou hodnoty z zdola nahoru, nastavením vlastnosti na hodnotu `reverse` tento směr obrátíme.

ZGrid `on` | {`off`}

Režim čar sítě z-ové osy. Je-li `Zgrid on`, kreslí MATLAB v každé vynášecí čárce kolmo na z-ovou osu čáry sítě, tj. čáry s konstantní hodnotou z.

ZLabel *identifikátor textu*

Popis z-ové osy. Tato vlastnost je identifikátor objektu text, který popisuje z-ovou osu. Pro popis osy musíme vytvořit textový objekt, abychom obdrželi jeho identifikátor. To provádí zároveň s určením popisu následující příkaz

```
set(gca,'zlabel',text(0,0,'axis label'))
```

Zatímco funkce text vyžaduje zadání pozice textového řetězce, není v tomto případě tato pozice pro umístění textu použita. MATLAB sám umístí řetězec `'axis label'` na vhodné místo vedle z-ové osy.

Textový objekt můžeme definovat také na libovolném místě a získat přímo jeho identifikátor. V tomto případě pak MATLAB přemístí textový řetězec na správnou pozici pro popis z-ové osy. Jednoduchý způsob popisu z-ové osy provádí funkce `zlabel`.

ZLim `[zmin zmax]`

Meze z-ové osy. Tato vlastnost je vektor o dvou prvcích, které určují minimální a maximální hodnotu z-ové osy. Změna `ZLim` má vliv jak na rozsah z-ové osy, tak na umístění popisu osy a vynášecích čárek.

ZLimMode {`auto`} | `manual`

Režim mezí z-ové osy. Je-li `ZLimMode` nastavena na `auto`, počítá MATLAB meze z-ové osy (`ZLim`) podle rozpětí hodnot `ZData`, která příslušejí dětem objektu axes, a výsledek zaokrouhlí. Je-li `ZLimMode manual`, jsou meze z-ové osy určeny vlastností `ZLim` a nezávisí na `ZData` v objektech dětí. Nastavení hodnot pro `ZLim` nastaví tuto vlastnost na `manual`.

ZScale {`linear`} | `log`

Měřítko z-ové osy. Tato vlastnost určuje buď lineární nebo logaritmickou stupnici z-ové osy.

ZTick *vektor*

Odstup vynášecích čárek na z-ové ose. Tato vlastnost řídí existenci a umístění vynášecích čárek na z-ové ose. Místa na z-ové ose, ve kterých chceme umístit vynášecí čárky, zapíšeme do vektoru `ZTick`.

Nechceme-li čárky zobrazit, zadáme do `ZTick` prázdný vektor `[ ]`.

ZTickMode {`auto`} | `manual`

Režim vynášecích čárek z-ové osy. Je-li `ZTickMode` nastaveno na `auto`, počítá MATLAB odstup vynášecích čárek z-ové osy (`ZTick`) podle rozpětí hodnot `ZData`, která příslušejí dětem objektu axes. Je-li `ZTickMode manual`, jsou meze z-ové osy určeny vlastností `ZTick` a nezávisí na `ZData` v objektech dětí. Nastavení hodnot pro `ZTick` nastaví tuto vlastnost na `manual`.

ZTickLabels *matice řetězců* nebo *řetězec* nebo *vektor čísel*

Popis vynášecích čárek. Tato vlastnost určuje text, kterým jsou označeny vynášecí čárky na z-ové ose. Tyto popisy nahrazují numerické popisy, které jsou generovány MATLABem. Není-li určen dostatek textových popisů, použije MATLAB všechny definované popisy, a zbývající vynášecí čárky značí opět od začátku již definovanými popisy (cyklicky). Následující příkaz např. označí první dvě vynášecí čárky z-ové osy popisem `'Stará data'` a `'Nová data'`.

```
set(gca,'ZTickLabels',['Stará data'; 'Nová data'])
```

POZOR! Všechny znakové řetězece musí mít shodný počet znaků, protože MATLAB je ukládá do matic.

Od verze MATLABu 4.2 je možno používat jako parametru též řetězec, kde jsou jednotlivé položky odděleny svislou čárkou `'|'`, např.

```
set(gca,'ZTickLabels','psi|kočky|ptáci')
```

nebo vektor čísel, např.

```
set(gca,'ZTickLabels',[1:0.5:7])
```

Tato vlastnost neovlivní ani počet vynášecích čárek ani jejich umístění.

ZTickLabelMode {`auto`} | `manual`

Režim popisu vynášecích čárek z-ové osy. Je-li tato vlastnost `auto`, počítá MATLAB popis vynášecích čárek z-ové osy (`ZTickLabels`) podle rozpětí hodnot `ZData`, která příslušejí dětem objektu axes. Je-li `ZTickMode manual`, jsou popisy vynášecích čárek z-ové osy určeny vlastností `ZTickLabels` a nezávisí na `ZData` v objektech dětí. Nastavení hodnot pro `ZTickLabels` nastaví tuto vlastnost na `manual`.

Viz též `axis`, `subplot`, `figure`, `gca`, `clf`, `cla`

Funkce Rozsah a vzhled os.

Syntaxe

```
axis([xmin xmax ymin ymax])
axis([xmin xmax ymin ymax zmin zmax])
axis('auto')
axis(axis)
v=axis
axis('ij')
axis('xy')
axis('square')
axis('equal')
axis('normal')
axis('off')
axis('on')
[s1,s2,s3]=axis('state')
```

Popis `axis` poskytuje jednoduchý způsob, jak pracovat s většinou vlastností objektu axes.

`axis([xmin xmax ymin ymax])` nastaví rozsah aktuálních os x a y.

`axis([xmin xmax ymin ymax zmin zmax])` nastaví rozsah aktuálních os x, y a z.

`axis('auto')` nastaví implicitní režim automatického nastavení mezí os, tj. optimální meze os jsou vypočítány automaticky.

`axis(axis)` zmrazí aktuální hodnotu nastaveného rozsahu os tak, aby následné grafy mohly použít stejný rozsah os, je-li `hold` zapnuto na `on`.

`v=axis` vrátí řádkový vektor, který obsahuje rozsah aktuálních os. Pro 2D graf má vektor `v` čtyři prvky, u prostorového grafu má šest prvků.

`axis('ij')` překreslí graf do maticových souřadnic. Počátek souřadnic nyní leží v levém horním rohu. Osa `i` je svislá osa a je číslována shora dolů. Vodorovná osa je osa `j` a je číslována zleva doprava.

`axis('xy')` vrátí graf do implicitního kartézského souřadného systému s počátkem v levém dolním rohu. Osa `x` je vodorovná a je číslována zleva doprava, osa `y` je svislá a je číslována zdola nahoru.

`axis('square')` vytvoří čtvercovou oblast pro aktuální objekt axes.

`axis('equal')` udává, že měřítkovací faktor a přírůstek osových značek je shodný pro x-ovou osu i y-ovou osu.

`axis('normal')` obnoví původní velikost aktuálního objektu axes a odstraní účinky příkazů `axis('square')` a `axis('equal')`.

`axis('off')` vypne všechny popisy os a vynášecích čárek (totéž jako vlastnost `Visible off` u objektu axes).

`axis('on')` zapne všechny popisy os a vynášecích čárek (totéž jako vlastnost `Visible on` u objektu axes).

`[s1,s2,s3]=axis('state')` vrací tři řetězce, které udávají aktuální nastavení tří vlastností objektu axes.

```
s1='auto' nebo 'manual'
s2='on'   nebo 'off'
s3='xy'   nebo 'ij'
```

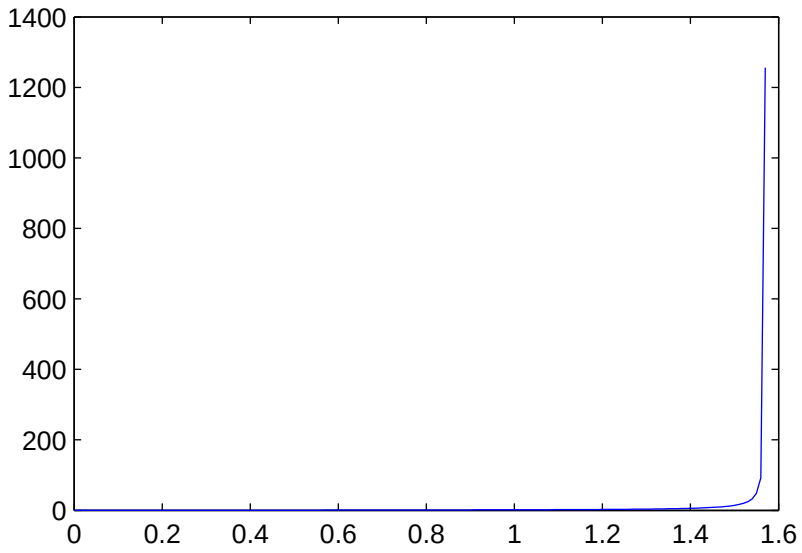
`axis(s1,s2,s3)` obnoví vlastnosti objektu axes na hodnoty udané třemi řetězci. Implicitně je

```
axis('auto','on','xy')
```

Příklady Následující dva příkazy

```
x=0:0.01:pi/2;
plot(x,tan(x))
```

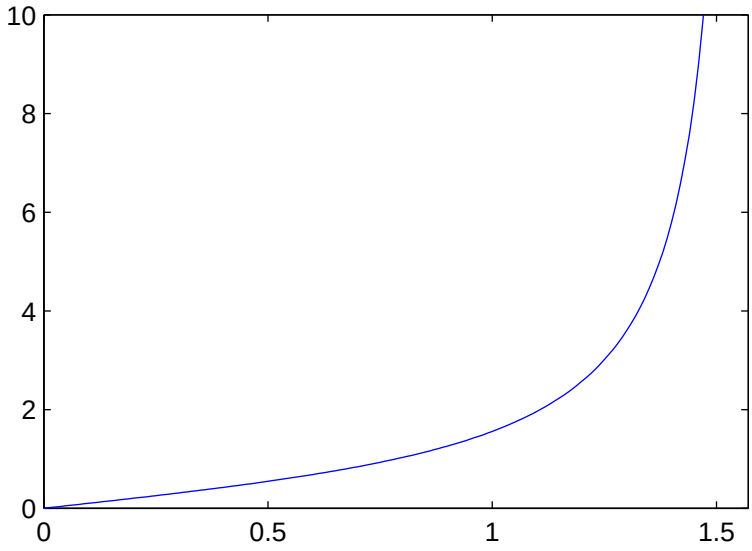
nedávají příliš uspokojivý výsledek, protože automatické nastavení rozsahu y-ové osy vychází z maximální hodnoty `ymax=tan(1.57)`, což je číslo větší než 1000.



Následuje-li za těmito příkazy příkaz

```
axis([0 pi/2 0 10])
```

dostáváme mnohem lepší výsledek.



Uvažujme matici `Z` typu $(10, 5)$ o prvcích `Z(i,j)=j/i`. Nejjednodušší způsob generování této matice je dvojnásobný cyklus `for`

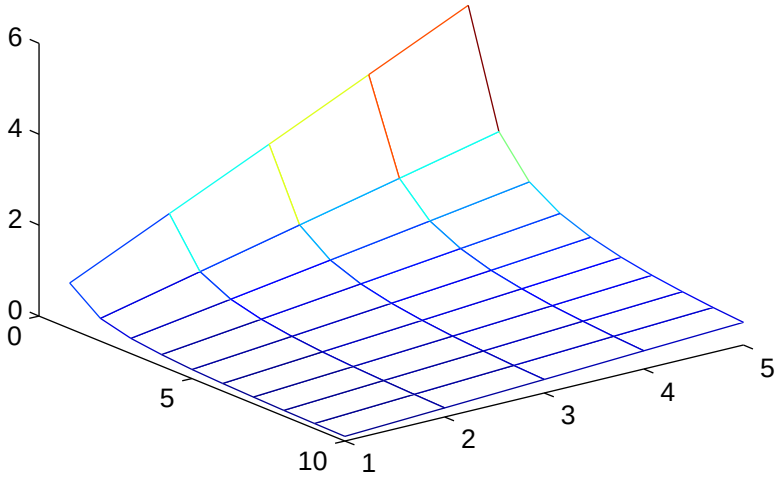
```
m=10;
n=5;
for i=1:m
  for j=1:n,
    Z(i,j)=j/i;
  end
end
```

Výsledná matice je

```
Z =

1.0000 2.0000 3.0000 4.0000 5.0000
0.5000 1.0000 1.5000 2.0000 2.5000
0.3333 0.6667 1.0000 1.3333 1.6667
. . . . . .
0.1111 0.2222 0.3333 0.4444 0.5556
0.1000 0.2000 0.3000 0.4000 0.5000
```

Matici můžeme zobrazit v maticových souřadnicích příkazy

```
mesh(Z), axis('ij')
```



Graf ukazuje 10 řádků a 5 sloupců matice `Z`, první prvek

```
Z(1,1)=1.0
```

leží vlevo nahoře, největší prvek

```
Z(1,5)=5.0
```

leží vpravo nahoře a hodnoty lineárně vzrůstají s indexem sloupců `j`.

Uvažujeme naopak funkci $z = f(x, y) = x/y$ v kartézských souřadnicích v oblasti $0 \leq x \leq 1$, $0 \leq y \leq 2$. Funkci zobrazíme na síti o 10 prvcích ve směru x-ové osy a 5 prvcích ve směru y-ové osy.

```
m=10;
n=5;
x=(1:n)/n;
y=2*(1:m)'/m;
[X,Y]=meshgrid(x,y);
Z=X./Y;
```

Vektory `x` a `y` jsou

```
x = [0.20 0.40 0.60 0.80 1.00]
```

a

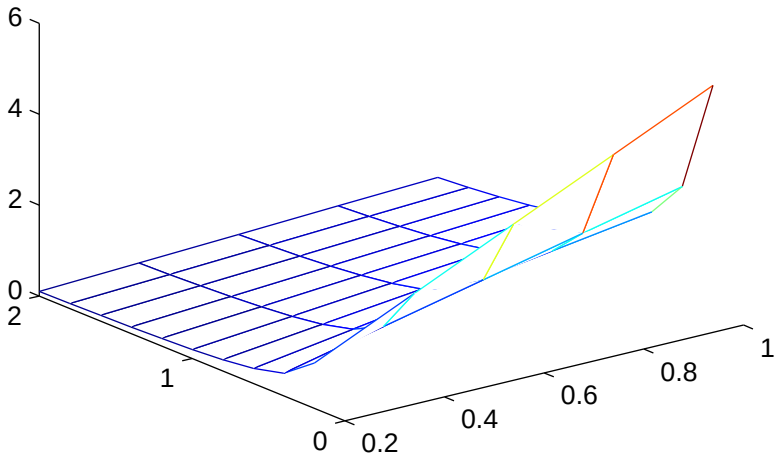
```
y = [0.20 0.40 0.60 ... 1.80 2.00]'
```

Příklad je navíc sestaven tak, že pole `Z` typu $(10, 5)$ je shodné s maticí `Z` z minulého příkladu.

Nyní jsme v kartézském souřadném systému, proto příkaz

```
mesh(x,y,Z)
```

vygeneruje graf



Implicitní kartézský souřadný systém má počátek v levém dolním rohu, maximální hodnota je umístěna v pravém dolním rohu. Navíc se liší popisy os x a y, protože jsme použili vektory `x` a `y`.

Algoritmus `axis` nastavuje vlastnosti objektu axes `XLim`, `YLim`, `ZLim`, `XLimMode`, `YLimMode`, `ZLimMode`, `YDir` a `Position`.

Viz též `axes`, `subplot`, `set`, `get`
Všechny vlastnosti objektu axes.

Funkce Sloupcový graf.

Syntaxe
```
bar(y);
bar(x,y);
[xb,yb]=bar(...)
```

Popis `bar(y)` kreslí sloupcový graf prvků vektoru `y`.

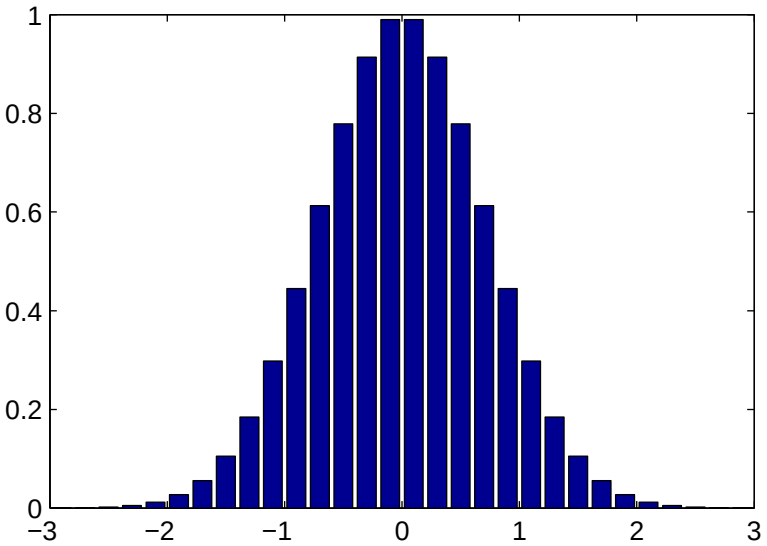
`bar(x,y)` kreslí sloupcový graf prvků vektoru `y` v místech určených vektorem `x`.

Jsou-li `x` a `y` matice téhož typu, je vykreslen sloupcový graf pro každý sloupec.

`[xb,yb]=bar(y)` a `[xb,yb]=bar(x,y)` nekreslí grafy, ale vrací vektory `xb` a `yb`, které se použijí pro generování sloupcového diagramu příkazem `plot(xb,yb)`. To je vhodné zvláště v takových případech, kdy pro generování grafu potřebujeme nastavit více vlastností objektu, např. při kombinaci sloupcového diagramu se složitějším příkazem `plot`.

Příklady Vykreslení křivky tvaru zvonu:

```
x=-2.9:0.2:2.9;
bar(x,exp(-x.*x))
```



Viz též `stairs`, `hist`

Funkce Generuje řetězec mezer.

Syntaxe `b=blanks(n)`

Popis `b=blanks(n)` generuje řetězec `b`, který obsahuje `n` mezer.

Poznámka Implementováno od verze 4.1.

Příklady Použití s funkcí `disp`

```
disp(['xxx' blanks(20) 'yyy'])
disp(blanks(n)'); % přemístí kurzor o n řádků níže
```

Viz též `clc`, `home`, `format`, `compact`

Funkce Zesvětlení nebo ztmavení mapy barev.

Syntaxe

```
brighten(beta)
map=brighten(beta)
newmap=brighten(map,beta)
```

Popis `brighten(beta)` nahradí aktuální mapu barev světlejší nabo tmavší mapou, která ale obsahuje tytéž základní barvy. Je-li `0<beta<1`, je mapa světlejší, pro `-1<beta<0` je mapa tmavší.

`brighten(beta)` následováno příkazem `brighten(-beta)` obnoví původní mapu.

`map=brighten(beta)` vrací světlejší nebo tmavší verzi aktuální mapy barev (neprojeví se na obrazovce).

`newmap=brighten(map,beta)` vrací světlejší nebo tmavší verzi zvolené mapy barev (neprojeví se na obrazovce).

Viz též `colormap`, `rgbplot`

Funkce Transformace barevné osy.

Syntaxe
```
caxis([cmin cmax])
caxis('auto')
v=caxis;
caxis(caxis)
```

Popis `caxis` umožňuje jednoduchým způsobem ovládat vlastnosti `CLim` a `CLimMode` objektů axes. Grafické funkce používající mapu barev, jako např. `mesh`, `surf`, `pcolor` a další, vytvářejí objekty surface a patch a transformují určená pole barev na mapy barev tak, aby vytvořily hodnoty barev. Tato transformace je řízena funkcí `colormap` a vlastnostmi `CLim` a `CLimMode`.

`caxis([cmin cmax])` nastavuje parametr transformace barevné osy `CLim` na předepsanou hodnotu. Všechny hodnoty v barevném poli, které leží vně intervalu `cmin` $\leq$ `c` $\leq$ `cmax` jsou ustřiženy, tj. odpovídající plošky nebo body nejsou zobrazeny. Od verze MATLABu 4.2 jsou data, jejichž hodnoty jsou menší než `cmin`, zobrazana, jakoby měla hodnotu `cmin`; tj. jsou kreslena první barvou z mapy barev, a data, jejichž hodnota je větší než `cmax`, zobrazena, jakoby měla hodnotu `cmax`; tj. jsou kreslena poslední barvou z mapy barev.

`caxis('auto')`, implicitní nastavení, nastavuje vlastnost `CLimMode` tak, aby se parametry transformace barevné osy `CLim` vypočetly automaticky podle minimální a maximální hodnoty barevného pole, které je určeno konkrétní grafickou funkcí. V tomto případě jsou uříznuty pouze hodnoty nastavené na `Inf` nebo `NaN` (Not-a-Number).

`v=caxis` vrací dvouprvkový řádkový vektor `v=[cmin cmax]`.

`caxis(caxis)` zmrazí aktuální rozsah barevné osy, takže následující grafy, je-li `hold` zapnuto na `on`, použijí tento rozsah.

Algoritmus Nechť `c`, poslední argument funkcí `mesh`, `surf` nebo `pcolor`, je pole barev. Nechť je nastaveno `caxis('auto')`, nechť `cmin=min(c)`, `cmax=max(c)`, `m=length(map)`.

Následující lineární transformace transformuje pole barev `c` do pole indexů `k`, kde 1 $\leq$ `k` $\leq$ `m`. Hodnota `c=cmin` se transformuje na `k=1`, hodnota `c=cmax` na `k=m`.

`k=fix((c-cmin)/(cmax-cmin)*m)+1`,	je-li `cmin` $\leq$ `c` $<$ `cmax`,
`k=m`,	je-li `c==cmax`,
`k='invisible'`,	je-li `c` $<$ `cmin`, `c` $>$ `cmax` nebo `c==NaN`.

Od verze MATLABu 4.2 je algoritmus následující:

`k=fix((c-cmin)/(cmax-cmin)*m)+1`,	je-li `cmin` $\leq$ `c` $<$ `cmax`,
`k=m`,	je-li `c` $\geq$ `cmax`,
`k=1`,	je-li `c` $\leq$ `cmin`,
`k='invisible'`,	je-li `c==NaN`.

Příklady Vytvoříme kouli o poloměru 1

```
[X,Y,Z]=sphere(32);
C=Z;
```

Prvky matice `C` mají v tomto případě rozsah $[-1\,1]$. Příkazem

```
surf(X,Y,Z,C)
```

zobrazíme všechna data. Prvky pole `C` ležící poblíž hodnoty -1 jsou přiřazeny nejnižším hodnotám v tabulce barev, prvky pole `C` ležící blízko hodnoty +1 jsou přiřazeny nejvyšším hodnotám v tabulce barev.

Zobrazení pouze dolní poloviny koule zajistí příkaz

```
caxis([-1 0])
```

Horní polovina je odstřižena. Příkazem

```
caxis([-1 3])
```

se naopak zobrazí celá koule, ale pomocí pouze poloviční tabulky barev. Protože data v poli `C` jsou v rozsahu $[-1\,1]$, jsou prvkům matice `C` přiřazeny barvy pouze ze spodní části tabulky barev.

Příkazem

```
caxis('auto')
```

nastavíme rozsah barevné osy zpět na implicitní hodnotu a opět se zobrazí celá koule v úplném rozsahu barev.

POZOR! Tento příklad používá algoritmus verze MATLABu 4.0.

```
z=caxis
```

vytvoří vektor `z=[-1 1]`.

Viz též `axes`, `axis`, `colormap`, `set`, `get`
Vlastnosti `CLim` a `CLimMode` objektu axes.
Vlastnost `ColorMap` objektu figure.

Funkce Vyčištění aktuálních os.

Syntaxe

```
cla
cla reset
cla('reset')
```

Popis `cla` zruší všechny objekty (line, text, patch, surface, image) aktuálního objektu axes.

`cla reset` zruší vše a zároveň znovu nastaví všechny vlastnosti objektu axes na jejich implicitní hodnoty, kromě vlastnosti `Position`.

`cla('reset')` je ekvivalentní `cla reset`.

Viz též `clf`, `reset`, `hold`

Funkce Popis vrstevnic jejich výškovou hodnotou.

Syntaxe
```
clabel(C)
clabel(C,v)
clabel(C,'manual')
```

Popis `clabel(C)` přidá do aktuálního grafu, v němž byly funkcí `contour` vytvořeny vrstevnice matice `C`, popis jejich výškové hodnoty. Umístění popisu je zvoleno náhodně.

`clabel(C,v)` provede popis pouze u těch vrstevnic, které jsou zadány vektorem `v`.

`clabel(C,'manual')` umístí popis vrstevnic do pozice vybrané myší. Stiskem mezerníku nebo stiskem levého tlačítka myši se popis provede. Stlačením klávesy **Return** popis ukončíme.

Příklady Generování, vykreslení a popis jednoduchého grafu vrstevnic

```
[X,Y]=meshgrid(-2:.2:2);
Z=X.^exp(-X.^2-Y.^2);
C=contour(X,Y,Z);
clabel(C);
```

Viz též `contour`, `contourc`, `ginput`

Funkce Vyčištění příkazového okna MATLABu.

Syntaxe `clc`

Popis `clc` vyčistí příkazové okno MATLABu, kurzor je nyní nahoře na obrazovce.

Příklady Zobrazení posloupnosti náhodných matic v příkazovém okně (jednu matici přes druhou stále na stejném místě)

```
clc
for i=1:25,
  home
  A=rand(5)
end|
```

Viz též `home`, `clf`

Funkce Vyčištění aktuálního grafického okna.

Syntaxe

```
clf
clf reset
clf('reset')
```

Popis `clf` zruší všechny objekty (axes, uicontrol a uimenu) aktuálního grafického okna.

`cla reset` zruší vše a zároveň znovu nastaví všechny vlastnosti objektu figure na jejich implicitní hodnoty, kromě vlastnosti `Position`.

`clf('reset')` je ekvivalentní `clf reset`.

Viz též `cla`, `reset`, `hold`, `clc`

Funkce Zavření grafického okna.

Syntaxe
```
close
close(h)
```

Popis `close` zavře aktuální grafické okno.

`close(h)` zavře okno s identifikátorem `h`.

Příkaz `close` zavírá okna bezpodmínečně a bez upozornění.

Příklady Jiný způsob, jak uzavřít aktuální okno

```
close(gcf)
```

Viz též `gcf`, `figure`, `delete`

Funkce Zobrazí barevný sloupec (barevné měřítko).

Syntaxe

```
colorbar
colorbar('horiz')
colorbar('vert')
colorbar(h)
h=colorbar(...)
```

Popis `colorbar` bez argumentů buď přidá nové vertikální barevné měřítko nebo změní existující.

`colorbar('horiz')` přidá k aktuální ose horizontální barevné měřítko.

`colorbar('vert')` přidá k aktuální ose vertikální barevné měřítko.

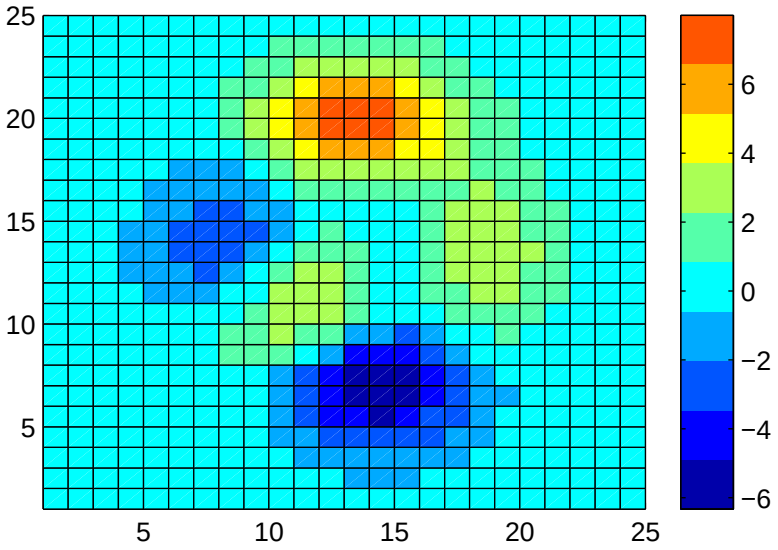
`colorbar(h)` umístí barevné měřítko do os `h`. Barevné měřítko bude horizontální, pokud šířka os `h` bude větší než jejich výška.

`h=colorbar(...)` vrátí identifikátor os barevného měřítka.

Poznámka Implementováno od verze 4.2.

Příklady Doplnění mozaikového grafu funkce `peaks` o její barevné měřítko:

```
colormap(jet(10))
pcolor(peaks(25))
colorbar
```



Viz též `legend`

Funkce Mapa barev.

Syntaxe
```
colormap(map)
colormap('default')
map=colormap
```

Popis Mapa barev je matice typu $(m, 3)$, která obsahuje reálná čísla v rozsahu od 0 do 1. V k-tém řádku mapy barev je prostřednictvím intenzit červené, zelené a modré barvy definována k-tá barva, tj. `map(k,:)=[r(k) g(k) b(k)]`. Např. [0 0 0] je černá barva, [1 1 1] je bílá, [1 0 0] je čistá červená barva, [0.5 1.0 0.83] je akvamarinová.

`colormap(map)` nastaví mapu barev na zadanou matici. Je chyba, leží-li nějaká hodnota vně intervalu $\langle 0, 1\rangle$.

`map=colormap` obnovuje aktuální mapu barev. Navrácené hodnoty jsou v intervalu $\langle 0, 1\rangle$.

`colormap('default')` a `colormap(hsv)` nastaví aktuální mapu barev na její implicitní hodnotu, tj. na mapu `hsv`, ve které se mění odstíny barev podle barevného modelu `hsv` (hue-saturation-value).

V adresáři `..\toolbox\matlab\color` jsou pomocí m-souborů definovány další mapy barev (např. `gray`, `hot`, `cool`, `bone`, `copper`, `pink`, `prism`, `jet`, `flag`). Velikost map barev můžeme určit argumentem těchto funkcí. Implicitní hodnota je 64. Následující příkaz vytvoří mapu barev délky 128, která obsahuje barvy od černé přes odstíny červené, oranžové a žluté až po bílou

```
colormap(hot(128))
```

Algoritmus Grafické funkce používající mapy barev – `mesh`, `surf`, `pcolor` a další - transformují matici barev c, jejíž hodnoty leží v rozsahu $[cmin,\ cmax]$, na pole indexů k, které mají rozsah $[1,\ m]$. Hodnoty $cmin$ a $cmax$ jsou buď $min(min(c))$ a $max(max(c))$ nebo jsou určeny pomocí `caxis`. Tato transformace je lineární, $cmin$ je transformováno na index 1 a $cmax$ na index m. Ke každému prvku matice je potom pomocí indexů přiřazena odpovídající barva mapy barev. Podrobnosti viz `caxis`.

Každé grafické okno má svoji vlastnost `Colormap`. `colormap` je jednoduchý m-soubor, který nastavuje a vrací tuto vlastnost.

Příklady V demonstračních příkladech, `imagedemo`, jsou předvedeny různé mapy barev. Vybereme-li z menu položku **ColorSpiral**, je funkcí `pcolor` zobrazena matice řádu 16, jejíž prvky se mění od 0 do 256 v pravoúhlé spirále. Implicitní mapa barev `hsv` začíná ve středu spirály červenou barvou, mění se postupně ve žlutou, zelenou, tyrkysovou, modrou a fialovou a vrací se na konci spirály zpět k červené barvě. Z nabídky map barev můžeme vybrat a zobrazit i další mapy barev.

Další informace o mapách barev dává funkce `rgbplot`. Např. `rgbplot(hsv)`, atd.

Viz též `surf`, `image`, `mesh`, `pcolor`, `caxis`, `brighten`, `rgbplot`, ColorSpec
Vlastnost `Colormap` u objektu figure.

ColorSpec

Funkce Určení barev.

Popis *ColorSpec* není příkaz, ale ukazuje tři způsoby, kterými lze definovat barvu v MATLABu:

- zkráceným názvem,
- dlouhým názvem,
- RGB vektorem.

Zkrácené a dlouhé názvy jsou řetězce MATLABu, které určují jednu z osmi předdefinovaných barev. V následující tabulce je uveden seznam předdefinovaných barev a jejich RGB ekvivalenty.

RGB hodnoty	Zkrácený název	Dlouhý název
1, 1, 0	y	yellow
1, 0, 1	m	magenta
0, 1, 1	c	cyan
1, 0, 0	r	red
0, 1, 0	g	green
0, 0, 1	b	blue
1, 1, 1	w	white
0, 0, 0	k	black

Trojice RGB je tříprvkový řádkový vektor, jehož prvky určují intenzitu červené, zelené a modré složky barvy. Intenzita musí být číslo v rozsahu $\langle 0, 1 \rangle$.

Pokud určíme takovou trojici RGB, která není ve výše uvedeném seznamu osmi barev, přidělí MATLAB nové barvě místo v systémové tabulce barev. Proto takto definované barvy jsou skutečné barvy na rozdíl od barev, které jsou transformovány do uzavřeného seznamu, který již existuje v tabulce barev.

Osm předdefinovaných barev a libovolné další, které definujeme RGB hodnotami, nejsou ani součástí mapy barev objektu figure, ani na ni nemají vliv. Jsou to tzv. *stálé* barvy na rozdíl od barev mapy barev.

Příklady Pro vytvoření grafu, který použije zelenou čáru, můžeme definovat barvu zkráceným názvem, dlouhým názvem nebo RGB vektorem.

Následující příkazy jsou ekvivalentní:

```
plot(xdata,ydata,'g')

plot(xdata,ydata,'green')
```

```
h=plot(xdata,ydata)
set(h,'Color',[0,1,0])
```

ColorSpec můžeme použít, kdykoliv potřebujeme definovat barvu.

Následující příkaz např. mění pozadí objektu figure na růžovou barvu:

```
set(gcf,'Color',[1,0.4,0.6])
```

Viz též `colormap`

Funkce Pohybující se bod (kometa).

Syntaxe
```
comet(Y)
comet(X,Y)
comet(X,Y,p)
```

Popis `comet(Y)` zobrazuje bod, který se pohybuje po trajektorii popsané vektorem `Y`.

`comet(X,Y)` zobrazuje bod, který se pohybuje po trajektorii popsané vektorem `Y` vzhledem k vektoru `X`.

`comet(X,Y,p)` používá 'chvost komety' o délce `p*length(Y)`. Implicitně je `p=0.10`.

`comet`, samostatně, spustí svoje demo.

Poznámka Implementováno od verze 4.1.

Příklady
```
t=-pi:pi/200:pi;
comet(t,tan(sin(t))-sin(tan(t)))
```

Viz též `comet3`, (`quakedemo`)

Funkce Pohybující se bod ve 3D (kometa).

Syntaxe
```
comet3(Z)
comet3(X,Y,Z)
comet3(X,Y,Z,p)
```

Popis `comet3(Z)` zobrazuje bod, který se pohybuje po trajektorii popsané vektorem `Z`.

`comet3(X,Y,Z)` zobrazuje bod, který se pohybuje po trajektorii popsané body `[X(i), Y(i), Z(i)]`.

`comet3(X,Y,Z,p)` používá 'chvost komety' o délce `p*length(Z)`. Implicitně je `p=0.10`.

`comet`, samostatně, spustí svoje demo.

Poznámka Implementováno od verze 4.1.

Příklady
```
t=-pi:pi/500:pi;
comet3(sin(5*t),cos(3*t),t)
```

Viz též `comet`, (`quakedemo`)

Funkce Graf tvaru růžice.

Syntaxe

```
compass(Z)
compass(X,Y)
compass(...,linetype)
```

Popis

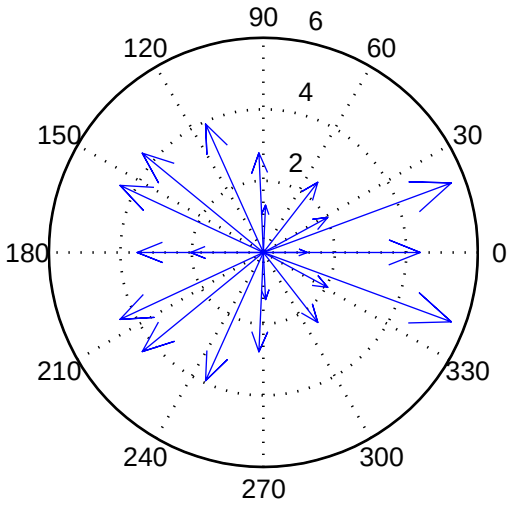
`compass(Z)` kreslí graf, který zobrazuje úhel a velikost komplexních prvků matice `Z` ve tvaru šipek vycházejících z počátku souřadného systému.

`compass(X,Y)` je ekvivalentní příkazu `compass(X+i*Y)` a zobrazí růžicový graf pro úhly a velikosti prvků matic `X` a `Y`.

`compass(...,linetype)` kreslí graf zadaným typem čáry (viz `plot`).

Příklady Pro zobrazení vlastních čísel matice náhodných prvků ve tvaru růžicového grafu použijeme následující příkazy:

```
Z=eig(randn(20,20))
compass(Z)
```



Viz též `rose`, `feather`, `plot`, `quiver`

Funkce Kreslení vrstevnic.

Syntaxe

```
contour(Z)
contour(Z,n)
contour(Z,v)
contour(x,y,Z)
contour(x,y,Z,n)
contour(x,y,Z,v)
contour(...,linetype)
[C,h]=contour(...)
```

Popis `contour(Z)` kreslí vrstevnice matice `Z` (v datových jednotkách pole `Z`). Levý dolní roh grafu odpovídá prvku `Z(1,1)`. Počet vrstevnic a jejich hodnoty jsou funkcí `contour` vybrány automaticky.

`contour(Z,n)` vytváří graf s `n` vrstevnicemi matice `Z` .

`contour(Z,v)` kreslí takové vrstevnice matice `Z`, jejichž hodnoty jsou zadány vektorem `v`.

`contour(x,y,Z)`, `contour(x,y,Z,n)` a `contour(x,y,Z,v)` vytvářejí grafy vrstevnic matice `Z`, pro rozsah os x a y používají data z vektorů `x` a `y`. Prvky těchto vektorů jsou rovnoměrně rozloženy.

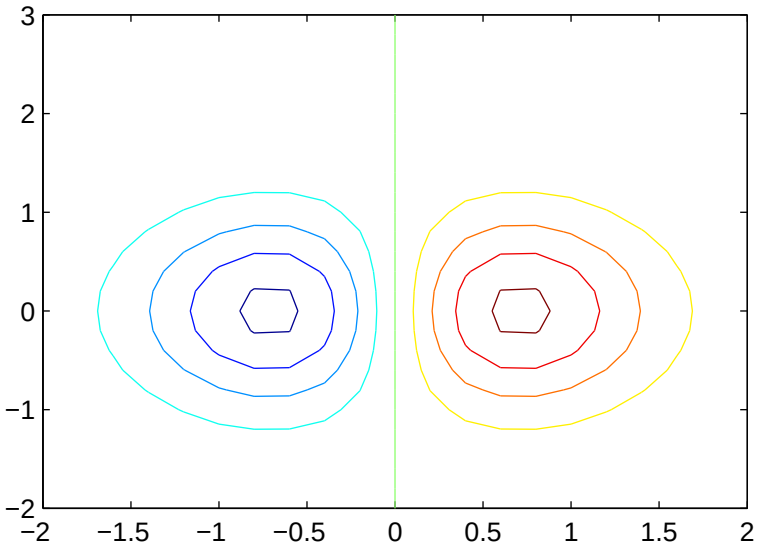
`contour(...,linetype)` kreslí vrstevnice zadaným typem čáry.

`[C,h]=contour(...)` vrací matici vrstevnic `C`, která je popsána u funkce `contourc` a používá ji funkce `clabel`, a vektor identifikátorů jednotlivých vrstevnic.

Popis jednotlivých úrovní vrstevnic viz `clabel`.

Příklady Vrstevnice funkce $z = x\,e^{-x^2-y^2}$ v oblasti $-2 \leq x \leq 2$, $-2 \leq y \leq 3$ zobrazíme následujícími příkazy

```
x=-2:.2:2;
y=-2:.2:3;
[X,Y]=meshgrid(x,y);
Z=X.*exp(-X.^2-Y.^2);
contour(x,y,Z)
```



Chybové hlášení

Je-li nejmenší rozměr matice `Z` menší než 2:

```
Matrix must be 2-by-2 or larger
```

Omezení `contour` předpokládá, že jsou vektory `x` a `y` monotónně rostoucí. Nemají-li prvky vektorů `x` a `y` rovnoměrný přírůstek, jsou vrstevnice *napnuté* a neshodují se s vrstevnicemi, které bychom získali pro tutéž matici při rovnoměrném rozložení prvků vektorů `x` a `y`.

Viz též `contour3`, `contourc`, `clabel`, `quiver`

Funkce Kreslení vrstevnic ve 3D.

Syntaxe

```
contour3(Z)
contour3(Z,n)
contour3(X,Y,Z)
contour3(X,Y,Z,n)
contour3(...,linetype)
[C,h]=contour3(...)
```

Popis `contour3` vytváří prostorové vrstevnice plochy definované na obdélníkové síti.

`contour3(Z)` kreslí vrstevnicové čáry matice `Z` v prostoru.

`contour3(Z,n)` kreslí `n` vrstevnic matice `Z` v prostoru.

`contour3(X,Y,Z)` a `contour3(X,Y,Z,n)` užívají matic `X` a `Y` pro definování rozsahu os. `X`, resp. `Y` mohou být též vektory, pak jsou ale doplněny na matice opakováním řádků, resp. sloupců.

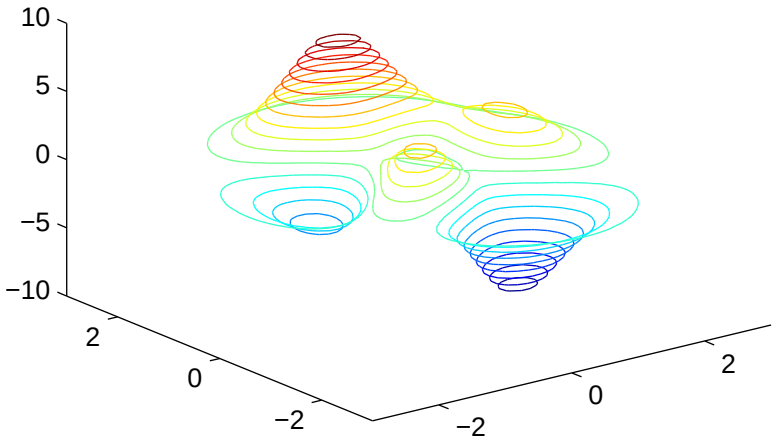
`contour3(...,linetype)` kreslí vrstevnice zadaným typem čáry.

`[C,h]=contour3(...)` vrací matici vrstevnic `C`, která je popsána u funkce `contourc` a používá ji funkce `clabel`, a vektor identifikátorů jednotlivých vrstevnic.

Popis jednotlivých úrovní vrstevnic viz `clabel`.

Příklady Dvacet vrstevnic funkce `peaks` zobrazíme následujícími příkazy

```
x=-3:.125:3;
y=x;
[X,Y]=meshgrid(x,y);
Z=peaks(X,Y);
contour3(X,Y,Z,20)
```



Omezení `contour3` předpokládá, že jsou vektory `x` a `y` monotónně rostoucí. Nemají-li prvky vektorů `x` a `y` rovnoměrný přírůstek, jsou vrstevnice *napnuté* a neshodují se s vrstevnicemi, které bychom získali pro tutéž matici při rovnoměrném rozložení prvků vektorů `x` a `y`.

Viz též `meshc`, `surfc`, `meshgrid`, `contour`

Funkce Výpočet vrstevnic.

Syntaxe

```
C=contourc(Z)
C=contourc(Z,n)
C=contourc(Z,v)
C=contourc(x,y,Z)
C=contourc(x,y,Z,n)
C=contourc(x,y,Z,v)
```

Popis `contourc` počítá matici vrstevnic `C`, kterou používá funkce `contour` pro vykreslení aktuálních vrstevnic

`C=contourc(Z)` počítá matici vrstevnic `C` pro vykreslení vrstevnic matice `Z` (v datových jednotkách pole `Z`). Počet vrstevnic a jejich hodnoty jsou funkcí `contourc` vybrány automaticky.

`C=contourc(Z,n)` počítá `n` vrstevnic matice `Z`.

`C=contourc(Z,v)` počítá takové vrstevnice matice `Z`, jejichž hodnoty jsou zadány vektorem `v`.

`C=contourc(x,y,Z)`, `C=contourc(x,y,Z,n)` a `C=contourc(x,y,Z,v)` počítají vrstevnice matice `Z`, pro rozsah os x a y používají data z vektorů `x` a `y`. Prvky těchto vektorů mají rovnoměrné přírůstky.

Matice vrstevnic `C` má dva řádky, pro každou kreslenou vrstevnici obsahuje její hodnotu, počet kreslených párů (x, y) a jednotlivé páry. Z těchto segmentů je matice `C` složena

```
C=[úroveň1 x1 x2 x3 ... úroveň2 x1 x2 x3 ... ;
   pár1    y1 y2 y3 ... pár2    y1 y2 y3 ...]
```

Omezení `contourc` předpokládá, že jsou vektory `x` a `y` monotónně rostoucí. Nemají-li prvky vektorů `x` a `y` rovnoměrný přírůstek, jsou vrstevnice *napnuté* a neshodují se s vrstevnicemi, které bychom získali pro tutéž matici při rovnoměrném rozložení prvků vektorů `x` a `y`.

Viz též `contour`, `contour3`

Funkce Řízení kontrastu.

Syntaxe `cmap=contrast(X,M)`

Popis `cmap=contrast(X,M)` vrací šedou mapu barev, což je matice typu $(M, 3)$ se třemi stejnými sloupci, při které příkazy

```
image(X)
colormap(cmap)
```

vytvoří objekt image se zhruba vyrovnaným jasovým histogramem. Pokud není `M` specifikováno, je použita implicitní hodnota, tj. 64.

Poznámka Implementováno od verze 4.1.

Příklady Zobrazení obrázku zeměkoule s využitím funkce `contrast`:

```
load earth
image(X)
cmap=contrast(X)
colormap(cmap)
```

Viz též `brighten`

Funkce Čte čárkou oddělená čísla ze souboru do matice.

Syntaxe
```
M=csvread(filename,r,c)
M=csvread(filename)
M=csvread(filename,r,c,rng)
```

Popis `M=csvread(filename,r,c)` načítá soubor `filename`, v němž jsou hodnoty oddělené čárkou, do matice `M`. Volitelné argumenty `r` a `c` udávají první řádek a sloupec spreadsheetu (matice), kam se má soubor načíst.

`M=csvread(filename)` načítá soubor `filename`, v němž jsou hodnoty oddělené čárkou, do matice `M`. Toto je ekvivalentní `r=c=0`, poněvadž horní levá buňka ve spreadsheetu (matici) má index $(0, 0)$.

Volitelný čtvrtý argument, `rng`, může být použit ke stanovení požadované části spreadsheetu. Chcete-li určit část spreadsheetu, specifikujte `rng` takto:

`rng=[UpperLeftRow UpperLeftColumn LowerRightRow LowerRightColumn]`.

Poznámka Implementováno od verze 4.2.

Příklady
```
A=csvread('c:/pokus.dat');
```

Viz též `csvwrite`, `wk1read`, `wk1write`

Funkce Zapisuje matici do souboru ve formátu čárkou oddělených čísel.

Syntaxe
```
csvwrite(filename,M,r,c)
csvwrite(filename,M)
```

Popis `csvwrite(filename,M,r,c)` zapisuje matici `M` do souboru `filename`, v němž jsou hodnoty oddělené čárkou. Volitelné argumenty `r` a `c` udávají první řádek a sloupec spreadsheetu (souboru), kam se má matice zapsat.

`csvwrite(filename,M)` zapisuje matici `M` do souboru `filename`, v němž jsou hodnoty oddělené čárkou. Toto je ekvivalentní `r=c=0`, poněvadž horní levá buňka ve spreadsheetu má index $(0, 0)$.

POZOR! Některé prvky, jejichž hodnota je nulová, budou chybět. Např. matice [1 0 2] se v souboru objeví jako $'1, , 2'$.

Poznámka Implementováno od verze 4.2.

Příklady
```
csvwrite('c:/pokus.dat',M);
```

Viz též `csvread`, `wk1read`, `wk1write`

Funkce Generování válce.

Syntaxe

```
[X,Y,Z]=cylinder(r)
[X,Y,Z]=cylinder(r,n)
[X,Y,Z]=cylinder
cylinder(...)
```

Popis `cylinder` generuje souřadnice (x, y, z) jednotkového válce pro funkce `surf` nebo `mesh`.

`[X,Y,Z]=cylinder(r,n)` vrátí souřadnice (x, y, z) rotačního tělesa, které vzniklo rotací křivky definované vektorem `r` kolem osy rovnoběžné s osou z a procházející počátkem $(0, 0, 0)$. Vektor `r` obsahuje poloměry rotačního tělesa v bodech rovnoměrně rozložených na ose z s jednotkovým přírůstkem. Rotační těleso může být zobrazeno funkcí `surf` nebo `mesh` (viz následující příklad). Číslo `n` udává počet bodů po obvodě rotačního tělesa.

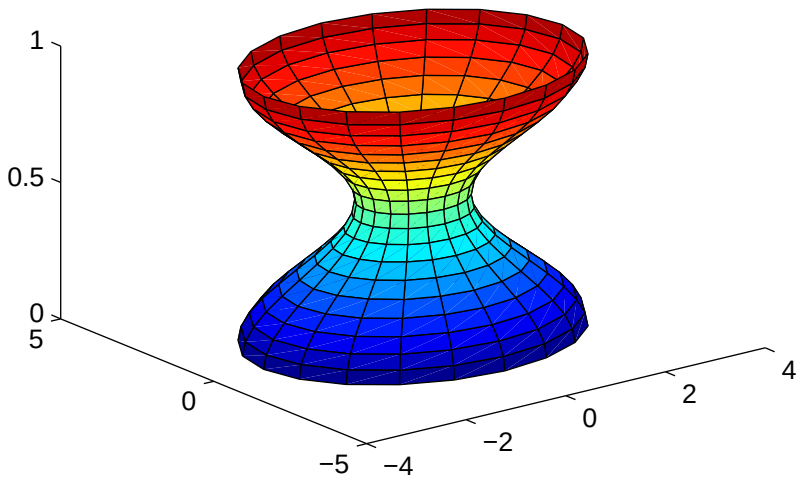
`[X,Y,Z]=cylinder(r)` použije implicitní hodnotu `n=20`.

`[X,Y,Z]=cylinder` bez vstupních argumentů použije `n=20` a `r=[1 1]`.

`cylinder(...)` bez výstupu vykreslí funkcí `surf` rotační těleso na obrazovku.

Příklady Vygenerování rotačního tělesa definovaného funkcí $2 + \cos(t)$:

```
t=0:pi/10:2*pi;
[X,Y,Z]=cylinder(2+cos(t));
surf(X,Y,Z)
```



Viz též `sphere`

Funkce Odstraní zbytkové mezery z konce řetězce.

Syntaxe `deblank(S)`

Popis `deblank(S)` odstraní zbytkové mezery a nulové znaky z řetězce `S`. Nulový znak je znak, jehož absolutní hodnota je 0.

Poznámka Implementováno od verze 4.1.

Viz též `blanks`

Funkce Konverze dekadického čísla na číslo hexadecimální.

Syntaxe `s=dec2hex(n)`

Popis `dec2hex(n)` transformuje dekadické celé číslo `n` do jeho hexadecimální reprezentace uložené v řetězci MATLABu.

Příklady `dec2hex(1023)` vrátí řetězec `'3ff'`

Viz též `hex2dec`, `hex2num`

Funkce Vymazání souborů nebo grafických objektů.

Syntaxe
```
delete(filename)
delete(h)
```

Popis `delete(filename)` vymaže soubor určený řetězcovou proměnnou `filename`.

`delete(h)` vymaže grafický objekt s identifikátorem `h`. Je-li uvedeným objektem objekt figure, je grafické okno uzavřeno a vymazáno bez upozornění.

Viz též (`cd`, `dir`, `type`, `who`, `!`)

Funkce Ukládá kopii příkazového okna.

Syntaxe
```
diary(filename)
diary
diary on
diary off
diary('on')
diary('off')
```

Popis `diary(filename)` ukládá kopii příkazového okna (všechny znaky z klávesnice a většina výsledných výstupů; kromě grafických oken) do souboru určeného řetězcovou proměnnou `filename`. Pokud soubor již existuje, je výstup připojen na konec souboru.

`diary`, samostatně, přepíná nastavení této funkce.

`diary on` zapíná ukládání kopie příkazového okna buď do aktuálního souboru určeného příkazem `diary filename` nebo do souboru `diary`, pokud nebyl příkaz `diary filename` použit.

`diary off` vypíná ukládání kopie příkazového okna.

Výstupní soubor tohoto příkazu je textovým souborem.

`diary('on')` je ekvivalentní `diary on` a `diary('off')` je ekvivalentní `diary off`.

Omezení Jméno výstupního souboru nesmí být `'on'` ani `'off'`.

Viz též `save`, `fprintf`, (`disp`)

Funkce Odrazivost difúzní plochy (Lambertův zákon).

Syntaxe `r=diffuse(Nx,Ny,Nz,S)`

Popis `diffuse` vrací odrazivost difúzní (Lambertovy) plochy od složek vektoru normály. Odrazivost udává, jaká část světla se odráží od plochy směrem k pozorovateli. Odrazivost se mění od 0 (žádné světlo se neodráží) do 1 (všechno světlo je odráženo).

`r=diffuse(Nx,Ny,Nz,S)` vrací odrazivost ploch s vektorem normály o složkách `[Nx, Ny, Nz]`. Složky vektoru normály mohou být matice, takže normála je

```
n(i,j)=[Nx(i,j),Ny(i,j),Nz(i,j)]
```

Tyto složky vektoru normály mohou být vypočteny prostřednictvím funkce `surfnorm`. Zdroj světla `S=[Sx,Sy,Sz]` je vektor o třech složkách, které určují směr, ze kterého je plocha osvětlena. Vektor `S` může mít též délku 2 a být zadán azimutem a elevací.

`diffuse` je volána ve funkci `surfl`, která vytváří objekt surface včetně stínů.

Algoritmus `diffuse` používá pro difúzní plochy Lambertova zákona $r = \cos(\theta)$, kde θ je úhel mezi normálou plochy a světelným zdrojem. Difúze nezávisí na směru pohledu.

Viz též `surfl`, `specular`, `surfnorm`

Funkce Čte ASCII znakem oddělená čísla ze souboru do matice.

Syntaxe
```
M=dlmread(filename,dlm,r,c)
M=dlmread(filename,dlm)
M=dlmread(filename,dlm,r,c,rng)
```

Popis `M=dlmread(filename,dlm,r,c)` načítá soubor `filename`, v němž jsou hodnoty oddělené ASCII znakem `dlm`, do matice `M`. Volitelné argumenty `r` a `c` udávají první řádek a sloupec spreadsheetu (matice), kam se má soubor načíst.

`M=dlmread(filename,dlm)` načítá soubor `filename`, v němž jsou hodnoty oddělené ASCII znakem `dlm`, do matice `M`. Toto je ekvivalentní `r=c=0`, poněvadž horní levá buňka ve spreadsheetu (matici) má index $(0, 0)$.

Volitelný čtvrtý argument, `rng`, může být použit ke stanovení požadované části spreadsheetu. `rng` může být zadán buď indexy nebo jménem. Při použití indexů zadejte `rng` takto:

`rng=[UpperLeftRow UpperLeftColumn LowerRightRow LowerRightColumn]`.

Poznámka Implementováno od verze 4.2.

Příklady
```
M=dlmread('c:/pokus.dat',';');
```

Viz též `csvread`, `csvwrite`, `dlmwrite`, `wk1read`, `wk1write`

Funkce Zapisuje matici do souboru, ve kterém jsou čísla oddělena zvoleným ASCII znakem.

Syntaxe
```
dlmwrite(filename,M,dlm,r,c)
dlmwrite(filename,M)
```

Popis `dlmwrite(filename,M,dlm,r,c)` zapisuje matici `M` do souboru `filename`, v němž jsou čísla oddělena ASCII znakem `dlm`. Volitelné argumenty `r` a `c` udávají první řádek a sloupec spreadsheetu (souboru), kam se má matice zapsat.

`dlmwrite(filename,M)` zapisuje matici `M` do souboru `filename`, v němž jsou čísla oddělena čárkou. Toto je ekvivalentní `r=c=0`, poněvadž horní levá buňka ve spreadsheetu má index $(0, 0)$.

POZOR! Některé prvky, jejichž hodnota je nulová, budou chybět. Např. matice [1 0 2] se v souboru objeví jako $'1, , 2'$.

Poznámka Implementováno od verze 4.2.

Příklady
```
dlmwrite('c:/pokus.dat',M,';');
```

Viz též `csvread`, `csvwrite`, `dlmread`, `wk1read`, `wk1write`

Funkce Zpracování nevyřízených žádostí o zobrazení.

Syntaxe
```
drawnow
drawnow discard
drawnow('discard')
```

Popis `drawnow` vyprázdní frontu a přinutí MATLAB, aby aktualizoval obrazovku.

Běží-li nějaký m-soubor, nezobrazuje MATLAB na obrazovce každý grafický příkaz samostaně. Probíhá-li např. m-soubor

```
plot(x,y)
axis([0 10 0 10])
title('Krátký nadpis')
grid
```

nekreslí MATLAB na obrazovce do té doby, dokud se neprovede poslední příkaz m-souboru a MATLAB se nevrátí na svoji příkazovou řádku. Tím se docílí efektivní provedení posloupnosti grafických příkazů. V předcházejícím příkladě např. nejsou kresleny osy dvakrát, jak by se stalo, kdybychom psali jednotlivé příkazy do příkazové řádky a tyto příkazy se přímo vykonávaly.

Následující čtyři události přinutí MATLAB, aby vyprázdnil frontu a kreslil hned na obrazovku:

- návrat na příkazovou řádku MATLABu,
- příkaz `pause`,
- vykonání příkazu `getframe`,
- vykonání příkazu `drawnow`.

`drawnow discard` provádí v podstatě opak operace `drawnow` – vzdá se všech nevyřízených událostí včetně kreslení, akcí myši a klávesnice. Tato volba může být užitečná tehdy, chceme-li přechodně změnit vlastnosti objektu v době provádění operace, a poté je změnit zpět, aniž by došlo k překreslení grafického okna. Např. před tiskem můžeme chtít změnit vlastnosti objektu figure, ale nechceme vidět obsah překresleného okna s těmito vlastnostmi (nebo nechceme čekat na jeho překreslení), a překreslit jej opět, až budeme resetovat vlastnosti.

Napíšeme-li

```
set(gcf,'Color','r'),drawnow discard
```

nezmění se v grafickém okně pozadí objektu figure, ale vypíšeme-li si poté nastavenou hodnotu vlastnosti `Color`, je vrácena jako červená:

```
get(gcf,'Color')

ans=
1 0 0
```

Pokud změníme velikost grafického okna, generujeme tím událost, která způsobí překreslení okna a pozadí objektu figure je překresleno červeně.

`drawnow('discard')` je ekvivalentní `drawnow discard`.

Funkce Chybový graf.

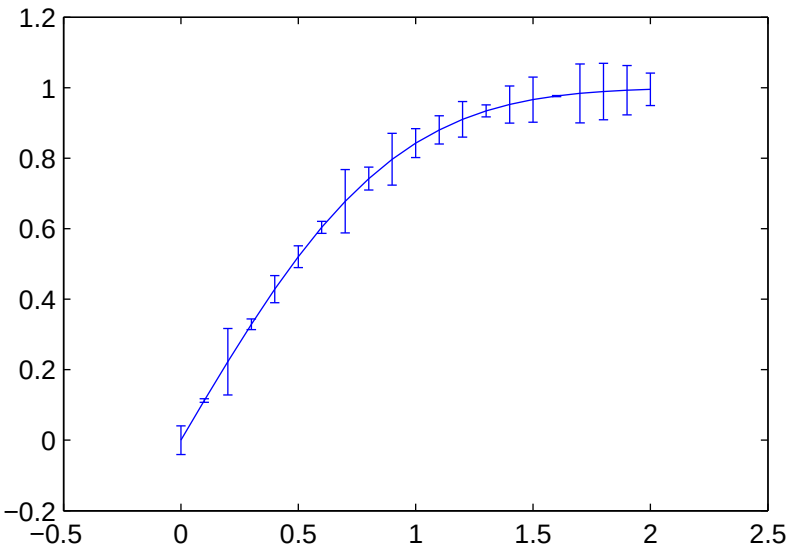
Syntaxe
```
errorbar(y,e)
errorbar(x,y,e)
```

Popis `errorbar(y,e)` kreslí graf vektoru `y` s chybovými sloupci určenými vektorem `e`. Vektory `y` a `e` musí mít stejnou délku. Délka chybového sloupce je `2*e`.

`errorbar(x,y,e)` kreslí graf závislosti vektoru `y` na `x` s chybovými sloupci určenými vektorem `e`. Vektory `x`, `y` a `e` musí mít stejnou délku. Délka chybového sloupce je `2*e` se středem v bodě `(x,y)`.

Jsou-li `x`, `y` a `e` matice stejné velikosti, vykreslí se chybový graf pro každý sloupec.

Příklady
```
x=0:0.1:2;
y=erf(x);
e=rand(size(x))/10;
errorbar(x,y,e)
```



Viz též `plot`, `bar`, (`std`)

Funkce Vytvoří dialogový box popisu chyby.

Syntaxe `h=errordlg(errorstr,dlgname,replace)`

Popis `h=errordlg(errorstr,dlgname,replace)` vytvoří dialogový box popisu chyby, který zobrazí textový řetězec `errorstr` v objektu figure, který bude mít název `dlgname`. Chcete-li dialogové okno odstranit, musíte stisknout tlačítko **OK**.
Pokud je `replace='on'` a dialogový box se jménem `dlgname` již existuje, přesune se dialogový box do popředí (žádné nové dialogové okno se nevytvoří). `errordlg` vrací identifikátor `h` objektu figure.

Poznámka Implementováno od verze 4.2.

Příklady `errordlg('Division by zero','ERROR')`

Viz též `dialog`

Funkce Interpretuje řetězce obsahující výrazy MATLABu.

Syntaxe

```
eval(s)
eval(s1,s2)
[arg1,arg2,arg3,...]=eval(s)
```

Popis `eval(s)` interpretuje textový řetězec `s`. Pokud `s` představuje výraz, vrátí `eval(s)` hodnotu tohoto výrazu. Pokus `s` představuje příkaz, `eval(s)` tento příkaz vykoná. Řetězec `s` je často vytvořen spojením dílčích řetězců a proměnných uvnitř hranatých závorek.

`eval(s1,s2)` interpretuje textový řetězec `s1`. Pokud dojde při interpretaci k chybě, vyhodnotí se řetězec `s2`.

`[arg1,arg2,arg3,...]=eval(s)` vrací výstupní argumenty po vyhodnocení řetězce `s`.

Příklady Příkazy

```
s='4*atan(1)';
pi=eval(s);
```

nastaví hodnotu `pi`.

Následující smyčka generuje posloupnost 12 matic nazvaných `M1` až `M12`:

```
for n=1:12
  eval(['M' int2str(n)'=magic(n)'])
end
```

Další příklad spustí zvolený dávkový m-soubor. Povšimněme si, že řetězce tvořící řádky matice `D` mají všechny stejnou délku.

```
D=['odedemo ' 'quaddemo' 'zerodemo' 'fitdemo '];
n=input('Zvol číslo demonstračního souboru: ');
eval(D(n,:))
```

Následující příklad demonstruje použití druhého řetězcového argumentu pro detekci chyb. Příkaz

```
eval('cd new/dir', 'disp(''cd was unsuccessful '')');
```

buď nastaví aktuální adresář na `new/dir` nebo v případě chyby zobrazí hlášení

```
'cd was unsuccessful'.
```

Poslední příklad načítá a zpracovává data z několika souborů, jejichž jména jsou `data1.dat`, `data2.dat`, ...

```
k=0;
while 1
  k=k+1;
  datak=['data' int2str(k)];
  filename=[datak '.dat'];
  if ~exist(filename), break, end
  eval(['load ' filename]);
  X=eval(datak);

  % Zpracuj data v matici X.

end
```

Viz též `feval`

Funkce Uzavírá jeden nebo více otevřených souborů.

Syntaxe
```
status=fclose(fid)
status=fclose('all')
```

Popis `fclose(fid)` uzavírá zvolený soubor, pokud je otevřený, a vrací 0 v případě úspěšné operace a -1 v případě neúspěchu.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

`fclose('all')` uzavírá všechny otevřené soubory, kromě standardního vstupu (0), standardního výstupu (1) a standardního chybového výstupu (2), a vrací 0 v případě úspěšné operace a -1 v případě neúspěchu.

Viz též `fopen`, `ferror`, `fread`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Graf tvaru *ptačího pera*.

Syntaxe

```
feather(z)
feather(x,y)
feather(...,linetype)
```

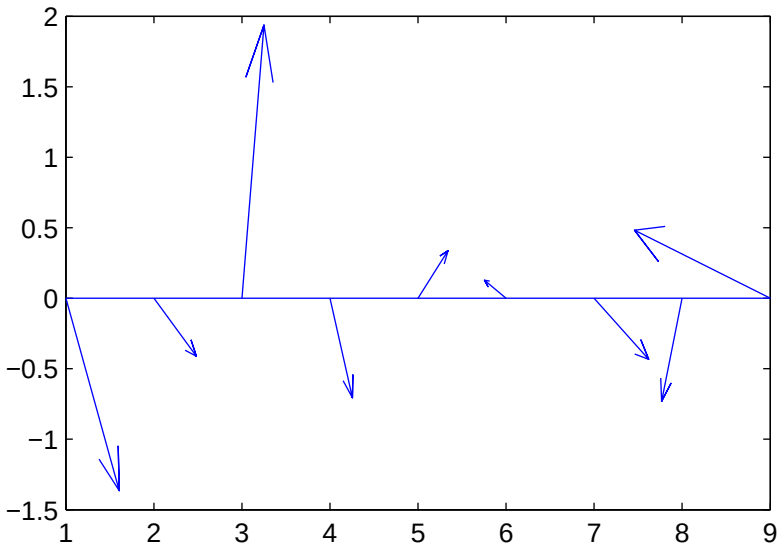
Popis `feather(z)` kreslí graf, který zobrazuje úhel a velikost komplexních prvků pole `z` pomocí šipek vycházejících z bodů rovnoměrně rozložených ve směru vodorovné osy.

`feather(x,y)` je ekvivalentní příkazu `feather(x+i*y)`.

`feather(...,linetype)` kreslí graf zadaným typem čáry (viz `plot`).

Příklady Vykreslení jednoduchého grafu *ptačí pero* pro náhodná čísla

```
z=randn(3,3)+randn(3,3)*i;
feather(z)
```



Viz též `compass`, `rose`

Funkce Test konce souboru.

Syntaxe `feof(fid)`

Popis `feof(fid)` testuje, zda je u daného souboru (`fid`) nastaven indikátor konce souboru.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

`foef(fid)` vrací 1, pokud je indikátor konce souboru nastaven, nebo 0, pokud nastaven není.

Indikátor konce souboru (EOF) je nastaven, když se funkce `fread` pokusí číst po posledním znaku v souboru. Přesunutí indikátoru pozice v souboru na konec souboru příkazem `fseek(fid,0,'eof')`, nenastaví indikátor EOF. Např.

```
fseek(fid,0,'eof')
result=feof(fid);

result=
0
```

Poznámka Implementováno od verze 4.1.

Viz též `fopen`, `ferror`, `fread`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Dotaz MATLABu na chybu v souborovém vstupu nebo výstupu.

Syntaxe
```
ferror(fid)
[message,errnum]=ferror(fid,'clear')
```

Popis `ferror` vrací v proměnné `message` chybové hlášení a volitelně číslo chyby `errnum` poslední souborové operace sdružené s daným souborem.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Řetězec `'clear'` je volitelný. Pokud je uveden, příkaz maže indikátor chyby daného souboru.

Pokud poslední souborová operace vykonaná na daném souboru byla úspěšná, `ferror` vrací v `errnum` nulu a proměnná `message` je prázdná.

Nenulová hodnota `errnum` naznačuje, že se při poslední souborové operaci s daným souborem vyskytla chyba. Existují dva způsoby, jak se můžeme o povaze chyby dozvědět více:

- Proměnná `message` je řetězec, který obsahuje informaci o povaze chyby.
- Pokud máme přístup k referenčnímu manuálu jazyka C, můžeme si prohlédnout chybové kódy generované souborovými operacemi jazyka C (knihovna `stdio`).

Viz též `fopen`, `fclose`, `fread`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Vrací další řádek souboru jako řetězec.

Syntaxe `fgetl(fid)`

Popis `fgetl(fid)` vrací další řádek souboru, sdruženého s identifikátorem souboru `fid`, jako řetězec MATLABu. Znak nového řádku (LF) a návratu vozíku (CR), pokud se vyskytují, nejsou zahrnuty. Pokud chceme tyto znaky zahrnout, použijeme funkci `fgets()`. Pokud je dosaženo konce souboru, vrací funkce -1.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Důležité upozornění:

Tato funkce je určena pouze pro textové soubory.

Poznámka Implementováno od verze 4.1.

Příklady Následující příklad je ekvivalentní příkazu: `'type fgetl.m'`

```
fid=fopen('fgetl.m');
while 1
  line=fgetl(fid);
  if ~isstr(line), break, end
  disp(line)
end
fclose(fid);
```

Viz též `fgets`

Funkce Vrací další řádek souboru jako řetězec.

Syntaxe `fgets(fid)`

Popis `fgets(fid)` vrací další řádek souboru, sdruženého s identifikátorem souboru `fid`, jako řetězec MATLABu. Znak nového řádku (LF), pokud se vyskytuje, je zahrnut. Pokud nechceme tento znak zahrnout, použijeme funkci `fgetl()`. Pokud je dosaženo konce souboru, vrací funkce -1.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Důležité upozornění:

Tato funkce je určena pouze pro textové soubory.

Poznámka Implementováno od verze 4.1.

Viz též `fgetl`

Funkce Test, zda existuje daný objekt figure.

Syntaxe
```
flag=figflag(str)
flag=figflag(str,silent)
[flag,fig]=figflag(str)
[flag,fig]=figflag(str,silent)
```

Popis `flag=figflag(str)` kontroluje, zda existuje objekt figure se jménem `str`. Pokud se takový objek figure existuje, `flag=1`, jinak `flag=0`.

`flag=figflag(str,silent)` používá navíc volitelný argument `silent`, kterým se řídí přenášení objektu figure do popředí. Pokud se `silent=0`, je objekt figure přenesen do popředí. Implicitně je `silent=0`.

`[flag,fig]=figflag(str)` a `[flag,fig]=figflag(str,silent)` vrací ve volitelném výstupním argumentu `fig` identifikátor objektu figure, který má jméno str. Pokud takový objekt neexistuje (`flag=0`), `fig=[]`.

Poznámka Implementováno od verze 4.2.

Viz též `figure`, `gcf`

Funkce Otevření nového grafického okna vytvořením objektu figure

Syntaxe

```
figure
figure(h)
h=figure
h=figure(PropertyName,PropertyValue,...)
```

Popis Objekty figure jsou dětmi objektu root a rodiči objektů axes, uimenu a uicontrol. Jsou to samostatná grafická okna na obrazovce, v nichž MATLAB zobrazuje grafické výstupy. Vytváříme-li objekt figure, vytvoříme nové grafické okno, jehož charakteristiky lze ovládat jak vlastnostmi objektu figure, tak systémem.

`figure`, samostatně, otevře nové grafické okno.

`h=figure` otevře nové grafické okno a vrátí identifikátor objektu figure. Identifikátory objektu figure jsou přirozená čísla začínající číslem 1 a jsou zobrazena na horním okraji grafického okna.

`figure(h)` zaktualizuje objekt figure s identifikátorem `h` pro následné grafické příkazy. Pokud tento objekt neexistuje, je vytvořen objekt figure s prvním dostupným identifikátorem pro tento objekt.

`figure` je funkce vytvářející objekt figure. Jejími vstupními argumenty jsou dvojice parametrů PropertyName/PropertyValue. Tyto vlastnosti jsou popsány v části *Vlastnosti objektu*. Nastavit resp. získat informace o nastavených vlastnostech lze též použitím funkce `set` resp. `get`.

Identifikátor aktuálního objektu figure obdržíme také příkazem `gcf` (**g**et **c**urrent **f**igure). Aktuální objekt figure je to grafické okno, do něhož jsou směrovány grafické příkazy.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď hned při vytváření objektu zadáním dvojic PropertyName/PropertyValue do vstupních argumentů funkce vytvářející objekt, nebo hodnoty vlastností specifikujeme až po vytvoření objektu identifikací objektu pomocí identifikátoru a funkcemi `set` a `get`.

Následující seznam udává přehled všech vlastností objektu figure s jejich přípustnými hodnotami. Pokud jsou nastaveny implicitní hodnoty, jsou uvnitř složených závorek.

BackingStore `{on} | off`

Režim ukládání kopie grafického okna do vyrovnávací paměti. Má-li `BackingStore` hodnotu `on`, uloží MATLAB kopii každého grafického okna v obrazovkových bodech (pixel) do vyrovnávací paměti. Má-li být potom grafické okno rozsvíceno, je jeho

obsah kopírován z této paměti. Tím se na rozdíl od jeho úplné regenerace zvýší rychlost vykreslení na obrazovku.

Ale tyto vyrovnávací paměti spotřebovávají systémovou paměť. Objeví-li se omezení pro systémovou paměť, musíme nastavit `BackingStore` na `off`, abychom tuto vlastnost zrušili a vymazali paměť.

Ne všechny stroje, na kterých MATLAB běží, podporují `BackingStore`. Pokud počítač tuto vlastnost nepodporuje, způsobí nastavení `BackingStore` pouze vypsání zprávy. Žádný jiný vliv nemá.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy kurzor je na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu. Všimněme si, že vlastnost `CallBack` pro objekt uimenu nahrazuje `ButtonDownFcn`, ale objekty uicontrol mají jak vlastnost `CallBack`, tak i vlastnost `ButtonDownFcn`.

Children *vektor identifikátorů*

Děti objektu figure. Tato vlastnost je vektor identifikátorů, které identifikují objekty zobrazené v objektu figure. Dětmi objektu figure jsou objekty axes, uicontrol a uimenu. Objekty axes mají jako děti grafické objekty (line, surface , atd).

Clipping {`on`} | `off`

Režim ořezávání. V objektu figure má vždy hodnotu `on`.

Color *ColorSpec*

Barva pozadí. Tato vlastnost určuje barvu pozadí grafického okna. Barvu můžeme určit buď pomocí RGB hodnot nebo jedním z jmen, které je v MATLABu předdefinováno. Více informací viz část `ColorSpec`.

ColorMap *matice RGB hodnot typu* $m, 3$

Mapa barev objektu figure. Tato vlastnost je matice typu $(m, 3)$ obsahující hodnoty intenzit červené, zelené a modré (RGB), které definují m samostatných barev. Grafické příkazy mají k barvám v mapě barev přístup prostřednictvím indexů. Barvy jsou indexovány číslem jejich řádku. Např. index 1 určuje první RGB vektor, index 2 druhý vektor atd. Mapa barev může mít libovolnou délku, ale musí mít tři sloupce. Implicitní mapa barev objektu figure obsahuje 64 předdefinovaných barev.

Implicitní mapu barev můžeme nahradit naší vlastní mapou nebo si můžeme vybrat některou z předdefinovaných map barev. Chceme-li definovat vlastní mapu barev, musíme určit v prvním sloupci intenzitu červené složky, ve druhém sloupci intenzitu zelené a ve třetím sloupci intenzitu modré složky. Intenzity barev jsou reálná čísla

z intervalu $\langle 0, 1 \rangle$. Hodnota 0 značí žádnou intenzitu a hodnota 1 plnou intenzitu. Trojice RGB s hodnotami $[0, 0, 0]$ určuje černou barvu, $[1, 1, 1]$ definuje bílou barvu.

Mapy barev mají vliv na vyobrazení objektů surface, image a patch, ale obecně neovlivňují ostatní objekty. Více informací viz popis `colormap` a `ColorSpec`

CurrentAxes *identifikátor objektu axes*

Aktuální objekt axes v objektu figure. Tato vlastnost je identifikátor aktuálního objektu axes v objektu figure (tj. identifikátor, který se získá funkcí `gca`, je-li tento objekt figure aktuální). Ve všech objektech figure, v nichž existují děti axes, existuje vždy aktuální objekt axes. Zaktuálnit konkrétní objekt axes můžeme buď funkcí axes nebo příkazem `set`.

Pokud objekt figure nemá žádné děti axes, lze je vytvořit příkazem

```
get(gcf,'CurrentAxes')
```

Zároveň se získá jejich identifikátor.

CurrentCharacter *znak (pouze pro čtení)*

Poslední stisknutá klávesa. MATLAB nastavuje tuto vlastnost na poslední stisknutou klávesu v aktuálním grafickém okně.

CurrentMenu *identifikátor menu*

Identifikátor aktuálního menu v objektu figure. Tato vlastnost obsahuje identifikátor posledního vybraného menu. Specifikujeme-li jeden identifikátor menu pro více než jednu položku menu, musíme se zeptat této vlastnosti, abychom mohli určit, které menu je tímto identifikátorem vybrané.

CurrentObject *identifikátor*

Identifikátor aktuálního objektu. MATLAB nastavuje tuto vlastnost na identifikátor objektu, který je pod aktuálním bodem (viz vlastnost `CurrentPoint`). Je to nejvýše položený objekt v zásobníku. Tuto vlastnost můžeme využít k tomu, abychom určili, jaký objekt uživatel vybral.

CurrentPoint `[x, y]`

Pozice posledního stisknutí nebo uvolnění tlačítka myši v tomto objektu figure. MATLAB nastavuje tuto vlastnost na souřadnice (x, y) bodu `CurrentPoint`. Tento bod můžeme vybrat stisknutím nebo uvolněním tlačítka myši, dokud je kurzor uvnitř grafického okna. `CurrentPoint` je měřen od levého dolního rohu grafického okna v jednotkách, které jsou zadány vlastností `Units`.

Tuto vlastnost využijeme tehdy, chceme-li určit, kde v grafickém okně stiskl uživatel tlačítko myši.

FixedColors *matice typu (n,3) (pouze pro čtení)*

Tato vlastnost udává seznam všech pevných barev definovaných pro objekt figure. Pevné barvy jsou nezávislé na mapě barev objektu figure. Jsou to přímo definované barvy, které MATLAB používá, když explicitně specifikujeme barvu objektu.

Např. pokud uvedeme následující příkaz

```
line('Color',[0.2 0.4 0.6])
```

a pak se zeptáme na vlastnost `FixedColors` příkazem

```
fc=get(gcf,'FixedColor')
```

jsou proměnné `fc` přiřazeny hodnoty

```
fc=
0.0000 0.0000 0.0000
1.0000 1.0000 1.0000
0.2000 0.4000 0.6000
```

Vidíme, že je vytvořena černá ([0 0 0]) a bílá ([1 1 1]) barva, protože objekt figure má implicitně černé pozadí a na něm bílý text. Chceme-li změnit vlastnosti `Color` objektu figure např. na zelenou, je černá položka nahrazena [0 1 0].

Interruptible `yes | {no}`

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

U objektů figure rozhoduje tato vlastnost také o tom, zda mohou být funkce `Window ButtonDownFcn`, `WindowButtonMotionFcn` a `WindowButtonUpFcn` v průběhu své činnosti přerušeny či nikoliv.

Implicitní hodnota je opět hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím zpětného volání pracovat , dokud není akce ukončena.

Z toho např. vyplývá, že uživatel nemůže zvolit aktuální grafické okno interaktivně (myší), a tím změnit hodnotu aktuálního objektu figure, tj. hodnotu vrácenou funkcí `gcf`. Toto je zvláště vhodné jako prevence proti roztržení funkce zpětného volání způsobené stisknutím tlačítka myši netrpělivým uživatelem v době, kdy je funkce zpětného volání v akci.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

InvertHardCopy {on} | off

Změna barev při tisku z bílé na černém pozadí na černou na bílém pozadí. Tato vlastnost ovlivní pouze tiskový výstup. Implicitní barvou pozadí objektu figure je barva černá. Proto pro výstup na tiskárnu je implicitně nastavena inverze. Je-li `InvertHardCopy on`, změní MATLAB černou barvu pozadí na bílou a bílé osy, vynášecí čárky, popis os, atd. na černou barvu. Tato inverze ale nemá vliv na barvy objektů surface a patch.

Pokud výstupní zařízení nepodporuje barvy, jsou čáry a texty, jejichž barvy jsou jiné než černé a bílé, změněny na bílé a černé (které dávají větší kontrast s pozadím papíru).

Nastavíme-li `InvertHardCopy` na `off`, odpovídá barva tištěného výstupu zobrazení na obrazovce.

KeyPressFcn *řetězec*

Funkce zpětného volání. Vlastnost `KeyPressFcn` je analogická vlastnosti `ButtonDown Fcn`, ve které můžeme specifikovat funkci zpětného volání, která má být vyvolána tehdy, je-li stisknuta klávesa a odpovídající grafické okno je aktuální. Procedura zpětného volání se může dotázat vlastnosti `CurrentCharacter` na klávesu, která byla stisknuta, a tím způsobila vykonávání funkce zpětného volání.

MenuBar `none | figure`

Tato vlastnost nám dovoluje zobrazit nebo skrýt pruh s popisem menu, který je umístěn v horní části grafického okna. Ne na všech systémech jsou tyto pruhy s popisem menu v grafickém okně podporovány. Ale tam, kde tomu tak je, je implicitně nastaveno jejich zobrazování.

MinColormap *skalár (implicitně=64)*

Minimální počet použitých vstupů v mapě barev. MATLAB zachovává tabulku indexů užívaných jinými aplikacemi pro své důležité barvy, a vyhýbá se je využít. Je-li v chodu aplikace bez barevných intenzit, pak může být přístupno více než 200 úseků (slots). To znamená, že na obrazovce může existovat několik map barev MATLABu o 64 barvách (implicitní hodnota) a mnoho dalších pevných barev spolu se zbylými aplikacemi, a barvy na obrazovce jsou zobrazovány správně bez ohledu na to, je-li grafické okno aktivní nebo ne.

Není-li zde dost volných úseků pro správnou instalaci požadované mapy barev objektu figure, tj. způsobí-li tato instalace vyvolání chybných barev v dalších aplikacích, změní objekt figure násilně tolik úseků, kolik jich potřebuje k tomu, aby zvýšil hodnotu prostorů `MinColormap`. Je-li výsledný počet úseků menší než délka mapy barev, jsou samostatné vstupy mapy barev seskupeny spolu se svými sousedy tak, že počet všech skupin odpovídá počtu dostupných úseků v tabulce barev. V tomto případě všechny prvky dané skupiny zobrazují barvu prostředního prvku skupiny.

Toto schema neinterpoluje ztracené barvy, ale pouze vzorkuje požadovanou mapu barev tak, aby odpovídala číslu prostoru tabulky barev. Tento způsob je vhodný pro jemně se měnící mapy barev, méně již pro mapy barev s nepravidelně rozmístěnými barvami. Pro objekty image, jejichž mapy barev často obsahují nepravidelně umístěné barvy, je lepší nastavit `MinColormap` na počet různých barev v objektu image, a tím se vyvarovat zobrazení objektu image v chybných barvách i v případě, kdy je odpovídající grafické okno aktivní.

Je-li otevřeno více grafických oken se stejnou mapou barev nebo s malými mapami barev, jsou zobrazovány objekty současně ve svých správných barvách. Jsou-li mapy barev příliš velké a/nebo jsou velmi rozdílné, je pravděpodobné, že se okna, která nejsou aktivní, zobrazují v chybných barvách. Kliknutí na neaktivní okno (tím se změní na aktivní) způsobí obnovení správných barev.

Name *řetězec*

Nadpis grafického okna. Tato vlastnost je řetězec, který je zobrazen v nadpisovém pruhu grafického okna. Implicitně je tento parametr prázdný řetězec a nadpis okna je zobrazen jako `Figure No. 1`, `Figure No. 2`, atd. Zadáme-li do tohoto parametru řetězec, změní se nadpis okna na `Figure No. 1: <řetězec>`.

NextPlot `new | {add} | replace`

Jak přidat další graf. Tato vlastnost říká vestavěným grafickým funkcím vyšší úrovně `plot`, `plot3`, `fill`, `fill3` a grafickým funkcím ve tvaru m-souborů `mesh`, `surf`, `bar`, atd. jaký objekt figure mají použít. Parametr `new` znamená, že bude před kreslením vytvořen nový objekt figure, parametrem `add` (implicitní hodnota) bude pro kreslení použit aktuální objekt figure. Parametr `replace` znamená, že budou před kreslením resetovány všechny vlastnosti objektu figure kromě vlastnosti `Position` a zrušeny všechny děti objektu figure.

Tuto vlastnost používá příkaz `subplot`; `subplot(1,1,1)` nastaví vlastnost `NextPlot` na `replace`. Vyvoláme-li grafický příkaz, který používá příkazy `subplot` a mění vlastnosti objektu figure, dovoluje nastavení `NextPlot` na hodnotu `replace` následným grafickým příkazům, aby se vrátily k implicitnímu chování jednoduchého příkazu `plot`.

K nastavení vlastnosti `NextPlot` lze též použít m-soubor `newplot`. Vyvoláním funkce `newplot` v grafických m-funkcích jako `mesh`, `surf`, `bar`, atd. se před kreslením grafů provedou v závislosti na nastavení vlastnosti `NextPlot` požadované akce. Při psaní vlastních grafických příkazů bychom měli volat funkci `newplot` hned na začátku. (Viz např. m-soubor `pcolor`). Viz též vlastnost `NextPlot` objektů axes.

NumberTitle `{on} | off`

Číslo nadpisu grafického okna. Tato vlastnost určuje, zda je řetězec `Figure No. N`, kde `N` je číslo objektu figure, zapsán v horním okraji grafického okna.

PaperOrientation {`portrait`} | `landscape`

Orientace papíru. Tato vlastnost určuje, jak jsou tištěná grafická okna orientována na papíru. Orientace `portrait` má delší rozměr stránky svisle, orientace `landscape` má delší rozměr stránky vodorovně.

PaperPosition *4-prvkových vektor*

Poloha na tištěné stránce. Tato vlastnost je obdélník, který určuje umístění grafického okna na stránce. Obdélník definujeme prostřednictvím vektoru `rect` ve tvaru

```
rect=[zleva, zdola, šířka, výška]
```

kde `zleva` znamená vzdálenost levého dolního rohu obdélníku od levé strany papíru, `zdola` vzdálenost spodní čáry obdélníku od dolního okraje stránky. Tyto dvě vzdálenosti společně definují levý dolní roh obdélníku. Parametry `šířka` a `výška` definují rozměry tohoto obdélníku.

Jednotky, ve kterých je obdélník zadán, jsou definovány vlastností `PaperUnits`.

PaperSize [*šířka výška*] *(pouze pro čtení)*

Velikost papíru. Tato vlastnost obsahuje velikost aktuální hodnoty `PaperType` měřenou v jednotkách `PaperUnits`.

PaperType {`usletter`} | `uslegal` | `a4letter`

Typ výstupního papíru. Je-li vlastnost `PaperUnits` nastavena na hodnotu `normalized`, používá MATLAB tuto vlastnost pro měřítkování tištěných objektů figure tak, aby správně souhlasily se stránkou.

PaperUnits `normalized` | {`inches`} | `centimeters` | `points`

Použité jednotky. Tato vlastnost určuje jednotky, které se používají v definicích vlastností `PaperPosition` a `PaperSize`. Všechny jednotky jsou počítány z levého dolního rohu papíru. Normalizované jednotky transformují levý dolní roh stránky na $(0, 0)$ a horní pravý roh na $(1, 1)$. Palce, centimetry a body jsou absolutní jednotky (1 bod = 1/72 palce).

Tato vlastnost má vliv na vlastnosti `PaperSize` a `PaperPosition`. Je-li vlastnost `PaperUnits` změněna, je dobrým zvykem ji po ukončení našich výpočtů vrátit na její implicitní hodnotu, aby neovlivnila ostatní funkce, které předpokládají implicitní nastavení této vlastnosti.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu figure. Rodičem objektu figure je objekt root. Tato vlastnost je identifikátor objektu root, který je vždy roven 0.

Pointer {`arrow`} | `crosshair` | `watch` | `cross` | `topl` | `topr` | `botl` | `botr` | `circle` | `fleur`

Symbol kurzoru. Tato vlastnost definuje symbol, kterým je značena pozice kurzoru v grafickém okně.

Position *4-prvkový vektor*

Pozice objektu figure. Tato vlastnost určuje velikost a polohu grafického okna na obrazovce. Polohu tohoto obdélníku určíme vektorem rect ve tvaru

```
rect=[zleva, zdola, šířka, výška]
```

kde `zleva` a `zdola` definuje vzdálenost levého dolního rohu grafického okna od levého dolního rohu obrazovky, parametry `šířka` a `výška` definují rozměry grafického okna. Informace o použitých jednotkách viz vlastnost `Units`. Prvky zleva a zdola mohou být v systémech, které mají více než jeden monitor, záporné.

K získání této vlastnosti, můžeme využít také funkci `get`. Podobně funkcí `set` lze přemístit grafické okno na uvedenou pozici.

Resize {`on`} | `off`

Režim změny velikosti grafického okna. Tato vlastnost určuje zda uživatel může či nemůže změnit velikost okna použitím myši. Hodnota `on` povoluje uživateli změnu velikosti okna, hodnota `off` nikoliv.

SelectionType `normal | extend | alt | open` *(pouze pro UNIX)*

Typ výběru pomocí myši. Tato vlastnost je nastavena MATLABem, aby poskytla informace o posledním stisknutí tlačítka myši. Tato informace označuje typ provedeného výběru. Na různých systémech se mohou vlastní akce požadované pro provedení těchto typů volby lišit. Všechny typy volby však existují na všech systémech. Následující seznam popisuje základní systém XWindows.

- volba `normal` značí kliknutí (stisknutí a uvolnění) tlačítka 1, když je kurzor na objektu, který chceme vybrat.
- volba `extend` je provedena při stisknutí a držení klávesy **Shift** se současným provedením volby `normal`.

  Nebo, má-li myš více tlačítek, můžeme vyvolat volbu `extend` kliknutím tlačítka 2 (obvykle prostřední tlačítko) u myši se třemi tlačítky nebo kliknutím obou tlačítek myši současně v případě dvoutlačítkové myši.
- volba `alt`(ernate) vyžaduje, abychom drželi stisknutou klávesu **Ctrl**, zatímco se provádí volba `normal`, nebo abychom stiskli tlačítko 3 (pravé tlačítko myši) u dvou- nebo tří-tlačítkové myši.
- volba `open` je tvořena dvojitým kliknutím myši, je-li kurzor na objektu, který chceme vybrat. Máme-li vícetlačítkovou myš, musíme stisknout totéž tlačítko pro obě kliknutí.

ShareColors {`yes`} | `no`

Je-li ShareColors (sdílení barev) nastaveno na hodnotu `yes`, využívají aplikace systémovou paletu barev. Paleta barev obsahuje barvy, které je možno současně zobrazit na obrazovce. V případě, že aplikace vyžaduje barvu, která není zobrazována, je do

systémové palety přidána. V případě, že požadavky jedné nebo více aplikací na nové barvy přesáhnou počet volných míst v paletě, projeví se to nesprávným zobrazením některých barev. Je-li některé okno aktivní, je mu poskytnuto tolik barev, kolik jich požaduje. Toto množství je pouze omezeno zobrazovacími schopnostmi. Barvy, které již nemohou být zobrazeny jsou porovnány s obsahem palety a je jim přiřazena nejbližší možná barva z této palety. Kromě toho jsou porovnávány barvy všech neaktivních oken a nastavovány stejným způsobem. Tento postup značně snižuje nežádoucí změny barev při přepínání aktivních oken.

V některých případech může být ale toto chování nevhodné, např. má-li aplikace svou vlastní mapu barev a čas potřebný ke znovunastavení přidělování palety barev pro všechna ostatní okna je příliš velký. Potom je žádoucí, aby okno, jehož mapa barev má být nastavena, nesdílelo svoji paletu barev s ostatními okny, a tudíž nastavíme hodnotu vlastnosti `ShareColors` na `no`.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
set(gcf,'Tag','Pokusný obrázek')
```

označí aktuální objekt figure visačkou `'Pokusný obrázek'`. Je-li potom potřeba se někdy na tento objekt figure odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','Pokusný obrázek')
```

Type *řetězec (pouze pro čtení)*

Typ grafického objektu. Tato vlastnost identifikuje druh grafického objektu. U objektů figure je `Type` vždy řetězec `'figure'`.

Units {`pixels`} | `normal` | `inches` | `centimeters` | `points`

Použité jednotky. Tato vlastnost definuje jednotky, které MATLAB používá pro interpretaci velikosti a umístění dat. Všechny jednotky jsou měřeny od levého dolního rohu okna. Normalizované jednotky transformují levý dolní roh stránky na $(0, 0)$ a horní pravý roh na $(1, 1)$. Palce, centimetry a body jsou absolutní jednotky (1 bod = 1/72 palce).

Tato vlastnost má vliv na vlastnosti `CurrentPoint` a `Position`. Změníme-li hodnotu vlastnosti `Units`, je dobrým zvykem ji po ukončení našich výpočtů vrátit na její implicitní hodnotu, aby neovlivnila ostatní funkce, které předpokládají implicitní nastavení této vlastnosti.

Definujeme-li jednotky pomocí dvojic NázevVlastnost/PropertyValue během vytváření objektu, musíme nastavit vlastnost `Units` před určením těch vlastností, u nichž chceme, aby tyto jednotky používaly.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice, kterou chceme spojit s objektem. Objekt tato data nepoužívá, ale my je můžeme získat příkazem `get`.

Visible {`on`} | `off`

Viditelnost objektu. Vlastnost `Visible` určuje zda je či není objekt zobrazen na obrazovce. Je-li vlastnost `Visible` objektu figure `off`, je celé grafické okno neviditelné.

WindowButtonDownFcn *řetězec*

Funkce zpětného volání. Tato vlastnost nám dovoluje definovat funkci pro konkrétní grafické okno, kterou MATLAB provede tehdy, pokud se v tomto okně vyskytne událost *button down* (tj. když je stisknuto tlačítko myši v době, kdy je kurzor uvnitř okna).

MATLAB provádí vyhodnocení řetězce příkazem `eval`. Řetězec může být libovolný platný příkaz MATLABu nebo jméno m-souboru.

WindowButtonMotionFcn *řetězec*

Funkce zpětného volání. Tato vlastnost nám dovoluje definovat funkci pro konkrétní grafické okno, kterou MATLAB provede tehdy, pokud se v tomto okně vyskytne událost *motion* (tj. pohybuje-li se kurzor uvnitř grafického okna).

MATLAB provádí vyhodnocení řetězce příkazem `eval`. Řetězec může být libovolný platný příkaz MATLABu nebo jméno m-souboru.

WindowButtonUpFcn *řetězec*

Funkce zpětného volání. Tato vlastnost nám dovoluje definovat funkci pro konkrétní grafické okno, kterou MATLAB provede tehdy, pokud se v tomto okně vyskytne událost *button up* (tj. když je tlačítko myši uvolněno).

Akce uvolnění tlačítka je spojena s tím oknem, ve kterém se vyskytla akce stisknutí tlačítka. Proto, když je tlačítko uvolněno, aby generovalo akci *button up*, nemusí být kurzor nutně uvnitř grafického okna.

MATLAB provádí vyhodnocení řetězce příkazem `eval`. Řetězec může být libovolný platný příkaz MATLABu nebo jméno m-souboru.

Viz též `close`, `clf`, `gcf`, `axes`, `subplot`

Funkce Vyplnění mnohoúhelníků ve 2D.

Syntaxe

```
fill(x,y,c)
fill(x,y,'c')
fill(X,Y,C)
fill(X1,Y1,C1,X2,Y2,C2,...)
h=fill(...)
fill(x,y,c,PropertyName,PropertyValue,...)
```

Popis `fill(x,y,c)` vyplní barvou určenou vektorem `c` (délky alespoň 3) mnohoúhelníky ve 2D definované vektory `x` a `y`. Vrcholy mnohoúhelníku jsou určeny dvojicemi prvků vektorů `x` a `y`. Je-li to nezbytné, je mnohoúhelník uzavřen spojením posledního vrcholu s prvním.

Je-li `c` jednoduchý znakový řetězec ze seznamu `'r'`, `'g'`, `'b'`, `'c'`, `'m'`, `'y'`, `'w'`, `'k'` nebo jeden řádkový vektor ve tvaru trojice `[r g b]`, je mnohoúhelník vyplněn konstantní určenou barvou.

Je-li `c` vektor téže délky jako vektory `x` a `y`, jsou jeho prvky transformovány pomocí `caxis` a použity jako indexy pro určení barev vrcholů mnohoúhelníka aktuální mapy barev. Barva uvnitř mnohoúhelníku je získána bilineární interpolací barev vrcholů.

Jsou-li `X` a `Y` matice shodného typu, kreslí `fill(X,Y,C)` pro každý sloupec jeden mnohoúhelník. V tomto případě je pole `C` pro barvy mnohoúhelníku typu `flat` řádkový vektor, pro barvy mnohoúhelníku typu `interp` je `C` matice.

Je-li jedno z polí `X` nebo `Y` matice a druhé sloupcový vektor se stejným počtem řádků, je ze sloupcového vektoru opakováním sloupce vygenerována matice daného typu.

`fill(X1,Y1,C1,X2,Y2,C2,...)` je jiný způsob definování několika plných ploch.

`fill` nastaví v závislosti na hodnotách matice `C` vlastnost `FaceColor` u objektu patch na hodnoty `'flat'`, `'interp'` nebo *ColorSpec*.

`h=fill(...)` vrací sloupcový vektor identifikátorů objektů patch, pro každý objekt patch jeden identifikátor. k určení dalších vlastností mohou za trojicí `x`, `y`, `c` následovat dvojice PropertyName/PropertyValue. `fill` nereaguje na vlastnost `NextPlot`.

Příklady Vytvoření červené značky STOP (bez textu)

```
t=(1/16:1/8:1)'*2*pi;
x=sin(t); y=cos(t);
fill(x,y,'r')
axis('square')
```

Viz též `patch`, `fill3`, `colormap`

Funkce Vyplnění mnohoúhelníků ve 3D.

Syntaxe

```
fill3(x,y,z,c)
fill3(X,Y,Z,C)
fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,...)
h=fill3(...)
fill3(x,y,z,c,PropertyName,PropertyValue,...)
```

Popis `fill3(x,y,z,c)` vyplní barvou určenou vektorem `c` (délky alespoň 3) mnohoúhelníky ve 3D definované vektory `x`, `y` a `z`. Vrcholy mnohoúhelníku jsou určeny trojicemi prvků vektorů `x`, `y` a `z`. Je-li to nezbytné, je mnohoúhelník uzavřen spojením posledního vrcholu s prvním.

Je-li `c` jednoduchý znakový řetězec ze seznamu `'r'`, `'g'`, `'b'`, `'c'`, `'m'`, `'y'`, `'w'`, `'k'` nebo jeden řádkový vektor ve tvaru trojice `[r g b]`, je mnohoúhelník vyplněn konstantní určenou barvou.

Je-li `c` vektor téže délky jako vektory `x`, `y` a `z`, jsou jeho prvky transformovány pomocí `caxis` a použity jako indexy pro určení barev vrcholů mnohoúhelníku do aktuální mapy barev. Barva uvnitř mnohoúhelníku je získána bilineární interpolací barev vrcholů.

Jsou-li `X`, `Y` a `Z` matice shodného typu, kreslí `fill3(X,Y,Z,C)` pro každý sloupec jeden mnohoúhelník. V tomto případě je pole `C` pro barvy mnohoúhelníku typu `flat` řádkový vektor, pro barvy mnohoúhelníku typu `interp` je `C` matice.

Je-li některé z polí `X`, `Y` nebo `Z` matice a ostatní jsou sloupcové vektory se stejným počtem řádků, jsou ze sloupcových vektorů vygenerovány matice požadovaného typu opakováním sloupce.

`fill3(X1,Y1,Z1,C1,X2,Y2,Z2,C2,...)` je jiný způsob definice několik plných ploch.

`fill3` nastaví v závislosti na hodnotách matice `C` vlastnost `FaceColor` u objektu patch na hodnoty `'flat'`, `'interp'` nebo *ColorSpec*.

`h=fill3(...)` vrací sloupcový vektor identifikátorů objektů patch, pro každý objekt patch jeden identifikátor. k určení dalších vlastností objektu patch mohou za čtveřicí `x`, `y`, `z`, `c` následovat dvojice parametrů PropertyName/PropertyValue. `fill3` nereaguje na vlastnost `NextPlot`.

Příklady Vyplnění čtyř náhodných trojúhelníků barvou

```
colormap(cool)
fill3(rand(3,4),rand(3,4),rand(3,4),rand(3,4))
```

Viz též `patch`, `fill`, `colormap`

Funkce Nalezení objektů se zvolenými vlastnostmi.

Syntaxe

```
h=findobj('PropertyName1',PropertyValue1,...)
h=findobj(ObjectHandles,'PropertyName1',PropertyValue1,...)
h=findobj(ObjectHandles,'flat','PropertyName1',PropertyValue1,...)
h=findobj(ObjectHandles)
h=findobj
```

Popis

`h=findobj('PropertyName1',PropertyValue1,...)` vrací identifikátory objektů od úrovně objektu root dolů, jejichž vlastnosti odpovídají vlastnostem uvedeným v párových argumentech PropertyName/PropertyValue (jméno vlastnosti/hodnota vlastnosti) funkce `findobj`.

`h=findobj(ObjectHandles,'PropertyName1',PropertyValue1,...)` omezuje hledání na objekty, jejichž identifikátory jsou uvedeny v argumentu `ObjectHandles`, a jejich potomky.

`h=findobj(ObjectHandles,'flat','PropertyName1',PropertyValue1,...)` omezuje hledání pouze na objekty, jejichž identifikátory jsou uvedeny v argumentu `Object Handles`. Jejich potomci nejsou hledáni.

`h=findobj(ObjectHandles)` vrací identifikátory uvedené v argumentu `Object Handles` a identifikátory všech jejich potomků.

`h=findobj` vrací identifikátor objektu root (0) a identifikátory všech jeho potomků.

Poznámka Implementováno od verze 4.2.

Příklady Následující příkaz

```
h=findobj([2 3],'FontSize',12,'Color','g')
```

prohledá objekty figure 2 a 3 a všechny jejich potomky a ve vektoru `h` vrátí identifikátory těch prohledávaných objektů, u nichž bude existovat vlastnost `FontSize` a `Color` a tyto vlastnosti budou mít odpovídající hodnoty, tj. 12 resp. `'g'`.

Viz též `set`, `get`, `gcf`, `gca`

Funkce Hledá výskyt jednoho řetězce v druhém.

Syntaxe `k=findstr(s1,s2)`

Popis `k=findstr(s1,s2)` vrací všechny pozice výskytu kratšího ze dvou řetězců v delším řetězci.

Poznámka Implementováno od verze 4.1.

Příklady

```
s='How much wood would a woodchuck chuck?';

findstr(s,'a')
21
findstr(s,'wood')
[10 23]
findstr(s,'Wood')
[ ]
findstr(s,' ')
[4 9 14 20 22 32]
```

Viz též `strcmp`

Funkce Otevírá soubor nebo získává informaci o otevřených souborech.

Syntaxe

```
fid=fopen(filename)
fid=fopen(filename,permission)
[fid,message]=fopen(filename,permission,architecture)
fids=fopen('all')
[filename,permission,architecture]=fopen(fid)
```

Popis `fopen(filename,permission)` otevírá soubor, jehož jméno je uvedeno v řetězci `filename`, v módu určeném řetězcem `permission`.

Legální přístupové řetězce `permission` jsou:

- `'r'` Otevře soubor pro čtení.
- `'r+'` Otevře soubor pro čtení a zápis.
- `'w'` Smaže obsah existujího souboru nebo vytvoří nový soubor a otevře ho pro zápis.
- `'w+'` Smaže obsah existujího souboru nebo vytvoří nový soubor a otevře ho pro čtení a zápis.
- `'a'` Vytvoří a otevře nový soubor pro zápis nebo otevře existující soubor pro zápis na konec souboru.
- `'a+'` Vytvoří a otevře nový soubor pro čtení a zápis nebo otevře existující soubor pro čtení a zápis na konec souboru.
- `'W'` Jako 'w', ale bez automatického vyprázdnění vyrovnávací paměti.
- `'A'` Jako 'a', ale bez automatického výprázdnění vyrovnávací paměti.

Přístupové módy `'W'` a `'A'` jsou určeny pro použití s páskovými jednotkami a neprovádí automatické vyprázdnění aktuální výstupní vyrovnávací paměti po výstupních operacích. Např. pro otevření 1/4" páskové jednotky na pracovní stanici SPARC pro zápis bez vyprazdňování: `fid=fopen('/dev/rst0','W')`

Pokud přístupový řetězec chybí, funkce předpokládá `'r'`. K přístupovému řetězci lze též přidat `'b'` nebo `'t'` (např. `'rb'`) pro rozlišení textových a binárních souborů. Implicitně jsou soubory otevírány v binárním módu. (Na UNIXových systémech jsou textové a binární soubory stejné, takže toto rozlišení nemá smysl, ale na systémech PC, Macintosh a VMS, které rozlišují textové a binární soubory, je to velice důležité).

Pokud `fopen` úspěšně otevře soubor, vrátí identifikátor souboru `fid`, což je celé číslo větší než 2, a proměnná `message` bude prázdná. `fid` je používán ostatními souborovými funkcemi (`fid` je jejich prvním parametrem) k identifikaci souboru, na kterém mají vykonat své operace.

Automaticky jsou využitelné tři identifikátory souborů, které se nemusí otevírat. Jsou to: standardní vstup (`fid=0`), standardní výstup (`fid=1`) a standardní výstup chyb (`fid=2`).

Pokud `fopen` neotevře soubor, nastaví hodnotu `fid` na -1. V tomto případě nám řetězec `message` pomůže určit typ chyby, ke které došlo.

Pokud je soubor otevřen v módu `'r'` a není nalezen v aktuálním adresáři, funkce `fopen` začne prohledávat cesty MATLABu.

Volitelný řetězcový parametr `architecture` definuje číselný formát souboru, který nám umožňuje sdílet soubory na počítačích různých architektur. Parametr může být jedním z těchto řetězců:

`'native'` nebo `'n'`	pro číselný formát aktuálního počítače,
`'ISIEEE-LE'` nebo `'l'`	pro IEEE Little Endian formát,
`'ISIEEE-BE'` nebo `'b'`	pro IEEE Big Endian formát,
`'vaxd'` nebo `'d'`	pro VAX D-float formát,
`'vaxg'` nebo `'g'`	pro VAX G-float formát,
`'cray'` nebo `'c'`	pro Cray číselný formát.

Pokud nezadáme parametr `architecture`, je použit číselný formát lokálního počítače. Individuální volání funkce `fread` nebo `fwrite` může zastínit číselný formát určený při volání funkce `fopen`.

`fopen('all')` vrací řádkový vektor obsahující identifikátory všech uživatelem otevřených souborů (ne 0, 1 a 2). Počet prvků ve vektoru je roven počtu uživatelem otevřených souborů.

`[filename,permission,architecture]=fopen(fid)` vrací proměnné sdružené s daným souborem: `filename`, `permission` a `architecture`. V případě chybného parametru `fid` jsou vrácené řetězce prázdné. Řetězce `permission` a `architecture` jsou volitelné.

Viz též `fclose`, `ferror`, `fread`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Vykreslení grafu funkce.

Syntaxe

```
fplot(fun,limits)
fplot(fun,limits,n)
fplot(fun,limits,n,angle)
fplot(fun,limits,n,angle,subdiv)
[x,y]=fplot(fun,limits,...)
```

Popis `fplot` kreslí automaticky funkci v mezích daných argumentem `limits`. Funkce musí být ve tvaru `y=fun(x)`, kde `x` je vektor a `fun` je jméno funkce, která vrací vektor (stejné délky jako vektor `x`) obsahující vypočtené funkční hodnoty v bodech zadaných vektorem `x`. Je-li `fun` jméno vektorové funkce, vrací tato funkční hodnotu v bodech `x` tak, že každý prvek `fun(x)` je sloupec. Např. vrací-li `fun` vektor se třemi složkami a je-li `length(x)=10`, je `fun(x)` matice typu $(10, 3)$.

`fplot(fun,limits)` kreslí funkci definovanou proměnnou `fun` v mezích x-ové osy, které jsou zadány pomocí `limits=[xmin xmax]`.

`fplot(fun,limits,n)` kreslí funkci s minimálním počtem `n` bodů vzorkování. Implicitní hodnota `n` je 25.

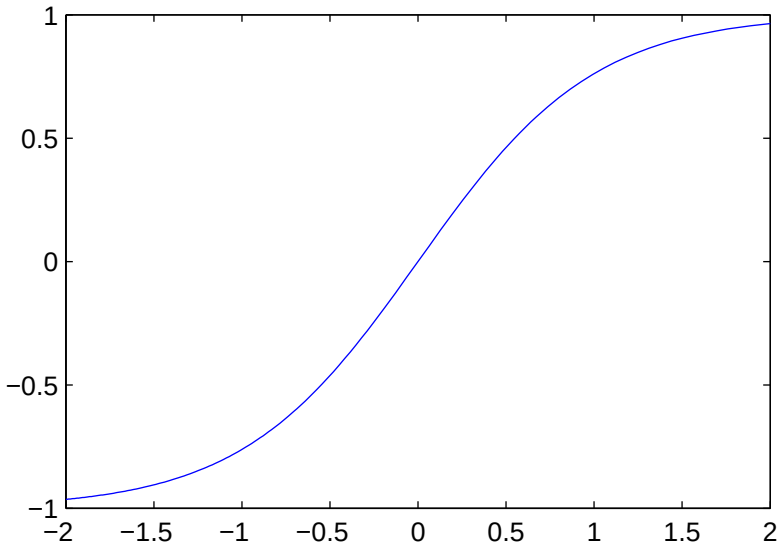
`fplot(fun,limits,n,angle)` kreslí funkci tak, že největší změna úhlu mezi jednotlivými segmenty kreslené funkce je menší než hodnota parametru `angle`. Implicitní hodnota pro parametr `angle` je 10 stupňů.

`fplot(fun,limits,n,angle,subdiv)` kreslí funkci tak, že největší změna v úhlu je dána hodnotou parametru `angle`, ale počet iterací nesmí překročit hodnotu zadanou parametrem `subdiv`. Implicitně je `subdiv=20`.

`[x,y]=fplot(fun,limits,...)` vrací souřadnice a pořadnice funkce `fun` ve sloupcových vektorech `x` a `y`. Na obrazovce není funkce vykreslena. Funkce pak může být zobrazena příkazem `plot(x,y)`.

Příklady Vykreslení funkce tangens hyperbolický v mezích od -2 do 2.

```
fplot('tanh',[-2 2])
```

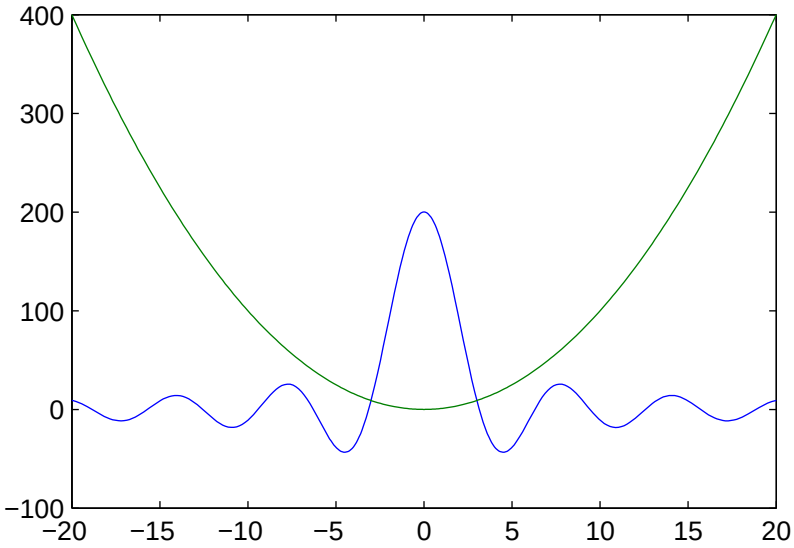


Funkce definovaná m-souborem

```
function y=myfun(x)
y(:,1)=200*sin(x(:))./x(:);
y(:,2)=x(:).^2;
```

je zobrazena příkazem

```
fplot('myfun',[-20 20],50,2)
```



Poznámka Ve verzi 4.2 byla tato funkce modifikována a používá místo úhlové techniky adaptivní lineární extrapolaci a techniku relativní chyby, což je spolehlivější algoritmus, který užívá méně bodů. Funkce nyní umožňuje specifikovat také y-ové meze, typ čáry nebo symbolu a jejich barvu. `fplot` dále povoluje zadávat funkce i s argumenty, tj. `'sin(x)'` nebo `'diric(x,10)'`.

Syntaxe[4.2]

```
fplot(fname,lims)
fplot(fname,lims,marker,tol)
[x,Y]=fplot(fname,lims,...)
```

Popis[4.2] `fplot(fname,lims)` kreslí funkci určenou řetězcovou proměnnou `fname` v mezích daných argumentem `lims=[xmin xmax]`. Tento argument může obsahovat i meze pro y-ovou osu `lims=[xmin xmax ymin ymax]`.

Funkce `fname` musí dodržovat sloupcový standard MATLABu. Např. pokud je `fname` funkce značící $f = [f_1(x)\ f_2(x)]$, potom pro vstup $[x_1\ x_2\ x_3]$ musí vrátit matici

$$f(x) = \begin{matrix} f_1(x_1) & f_2(x_1) \\ f_1(x_2) & f_2(x_2) \\ f_1(x_3) & f_2(x_3) \end{matrix}$$

Popř. `fname` smí být nějaký vyhodnotitelný řetězec s proměnnou `x`, např. `'sin(x)'`, `'diric(x,10)'` nebo `'[sin(x),cos(x)]'`.

`fplot` akceptuje dva volitelné argumenty: `marker` a `tol`. `marker` je řetězec, který určuje typ čáry nebo symbolu a jejich barvu. Implicitní hodnota je `'-'`. Oproti standardním typům funkce `plot` akceptuje `fplot` ještě: `'-+'`, `'-x'`, `'o'`, `'-*'` (nebo `'+-'`, `'x-'`, `'o-'`, a `'*-'`).

`tol` je přípustná relativní chyba. Implicitní hodnota je 2e-3. Maximální počet kroků je `(1/tol)+1`.

`[x,Y]=fplot(fname,lims,...)` vrací souřadnice a pořadnice funkce `fname` ve sloupcovém vektoru `x` a sloupcích matice `Y`.

Příklady[4.2]

```
fplot('sin',[0 4*pi])
fplot('sin(x)',[0 4*pi],'-+')
fplot('[sin(x),cos(x)]',[0 4*pi],'-x')
fplot('abs(exp(-j*x*(0:9))*ones(10,1))',[0 2*pi],'-o')
fplot('tan',[-2*pi 2*pi -2*pi 2*pi],'-*')
fplot('[tan(x),sin(x),cos(x)]',[-2*pi 2*pi -2*pi 2*pi])
fplot('sin(1./x)',[0.01 0.1],1e-3)
```

Viz též `plot`

Funkce Zapisuje formátovaná data do souboru.

Syntaxe
```
count=fprintf(fid,format,A,...)
fprintf(format,A,...)
```

Popis `fprintf(fid,format,A,...)` formátuje data z matice `A` (popř. dalších matic) podle formátovacího řetězce `format` a zapisuje je do souboru, který je sdružen s identifikátorem souboru `fid`.

`fid` je identifikátor souboru získaný funkcí `fopen` při otvírání souboru. Pro standardní výstup (obrazovka) má hodnotu 1 a pro standardní výstup chyb hodnotu 2. Na některých počítačích může být `fopen` použito se jmény zařízení; např. v prostředí MS-DOSu může `COM1` definovat sériový port odpovídající modemu. K odesílání znaků do modemu pak použijeme příkazy `fopen('COM1')` a `fprintf`.

Pokud nebude `fid` zadáno v seznamu argumentů funkce `fprintf`, použije funkce standardní výstupní zařízení, tj. `fid=1`.

`count` vrací počet úspěšně zapsaných bytů.

`format` je řetězec obsahující obyčejné znaky a formátovací specifikace. Obyčejné znaky zahrnují normální alfanumerické znaky a escape znaky. Escape znaky jsou

- `\n` nový řádek,
- `\t` horizontální tabelátor,
- `\b` znak zpět (backspace),
- `\r` začátek řádky (carriage return),
- `\f` odstránkování (form feed),
- `\\` zpětné lomítko,
- `\'` jednoduchá uvozovka.

Formátovací specifikace vyžaduje znak %, volitelně příznak, volitelně šířku pole, volitelně přesnost, volitelně podrobnější specifikaci typu a konverzní znak. Konverzní znaky jsou:

- `d` dekadické celé číslo,
- `i` dekadické celé číslo,
- `o` oktalové neznaménkové celé číslo,
- `u` dekadické neznaménkové celé číslo,
- `x` hexadecimální neznaménkové celé číslo (s **a,b,c,d,e,f**),
- `X` hexadecimální neznaménkové celé číslo (a **A,B,C,D,E,F**),
- `e` číslo s pohyblivou řádovou čárkou ([-]d.dddd **e**[-]ddd),
- `f` číslo s pohyblivou řádovou čárkou ([-]dddd.dddd),
- `g` číslo s pohyblivou řádovou čárkou ve tvaru **e** nebo **f**, co je kratší,

`E` číslo s pohyblivou řádovou čárkou (jako **e**, ale s exponenten **E**),
`G` číslo s pohyblivou řádovou čárkou (jako **g**, ale s exponentem **E**),
`c` jediný znak,
`s` řetězec.

Mezi znak `%` a konverzní znak (`e`, `f` nebo `g`) můžeme přidat jeden nebo více následujících znaků:

známénko mínus (-)	zarovnání výsledku doleva,
číselný řetězec	minimální počet znaků,
tečku	oddělení následujícího číselného řetězce,
číselný řetězec	přesnost (počet číslic vpravo od desetinné tečky).

Více informací o formátovacích řetězcích lze získat z referenčního manuálu jazyka C.

`fprintf` se chová jako jeho jmenovec v ANSI C s jistými výjimkami a rozšířeními:

Pokud se číslo MATLABu (přesnost double) netransformuje přesně na datový typ odpovídající konverznímu znaku, je potom použit formát `e`. Např.

```
A=[2 3 4; pi 2*pi 5.235; NaN Inf NaN];
fprintf(1,'%d\n',A)
2
3.141593e+00
NaN
3
6.283185e+00
Inf
4
5.235000e+00
NaN
```

Jestliže plánujeme použít celočíselnou konverzi, jako např. `d`, musíme explicitně transformovat reálnou hodnotu na hodnotu celočíselnou (funkce `floor`, `ceil`, `round` nebo `fix`).

Pro konverzní znaky `o`, `u`, `x` a `X` jsou podporovány následující nestandardní podrobnější specifikace typu:

`t` podporovaný datový typ jazyka C je float spíše než neznaménkové celé číslo,
`b` podporovaný datový typ jazyka C je double spíše než neznaménkové celé číslo.

Např. pro tisk čísla double v hexadecimálním tvaru použijeme `'%bx'`.

`fprintf` se liší od svého jmenovce z jazyka C ještě v jedné důležité vlastnosti a tou je vektorizace. Pokud vstupní matice `A` není skalární, formátovací řetězec se cyklicky prochází dokud se nevyčerpají všechny prvky matice `A`. Matice `A` se prochází po sloupcích.

Příklady Příkazy

```
x=0:0.1:1;
y=[x;exp(x)];
fid=fopen('exp.txt','w')
fprintf(fid,'%6.2f %12.8f\n',y);
fclose(fid);
```

vytvoří textový soubor pojmenovaný `exp.txt`, který obsahuje krátkou tabulku exponenciální funkce:

```
0.00 1.00000000
0.10 1.10517092
. . . . . .
1.00 2.71828183
```

Příkaz

```
fprintf('Jednotková kružnice má obvod %g.\n', 2*pi)
```

zobrazí na obrazovce řádek:

```
Jednotková kružnice má obvod 6.283186.
```

Příkazy

```
B=[8.8 7.7; 8800 7700]
fprintf(1,'X is %6.2f m nebo %8.3f mm\n',9.9,9900,B)
```

zobrazí řádky:

```
X is 9.9 m nebo 9900.000 mm
X is 8.8 m nebo 8800.000 mm
X is 7.7 m nebo 7700.000 mm
```

Viz též `fopen`, `fclose`, `fscanf`, `sprintf`, `ferror`, `fread`, `fwrite`, `fseek`, `ftell`, `diary`, `save`

Funkce Čte binární data ze souboru.

Syntaxe
```
A=fread(fid)
[A,count]=fread(fid,size,precision)
[A,count]=fread(fid,size,precision,skip)
```

Popis `fread` čte binární data z daného souboru a zapisuje je do matice `A`. `fread` se pokouší číst tolik čísel požadované přesnosti (`precision`), kolik je určeno parametrem `size`. Jestliže je použit výstupní parametr `count`, `fread` vrací počet úspěšně načtených prvků.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Parametr `size` je volitelný. Pokud je zadán, může mít následující tvar

`n`	Čte `n` prvků do sloupcového vektoru.
`inf`	Čte do konce souboru. Výsledkem je sloupcový vektor obsahující stejný počet prvků, jako je v souboru.
`[m,n]`	Čte tolik prvků, aby jimi (po sloupcích) vyplnil matici o rozměrech $[m, n]$. Pokud je prvků v souboru málo, doplní matici nulami.

Pokud není parametr `size` uveden, funkce předpokládá `inf`.

Parametr `precision` je volitelný. Pokud není uveden, funkce předpokládá `'uchar'`. Parametr určuje přesnost načítaných dat. `precision` řídí počet bitů načtených pro každou hodnotu a interpretaci těchto bitů (celé číslo, reálné číslo, znak, atd.). Ve všech případech prvky výsledné matice jsou uloženy v přesnosti double, tak jako je tomu u všech ostatních dat MATLABu.

Poznamenejme, že pokud `fread` dostihne konce souboru a současný vstupní proud neobsahuje dosti bitů k vypsání úplného maticového prvku požadované přesnosti, `fread` vycpává poslední byte nebo prvek nulovými bity, dokud není získána kompletní hodnota. Pokud se objeví chyba, je čtení zastaveno na poslední kompletní hodnotě.

Důležité upozornění: Numerické přesnosti jsou hardwarově závislé. Přesnosti uvedené v následující tabulce nemusí být podporovány na všech systémech. (Na počítačích CRAY jsou podporovány pouze přesnosti: `'char'`, `'schar'`, `'uchar'`, `'float'`, `'double'` a `'float64'`.)

Tato část předpokládá, že známe, jaká je reprezentace čísel na našem počítači. Pokud to nevíme, nahlédneme do referenčního manuálu k našemu počítači.

MATLAB podporuje všechny datové typy jazyka C i Fortranu vypsané v tabulce.

MATLAB	C nebo FORTRAN	Interpretace
'char'	'char', 'char*1'	znak; 8 bitů
'schar'	'signed char'	znaménkový znak; 8 bitů
'uchar'	'unsigned char'	neznaménkový znak; 8 bitů
'short'	'short'	celé číslo; 16 bitů
'ushort'	'unsigned short'	neznaménkové celé číslo; 16 bitů
'int'	'int'	celé číslo; 16 nebo 32 bitů
'uint'	'unsigned int'	neznaménkové celé číslo; 16 nebo 32 bitů
'long'	'long'	celé číslo; 32 bitů
'float'	'float'	reálné číslo; 32 bitů
'ulong'	'unsigned long'	neznaménkové celé číslo; 32 bitů
'float32'	'real*4'	32 bitové reálné číslo
'double'	'double'	reálné číslo; 64 bitů
'float64'	'real*8'	64 bitové reálné číslo
'intN'		znaménkové celé číslo; N bitů široké
'uintN'		neznaménkové celé číslo; N bitů široké

N představuje číslo mezi 1 a 64. Převládají následující ekvivalenty:

'int8'	'integer*1'	celé číslo; 8 bitů
'int16'	'integer*2'	celé číslo; 16 bitů
'int32'	'integer*4'	celé číslo; 32 bitů
'int64'	'integer*8'	celé číslo; 64 bitů

skip je volitelný parametr udávající kolik bytů se má přeskočit po každém čtení. Např. uvažujme soubor new_data skládající se z řady záznamů, jejichž položky jsou dlouhé 8, 16 a 4 byty. Z tohoto souboru chceme přečíst pouze 4-bytové položky. To provedeme následujícími příkazy:

```
fid=fopen('new_data');       % otevře soubor pro čtení
status= fseek(fid,24,'bof'); % nastaví identifikátor pozice
A=fread(fid,'float',24);     % čte až do konce souboru
```

Příklady Příkazy

```
fid=fopen('fread.m','r');
F=fread(fid);
s=setstr(F')
```

otevřou soubor 'fread.m', potom ho načtou a zobrazí; použije se implicitní velikost (size=inf) a přesnost (precision='uchar'). length(F) udává počet znaků v souboru.

Viz též `fopen`, `fclose`, `ferror`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`, `load`

Funkce Přetáčí otevřený soubor na začátek.

Syntaxe `frewind(fid)`

Popis `frewind(fid)` nastavuje identifikátor pozice souboru na začátek souboru sdruženého s identifikátorem souboru `fid`.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Varování:

Přetočení souboru, který je sdružen s páskovým zařízením, nemusí pracovat; nemusí být dokonce generována ani žádná chyba!

Poznámka Implementováno od verze 4.1.

Viz též `fopen`, `ferror`, `fread`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Čte formátovaná data ze souboru.

Syntaxe
```
[A,count]=fscanf(fid,format,size)
[A,count]=fscanf(fid,format)
```

Popis `[A,count]=fscanf(fid,format,size)` čte data ze souboru sdruženého s identifikátorem souboru `fid`, převádí je podle daného formátovacího řetězce `format` a vrací je do matice `A`. `count` je volitelný výstupní parametr, který vrací počet úspěšně načtených prvků.

`fid` je identifikátor souboru získaný funkcí `fopen`.

Parametr `size` je volitelný. Pokud je zadán, může mít následující tvar

`n`	Čte `n` prvků do sloupcového vektoru.
`inf`	Čte do konce souboru. Výsledkem je sloupcový vektor obsahující stejný počet prvků, jako je v souboru.
`[m,n]`	Čte tolik prvků, aby jimi (po sloupcích) vyplnil matici o rozměrech $[m, n]$. `n` může být `inf`, ale `m` nikoli.

Pokud není parametr `size` uveden, funkce předpokládá `inf`.

Řetězec `format` určuje formát dat, která se mají načíst. Když MATLAB čte daný soubor, pokouší se přizpůsobit data ze souboru formátovacímu řetězci. Pokud dojde k přizpůsobení, data jsou zapsána ve sloupcovém pořadí do matice. Pokud dojde jen k částečné shodě, jsou do matice zapsány pouze přizpůsobené části a čtení se zastaví.

`format` je řetězec obsahující obyčejné znaky a formátovací specifikace.

Formátovací specifikace určuje typ načítaných dat a vyžaduje znak %, volitelně hvězdičku potlačující přiřazení, volitelně šířku pole a konverzní znak. Konverzní znaky jsou

`d`	dekadické celé číslo,
`i`	dekadické celé číslo,
`o`	oktalové neznaménkové celé číslo,
`u`	dekadické neznaménkové celé číslo,
`x`	hexadecimální neznaménkové celé číslo (s **a,b,c,d,e,f**),
`X`	hexadecimální neznaménkové celé číslo (a **A,B,C,D,E,F**),
`e`	číslo s pohyblivou řádovou čárkou ([-]d.dddd **e**[-]ddd),
`f`	číslo s pohyblivou řádovou čárkou ([-]dddd.dddd),
`g`	číslo s pohyblivou řádovou čárkou ve tvaru **e** nebo **f**, co je kratší,
`E`	číslo s pohyblivou řádovou čárkou (jako **e**, ale s exponenten **E**),
`G`	číslo s pohyblivou řádovou čárkou (jako **g**, ale s exponentem **E**),

`c` jediný znak,
`s` řetězec.

Mezi znak `%` a konverzní znak můžete přidat jeden nebo více následujících znaků:

hvězdičku (*)	potlačuje přiřazení dalšího vstupního pole,
číselný řetězec	maximální počet znaků,
písmeno	potlačuje implicitní typ adresového argumentu; např. `h` (short) v `'%hd'` slouží k načtení krátkého celého čísla (short integer), `l` (long) v `'%ld'` k načtení dlouhého celého čísla (long integer) a v `'%lg'` k načtení reálného čísla v dvojnásobné přesnosti (double).

Více informací o formátovacích řetězcích lze získat z referenčního manuálu jazyka C.

`fscanf` se liší od svého jmenovce z jazyka C v jedné důležité vlastnosti a tou je vektorizace. Dokud není dosaženo konce souboru nebo počtu dat určeném parametrem `size`, formátovací řetězec je cyklicky aplikován na načítaná data.

Příklady Příklad v `fprintf` generuje ASCII soubor `exp.txt`, který vypadá takto:

```
0.00 1.00000000
0.10 1.10517092
. . . . . .
1.00 2.71828183
```

Načtěme tento soubor zpátky do dvousloupcové matice:

```
  fid=fopen('exp.txt');
  a=fscanf(fid,'%g %g',[2 inf]);
  a=a';
fclose(fid);
```

Viz též `fopen`, `fclose`, `ferror`, `fread`, `fwrite`, `fseek`, `ftell`, `fprintf`, `sprintf`, `sscanf`

Funkce Nastaví indikátor pozice v souboru.

Syntaxe `status=fseek(fid,offset,origin)`

Popis `fseek` posune indikátoru pozice v daném souboru (`fid`) o daný počet bytů (`offset`) vzhledem k počátku (`origin`).

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Hodnoty parametru `offset` jsou interpretovány následovně:

`offset > 0`	posune indikátor pozice o offset bytů směrem ke konci souboru,
`offset = 0`	nemění pozici,
`offset < 0`	posune indikátor pozice o offset bytů směrem k počátku souboru.

Hodnoty řetězcového parametru origin jsou interpretovány následovně

`'bof' (-1)`	od začátku souboru,
`'cof' (0)`	od současné pozice,
`'eof' (1)`	od konce souboru.

Hodnota výstupního parametru `status` je nastavena na 0, pokud proběhla operace úspěšně, a na -1 při chybě. Pokud se vyskyne chyba, použijte funkci `ferror` k získání více informací o povaze chyby.

Příklady Přetočení souboru na začátek:

```
fseek(fid,0,-1)
```

Viz též `fopen`, `fclose`, `ferror`, `fread`, `fwrite`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Vrátí hodnotu indikátoru pozice v souboru.

Syntaxe `position=ftell(fid)`

Popis `ftell` vrací pozici indikátoru pozice daného souboru (`fid`). Hodnota `position` udává počet bytů od začátku souboru.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Hodnota `position` je nezáporné celé číslo, pokud je dotaz úspěšný. Pokud je vráceno -1, znamená to, že dotaz byl neúspěšný; použijeme `ferror` k určení povahy chyby.

Viz též `fopen`, `fclose`, `ferror`, `fread`, `fwrite`, `fseek`, `fprintf`, `sprintf`, `fscanf`, `sscanf`

Funkce Zapisuje binární data z matice MATLABu do souboru.

Syntaxe
```
count=fwrite(fid,A,precision)
count=fwrite(fid,A,precision,skip)
```

Popis `fwrite` zapisuje prvky matice `A` do daného souboru, hodnoty jsou ukládány v přesnosti dané řetězcovým parametrem `precision`. Data jsou zapisována do souboru po sloupcích. `count` udává počet úspěšně zapsaných prvků.

`fid` je identifikátor souboru sdružený s otevřeným souborem. (Viz `fopen` pro úplný popis `fid`.)

Parametr `precision` udává numerickou přesnost dat (Viz `fread` pro více informací o `precision`).

`skip` je volitelný parametr udávající, kolik bytů se má přeskočit před každým zápisem. Když chcete přeskočit nějakou část souboru (např. hlavičku), nemusíte volat funkci `fseek`, jako tomu je při čtení dat (viz `fread`). Např. zápis do 4-bytové položky prvního záznamu popisovaného u funkce `fread` se provede následujícími příkazy:

```
fid=fopen('new_data','w');      % otevře soubor new_data pro zápis
count=fwrite(fid,A,'float',24); % čte až do konce souboru
```

Příklady Příkazy

```
fid=fopen('magic5.bin','wb')
fwrite(fid,magic(5),'integer*4')
```

vytvoří 100-bytový binární soubor, který obsahuje 25 prvků magického čtverce 5x5 uložených jako 4-bytová celá čísla.

Viz též `fopen`, `fclose`, `ferror`, `fread`, `fseek`, `ftell`, `fprintf`, `sprintf`, `fscanf`, `sscanf`, `save`, `diary`

Funkce Poskytuje identifikátor aktuálního objektu axes.

Syntaxe `h=gca`

Popis `gca` vrací identifikátor aktuálního objektu axes. Aktuální objekt axes je takový objekt, kde grafické příkazy jako `plot`, `title`, `surf`, atd. vytvářejí své výsledky.

Každý objekt figure má svůj vlastní aktuální objekt axes. Změna aktuálního objektu figure způsobí, že `gca` vrátí identifikátor objektu axes nového aktuálního objektu figure.

Ke změně aktuálního objektu axes na jiný objekt axes nebo k vytvoření nových objektů axes použijeme příkazy `axes` nebo `subplot`. Příkazem `cla` se resetuje aktuální objekt axes.

Viz též `axes`, `subplot`, `delete`, `cla`, `hold`, `gcf`

Funkce Poskytuje identifikátor aktuálního objektu figure.

Syntaxe `h=gcf`

Popis `gcf` vrací identifikátor aktuálního objektu figure. Aktuální objekt figure je objekt figure, do kterého jsou směrovány výsledky grafické příkazů jako `plot`, `title`, `surf`, atd.

Každý objekt figure má aktuální objekt axes.

Ke změně aktuálního objektu figure na jiný objekt figure nebo k vytvoření nového objektu figure použijeme příkaz `figure`. Příkazem `clf` resetujeme aktuální objekt figure.

Viz též `figure`, `subplot`, `delete`, `cla`, `hold`, `gcf`

Funkce Poskytuje identifikátor aktuálního objektu.

Syntaxe `object=gco`

Popis `object=gco` vrací identifikátor aktuálního grafického objektu aktuálního objektu figure.

`object=gco(figure)` vrací identifikátor aktuálního grafického objektu v objektu figure s identifikátorem `fig`.

Aktuální objekt daného objektu figure je objekt, na který bylo naposledy kliknuto myší.

Poznámka Implementováno od verze 4.1.

Viz též `figure`, `subplot`, `delete`, `cla`, `hold`, `gcf`

Funkce Poskytuje vlastnosti objektu.

Syntaxe

```
V=get(h,PropertyName)
get(h)
p=get(h)
```

Popis Funkci `get` použijeme k získání aktuálních hodnot vlastností objektu. Argument `h` určuje objekt, na jehož vlastnosti se ptáme.

POZOR! Na rozdíl od funkce `set`, zde nesmí být `h` vektor, pouze skalár.

`V=get(h,PropertyName)` vrací aktuální hodnotu vlastnosti objektu určené parametrem `PropertyName`, kterému přísluší identifikátor `h`.

`get(h)` bez názvu vlastnosti, uvede seznam všech vlastností, které patří k objektu s identifikátorem `h`, společně s jejich aktuálními hodnotami.

Dotaz na hodnotu libovolné implicitní vlastnosti provedeme zřetězením slova `Default` s typem objektu a názvem vlastnosti. Např. implicitní hodnotu vlastnosti `Color` pro objekt figure obdržíme následujícím příkazem

```
get(0,'DefaultFigureColor')
```

Chceme-li získat seznam všech implicitních hodnot aktuálně definovaných objektem pro své potomky, použijeme příkaz

```
get(h,'Default')
```

kde `h` je identifikátor objektu.

Příklady K obdržení identifikátoru objektu použijeme příkaz `get`. Např. následující příkaz vytváří objekt surface a objekt text

```
surf(peaks);
text(26,50,7,'Vrchol funkce peaks')
```

Jestliže chceme změnit barvu textu, ale při vytváření jsme neuložili jeho identifikátor, použijeme `get` k dotazu vlastností objektu v aktuálním objektu axes.

```
h=get(gca,'Children');
get(h(1),'Type')

ans=
text

get(h(2),'Type')
```

```
ans=
surface
```

V tomto případě `h(1)` obsahuje identifikátor objektu text. Můžeme tedy již nyní použít tento identifikátor pro nastavení barvy textu:

```
set(h(1),'Color',[1 0 0])
```

Seznam vlastností objektu surface a jejich aktuálních hodnot obdržíme použitím `get` s identifikátorem objektu surface:

```
get(h(2))

  CData=[(49 by 49)]
  EdgeColor=[0 0 0]
  EraseMode=normal
  FaceColor=flat
  LineStyle=-
  LineWidth=[0.5]
  MarkerSize=[6]
  MeshStyle=both
  XData=[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
  22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
  40 41 42 43 44 45 46 47 48 49]
  YData=[(49 by 1)]
  ZData=[(49 by 49)]

  ButtonDownFcn=
  Children=[]
  Clipping=on
  Interruptible=no
  Parent=[0.000610352]
  Type=surface
  UserData=[]
  Visible=on
```

Viz též `set`, `gca`, `gcf`

Funkce Získání dat pro animaci.

Syntaxe

```
M=getframe
M=getframe(h)
M=getframe(h,rect)
```

Popis `getframe` vrací sloupcový vektor s jedním animačním rámečkem. Rámeček je snímek (bitmapa) aktuálního objektu axes. `getframe` se většinou používá uvnitř smyčky `for` pro sestavení animační matice `M`, kterou používá `movie` pro animaci.

`getframe(h)` vezme rámeček z objektu `h`, kde `h` je identifikátor jednoho z objektů root, figure nebo axes.

`getframe(h,rect)` specifikuje navíc obdélník, vztahující se k dolnímu levému rohu objektu `h` v jednotkách určených ve vlastnosti `Units` tohoto objektu, z kterého se kopíruje bitmapa.

```
rect=[zleva zdola šířka výška]
```

Abychom zabránili použití nadměrné paměti, je vhodnější před generováním animace vytvořit nulovou animační matici `M` požadované velikosti funkcí `moviein`.

Příklady Kmitání funkce peaks:

```
z=peaks;
surf(z);
lim=axis;
M=moviein(20);
for j=1:20  % Záznam animace
  surf(sin(2*pi*j/20)*z,z)
  axis(lim)
  M(:,j)=getframe;
end
movie(M,20) % Přehrání animace dvacetkrát
```

Viz též `movie`, `moviein`

Funkce Grafický vstup pomocí myši v grafickém okně.

Syntaxe
```
[x,z]=ginput(n)
[x,z]=ginput
[x,z,button]=ginput(n)
[x,z,button]=ginput
```

Popis `ginput` poskytuje prostředky pro výběr bodů z grafického okna pomocí myši nebo kurzorových kláves. Kurzor musí být v grafickém okně, aby `ginput` vrátil souřadnice vybraných bodů.

`[x,z]=ginput(n)` bere `n` bodů z aktuálního objektu axes a vrací ve sloupcových vektorech `x` a `y` jejich souřadnice *(x,y)*.

Pro pohyb kurzoru použijeme myš (nebo na některých systémech kurzorové klávesy). Vstupní data (body) označujeme stisknutím tlačítka myši nebo klávesou na klávesnici. Vstup můžeme ukončit dříve než označíme všech `n` bodů stisknutím klávesy **Return**.

`[x,y]=ginput` shromažďuje neomezený počet bodů, dokud nestiskneme klávesu **Return**.

`[x,y,button]=ginput(n)` a `[x,y,button]=ginput` vrací navíc třetí parametr – button, který obsahuje vektor celých čísel, které určují, jaké bylo použito tlačítko myši (1, 2, 3 zleva) nebo jaká klávesa byla použita (v ASCII kódu). Neurčíme-li vstupní argument, shromažďuje `ginput` neomezený počet bodů až do stisknutí klávesy **Return**.

Viz též `plot`, `gtext`

Funkce Vykreslení teoretického grafu.

Syntaxe
```
gplot(A,xy)
gplot(A,xy,linetype)
[X,Y]=gplot(A,xy)
```

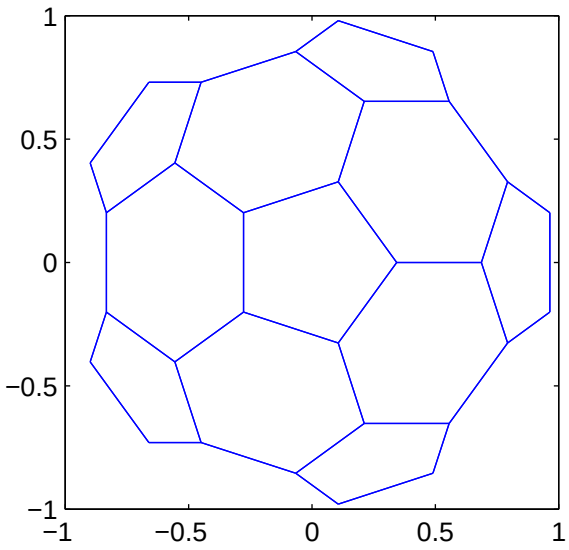
Popis `gplot(A,xy)` kreslí graf specifikovaný pomocí `A` a `xy`. Matice `A` je maticí spojení, která má nenulový prvek `A(i,j)` tehdy, je-li uzel `i` spojen s uzlem `j`. Pole `xy` je matice typu $(n, 2)$, která udává v i-tém řádku pozici pro uzel `i`, `xy(i,:)=[x(i) y(i)]`. n udává počet uzlů grafu.

`gplot(A,xy,linetype)` používá pro hrany místo implicitní hodntoty `'r-'` typ čáry a barvu definovanou parametrem `linetype` (viz `plot`). Např. `linetype='g:'`.

`[X,Y]=gplot(A,xy)` negeneruje graf, ale vrací vektory `X` a `Y` s využitím hodnot `NaN` k oddělení jednotlivých čar. Tyto vektory mohou být vykresleny později funkcí `plot`.

Příklady Graf odpovídající polovině míče vykreslíme následujícími příkazy:

```
[B,xy]=bucky;
gplot(B(1:30,1:30),xy(1:30,:))
```



Viz též `plot`, (`spy`)

Funkce Nastavení implicitních vlastností objektu figure při použití monitorů, které jsou schopny zobrazit pouze odstíny šedi.

Syntaxe `graymon`

Popis `graymon` mění implicitní grafické vlastnosti objektů figure z důvodu čitelného zobrazení na monitorech, které jsou schopny zobrazit pouze odstíny šedi.

Poznámka Implementováno od verze 4.1.

Funkce Čáry sítě pro grafy ve 2D a 3D.

Syntaxe
```
grid on
grid off
grid('on')
grid('off')
grid
```

Popis `grid on` přidá do aktuálního objektu axes čáry sítě.

`grid off` vypne čáry sítě (nezobrazí je).

`grid`, samostatně, slouží jako přepínač.

`grid('on')` je ekvivalentní `grid on` a `grid('off')` je ekvivalentní `grid off`.

Algoritmus `grid` nastavuje vlastnosti `XGrid`, `YGrid` a `ZGrid` aktuálního objektu axes.

Viz též `title`, `xlabel`, `text`, `plot`, `axes`

Vlastnosti `XGrid`, `YGrid` a `ZGrid` objektů axes.

Funkce Vzorkování dat.

Syntaxe
```
ZI=griddata(x,y,z,XI,YI)
[XI,YI,ZI]=griddata(x,y,z,XI,YI)
```

Popis `ZI=griddata(x,y,z,XI,YI)` vrací matici `ZI` s prvky, které odpovídají prvkům pravidelné sítě definované maticemi `XI` a `YI` a které jsou určeny interpolací (2D funkcí) z obvykle nerovnoměrně dělených vektorů `x`, `y` a `z`.

`XI` může být řádkový vektor, který v tomto případě určuje matici s konstantními sloupci. Podobně `YI` může být sloupcový vektor, který určuje matici s konstantními řádky.

`[XI,YI,ZI]=griddata(x,y,z,XI,YI)` vrací `XI` a `YI` tvořené tímto způsobem, které jsou stejné jako matice navrácené funkcí `meshgrid`.

`griddata` používá inverzní distanční metodu.

Příklady Navzorkování funkce ve 100 náhodných bodech mezi -2.0 a +2.0:

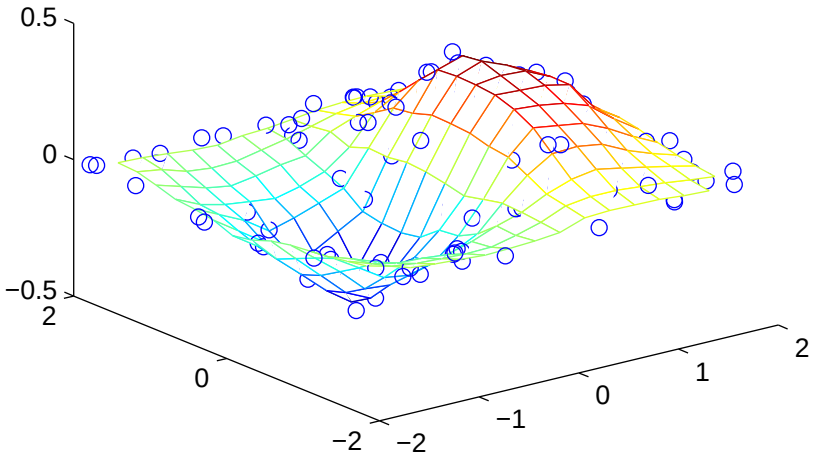
```
x=rand(100,1)*4-2;
y=rand(100,1)*4-2;
z=x.*exp(-x.^2-y.^2);
```

`x`,`y` a `z` jsou nyní vektory obsahující nerovnoměrně rozmístěná data. Definujme pravidelnou síť a vygenerujme na ní odpovídající data:

```
ti=-2:0.25:2;
[XI,YI]=meshgrid(ti,ti);
ZI=griddata(x,y,z,XI,YI);
```

Vykreslení dat na síti společně s nerovnoměrně rozloženými datovými body, které byly použity k jejich generování:

```
mesh(XI,YI,ZI)
hold on
plot3(x,y,z,'o')
hold off
```



Viz též (`interp1`, `interp2`)

Funkce Umístění textu do grafu pomocí myši.

Syntaxe
```
gtext(s)
h=gtext(s)
```

Popis `gtext(s)` čeká na stisknutí tlačítka myši nebo klávesy na klávesnici, zatímco kurzor myši je uvnitř grafického okna. Stisknutím tlačítka myši nebo klávesy na klávesnici se na zvolené místo v grafu zapíše textový řetězec `s`.

`h=gtext(s)` umístí textový řetězec `s` na zvolenou pozici v grafu a vrátí jeho identifikátor.

POZOR! `gtext` umístí text pouze do oblasti aktuálního objektu axes.

Algoritmus `gtext` používá funkce `ginput` a `text`.

Příklady Označení zajímavých částí aktuálního grafu řetězcem

```
gtext('Toto je maximum funkce!')
```

Viz též `ginput`, `text`

Funkce Vytvoří dialogový box nápovědy.

Syntaxe `h=helpdlg(helpstring,dlgname)`

Popis `h=helpdlg(helpstring,dlgname)` vytvoří dialogový box nápovědy, který zobrazí řetězec `helpstring` v objektu figure se jménem `dlgname`. Má-li dialogové okno zmizet, musíte stisknout tlačítko **OK**. Pokud dialogový box se jménem `dlgname` již existuje, přesune se pouze do popředí (nové dialogové okno se nevytvoří). `helpdlg` vrací identifikátor `h` objektu figure.

Poznámka Implementováno od verze 4.2.

Příklady `helpdlg('HELP message','HELP')`

Viz též `dialog`

Funkce Transformace hexadecimálního čísla na dekadické.

Syntaxe `d=hex2dec(s)`

Popis `hex2dec(s)` transformuje hexadecimální celé číslo `s` uložené jako řetězec do jeho dekadického tvaru.

Příklady `hex2dec('3ff')` vrací `1023`

Viz též `dec2hex`, `hex2num`, `format hex`

Funkce Transformace hexadecimálního čísla na číslo double.

Syntaxe `f=hex2num(S)`

Popis `hex2num(S)`, kde `S` je 16-ti znakový řetězec obsahující hexadecimální číslo, vrací reálné číslo v IEEE double přesnosti.

Pokud je znaků méně než 16, jsou doplněny na správnou délku nulami.

Příklady `hex2num('400921fb54442d18')` vrací π

`hex2num('bff')` vrací -1

Viz též `hex2dec`, `dec2hex`, `format hex`

Funkce Režim zobrazování skrytých čar grafu typu `mesh`.

Syntaxe
```
hidden on
hidden off
hidden('on')
hidden('off')
hidden
```

Popis `hidden on` zapíná odstranění skrytých (neviditelných) čar v aktuálním grafu, takže čáry v zadních ploškách `mesh` jsou skryty předními ploškami. Tato hodnota je implicitní.

`hidden off` vypíná odstraňování neviditelných čar.

`hidden`, samostatně, slouží jako přepínač.

`hidden('on')` je ekvivalentní `hidden on` a `hidden('off')` je ekvivalentní `hidden off`.

Algoritmus `hidden` nastavuje vlastnost `FaceColor` objektu surface. Odstranění skrytých čar odpovídá nastavení `FaceColor=BackgroungColor`, které je obvykle `black`. Zobrazení skrytých čar odpovídá nastavení `FaceColor=none`.

Viz též `shading`
Vlastnosti `FaceColor` a `EdgeColor` objektu surface.

Funkce Histogram.

Syntaxe

```
hist(y)
hist(y,nb)
hist(y,x)
[n,x]=hist(y)
[n,x]=hist(y,nb)
[n,x]=hist(y,x)
```

Popis `hist(y)` vykreslí histogram o 10 sloupcích pro data vektoru `y`. Sloupce jsou rovnoměrně rozmístěny mezi minimem a maximem vektoru `y`.

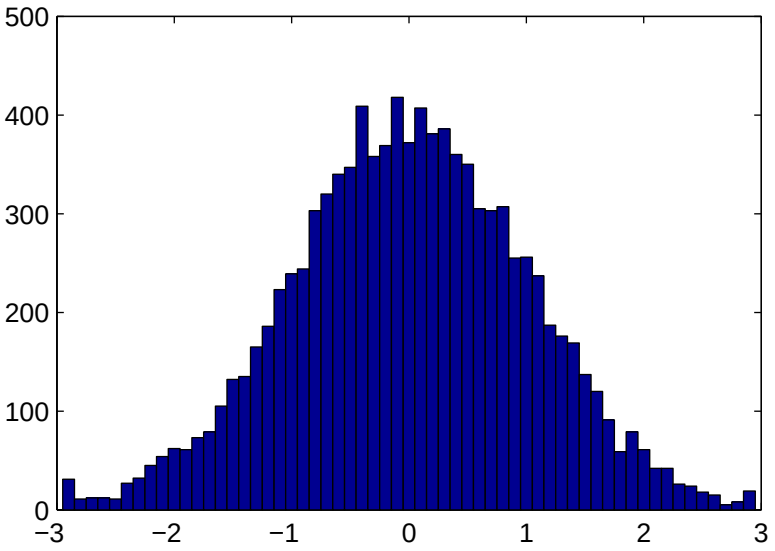
`hist(y,nb)` kreslí histogram s `nb` sloupci.

`hist(y,x)` kreslí histogram pomocí sloupců specifikovaných vektorem `x`.

`[n,x]=hist(y,...)`, `[n,x]=hist(y,nb)` a `[n,x]=hist(y,x)` nekreslí grafy, ale vrací vektory `n` a `x`, které obsahují počty frekvencí a umístění sloupců, takže histogram lze pak vykreslit příkazem `bar(x,n)`. To je užitečné v situacích, kdy vytváříme složitější graf s nastavením více vlastností.

Příklady Generování histogramu z Gaussových dat.

```
x=-2.9:0.1:2.9;
y=randn(10000,1);
hist(y,x)
```



Viz též `bar`, `stairs`

Funkce Drží aktuální graf.

Syntaxe
```
hold on
hold off
hold('on')
hold('off')
hold
```

Popis `hold on` podrží aktuální graf a všechny vlastnosti objektu axes, takže následné grafické příkazy přidávají své výsledky do existujícího grafu.

`hold off` přepíná do implicitního režimu, čímž nové příkazy pro kreslení vymažou předcházející graf a resetují před kreslením nových grafů všechny vlastnosti objektu axes.

`hold`, samostatně, slouží jako přepínač.

`hold` je vlastnost objektu axes. Existuje-li několik objektů axes, má každý svůj vlastní stav `hold`.

`hold('on')` je ekvivalentní `hold on` a `hold('off')` je ekvivalentní `hold off`.

Algoritmus `hold` pracuje s vlastností `NextPlot` objektů figure a axes.

`hold on` nastavuje vlastnost `NextPlot` aktuálních objektů figure a axes na `add`.

`hold off` nastaví vlastnost `NextPlot` aktuálního objektu axes na `replace`.

Viz též `axis`, `cla`
Vlastnost `NextPlot` objektů axes.

Funkce Mapa barev hsv.

Syntaxe

```
map=hsv
map=hsv(m)
```

Popis `hsv` je používána ve spojení s `colormap` pro určení mapy barev, která střídá složky barev barevného modelu `hsv` (hue-saturation-values). Barvy začínají červenou, pokračují přes žlutou, zelenou, tyrkysovou, modrou, fialovou a vrací se zpět na červenou. Tato mapa je zvláště vhodná k zobrazování periodických funkcí.

`hsv(m)` vrací matici typu $(m, 3)$, která obsahuje mapu barev `hsv`.

`hsv` samostatně je ekvivalentní příkazu `hsv(64)`.

Příklady Příkazy

```
format rat
M=hsv(19)
```

vytvoří

```
M =

1 0 0
1 1/3 0
1 2/3 0
1 1 0
2/3 1 0
1/3 1 0
0 1 0
0 1 1/3
0 1 2/3
0 1 1
0 2/3 1
0 1/3 1
0 0 1
1/3 0 1
2/3 0 1
1 0 1
1 0 2/3
1 0 1/3
1 0 0
```

Řádky matice `M`, které obsahují pouze hodnoty 0 a 1 reprezentují základní barvy – červenou, žlutou, zelenou, tyrkysovou, modrou, fialovou a opět červenou. Ostatní řádky představují pomocné přechodové barvy.

Algoritmus `hsv(m)` je totéž jako `hsv2rgb([h ones(m,2)])`, kde `h` je lineárně rostoucí.

```
h=(0:m-1)'/max(m-1,1);
```

V m-souboru `hsv.m` je nutno opravit odpovídající řádku podle výše uvedeného vztahu!

Viz též `hsv2rgb`, `rgb2hsv`, `colormap`

příslušné položky on-line helpu pro mapy barev `gray`, `hot`, `cool`, `bone`, `copper`, `pink`, `flag`, `jet`

Funkce Konverze HSV hodnot (hue-saturation-value) na RGB hodnoty (red-green-blue).

Syntaxe `M=hsv2rgb(H)`

Popis `M=hsv2rgb(H)` konvertuje mapu barev `hsv` na mapu barev `rgb`. Každá mapa je matice s libovolným počtem řádků, přesně třemi sloupci a s prvky v intervalu od 0 do 1. Sloupce vstupní matice `H` reprezentují barevný tón, sytost a jasovou hodnotu. Sloupce výsledné výstupní matice `H` reprezentují intenzitu červené, zelené a modré.

Protože první sloupec matice `H(:,1)` – barevný tón – probíhá hodnoty od 0 do 1, mění se výsledná barva od červené přes žlutou, zelenou, tyrkysovou, modou a fialovou zpět na červenou. Je-li `H(:,2)` – sytost – nulová, nejsou barvy saturované, jsou tvořeny pouze odstíny šedé. Je-li `H(:,2)` rovno 1, barvy jsou plně saturované, neobsahují žádné složky bílé barvy. Mění-li se `H(:,3)` – jasové hodnoty – od 0 do 1, zvyšuje se jas.

V MATLABu je implicitní mapou barev `hsv2rgb([h s v])`, kde `h` je lineární rostoucí od 0 do 1 a `s` i `v` jsou všechny rovny 1.

Viz též `colormap`, `hsv`, `rgb2hsv`, `brighten`

Funkce Práce s objektem image.

Syntaxe
```
image(C)
image(x,y,C)
h=image(x,y,C,PropertyName,PropertyValue,...)
```

Popis `image` je jak příkaz vyšší úrovně pro zobrazování obrazů, tak funkce nižší úrovně pro vytváření objektů image.

Objekty image jsou dětmi objektů axes. MATLAB zobrazuje objekt image pomocí transformace každého prvku v matici do mapy barev objektu figure. Funkce `image` dovoluje použít jako vstupní argumenty dvojice parametrů PropertyName/PropertyValue. Tyto vlastnosti, které ovládají různý vzhled objektu image, jsou popsány v oddíle *Vlastnosti objektu*. Hodnoty těchto vlastností lze též nastavit příkazem `set`. Příkazem `get` se můžeme dotázat na jejich aktuálně nastavenou hodnotu.

`image(C)` zobrazuje matici `C` jako obraz. Každý prvek matice `C` určuje barvu elementárního obdélníčku v objektu image. Prvky matice `C` jsou pro určení barvy použity jako indexy aktuální mapy barev.

`image(x,y,C)`, kde `x` a `y` jsou vektory, určuje hranice obrazu na osách x a y, a vytváří tentýž objekt image jako `image(C)`.

`h=image(...)` vrací identifikátor objektu image, který vytváří.

Za maticovými argumenty mohou následovat dvojice parametrů PropertyName/PropertyValue, které určují další vlastnosti objektu image. Maticové argumenty můžeme úplně vynechat a jen specifikovat vlastnosti dvojicemi parametrů PropertyName/PropertyValue.

`image` a `pcolor` jsou si podobné, ale mají některé důležité odlišnosti:

image(C)

- určuje barvy políček,
- používá pro vyplnění každého elementárního obdélníčku jednu barvu (pixel replication),
- není-li `hold` nastaveno na `on` (y-ová osa změněna), používá matici souřadnic,
- používá přímo indexy mapy barev.

pcolor(C)

- určuje barvy vrcholů,
- je-li zvolen odstín `interp`, používá pro vyplnění elementárních plošek bilineární interpolaci,

- přistupuje k mapě barev přes meze funkce `caxis`.

Z prvního bodu vyplývá, že počet obdélníčků u `image(C)` je shodný s počtem vrcholů u `pcolor(C)`.

Příklady Některé čistě matematické konstrukce generují zajímavé obrazy. Např.

```
colormap(cool)
image(((real(fft(eye(64)))+1)/2)*64)
```

Nebo můžeme načíst objekty image jako fotografie a vykreslit je z externích souborů:

```
load earth
colormap(map)
image(X)
```

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu pomocí dvojic parametrů PropertyName/PropertyValue jako argumentů funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy pointer je na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu. Všimněme si, že funkce `CallBack` pro objekt uimenu nahrazuje `ButtonDownFcn`, ale objekty uicontrol podporují jak své vlastní funkce `CallBack`, tak i funkci `ButtonDownFcn`.

CData *matice*

Barevná data. Tato vlastnost je matice hodnot, které určují barvu každého prvku objektu image. `image(C)` přiřadí matici `C` do `CData`. Každý prvek matice `CData` určuje barvu elementárního obdélníčku v objektu image. Prvky matice `CData` jsou použity jako indexy aktuální mapy barev pro určení barvy. Neceločíselné hodnoty jsou zaokrouhleny na nejbližší nižší celé číslo. Hodnoty vně rozsahu 1 až `length(colormap)` jsou uříznuty tím, že jsou zprůhledněny.

Clipping {`on`} | `off`

Režim ořezávání. Je-li `Clipping on`, nejsou žádné části objektu image, které leží mimo obdélník os zobrazeny.

Interruptible `yes | {no}`

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí ButtonDownFcn během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu image. Tato vlastnost je identifikátor rodiče objektu image, kterým vždy je objekt axes.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
load clown
colormap(map);
image(X,'Tag','klaun');
```

vytvoří objekt image (hlava klauna) a označí ho visačkou `'klaun'`. Je-li potom potřeba se někdy na tento obrázek odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','klaun')
```

Type *řetězec (pouze pro čtení)*

Typ objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekty image je `Type` vždy řetězec `'image'`.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice. kterou chceme přiřadit objektu. Objekt tato data nepoužije, ale my je můžeme získat příkazem `get`.

Visible `{on} | off`

Viditelnost objektu. Vlastnost `Visible` určuje, zda je či není objekt zobrazen na obrazovce.

XData `X[1 n]`
YData `Y[1 m]`

X a Y data. Tato vlastnost určuje pozice řádků a sloupců objektu image. Nejsou-li zadány, jsou místo `XData` a `YData` použita čísla řádek a čísla sloupců matice `CData`.

Viz též `pcolor`, `colormap`

Funkce Transformuje data a zobrazí je jako objekt image.

Syntaxe `imagesc(...)`

Popis `imagesc(...)` je stejné jako `image(...)` až na to, že jsou data transformována pro plné využití mapy barev.

Poslední volitelný argument `clims=[clow chigh]` je parametrem transformace.

POZOR! `imagesc` umisťuje informaci o transformačních parametrech do vlastnosti `UserData` příslušného objektu image, takže si funkce `colorbar` může zjistit údaje pro vytvoření smysluplných popisů.

Poznámka Implementováno od verze 4.1.

Viz též `image`, `colorbar`

Funkce Transformace celého čísla do řetězce.

Syntaxe `S=int2str(n)`

Popis `S=int2str(n)` transformuje celé číslo `n` do řetězcové reprezentace.

Viz též `num2str`, `sprintf`, `fprintf`

Funkce Test nastaveni funkce hold.

Syntaxe `ishold`

Popis `ishold` vrací 1, je-li `hold on`, a vrací 0, je-li `hold off`.

`hold on` znamená, že aktuální graf a všechny vlastnosti os jsou zachovány, takže následné grafické objekty jsou do aktuálního grafu přidány, tj. vlastnost `NextPlot` objektu figure i objektu axes má hodnotu `'add'`.

Poznámka Implementováno od verze 4.1.

Viz též `hold`
Vlastnost `NextPlot` objektů figure a axes.

Funkce Test znaku z abecedy.

Syntaxe `t=isletter(s)`

Popis Pro řetězec `s`, vrací `isletter(s)` vektor, jehož prvky mají hodnotu 1, pokud je odpovídající znak řetězce `s` obsažen v abecedě, jinak má hodnotu 0.

Poznámka Implementováno od verze 4.1.

Příklady Příkaz

```
isletter('***Ahoj***')
```

vrací vektor

```
[0 0 0 1 1 1 1 0 0 0]
```

Viz též `isstr`

Funkce Test *bílých* mezer.

Syntaxe `t=isspace(s)`

Popis Pro řetězec `s`, vrací `isspace(s)` vektor, jehož prvky mají hodnotu 1, pokud je odpovídající znak řetězce `s` *bílou* mezerou, jinak má hodnotu 0.

Bílou mezerou se rozumí mezera, znak nový řádek (LF), znak návrat vozíku (CR), tabulátor, vertikální tabulátor nebo znak odstránkování (FF).

Poznámka Implementováno od verze 4.2.

Příklady Příkaz

```
isspace('Ahoj !')
```

vrací vektor

```
[0 0 0 0 1 0]
```

Viz též `isletter`

Funkce Test řetězce.

Syntaxe `k=isstr(t)`

Popis Příkaz `t='Hello, World.'` vytvoří vektor, jehož složky jsou ASCII kódy znaků řetězce. Velikost vektoru `t` odpovídá počtu znaků. Tento vektor se nijak neliší od ostatních vektorů MATLABu, až na to, že je-li zobrazen, je zobrazen text místo dekadických ASCII kódů.

```
t=
Hello, World.
```

Každé proměnné MATLABu je přidělen příznak, který, pokud je nastaven, říká výstupním funkcím MATLABu, aby zobrazily proměnnou jako text.

`isstr(t)` vrací 1, je-li nastaven příznak zobrazení textu `t`, jinak vrací 0.

`isstr` pracuje také s maticemi.

Příklady

```
isstr('Hello')
1

isstr(abs('Hello'))
0
```

Viz též `setstr`, `strcmp`, `strings`

Funkce Legenda ke grafu.

Syntaxe

```
legend(string1,string2,string3,...)
legend(linetype1,string1,linetype2,string2,...)
legend(h,...)
legend(m)
legend(h,m)
legend off
legend(...,tol)
```

Popis `legend(string1,string2,string3,...)` vloží legendu do aktuálního grafu, tj. aktuálních os; zvolené textové řetězce použije jako popisky.

`legend(linetype1,string1,linetype2,string2,...)` specifikuje typ a barvu čar pro každý popis. Typ a barva čáry viz funkce `plot`.

`legend(h,...)` vloží legendu do grafu (os) s identifikátorem `h`.

`legend(m)`, kde `m` je řetězcová matice, použije pro popisky jednotlivé řádky matice `m` a `legend(h,m)`, kde `h` je vektor identifikátorů objektů line, popíše pouze čáry, jejichž identifikátory se nachází ve vektoru `h`.

`legend off` odstraní legendu z aktuálních os.

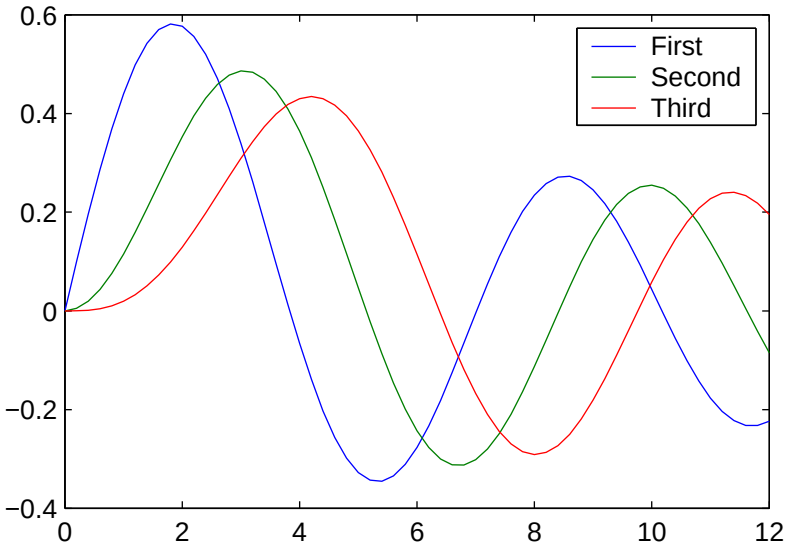
`legend(...,tol)` nastaví toleranci pro překrytí datových bodů. Pokud `legend` nenajde žádné místo, kde by překryl méně než `tol` datových bodů, zmenší `legend` graf a umístí legendu mimo. `tol=-1` způsobí umístění legendy mimo graf. `tol=0` umístí legendu na graf a neohlíží se na případné zakrytí datových bodů.

Když chcete legendu přesunout, stiskněte levé tlačítko myši na legendě a přesuňte ji na požadované místo.

Poznámka Implementováno od verze 4.2.

Příklady Popsání grafu Besselových funkcí prvních tří řádů.

```
x=0:.2:12;
plot(x,bessel(1,x),x,bessel(2,x),x,bessel(3,x));
legend('First','Second','Third');
```



Abyste se vyhnuli čárám sítě nebo grafu zakrývajícím legendu, vytvořte legendu aktuálních os před tiskem. Např.

```
h=legend('string')
axes(h)
print
```

Jsou-li osy legendy aktuální osy, nemusí být objekt figure překreslen. K vynucení překreslení použijte příkaz `refresh`.

Viz též `refresh`, `plot`

Funkce Vytváření objektu line.

Syntaxe
```
line(x,y)
line(x,y,z)
h=line(x,y,z,PropertyName,PropertyValue,...)
```

Popis `line` je grafická funkce nižší úrovně, která vytváří objekty line. Objekty line jsou dětmi objektů axes. Jsou to základní grafické objekty, které jsou použity pro generování grafů, vrstevnic a hran ploch. Funkce `line` dovoluje použít jako vstupní argumenty dvojice parametrů PropertyName/PropertyValue. Tyto vlastnosti, které řídí různý vzhled objektu line, jsou popsány v oddíle *Vlastnosti objektu*. Hodnoty těchto vlastností lze též nastavit příkazem `set`. Příkazem `get` se můžeme dotázat na jejich aktuálně nastavenou hodnotu.

Na rozdíl od funkcí jako je `plot`, nevymazává `line` osy, nenastavuje parametry pro prohlížení ani neprovádí jiné akce, než je generování objektu line v aktuálním objektu axes.

`line(x,y)` přidá do aktuálního objektu axes čáru zadanou vektory `x` a `y`. Jsou-li `x` a `y` matice stejného typu, vykreslí `line` čáru pro každý sloupec.

`line(x,y,z)` vytváří objekt line v 3D.

Za dvojicemi `x`, `y` (pro 3D za trojicemi `x`, `y`, `z`) mohou následovat dvojice parametrů PropertyName/PropertyValue, které určují další vlastnosti objektu line. Dvojice `x`, `y` (trojice `x`, `y`, `z`) můžeme úplně vynechat a jen specifikovat vlastnosti dvojicemi PropertyName/PropertyValue. Např. následující příkazy jsou ekvivalentní:

```
line('XData', x,'YData',y,'ZData',z)

line(x,y,z)
```

`line` je grafické funkce nižší úrovně, která není obvykle použita přímo. Místo ní jsou používány funkce `plot` a `plot3`.

`line` vrací sloupcový vektor identifikátorů, které odpovídají každému vytvořenému objektu line.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu dvojicemi PropertyName/PropertyValue, které jsou argumenty funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy kurzor je na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu. Všimněme si, že funkce `CallBack` pro objekt uimenu nahrazuje `ButtonDownFcn`, ale objekty uicontrol podporují jak své vlastní funkce `CallBack`, tak i funkci `ButtonDownFcn`.

Children *vektor identifikátorů*

Děti objektu. Objekty line nemají nikdy žádné děti, a proto je tato vlastnost vždy prázdná matice.

Clipping {`on`} | `off`

Režim ořezávání. Je-li `Clipping on`, nejsou žádné části objektu line, které leží mimo obdélník os zobrazeny.

Color *ColorSpec*

Barva čáry. Tato vlastnost určuje barvu čáry. Více informací o specifikaci barev viz `ColorSpec`.

EraseMode {`normal`} | `none` | `xor` | `background`

Režim mazání. Tato vlastnost řídí techniku MATLABu používanou pro kreslení a mazání objektů line. Tato vlastnost je užitečná při vytváření animovaných posloupností, protože řídí překreslování jednotlivých grafických objektů pro získání požadovaných efektů.

Režim `normal` překresluje danou oblast obrazovky tím, že provádí trojrozměrnou analýzu, která je nezbytná pro zabezpečení správného vynesení všech objektů. Tento režim přináší nejpřesnější obrázek, ale je také časově nejnáročnější. Ostatní režimy neprovádějí úplné překreslování, a proto jsou značně rychlejší, ale vytvářejí méně přesný obraz.

Je-li nastaven režim `none`, není objekt při přesunu nebo zničení vymazán.

Při režimu `xor` je objekt kreslen a mazán pomocí funkce `xor` aplikovanou na původní barvu. Je-li objekt vymazán, nepoškodí objekty pod ním. Ale je-li objekt kreslen v režimu `xor`, závisí jeho barva na barvě obrazovky pod ním. Je správně vybarven pouze, je-li nad barvou pozadí objektu figure.

Režim `background` vytváří správně barvené objekty. Avšak objekt je mazán svým vykreslováním na barvě pozadí objektu figure. Tím se poškodí objekty, které jsou za mazaným objektem.

Interruptible `yes` | {`no`}

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota

je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

LineStyle *typ čáry (implicitně '-')*

Typ čáry. Tato vlastnost určuje typ použité čáry. Požadovaný typ čáry určíme řetězcem, který obsahuje uvedené symboly z následujícího seznamu; např. `'-'` pro plnou čáru. Plus, bod, hvězdička, kroužek a značka x jsou měřítkovatelné značky.

plná	`-`	kroužek	`o`
čárkovaná	`--`	plus	`+`
tečkovaná	`:`	bod	`.`
čerchovaná	`-.`	hvězdička	`*`
		značka x	`x`

LineWidth *šířka*

Šířka čáry. Nová šířka je určena v bodech (1 bod = 1/72 palce). Implicitní hodnota je 0.5 bodu.

MarkerSize *skalár (implicitně 6 bodů)*

Měřítkovací faktor velikosti značek. Je-li vlastnost `LineStyle` nastavena na značky, měřítkuje tato vlastnost jejich velikost. Používá se pouze pro značky typu bod, plus, hvězdička, kroužek a značka x (1 bod = 1/72 palce).

Parent *identifikátor (pouze pro čtení)*

Rodič objektu line. Tato vlastnost je identifikátor rodiče objektu line, kterým vždy je objekt axes.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
line(sort(rand(1,100)),rand(1,100),'Tag','šum');
```

vytvoří objekt line a označí ho visačkou `'šum'`. Je-li potom potřeba se někdy na tento objekt odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','šum')
```

Type *řetězec (pouze pro čtení)*

Typ objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekty line je `Type` vždy řetězec `'line'`.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice. kterou chceme přiřadit objektu. Objekt tato data nepoužije, ale my je můžeme získat příkazem `get`.

Visible {`on`} | `off`

Viditelnost objektu. Vlastnost `Visible` určuje, zda je či není objekt zobrazen na obrazovce.

XData *vektor*
YData *vektor*
Zdata *vektor*

Souřadnicová data. Tyto vlastnosti určují data použitá ke generování čar. Jsou-li data ve formě matic, je každý sloupec interpretován jako samostatná čára. V tomto případě musí mít všechna data stejný počet řádků. Má-li jeden argument pouze jediný sloupec, udělá funkce line jeho kopii tak, aby počet odpovídal počtu sloupců specifikovaných v ostatních argumentech. Např. následující příkaz použije jeden sloupec specifikovaný v `ZData` dvakrát, aby mohl vytvořit dvě čáry, každou se čtyřmi body.

```
line(rand(4,2),rand(4,2),rand(4,1))
```

Mají-li všechna data stejný počet sloupců, ale pouze jediný řádek, transponuje MATLAB matice tak, aby obsahovala taková data, která lze vykreslit. Např.

```
line(rand(1,4),rand(1,4),rand(1,4))
```

je změněno na

```
line(rand(4,1),rand(4,1),rand(4,1))
```

Tento způsob se též aplikuje v případě, kdy pouze jedna nebo dvě matice mají jeden řádek. Např. příkaz

```
line(rand(2,4),rand(2,4),rand(1,4))
```

je ekvivalentní příkazu

```
line(rand(4,2),rand(4,2),rand(4,1))
```

Viz též `patch`, `text`, `plot`, `plot3`

Funkce Nahrává proměnné z disku.

Syntaxe

```
load
load filename
load filename -ascii
load filename -mat
```

Popis `Load` a `save` jsou příkazy MATLABu, které slouží pro nahrání a uložení proměnných ze/do souboru. Dále mohou sloužit k importu a exportu ASCII souborů.

Binární soubory MATLABu mají příponu `'.mat'` a obsahují proměnné v přesnosti double. Tyto soubory jsou vytvořeny příkazem `save` a čitelné příkazem `load`. Soubory mohou být vytvořeny na jednom počítači a později přečteny MATLABem na jiném počítači, který má jiný formát reálných čísel. Kromě MATLABu mohou s těmito soubory pracovat i jiné programy.

Příkaz

```
load
```

samostatně, načte všechny proměnné uložené v souboru `matlab.mat`.

Příkaz

```
load my_data
```

načte proměnné z binárního souboru `my_data.mat`.

Příkaz

```
load our_data.xyz
```

čte ASCII soubor `our_data.xyz`, který musí obsahovat obdélníkové pole numerických hodnot, uspořádaných do m řádků s n hodnotami v každém řádku. Výsledek je matice typu (m, n) se stejným jménem, jako je název souboru bez přípony.

Pokud chceme načíst ASCII soubor, který neobsahuje žádnou příponu, musíme použít příkaz `load filename -ascii`; např. `load my_data -ascii`. Jinak MATLAB přidá příponu `'.mat'` a pokusí se načíst soubor jako soubor MAT.

Pokud chceme načíst soubor MAT, který neobsahuje příponu `'.mat'`, musíme použít příkaz `load filename -mat`; např.

```
load my_data.dat -mat
```

Jestliže je parametr `filename` roven `'stdio'`, čte příkaz `load` ze standardního vstupu.

Příklady Jestliže soubor `sample.dat` obsahuje čtyři řádky numerického textu

```
1.1 1.2 1.3
2.1 2.2 2.3
3.1 3.2 3.3
4.1 4.2 4.3
```

potom příkaz

```
load sample.dat
```

vytvoří v pracovním prostoru matici typu $(4, 3)$ se jménem `sample`.

Algoritmus Příručka *External Interface Guide* popisuje podrobně strukturu souborů MAT. *External Interface Library* obsahuje procedury v jazyku C a Fortranu pro práci (čtení a zápis) se soubory MAT.

Viz též `save`, `spconvert`, `fscanf`, `fprintf`

Funkce Logaritmická stupnice os x a y.

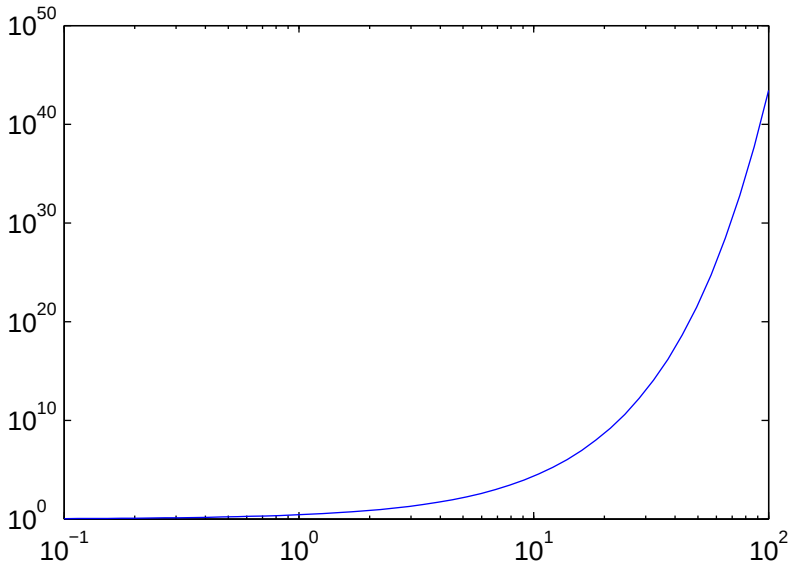
Syntaxe

```
loglog(x,y)
loglog(x,y,linetype)
loglog(x1,y1,linetype1,x2,y2,linetype2,...)
h=loglog(x,y,PropertyName,PropertyValue)
```

Popis `loglog(...)` je totéž jako `plot(...)`, ale pro obě osy x a y je použita logaritmická stupnice.

Příklady Jednoduchý graf:

```
x=logspace(-1,2);
loglog(x,exp(x))
```



Viz též `plot`, `semilogx`, `semilogy`

Funkce Transformuje velká písmena v řetězci na malá.

Syntaxe `t=lower(s)`

Popis `lower(s)` vrací řetězec vytvořený z původního řetězce `s` transformací velkých písmen na malá a ponecháním ostatních znaků beze změny.

Algoritmus Všechny hodnoty v rozsahu `'A':'Z'` jsou zmenšeny o `'A'-'a'`.

Příklady

```
lower('MathWorks')
'mathworks'
```

Viz též `upper`, `isstr`, `strcmp`

Funkce Drátový model plochy ve 3D.

Syntaxe

```
mesh(X,Y,Z,C)
mesh(X,Y,Z)
mesh(x,y,Z,C)
mesh(x,y,Z)
mesh(Z,C)
mesh(Z)
h=mesh(...)
meshc(...)
meshz(...)
```

Popis Úplná diskuse parametrických ploch je provedena u popisu funkce `surf`.

V nejobecnější podobě má `mesh` čtyři vstupní argumenty.

`mesh(X,Y,Z,C)` kreslí barvou určenou maticí `C` barevné čáry na parametrické ploše určené maticemi `X`, `Y` a `Z`. V jednodušším případě mohou `X` a `Y` být vektory, popř. je můžeme vynechat. Matici `C` lze též vynechat.

Bod pohledu je určen funkcí `view`. Popisy os jsou buď určeny rozsahem polí `X`, `Y` a `Z` nebo aktuálním nastavením `axis`. Transformace barev je dána rozsahem matice `C` nebo aktuálním nastavením `caxis`. Transformované barevné hodnoty jsou použity jako indexy aktuální mapy barev.

`mesh(X,Y,Z)` používá `C=Z`, takže barva je úměrná výšce plochy.

`mesh(x,y,Z,C)` a `mesh(x,y,Z)` s dvěma vektorovými argumenty na místo maticových argumentů musí mít `length(x)=n` a `length(y)=m`, kde `[m,n]=size(Z)`. V tomto případě jsou průsečíky čar trojice `(x(j),y(i),Z(i,j))`. Vidíme, že `x` odpovídá sloupcům a `y` řádkům matice `Z`.

`mesh(Z,C)` a `mesh(Z)` používá `x=1:n` a `y=1:m`.

`h=mesh(...)` vrací navíc identifikátor objektu surface. Objekty surface jsou dětmi objektů axes.

`meshc(...)` navíc kreslí pod `mesh(...)` vrstevnice.

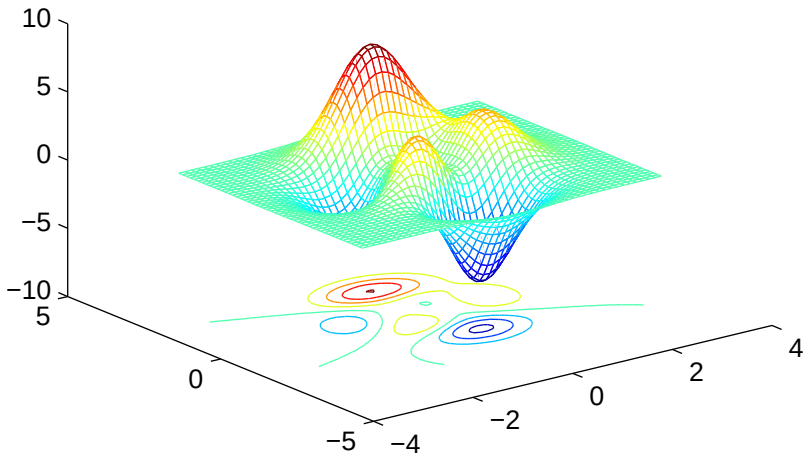
`meshz(...)` kreslí navíc pod `mesh(...)` referenční rovinu spojenou s plochou *záclonou*.

Algoritmus `meshc` volá `mesh`, nastavuje `hold` na `on`, a pak volá `contour` s posunutím. Pokud nám tento postup nevyhovuje, můžeme přímo použít odpovídající příkazy. Tímto způsobem můžeme kombinovat i další typy grafů, např. grafy vytvořené příkazy `pcolor` a `surf`.

Omezení `meshc` předpokládá, že jsou vektory `x` a `y` monotónně rostoucí. Jsou-li prvky vektorů `x` a `y` nerovnoměrně rozmístěny, jsou vrstevnice *protaženy*, což neodpovídá vrstevnicím kresleným pro rovnoměrně rozmístěné prvky vektorů `x` a `y`.

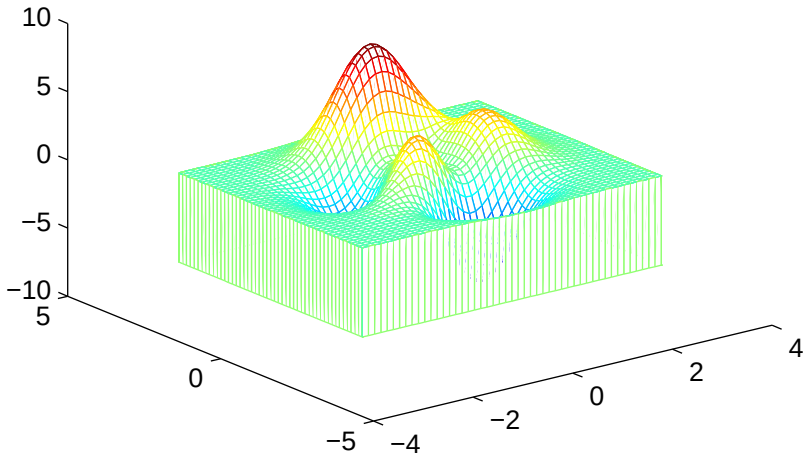
Příklady Vytvoření kombinovaného drátového modelu plochy `peaks` a jejích vrstevnic.

```
[x,y]=meshgrid(-3:0.125:3);
z=peaks(x,y);
meshc(x,y,z)
```



Generování drátového modelu plochy `peaks` se *záclonou*

```
[x,y]=meshgrid(-3:0.125:3);
z=peaks(x,y);
meshz(x,y,z)
```



Viz též `axis`, `caxis`, `colormap`, `hold`, `shading` a `view` nastavují vnitřní parametry MATLABu, které mají vliv na `mesh`.
`surf`, `surfc`, `surfl`, `pcolor`, `image` a `contour` poskytují alternativy.

Funkce Generování matic X a `Y` pro grafy 3D.

Syntaxe
```
[X,Y]=meshgrid(x,y)
[X,Y]=meshgrid(x)
```

Popis `[X,Y]=meshgrid(x,y)` transformuje oblast určenou vektory `x` a `y` do matic `X` a `Y`, které mohou být použity pro vyhodnocení funkcí dvou proměnných a grafů drátových modelů ploch, popř. ploch ve 3D. Řádky výstupní matice `X` jsou kopiemi vektoru `x` a sloupce výstupní matice `Y` jsou kopiemi vektoru `y`.

`[X,Y]=meshgrid(x)` je zkráceným zápisem příkazu `[X,Y]=meshgrid(x,x)`

Příklady Vyhodnocení a vykreslení funkce $z = x\,e^{-x^2-y^2}$ v oblasti $-2 \leq x \leq 2$, $2 \leq y \leq 2$.

```
[X,Y]=meshgrid(-2:0.2:2);
Z=X.*exp(-X.^2-Y.^2);
mesh(Z)
```

Viz též `surf`, `mesh`

Funkce Přehrávání zaznamenané animační matice.

Syntaxe

```
movie(M)
movie(M,n)
movie(M,n,fps)
movie(h,...)
movie(h,m,n,fps,loc)
M=moviein(n)
```

Popis

`movie(M)` přehraje jednu animaci z matice `M`. `M` musí být matice, jejíž sloupce jsou animační snímky (obvykle z `getframe`)

`movie(M,n)` přehraje animaci `n` krát. Je-li `n` záporné, je každé přehrání provedeno jednou vpřed a jednou vzad. Je-li `n` vektor, určují jeho prvky od druhého výše pořadí, ve kterém jsou snímky přehrány. Např. je-li `M` matice o třech sloupcích, `n=[10 3 2 1]` přehraje animaci desetkrát ve zpětném pořadí.

`movie(M,n,fps)` přehraje animaci rychlostí `fps` snímků za sekundu. Implicitně (pokud není `fps` zadáno) je `fps=12`. Počítače, které nemohou dosáhnout specifikované rychlosti, přehrají snímky takovou rychlostí, jaké jsou schopny.

`movie(h,...)` přehraje animaci v objektu `h`, kde `h` je identifikátor objektu figure nebo objektu axes.

`movie(h,m,n,fps,loc)` specifikuje navíc pozici k přehrání animace vzhledem k dolnímu levému rohu (`loc=[x y]`) objektu `h` v jednotkách jeho vlastnosti `Units`.

`M=moviein(n)` vytváří dostatečně velkou animační matici k uchování `n` snímků animace na základě aktuálního grafického okna. Matice má dostatek řádků tak, aby uložila `n` kopií výstupu z `getframe`, pro každý sloupec jednu. `moviein` je jednoduchý m-soubor, který obsahuje jedinou řádku:

```
M=zeros(length(getframe),n)
```

Příklady Kmitání funkce `peaks`:

```
z=peaks;
surf(z);
lim=axis;
M=moviein(20);
for j=1:20  % Záznam animace
  surf(sin(2*pi*j/20)*z,z)
  axis(lim)
  M(:,j)=getframe;
```

```
  end
  movie(M,20) % Přehrání animace dvacetkrát
```

Viz též getframe

Funkce Ovlivňuje chování vlastnosti NextPlot.

Syntaxe `h=newplot`

Popis `newplot` je standardní úvodní příkaz, který by měl být vložen na začátek m-souboru, který používá pro tvorbu grafů pouze grafické funkce nižší úrovně.

`h=newplot` provádí následující operace v závislosti na nastavení vlastnosti `NextPlot` objektů axes a figure, a vrací identifikátor odpovídajícího objektu axes:

- Otevře nové grafické okno, pokud má vlastnost `NextPlot` objektu figure hodnotu `'New'`.
- Vyčistí a resetuje aktuální grafické okno, pokud má vlastnost `NextPlot` objektu figure hodnotu `'Replace'`.
- Otevře nové osy, pokud má vlastnost `NextPlot` objektu axes hodnotu `'New'`.
- Vyčistí a resetuje aktuální osy, pokud má vlastnost `NextPlot` objektu axes hodnotu `'Replace'`.

Poznámka Implementováno od verze 4.1.

Viz též `hold`, `ishold`, `figure`, `axes`

Funkce Transformace čísla na řetězec.

Syntaxe `s=num2str(x)`

Popis `num2str` transformuje čísla do jejich řetězcové reprezentace. Tato funkce je velice užitečná pro popis os a grafů číselnými hodnotami.

`s=num2str(x)` transformuje skalární číslo `x` do jeho řetězcové reprezentace `s` (s přibližně čtyřmi číslicemi přesnosti a exponentem).

Algoritmus `num2str` používá funkci `sprintf`. Pokud si přejeme změnit počet číslic, můžeme soubor modifikovat.

Příklady

```
num2str(pi)
'3.142'

num2str(eps)
'2.22e-16'
```

Viz též `int2str`, `fprintf`, `sprintf`, `setstr`, `hex2num`

Funkce Orientace papíru při tisku.

Syntaxe

```
orient landscape
orient('landscape')
orient portrait
orient('portrait')
orient tall
orient('tall')
orient
```

Popis `orient` nastavuje vlastnosti `PaperOrientation` a `PaperPosition` aktuálního grafického okna.

`orient`, samostatně, vrací řetězec s aktuální orientací papíru (`portrait`, `landscape` nebo `tall`). Implicitní orientace je `portrait`, tj. taková orientace, kde největší rozměr papíru je shora dolů. Orientace `landscape` je orientace s největím rozměrem papíru zleva doprava.

`orient landscape` generuje výstup pro následné operace `print` z aktuálního grafického okna v orientaci `landscape` na celou stránku papíru.

`orient portrait` se vrací na implicitní orientaci papíru, kde grafické okno vyplňuje obdélník s poměrem vzhledu 4/3 ve středu stránky.

`orient tall` způsobí, že je grafické okno transformováno na celou stránku v orientaci `portrait`.

`orient('landscape')` je ekvivalentní `orient landscape`, `orient('portrait')` je ekvivalentní `orient portrait` a `orient('tall')` je ekvivalentní `orient tall`.

Viz též `print`

Funkce Vytváření objektu patch.

Syntaxe
```
patch(x,y,c)
patch(x,y,z,c)
h=patch(x,y,z,PropertyName,PropertyValue,...)
```

Popis `patch` je grafická funkce nižší úrovně, která vytváří objekty patch. Objekty patch jsou dětmi objektů axes. Objekt patch je vyplněná plocha mnohoúhelníku, která pro stínování akceptuje barevná data. Funkce `patch` dovoluje použít jako vstupní argumenty dvojice parametrů PropertyName/PropertyValue. Tyto vlastnosti, které řídí různý vzhled objektu patch, jsou popsány v oddíle *Vlastnosti objektu*. Hodnoty těchto vlastností lze též nastavit příkazem `set`. Příkazem `get` se můžeme dotázat na jejich aktuálně nastavenou hodnotu.

Na rozdíl od funkcí vyšší úrovně vytvářející plochy, jako je `fill`, nevymazává `patch` osy, nenastavuje parametry pro prohlížení ani neprovádí jiné akce, než je generování objektu patch v aktuálním objektu axes. Nedefinují-li body `(x,y)` uzavřený mnohoúhelník, `patch` jej uzavře. Body v polích `x` a `y` mohou definovat konkávní i vzájemně se protínající mnohoúhelník.

`patch(x,y,c)` přidá do aktuálního objektu axes plný 2D mnohoúhelník zadaný vektory `x` a `y`. Vektor `c` specifikuje barvu, která je použita k jeho vyplnění. Vrcholy mnohoúhelníku jsou určeny dvojicemi prvků vektorů `x` a `y`.

Je-li `c` skalár, je tak určena jediná barva mnohoúhelníku (vybarvení *flat*). Je-li `c` vektor stejné délky jako `x` a `y`, jsou jeho prvky transformovány funkcí `caxis` a použity jako indexy aktuální mapy barev pro určení barev vrcholů mnohoúhelníku; barva uvnitř mnohoúhelníku se získá bilineární interpolací barev vrcholů. (Délka vektoru `c` musí být větší než 3.)

Je-li `c` řetězec, jsou mnohoúhelníky vybarveny určenou barvou. Řetězec může být buď písmeno: `'r'`, `'g'`, `'b'`, `'c'`, `'m'`, `'y'`, `'w'` nebo `'k'` nebo jméno barvy: `'red'`, `'green'`, `'blue'`, `'cyan'`, `'magenta'`, `'yellow'`, `'white'` nebo `'black'`.

Jsou-li `x` a `y` matice téhož typu, kreslí funkce `patch` pro každý sloupec jeden mnohoúhelník . V tomto případě platí pro pole `c` jedna z následujících možností:

- Řádkový vektor, jehož velikost je rovna `size(x,2)` (tj. je rovna počtu sloupců vektoru `x` nebo `y`) pro stínování *flat*.
- Matice téhož typu jako `x` a `y` pro interpolované stínování.
- Sloupcový vektor, který má stejný počet řádků jako `x` a `y` . Pak tento sloupec používá k získání barvy vrcholů pro interpolované stínování každý objekt patch.

`patch` nastavuje svoji vlastnost `FaceColor` na `flat`, `interp` nebo `ColorSpec` v závislosti na hodnotách pole `c`.

`patch(x,y,z,c)` vytváří objekt patch v souřadnicích 3D.

`patch` vrací sloupcový vektor identifikátorů, které odpovídají každému vytvořenému objektu patch.

Za dvojicemi `x`, `y` (pro 3D za trojicemi `x`, `y`, `z`) mohou následovat dvojice parametrů PropertyName/PropertyValue, které určují další vlastnosti objektu patch. Dvojice `x`, `y` (trojice `x`, `y`, `z`) můžeme úplně vynechat a jen specifikovat vlastnosti dvojicemi PropertyName/PropertyValue.

`patch` je grafické funkce nižší úrovně, která není obvykle použita přímo. Místo ní jsou používány funkce `fill` a `fill3`.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu dvojicemi parametrů PropertyName/PropertyValue, které jsou argumenty funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy je kurzor na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu.

CData *vektor*

Barevná data. Tato vlastnost je vektor hodnot, které určují barvu každého bodu podél hrany objektu patch. Tyto hodnoty používá MATLAB pouze v případě, je-li `EdgeColor` nebo `FaceColor` nastaveno na `interp` nebo `flat`.

Children *vektor identifikátorů*

Děti objektu patch. Objekty patch nemají nikdy žádné děti, a proto je tato vlastnost vždy prázdná matice.

Clipping {`on`} | `off`

Režim ořezávání. Je-li `Clipping on`, nejsou žádné části objektu patch, které leží mimo obdélník os zobrazeny.

EdgeColor {*ColorSpec*} | `none` | `flat` | `interp`

Barva hrany objektu patch. Tato vlastnost nám umožňuje určit barvu hran objektu patch. `ColorSpec` definuje jednu barvu (implicitní je černá). Zvolíme-li hodnotu

`none`, nejsou hrany kresleny. Hodnota `flat` vybarví hrany jedinou barvou určenou průměrem z dat barevné osy pro objekt patch. Při hodnotě `interp` je barva hran získána bilineární interpolací hodnot definovaných ve vrcholech.

EraseMode {`normal`} | `none` | `xor` | `background`

Režim mazání. Tato vlastnost řídí techniku MATLABu používanou pro kreslení a mazání objektů patch. Tato vlastnost je užitečná při vytváření animovaných posloupností, protože řídí překreslování jednotlivých objektů pro získání požadovaných efektů.

Režim `normal` překresluje danou oblast obrazovky tím, že provádí trojrozměrnou analýzu, která je nezbytná pro zabezpečení správného vynesení všech objektů. Tento režim přináší nejpřesnější obrázek, ale je časově náročnější. Ostatní režimy neprovádějí úplné překreslování, a proto jsou značně rychlejší, ale vytvářejí méně přesný obraz.

Je-li nastaven režim `none`, není objekt při přesunu nebo zničení vymazán.

Při režimu `xor` je objekt kreslen a mazán pomocí funkce `xor` aplikovanou na původní barvu. Je-li objekt vymazán, nepoškodí objekty pod ním. Ale je-li objekt kreslen v režimu `xor`, závisí jeho barva na barvě obrazovky pod ním. Je správně vybarven pouze je-li nad barvou pozadí objektu figure.

Režim `background` vytváří správně barvené objekty. Avšak objekt je mazán svým vykreslováním na barvě pozadí objektu figure. Tím se poškodí objekty, které jsou za mazaným objektem.

FaceColor {*ColorSpec*} | `none` | `flat` | `interp`

Barva čela objektu patch. Tato vlastnost nám umožňuje určit barvu plochy objektu patch. Můžeme definovat jednu samostatnou barvu pro objekt patch (`ColorSpec`). Zvolíme-li hodnotu `none`, není objekt patch kreslen (ale můžeme ještě vykreslit hrany). Hodnota `flat` používá k určení jediné barvy pro každý objekt patch hodnoty v matici `c`. Při hodnotě `interp` je barva získána lineární interpolací z hodnot definovaných ve vlastnosti `CData`.

Interruptible `yes` | {`no`}

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

LineWidth *šířka*

Šířka čáry. Nová šířka je určena v bodech (1 bod = 1/72 palce). Implicitní hodnota je 0.5 bodu.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu patch. Tato vlastnost je identifikátor rodiče objektu patch, kterým vždy je objekt axes.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
patch(rand(4,1),rand(4,1),rand(4,1),'Tag','Pokusný patch');
```

vytvoří objekt patch a označí ho visačkou `'Pokusný patch'`. Je-li potom potřeba se někdy na tento objekt odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','Pokusný patch')
```

Type *řetězec (pouze pro čtení)*

Typ objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekty patch je `Type` vždy řetězec `'patch'`.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice. kterou chceme přiřadit objektu. Objekt tato data nepoužije, ale my je můžeme získat příkazem `get`.

Visible {`on`} | `off`

Viditelnost objektu. Vlastnost `Visible` určuje, zda je či není objekt zobrazen na obrazovce.

XData *vektor*
YData *vektor*
ZData *vektor*

Data vrcholů. Tyto vlastnosti určují body podél hrany objektu patch. Jsou-li data ve formě matic, je každý sloupec interpretován jako samostatný objekt patch. V tomto případě musí mít všechna data stejný počet řádků. Ale má-li jeden vstupní argument pouze jediný sloupec, udělá funkce `patch` jeho kopii tak, aby počet sloupců tohoto argumentu odpovídal počtu sloupců specifikovaných v ostatních argumentech. Např. následující příkaz použije k vytvoření dvou objektů patch (každý se čtyřmi vrcholy) jediný sloupec specifikovaný v `ZData`.

```
patch(rand(4,2),rand(4,2),rand(4,1))
```

Mají-li všechna data stejný počet sloupců, ale pouze jediný řádek, transponuje MATLAB matice tak, aby obsahovala taková data, která lze vykreslit. Např.

```
patch(rand(1,4),rand(1,4),rand(1,4))
```

je změněno na

```
patch(rand(4,1),rand(4,1),rand(4,1))
```

Tento způsob se též aplikuje v případě, kdy pouze jedna nebo dvě matice mají jeden řádek. Např. příkaz

```
patch(rand(2,4),rand(2,4),rand(1,4))
```

je ekvivalentní příkazu

```
patch(rand(4,2),rand(4,2),rand(4,1))
```

Viz též `line`, `text`, `fill`, `fill3`

Funkce Pseudobarevné grafy.

Syntaxe

```
pcolor(C)
pcolor(x,y,C)
pcolor(X,Y,C)
h=pcolor(...)
```

Popis `pcolor(C)` kreslí pseudobarevné nebo mozaikové grafy, tj. obdélníková pole buněk s barvami určenými prvky matice `C`. Při implicitním režimu stínování typu `'flat'` má každá buňka konstantní barvu a poslední řádek a poslední sloupec matice `C` nejsou použity. Při režimu stínování typu `'interp'` má každá buňka barvu, která je výsledkem bilineární interpolace barev v jejích čtyřech vrcholech a jsou použity všechny prvky matice `C`. Transformace matice `C` je definována pomocí `colormap` a `caxis`.

`pcolor(x,y,C)` s dvěma vektorovými argumenty musí mít délky `length(x)=n` a `length(y)=m`, kde `[m,n]=size(C)`. Rozestup čar sítě je nastaven pomocí `x` a `y`, takže čáry sítě jsou přímé, ale nemusí být rovnoměrně rozmístěny. Všimněme si, že `x` odpovídá sloupcům matice `C` a vektor `y` řádkům matice `C`.

`pcolor(X,Y,C)` má tři maticové argumenty, všechny stejného typu. 2D síť je nyní kreslena s vrcholy v bodech `[X(i,j),Y(i,j)]`. Barva v buňce `(i,j)` je určena buď prvkem `C(i,j)` v případě stínování `'faceted'` nebo `'flat'`, nebo interpolací mezi barvami ve čtyřech vrcholech pro stínování `'interp'`.

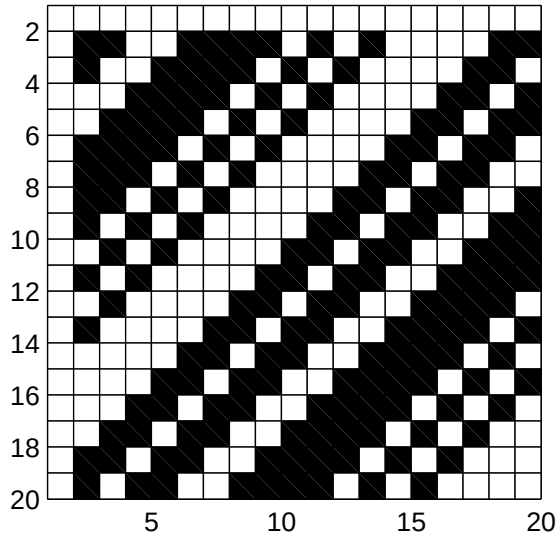
`pcolor` má úzkou vazbu na funkci `surf`; ve skutečnosti je `pcolor(X,Y,C)` totéž jako pohled shora na `surf`(X,Y,0*Z,C), tj. `view([0 90])`.

`pcolor` a `image` jsou podobné funkce, ale `pcolor(C)` určuje barvy vrcholů, zatímco `image(C)` určuje barvy buněk a přímo indexy mapy barev bez transformace. V důsledku toho je počet vrcholů pro `pcolor(C)` stejný jako počet buněk pro `image(C)`. `pcolor(X,Y,C)` se třemi argumenty může vytvořit parametrickou síť, což není u funkce `image` možné.

`h=pcolor(...)` vrací identifikátor objektu surface. Objekty surface jsou dětmi objektu axes.

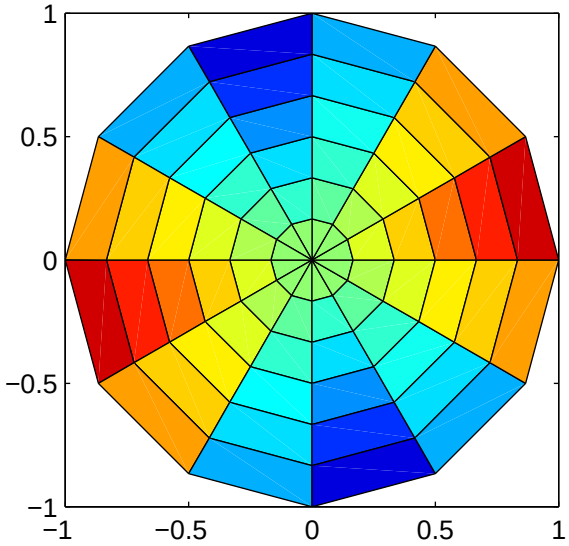
Příklady Hadamardova matice má pouze prvky +1 a -1, proto je vhodné použít mapu barev pouze se dvěma vstupy.

```
pcolor(hadamard(20))
colormap(gray(2))
axis('ij')
axis('square')
```



Jednoduché barevné kolo ilustruje polární souřadný systém.

```
n=6;
r=(0:n)'/n;
theta=pi*(-n:n)/n;
X=r*cos(theta);
Y=r*sin(theta);
C=r*cos(2*theta);
pcolor(X,Y,C)
axis('square')
```



Viz též `surf`, `image`, `view`

Funkce Lineární graf 2D.

Syntaxe

```
plot(y)
plot(y, linetype)
plot(x,y)
plot(x,y,linetype)
plot(x1,y1,linetype1,x2,y2,linetype2,...)
h=plot(x1,y1,[s1],x2,y2,[s2],PropertyName,PropertyValue,...)
```

Popis `plot(y)` vytváří graf sloupců `y` v závislosti na jejich indexech. Je-li `y` komplexní, je `plot(y)` ekvivalentní příkazu `plot(real(y),imag(y))`. Ve všech ostatních použitích funkce `plot` je imaginární část ignorována.

`plot(x,y)` vytváří graf vektoru `y` v závislosti na vektoru `x`. Je-li funkce `plot` použita se dvěma argumenty a má-li buď `X` nebo `Y` více než jednu řádku nebo jeden sloupec, potom

- Je-li `x` vektor a `Y` matice, `plot(x,Y)` kreslí řádky nebo sloupce matice `Y` vzhledem k vektoru `x`, pro každou čáru použije jinou barvu nebo jiný typ čáry. O tom, zda budou kresleny řádky nebo sloupce rozhoduje počet prvků vektoru `x`. Řádková nebo sloupcová orientace je vybrána podle shody počtu prvků řádků nebo sloupců matice `Y` s počtem prvků vektoru `x`. Pokud je matice `Y` čtvercová, vykreslí se její sloupce.
- Je-li `X` matice a `y` vektor, `plot(X,y)` kreslí každý řádek nebo sloupec matice `X` vzhledem k vektoru `y`.
- Jsou-li `X` i `Y` matice téže velikosti, `plot(X,Y)` zobrazí sloupce matice `X` vůči sloupcům matice `Y`.

Různé typy čar, symboly a barvy mohou být získány příkazem `plot(x,y,s)`, kde `s` je řetězec (1, 2 nebo 3 znaky) vytvořený z následujících znaků. (POZOR! Nelze kombinovat znaky z jednoho sloupce).

`y`	žlutá	`.`	bod
`m`	fialová	`o`	kroužek
`c`	tyrkysová	`x`	značka x
`r`	červená	`+`	plus
`g`	zelená	`*`	hvězdička
`b`	modrá	`-`	plná
`w`	bílá	`:`	tečkovaná
`k`	černá	`-.`	čerchovaná
		`--`	čárkovaná

Např. `plot(x,y,'c+')` vykreslí v každém bodě dat tyrkysové značky plus.

`plot(x1,y1,s1,x2,y2,s2,...)` kombinuje kreslení dat definované trojicemi `(x,y,s)`, kde `xi` a `yi` jsou vektory nebo matice a `si` jsou řetězce.

Např. `plot(x,y,'-',x,y,'go')` vykreslí data dvakrát, plnou žlutou čáru a datové body vyznačí zelenými kroužky.

Příkaz `plot`, který nemá určenou barvu, vybírá automaticky barvy z výše uvedené tabulky. Implicitní barvou pro jednu čáru je žlutá barva, pro násobné čáry se z uvedené tabulky bere cyklicky prvních šest barev.

`plot` vrací sloupcový vektor identifikátorů objektů line, každému objektu line přísluší jeden identifikátor. Objekty line vytvořené funkcí plot jsou dětmi aktuálního objektu axes.

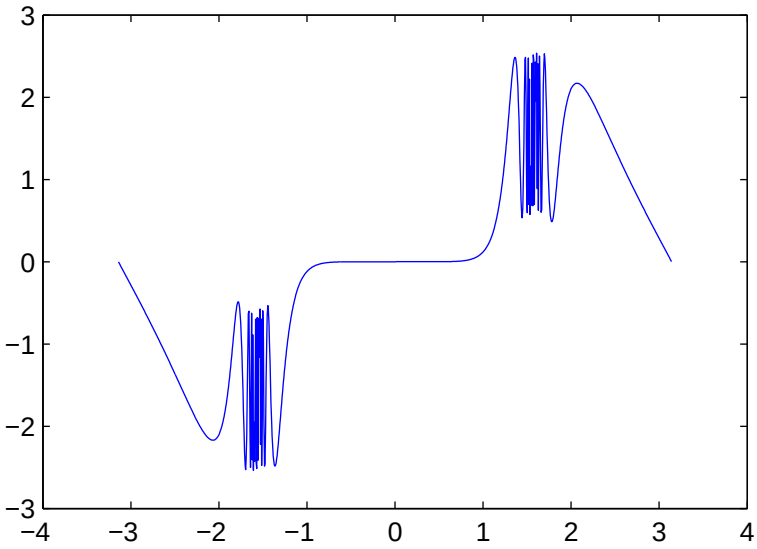
Za dvojicemi `x`, `y` (nebo trojicemi `x`, `y`, `s`) mohou následovat dvojice parametrů PropertyName/PropertyValue, které určují další vlastnosti objektů line.

`plot` ignoruje definování implicitních hodnot uživatelem; tyto implicitní hodnoty využívá funkce `line`.

Příklady Příkazy

```
x=-pi:pi/500:pi;
y=tan(sin(x))-sin(tan(x));
plot(x, y)
```

vytvoří



Viz též `semilogx`, `semilogy`, `loglog`, `polar`, `grid`, `title`, `xlabel`, `ylabel`, `axis`, `hold`, `line`, `subplot`

Funkce Kreslení čar a bodů ve 3D.

Syntaxe

```
plot3(x,y,z)
plot3(X,Y,Z)
plot3(x,y,z,linetype)
plot3(x1,y1,linetype1,x2,y2,linetype2,...)
h=plot3(x1,y1,[s1],x2,y2,[s2],PropertyName,PropertyValue,...)
```

Popis `plot3` je třídimenzionální analogie funkce `plot`.

`plot3(x,y,z)`, kde `x`, `y` a `z` jsou tři vektory stejné délky, kreslí čáru ve 3D, která prochází body, jejichž souřadnice jsou prvky vektorů `x`, `y` a `z`.

`plot3(X,Y,Z)`, kde `X`, `Y` a `Z` jsou tři matice stejné velikosti, kreslí několik čar ve 3D, získaných ze sloupců matic `X`, `Y` a `Z`.

Různé typy čar, symboly a barvy mohou být získány příkazem `plot3(X,Y,Z,s)`, kde `s` je řetězec (1, 2 nebo 3 znaky) vytvořený ze znaků uvedených u funkce `plot`.

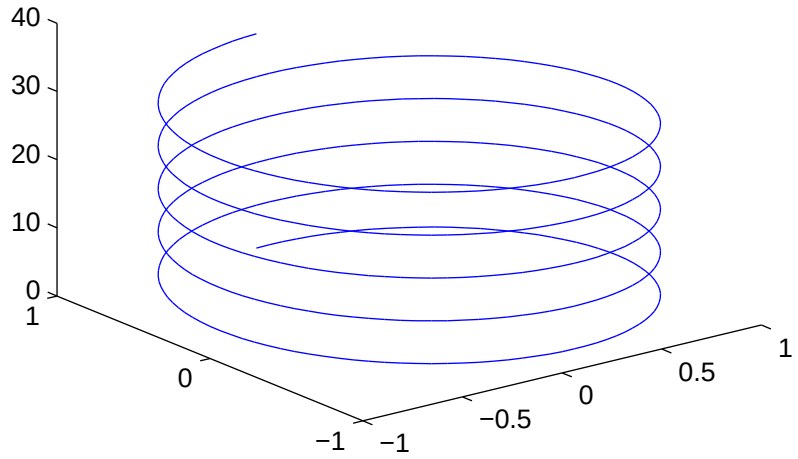
`plot3(x1,y1,z1,s1,x2,y2,z2,s2,x3,y3,z3,s3,...)` kombinuje kreslení dat definované čtveřicemi (`x`, `y`, `z`, `s`), kde `xi`, `yi` a `zi` jsou vektory nebo matice a `si` jsou retězce.

`plot3` vrací sloupcový vektor identifikátorů objektů line, každému objektu line přísluší jeden identifikátor. Objekty line vytvořené funkcí `plot3` jsou dětmi aktuálního objektu axes.

Za trojicemi `x`, `y`, `z` (nebo čtveřicemi `x`, `y`, `z`, `s`) mohou následovat dvojice parametrů PropertyName/PropertyValue, které určují další vlastnosti objektů line.

Příklady Vykreslení trojrozměrné spirály:

```
t=0:pi/50:10*pi;
plot3(sin(t),cos(t),t)
```



Viz též `plot`

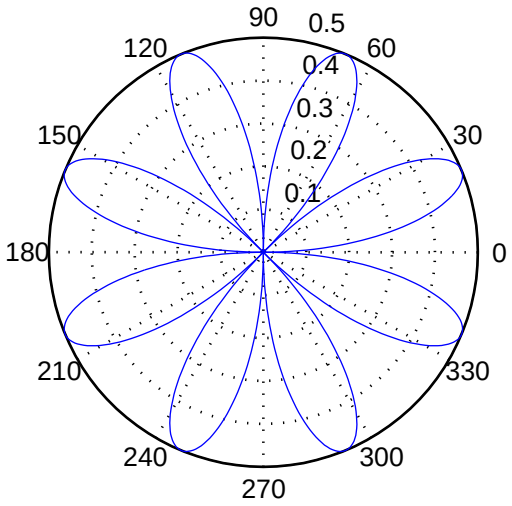
Funkce Graf v polárních souřadnicích.

Syntaxe
```
polar(theta,rho)
polar(theta,rho,linetype)
```

Popis `polar(theta,rho)` vytvoří graf v polárních souřadnicích s úhlem `theta` v radiánech a poloměrem `rho`. `linetype` je řetězec vytvořený ze znaků uvedených u funkce `plot`.

Příklady Jednoduchý graf v polárních souřadnicích:

```
t=0:0.01:2*pi;
polar(t,sin(2*t).*cos(2*t))
```



Viz též `plot`, `loglog`, `semilog`, `rose`, `compass`

Funkce Tiskne graf nebo ukládá graf do souboru.

Syntaxe

```
print
print filename
print filename -fmodulname
print -ddevice -v{olby} filename
```

Popis `print`, samostatně, posílá obsah aktuálního grafického okna na implicitní tiskárnu.

`print filename` ukládá aktuální grafické okno do souboru určeného proměnnou `filename` v implicitním formátu. Jestliže již soubor existuje, `print` ho přepíše, pokud nebylo použito volby `-append`. Pokud určený soubor nezahrnuje příponu, je odpovídající přípona `'.ps'` nebo `'.eps'` přidána. Pokud jméno souboru chybí, je graf poslán přímo na výstupní zařízení (tiskárnu) podle specifikace funkce `printopt`.

`print filename -fmodulname` vytiskne zvolený systém SIMULINKu.

Přímo podporovaná zařízení:

`-dps`	PostScript pro černobílé tiskárny
`-dpsc`	PostScript pro barevné tiskárny
`-dps2`	Level 2 PostScript pro černobílé tiskárny
`-dpsc2`	Level 2 PostScript pro barevné tiskárny
`-deps`	Encapsulated PostScript (EPSF)
`-depsc`	Encapsulated Color PostScript (EPSF)
`-deps2`	Encapsulated Level 2 PostScript (EPSF)
`-depsc2`	Encapsulated Level 2 Color PostScript (EPSF)

Obecně jsou soubory s formátem Level 2 PostScript menší a rychleji se vytisknou, ale ne všechny PostScriptové tiskárny tyto formáty podporují.

Dodatečně vestavěná zařízení (od verze 4.2):

`-dhpgl`	HPGL kompatibilní ploter
`-dill`	Soubor kompatibilní s Adobe Illustratorem 88
`-dmfile`	M-soubor, po jehož spuštění se obnoví uložený objekt figure a jeho děti

Dodatečná zařízení podporovaná přes postprocesor GhostScript (transformuje soubory PostScript na jiné formáty) jsou využitelná pouze na systémech UNIX a PC:

`-dlaserjet`	HP LaserJet
`-dljetplus`	HP LaserJet+
`-dljet2p`	HP LaserJet IIP
`-dljet3`	HP LaserJet III
`-dcdeskjet`	HP DeskJet 500C s 1 bitovou barvou
`-dcdjcolor`	HP DeskJet 500C s 24 bitovou barvou a kvalitním barevným (Floyd-Steinberg) rozptylováním (dithering)
`-dcdjmono`	HP DeskJet 500C - pouze černobílý tisk
`-ddeskjet`	HP DeskJet a DeskJet Plus
`-dpaintjet`	HP PaintJet - barevná tiskárna
`-dpjetxl`	HP PaintJet XL - barevná tiskárna
`-dbj10e`	Canon BubbleJet BJ10e
`-dln03`	DEC LN03
`-depson`	Epson-kompatibilní jehličková tiskárna (9- nebo 24-jehel)
`-deps9high`	Epson-kompatibilní 9-ti jehličková tiskárna, prokládané řádky (trojnásobné rozlišení)
`-depsonc`	Epson LQ-2550 a Fujitsu 3400/2400/1200, barevné tiskárny
`-dgif8`	GIF formát - 8-bitová barva
`-dpcx16`	Starší barevný PCX formát (EGA/VGA, 16 barev)
`-dpcx256`	Novější barevný PCX formát (256 barev)

Následující tiskové volby jsou podporované na všech systémech:

`-append`	přidá graf na konec souboru (nepřepisuje soubor)
`-epsi`	přidá 1-bitové bitmapové preview (EPSI formát)
`-ocmyk` [4.2]	použije v PostScriptu CMYK barvy místo RGB barev
`-P`*printer*	specifikace tiskárny (pouze UNIX)
`-f`*handle*	identifikátor objektu figure určeného pro tisk
`-s`*name*	jméno systémového okna SIMULINKu určeného pro tisk

Využitelná zařízení a tiskové volby pouze pro Windows:

`-dwin`	tisková služba Windows
`-dwinc`	tisková služba Windows pro barevný tisk
`-dmeta`	kopie do clipboardu ve formátu Windows metafile
`-dbitmap`	kopie do clipboardu ve formátu Windows bitmap
`-dsetup`	vyvolání dialogového boxu Print Setup pro nastavení tiskárny, ale bez tisku
`-v`	vyvolání dialogového boxu Print pro tisk, který je normálně potlačen.

POZOR! Pokud použijeme výše uvedených tiskových služeb Windows, musíme počítat s jistými odlišnostmi. Kopie z tiskárny se může lišit od originálu na obrazovce; např. nedodržení typu čar (všechny typy čar se vytisknou jako plná čára), obtíže při

přenosu přes clipboard ve formátu Windows metafile, nerespektování hodnot `NaN` v objektech surface atd.

Normálně MATLAB mění černé pozadí na bílé pozadí a bílé osy a popisy na černé. Tyto změny lze potlačit příkazem:

```
set(gcf,'InvertHardCopy','off')
```

Objekty uimenu a uicontrol nejsou tištěny.

Příklady Příkaz

```
print meshdata -depsc2
```

uloží obsah aktuálního grafického okna ve formátu Level 2 color Encapsulated PostScript do souboru `meshdata.eps`.

Viz též `printopt`, `orient`

Funkce Konfigurace implicitní tiskárny.

Syntaxe `[pcmd,dev]=printopt`

Popis `printopt` je m-soubor používaný funkcí `print`, který slouží pro nastavení implicitní tiskárny.

`[pcmd,dev]=printopt` vrací dva řetězce, `pcmd` a `dev`. `pcmd` je řetězec obsahující příkaz pro tisk, který používá funkce print. Implicitně je:

`lpr -r`	Unix
`PRINT`	Windows
`unused`	Macintosh
`PRINT/DELETE`	VMS
`lp`	SGI

`dev` je řetězec obsahující implicitní zařízení pro tisk, používané funkcí `print`. Implicitní hodnoty jsou:

`-dps`	Unix & VMS
`-dwin`	Windows
`-dmac`	Macintosh

Viz též `print`, `orient`

Funkce Vytvoří dialogový box dotazu.

Syntaxe `click=questdlg(q,yes,no,cancel,default)`

Popis `click=questdlg(q,yes,no,cancel,default)` vytvoří dialogový box dotazu, který zobrazí dotaz `q`. V dialogovém boxu se mohou dále objevit až tři tlačítka s popisem určeným argumenty `yes`, `no` a `cancel`. Dialogové okno zmizí po volbě některého tlačítka. `questdlg` vrací řetězec `click` v závislosti na stisknutém tlačítku. `default` je implicitní číslo tlačítka.

Poznámka Implementováno od verze 4.2.

Příklady

```
c=questdlg('Do you want to continue ?','Yes','No')
```

Funkce Graf tvaru jehelníčku.

Syntaxe

```
quiver(X,Y,DX,DY)
quiver(x,y,DX,DY)
quiver(DX,DY)
quiver(X,Y,DX,DY,s)
quiver(DX,DY,s)
quiver(...,linetype)
```

Popis `quiver(X,Y,DX,DY)` kreslí malé šipky v každé dvojici prvků matic `X` a `Y`. Dvojice prvků v maticích `DX` a `DY` určují směr a relativní velikost šipek.

`quiver(x,y,DX,DY)` s dvěma vektorovými argumenty nahrazujícími první dva maticové argumenty musí mít `length(x)=n` a `length(y)=m`, kde `[m,n]=size(DX) =size(DY)`. V tomto případě jsou šipky čtveřice `(x(j),y(j),DX(i,j),DY(i,j))`. `x` odpovídá sloupcům matic `DX` a `DY`, `y` odpovídá řádkům.

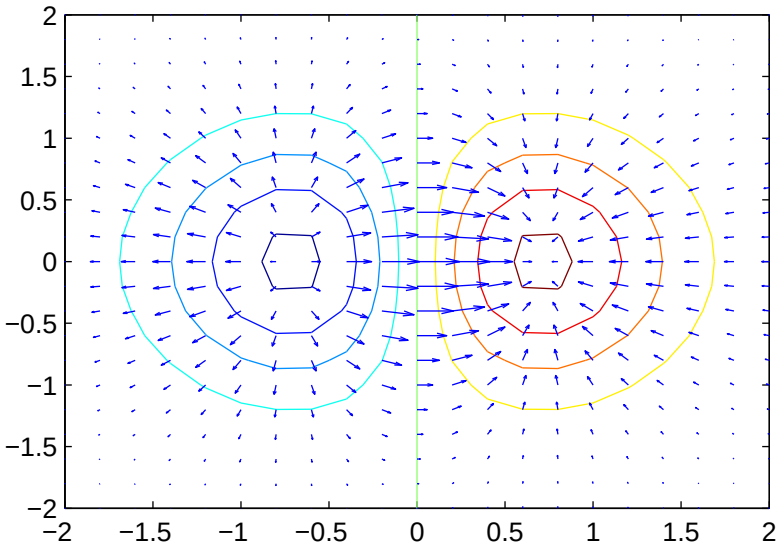
`quiver(DX,DY)` používá `x=1:n` a `y=1:m`. Zde jsou `DX` a `DY` definované nad geometricky obdélníkovou sítí.

`quiver(X,Y,DX,DY,s)` a `quiver(DX,DY,s)` používají skalár `s` jako měřítkovací faktor pro délky šipek. Např. `s=2` zdvojnásobí jejich relativní délku, `s=0.5` zkrátí jejich délku na polovinu.

Poslední řetězový argument specifikuje barvu a typ čáry (viz popis funkce `plot`).

Příklady Graf pole gradientů funkce $z = x\,e^{-x^2-y^2}$

```
[x,y]=meshgrid(-2:0.2:2);
z=x.*exp(-x.^2-y.^2);
[dx,dy]=gradient(z,0.2,0.2);
contour(x,y,z)
hold on
quiver(x,y,dx,dy)
hold off
```



Viz též `contour`, `plot`

Funkce Zaktualizuje (překreslí) požadovaný objekt figure.

Syntaxe
```
refresh
refresh(h)
```

Popis `refresh` zaktualizuje aktuální objekt figure.

`refresh(h)` zaktualizuje objekt figure s identifikátorem `h`.

Poznámka Implementováno od verze 4.2.

Viz též `legend`

Funkce Nastavuje vlastnosti objektů axes a figure na jejich implicitní hodnoty.

Syntaxe `reset(h)`

Popis `reset(h)` resetuje všechny vlastnosti (kromě vlastnosti `Position`) objektu s identifikátorem `h` na jeho implicitní hodnoty.

Příklady `reset(gca)` resetuje vlastnosti aktuálního objektu axes.

`reset(gcf)` resetuje vlastnosti aktuálního objektu figure.

Viz též `clf`, `cla`, `gcf`, `gca`, `hold`

Funkce Graf mapy barev.

Syntaxe `rgbplot(map)`

Popis `rgbplot(map)` vytváří graf mapy barev, tj. matice typu $(m, 3)$, která je příslušným vstupním parametrem pro funkci `colormap`. Tři sloupce matice map jsou kresleny červenými, zelenými a modrými čarami.

Příklady `rgbplot(hsv)` vytvoří graf implicitní mapy barev.

Viz též `colormap`, `spinmap`

Funkce Konverze RGB hodnot na HSV hodnoty.

Syntaxe \verbH=rgb2hsv(M)—

Popis `H=rgb2hsv(M)` konvertuje mapu barev `rgb` na mapu barev `hsv`. Každá mapa je matice s libovolným počtem řádků, přesně třemi sloupci a s prvky v intervalu od 0 do 1. Sloupce vstupní matice `M` reprezentují intenzitu červené, zelené a modré. Sloupce výsledné výstupní matice `H` representují barevný tón, sytost a jasovou hodnotu.

Viz též `colormap`, `hsv`, `hsv2rgb`, `brighten`

root

Funkce Vlastnosti objektu root.

Popis Objekt root odpovídá obrazovce počítače. Existuje pouze jediný objekt root a nemá žádné rodiče. Úkolem objektu root je být rodičem objektů figure.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu zadáním dvojic parametrů PropertyName/PropertyValue do vstupních argumentů funkce vytvářející objekt, nebo hodnoty vlastností specifikujeme až po vytvoření objektu identifikací objektu pomocí identifikátoru a funkcemi `set` a `get`.

V této části je uveden seznam názvů vlastností objektu root s jejich přípustnými hodnotami. Pokud jsou nastaveny implicitní hodnoty, jsou uvnitř složených závorek.

BlackAndWhite `on | {off}`

Potlačení automatické kontroly typu monitoru. Implicitně MATLAB automaticky určuje, běží-li na barevném nebo monochromatickém monitoru. Nastavení této vlastnosti na `on` jej zbaví schopnosti této kontroly a způsobí, že MATLAB uvažuje monochromatický monitor. To je užitečné v případě, kdy sice MATLAB běží na barevném monitoru, ale zobrazuje na monochromatickém terminálu. Taková situace může způsobit chybné určení typu monitoru.

ButtonDownFcn *řetězec*

Funkce zpětného volání. V objektu root je tento řetězec vždy prázdný.

CaptureMatrix *matice (pouze pro čtení)*

Matice obrazových dat. Tato vlastnost je matice, která obsahuje obraz dat oblasti uzavřené obdélníkem `CaptureRect`. Matici můžeme získat pomocí funkce `get`. To nám umožňuje importovat do MATLABu cokoliv, co se vyskytuje na obrazovce. Pro zobrazení této matice se použije funkce `image`.

CaptureRect *4-prvkový vektor*

Sběrný obdélník. Tato vlastnost je obdélník, který určuje oblast obrazovky, se kterou pracuje matice `CaptureMatrix`. Obdélník je definován vektorem

```
rect=[zleva, zdola, šířka, výška]
```

kde `zleva` a `zdola` definují umístění levého dolního rohu obdélníku a `šířka` a `výška` definují rozměry obdélníku. Vlastnost `Units` určuje, jaké jednotky jsou použity pro specifikaci těchto rozměrů.

Children *vektor identifikátorů*

Děti objektu root. Tato vlastnost je vektor identifikátorů všech objektů figure.

Clipping {on} | off

Režim ořezávání. V objektu root má vždy hodnotu on.

CurrentFigure *identifikátor*

Aktuální grafické okno. Tato vlastnost je identifikátor aktuálního objektu figure. Tento identifikátor je vrácen funkcí gcf. Grafické okno můžeme zaktuálnit buď vyvoláním

```
figure(h)
```

kde h je identifikátor objektu figure, který chceme zaktuálnit, nebo nastavením hodnoty této vlastnosti. Neexistuje-li žádný objekt figure, je vytvořen příkazem

```
get(0,'CurrentFigure')
```

Diary on | {off}

Režim vytváření deníku. Je-li tato vlastnost on, zachovává MATLAB soubor (deník), jehož jméno je specifikováno vlastností DiaryFile a který ukládá kopie všech vstupů z klávesnice a většinu výsledných výstupů. Vlastnost Diary drží stav vytváření deníku. Viz též příkaz diary.

DiaryFile *řetězec*

Jméno deníku. Tato vlastnost obsahuje jméno souboru – deníku.

Echo on | {off}

Režim zobrazování příkazů v souboru typu script. Je-li Echo nastaveno na hodnotu on, je každá řádka souboru typu script v době, kdy je vykonávána, zobrazena. Vlastnost Echo drží stav zobrazování souborů MATLABu typu script. Viz též příkaz echo.

Format short | {shortE} | long | longE | bank | hex | +

Režim výstupního formátu. Tato vlastnost nastavuje výstupní formát MATLABu. Viz též příkaz format.

short	formát s pevnou desetinnou tečkou s 5 číslicemi
shortE	formát v exponenciálním tvaru s 5 číslicemi
long	formát s pevnou desetinnou tečkou s 15 číslicemi
longE	formát v exponenciálním tvaru s 15 číslicemi
bank	pevný formát dolarů a centů
hex	formát v hexadecimálním tvaru
+	zobrazuje symboly + a -

FormatSpacing compact | {loose}

Mezery výstupního formátu. Tato vlastnost nastavuje výstupní formát MATLABu – kompaktní nebo *vzdušný*. Více informací viz příkaz format.

Interruptible yes | {no}

Režim přerušení. V objektu root má vždy hodnotu no.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu figure. Objekt root nemá žádné rodiče. Tato vlastnost je vždy prázdná matice.

PointerLocation `[x y]` *(pouze pro čtení)*

Aktuální pozice kurzoru. MATLAB nastavuje tuto vlastnost na aktuální polohu kurzoru. Poloha je určena vektorem obsahujícím souřadnice `x` a `y` kurzoru, které jsou měřené v jednotkách určených vlastností `Units` z levého dolního rohu obrazovky.

Této vlastnosti se můžeme kdykoliv dotázat, zda je či není kurzor v okně MATLABu. Vždy obsahuje okamžitou pozici kurzoru.

POZOR! Aktuální pozice však může být v době vracení této hodnoty již změněna.

PointerWindow *identifikátor (pouze pro čtení)*

Identifikátor okna obsahujícího kurzor. MATLAB nastavuje tuto vlastnost na identifikátor toho grafického okna, které obsahuje kurzor. Není-li kurzor uvnitř okna MATLABu, je hodnota vlastnosti 0.

ScreenDepth *počet bitů na pixel*

Hloubka bitmapy. Tato vlastnost signalizuje hloubku bitmapy neboli počet bitů na pixel. Maximální počet barev, které mohou být současně zobrazeny na aktuálním grafickém zařízení je tudíž `2^ScreenDepth`. `Screen Depth` může potlačit vlastnost `BlackAndWhite` popsanou výše. Nastavením vlastnosti `ScreenDepth` na hodnotu 1 je MATLAB donucen zobrazit grafiku černobíle bez ohledu na to, zda je či není monitor schopen zobrazovat barvy.

ScreenSize *4-prvkový vektor (pouze pro čtení)*

Velikost obrazovky. Tato vlastnost definuje obdélník, který odpovídá velikosti celé obrazovky. Obdélník je určen čtyřmi prvky

```
rect=[zleva, zdola, šířka, výška]
```

kde prvky `zleva` a `zdola` jsou vždy nulové. Prvky `šířka` a `výška` obsahují velikost obrazovky v jednotkách specifikovaných vlastností `Units`.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu.

Type *řetězec (pouze pro čtení)*

Typ grafického objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekty root je `Type` vždy řetězec `'root'`.

Units {`pixels`} | `normal` | `inches` | `centimeters` | `points`

Použité jednotky. Tato vlastnost definuje jednotky, které MATLAB používá pro interpretaci velikosti a umístění dat. Všechny jednotky jsou měřeny od levého dolního

rohu okna. Normalizované jednotky transformují levý dolní roh obrazovky na hodnotu $(0, 0)$ a horní pravý roh na $(1, 1)$. Palce, centimetry a body jsou absolutní jednotky (1 bod = 1/72 palce).

Tato vlastnost má vliv na vlastnosti `CaptureRect`, `PointerLocation` a `ScreenSize`. Změníme-li hodnotu vlastnosti `Units`, je dobrým zvykem ji po ukončení našich výpočtů vrátit na její implicitní hodnotu, aby neovlivnila ostatní funkce, které předpokládají implicitní nastavení této vlastnosti.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice, kterou chceme spojit s objektem. Objekt tato data nepoužívá, ale my je můžeme získat příkazem `get`.

Visible {`on`} | `off`

Viditelnost objektu. Vlastnost `Visible` určuje zda je či není objekt zobrazen na obrazovce. Nastavení viditelnosti pro objekt root nemá žádný výsledek.

Viz též `figure`, `gcf`, (`echo`, `diary`, `format`)

Funkce Úhlový histogram.

Syntaxe

```
rose(theta)
rose(theta,n)
rose(theta,x)
[t,r]=rose(...)
```

Popis `rose(theta)` kreslí úhlový histogram pro úhly ve vektoru `theta` (v radiánech). Úhlový histogram je graf v polárních souřadnicích, který zobrazuje počet vzorků `theta` uvnitř úhlového intervalu.

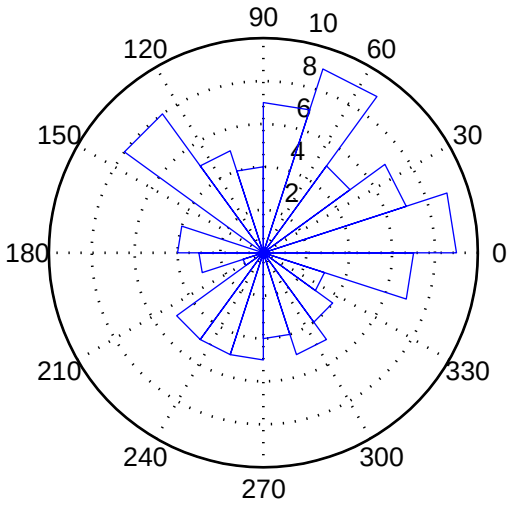
`rose(theta,n)`, kde `n` je skalár, používá `n` rovnoměrně rozmístěných úhlových intervalů od 0 do 2π. Implicitní hodnota pro `n` je 20.

`rose(theta,x)`, kde `x` je vektor, používá úhlové intervaly určené vektorem `x`. Hodnoty ve vektoru `x` určují střed každého úhlového intervalu.

`[t,r]=rose(...)` vrací vektory `t` a `r` (nekreslí histogram). Úhlový histogram lze pak vykreslit příkazem `polar(t,r)`.

Příklady

```
t=rand(100,1)*2*pi;
rose(t);
```



Viz též `hist`, `polar`

Funkce Rotuje objektem.

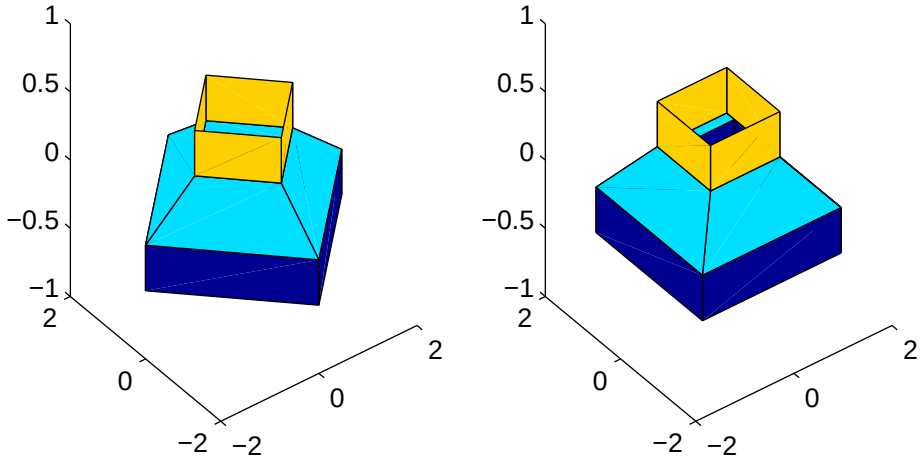
Syntaxe `rotate(h,azel,alpha,origin)`

Popis `rotate(h,azel,alpha,origin)` rotuje objektem `h` o úhel `alpha` okolo osy popsané argumentem `azel`, což je dvojprvkový vektor $(azimut, elevace)$ nebo trojprvkový vektor (x, y, z). Volitelný argument `origin` je trojprvkový vektor udávající střed rotace.

Poznámka Implementováno od verze 4.2.

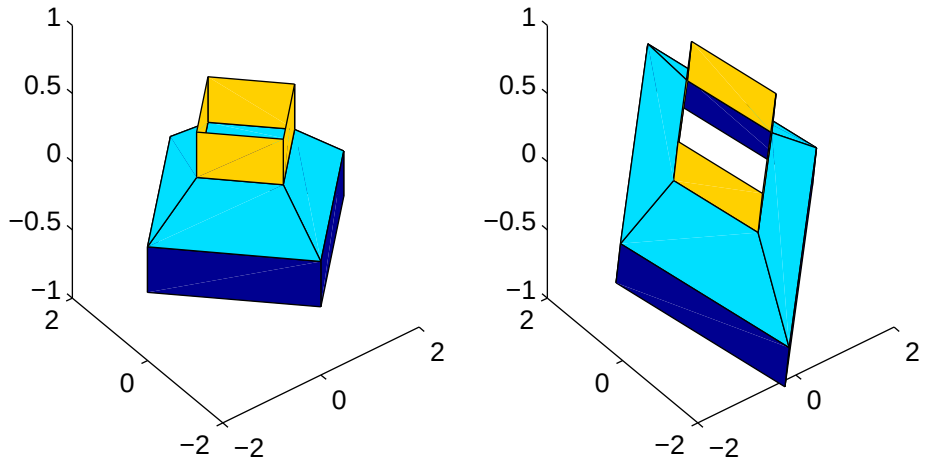
Příklady Následující tři příklady znázorňují rotaci kolem tří základních os

```
[X,Y,Z]=cylinder([2 2 1 1],4);

subplot(1,2,1);
h1=surf(X,Y,Z-0.5);
axis([-2 2 -2 2 -1 1]);
subplot(1,2,2);
h2=surf(X,Y,Z-0.5);
rotate(h2,[0,90],45);
axis([-2 2 -2 2 -1 1]);
```

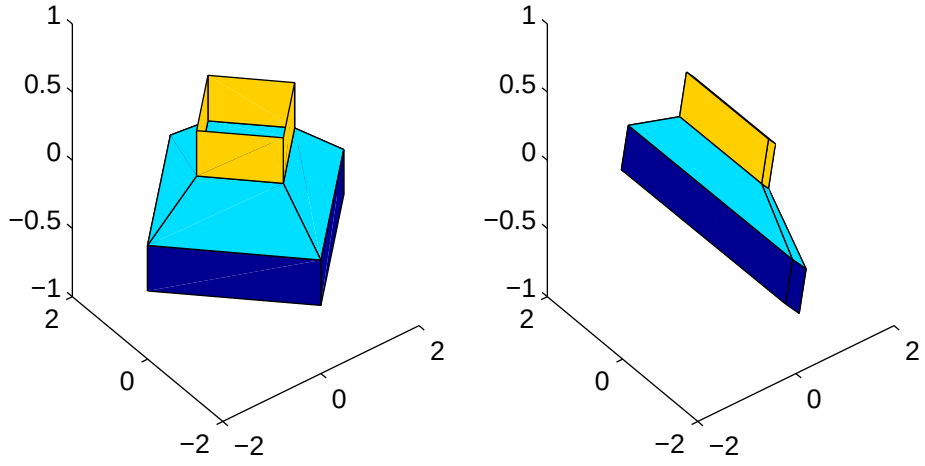



```
subplot(1,2,1);
h1=surf(X,Y,Z-0.5);
axis([-2 2 -2 2 -1 1]);
subplot(1,2,2);
```

```
h2=surf(X,Y,Z-0.5);
rotate(h2,[0,0],20);
axis([-2 2 -2 2 -1 1])
```




```
subplot(1,2,1);
h1=surf(X,Y,Z-0.5);
axis([-2 2 -2 2 -1 1]);
subplot(1,2,2);
h2=surf(X,Y,Z-0.5);
rotate(h2,[90,0],25);
axis([-2 2 -2 2 -1 1]);
```



Funkce Ukládá proměnné na disk.

Syntaxe

```
save
save filename
save filename variables
save filename variables keywords
```

Popis `save` a `load` jsou příkazy MATLABu, které slouží pro uložení a nahrání proměnných do/ze souboru. Dále mohou sloužit k importu a exportu ASCII souborů.

Binární soubory MATLABu mají příponu `'.mat'` a obsahují proměnné v přesnosti double. Tyto soubory jsou vytvořeny příkazem `save` a čitelné příkazem `load`. Soubory mohou být vytvořeny na jednom počítači a později přečteny MATLABem na jiném počítači, který má jiný formát reálných čísel. Kromě MATLABu mohou s těmito soubory pracovat i jiné programy.

Příkaz

```
save
```

samostatně, uloží všechny proměnné v binárním formátu do souboru `matlab.mat`. Data mohou být načtena příkazem `load`.

Příkaz

```
save my_data
```

použije soubor `my_data.mat` místo implicitního souboru `matlab.mat`.

Pokud chceme uložit pouze některé proměnné, zapíšeme jejich jména za jméno souboru `filename`; např.

```
save my_data X Y Z
```

Implicitní přípona `'.mat'` se nepoužije, pokud zadáme jméno souboru včetně přípony.

Klíčová slova `-ascii`, `-double` a `-tabs` způsobí, že jsou data uložena namísto v binárním formátu ve formátu ASCII.

Příkaz

```
save my_data X Y Z -ascii
```

používá 8-mi místný ASCII formát.

Příkaz

```
save my_data X Y Z -ascii -double
```

používá 16-ti místný ASCII formát.

Pokud přidáme klíčové slovo `-tabs`, budou data oddělena tabelátory.

```
save my_data X Y X -ascii -tabs
save my_data X Y X -ascii -double -tabs
```

Jestliže je —filename— —'stdio'—, posílá příkaz —save— data na standardní výstup.

POZOR! Při ukládání v ASCII formátu jsou pole zaznamenávána po řádcích bezprostředně za sebou, nelze je načíst funkcí `load`. Při ukládání v binárním formátu se uloží jednotlivá pole včetně jména a velikosti, takže je lze funkcí `load` načíst bez problémů.

Algoritmus Binární formát používaný funkcí `save` závisí na velikosti a typu každé matice. Matice s reálnými položkami a matice s 10000 prvky nebo méně jsou uloženy v double přesnosti (8 bytů na prvek). Matice se všemi celočíselnými prvky, kterých je více než 10000, jsou uloženy ve formátech, které vyžadují méně bytů na prvek:

Rozsah prvku	Počet bytů na prvek
$[0 : 255]$	1
$[0 : 65535]$	2
$[-32767 : 32767]$	2
$[-2^{31} + 1 : 2^{31} - 1]$	4
ostatní	8

Příručka *External Interface Guide* popisuje podrobně strukturu souborů MAT. *External Interface Library* obsahuje procedury v jazyku C a Fortranu pro práci (čtení a zápis) se soubory MAT.

Viz též `load`, `diary`, `fwrite`, `fprintf`

Funkce Graf s logaritmickou stupnicí x-ové, resp. y-ové osy.

Syntaxe

```
semilogx(x,y)
semilogy(x,y)
semilogx(x,y,linetype)
semilogy(x,y,linetype)
semilogx(x1,y1,linetype1,x2,y2,linetype2,...)
semilogy(x1,y1,linetype1,x2,y2,linetype2,...)
h=semilogx(x1,y1,s1,x2,y2,s2,...,PropertyName,PropertyValue)
h=semilogy(x1,y1,s1,x2,y2,s2,...,PropertyName,PropertyValue)
```

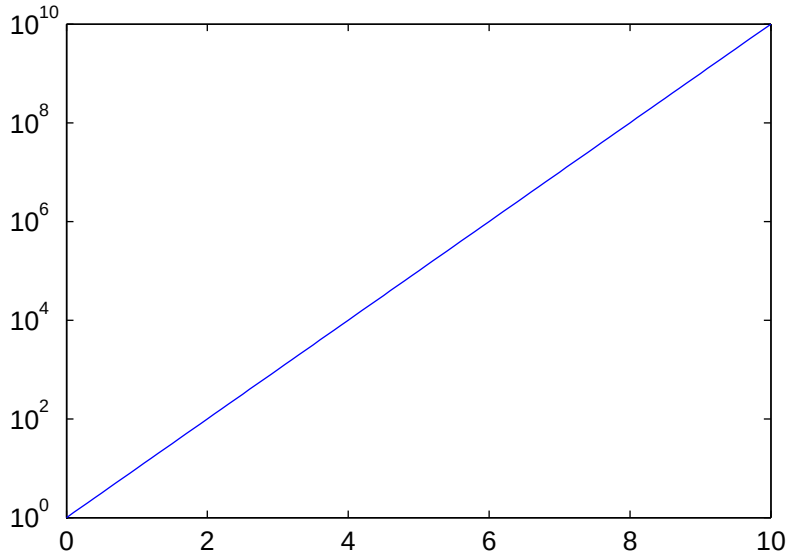
Popis `semilogx(x,y)` vytváří graf s logaritmickou stupnicí při základě 10 na x-ové ose a lineární stupnicí na y-ové ose.

`semilogy(x,y)` vytváří graf s logaritmickou stupnicí při základě 10 na y-ové a lineární stupnicí na x-ové ose.

Příkazy `loglog`, `semilogx` a `semilogy` jsou používány naprosto shodně s příkazem `plot`, ale zobrazují grafy v různých měřítkách.

Příklady Velice jednoduchý graf s použitím funkce `semilogy`:

```
x=0:0.1:10;
semilogy(x,10.^x)
```



Viz též `plot`, `loglog`, `xlabel`, `ylabel`

Funkce Nastavení vlastností objektu.

Syntaxe
```
set(h,PropertyName,PropertyValue,...)
set(h)
set(h,PropertyName)
```

Popis Funkci `set` použijeme k definování hodnot vlastností objektu (kromě případů, kdy je vlastnost určena pouze pro čtení)

```
v=set(h,PropertyName,PropertyValue,...)
```

V případě správného vykonání příkazu obsahuje hodnota `v` nulu.

Identifikátor `h` (jako argument funkce `set`) identifikuje objekt (nebo, je-li `h` vektor, objekty), jehož vlastnosti jsou nastavovány. V jediném vyvolání funkce `set` můžeme určit libovolný počet dvojic parametrů PropertyName/PropertyValue. MATLAB vykoná jedno *dlouhé* volání rychleji než několik *krátkých* volání.

Pro nastavení implicitních hodnot vlastností objektu přidáme k typu objektu a názvu vlastnosti slovo `Default`. Např. k nastavení implicitní barvy objektu axes pro nový objekt axes v aktuálním objektu figure použijeme příkaz

```
set(gcf,'DefaultAxesColor',[1 1 1])
```

Protože objekty kontrolují implicitní hodnoty vlastností svých rodičů, měli bychom nastavit implicitní barvu objektu axes v objektu figure, ve kterém chceme tuto hodnotu používat. Více informací o implicitních hodnotách viz uživatelská část této příručky.

`set(h)`, kde `h` je identifikátor objektu, zobrazí seznam všech nastavitelných vlastností pro tento objekt společně s možnými hodnotami pro každou vlastnost.

`set(h,PropertyName)` dává seznam možných hodnot pro uvedenou vlastnost.

Funkce `set` též jako parametr PropertyValue akceptuje řetězce `default`, `factory` a `remove`.

Specifikujeme-li hodnotu `default`, nastaví se vlastnost na první nalezenou implicitní hodnotu pro tuto vlastnost. Následující příkazy např. nastaví na zelenou barvu vlastnost objektu surface `EdgeColor` (barva hran):

```
h=surf(peaks)
set(0,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','default')
```

Pokud již na úrovni objektů axes nebo figure existuje implicitní hodnota pro vlastnost `EdgeColor`, nastavily by tyto příkazy `EdgeColor` objektu surface s identifikátorem `h` na tuto implicitní hodnotu.

Zadáme-li hodnotu `factory`, nastaví se vlastnost na svou hodnotu vestavěnou v MATLABu (factory setting). Následující příkazy nastaví vlastnost `EdgeColor` objektu surface s identifikátorem `h` na černou barvu (tj. factory setting) bez ohledu na definovanou implicitní hodnotu.

```
h=surf(peaks)
set(0,'DefaultSurfaceEdgeColor','g')
set(h,'EdgeColor','factory')
```

Řetězcem `remove` můžeme odstranit implicitní hodnoty nastavené uživatelem. Příkaz

```
set(0,'DefaultSurfaceEdgeColor','remove')
```

odstraní z úrovně root definici implicitně nastavené barvy hran. Hodnota pro `EdgeColor` objektu surface se vrátí zpět na svoji vestavěnou hodnotu (factory setting).

Chceme-li použít hodnoty `default`, `factory` nebo `remove` jako řetězce v popisech os, musíme před slovo umístit zpětné lomítko:

```
set(gca,'XLabel','\default')
```

Příklady Následující příkaz nastaví rozsahy os x, y a z aktuálního objektu axes na hodnoty 0 až 10 pro x-ovou osu, -25 až 25 pro y-ovou osu a -8 až 10 pro z-ovou osu:

```
set(gca,'XLim',[0 10],'YLim',[-25 25],'ZLim',[-8 10])
```

Funkce `set` poskytuje též seznam vlastností konkrétního určeného objektu. Např. následující příkaz udává vlastnosti, které se používají v objektu surface. Implicitní hodnoty jsou umístěny ve složených závorkách.

```
h=surf(peaks);
set(h)

Cdata
EdgeColor: [ none | {flat} | interp ] -or- a ColorSpec.
EraseMode: [ {normal} | background | xor | none ]
FaceColor: [ none | {flat} | interp | texturemap ] -or-  a ColorSpec
LineStyle: [ {-} | -- | : | -. | + | o | * | . | x ]
LineWidth
MarkerSize
MeshStyle: [ {both} | row | column ]
Xdata
Ydata
ZData

ButtonDownFcn
Clipping: [ {on} | off ]
```

```
Interruptible: [ {no} | yes ]
Parent
UserData
Visible: [ {on} | off ]
```

Viz též get, gca, gcf

Funkce Nastavuje příznak řetězce pro zobrazování.

Syntaxe `s=setstr(t)`

Popis `t=setstr(t)` nemění numerické hodnoty v matici `t`, ale říká MATLABu, aby tuto matici při tisku interpretoval jako ASCII znaky.

Prvky `t` jsou obvykle celá čísla v rozsahu 32:127, což jsou tisknutelné ASCII znaky, nebo v rozsaho 0:255, což jsou všechny 8-bitové hodnoty. Pokud nejsou prvky matice `t` celočíselné nebo jsou mimo rozsah 0:255, jsou tištěné znaky určeny vztahem

```
fix(rem(t,256))
```

Příklady Příkaz

```
ascii=setstr(reshape(32:127,32,3)')
```

tiskne 3x32 tisknutelných ASCII znaků

```
ascii=
 !"#$%&'()*+,-./0123456789:;<=>?
@ABCDEFGHIJKLMNOPQRSTUVWXYZ[/]^_
`abcdefghijklmnopqrstuvwxyz{|}~
```

Viz též `abs`, `isstr`, `strings`

Funkce Nastavení vlastnosti pro barevné odstíny.

Syntaxe

```
shading faceted
shading('faceted')
shading interp
shading('interp')
shading flat
shading('flat')
```

Popis `shading` řídí barevné odstíny objektů surface a patch. Objekty surface a patch jsou tvořeny funkcemi `surf`, `mesh`, `pcolor`, `fill` a `fill3`.

`shading flat` nastavuje odstíny aktuálního grafu na hodnotu `flat`. `shading interp` nastaví odstíny na hodnotu `interp`, `shading faceted` na hodnotu `faceted`, což je implicitní hodnota.

Odstín *flat* je po částech konstantní, každý segment drátové sítě nebo plochy má konstantní barvu určenou hodnotami barev v krajních bodech segmentu nebo v rozích plošky.

Interpolované odstíny, které jsou známé jako Gouraudovy odstíny, jsou po částech bilineární, barva každého segmentu nebo plošky se mění lineárně a interpoluje hodnoty na koncích segmentu nebo v rozích plošky.

Odstíny *faceted* jsou odstíny *flat* s černou drátovou sítí. Tato volba je často nejpůsobivější, a proto je nastavena jako implicitní hodnota.

`shading('faceted')` je ekvivalentní `shading faceted`, `shading('interp')` je ekvivalentní `shading interp` a `shading('flat')` příkazu `shading flat`.

Algoritmus Funkce `shading` nastavuje vlastnosti `EdgeColor` a `FaceColor` všech objektů surface v aktuálním objektu axes. Nastavuje je na správné hodnoty v závislosti na tom, zda objekty surface reprezentují drátové modely nebo plochy.

Viz též `surf`, `mesh`, pcolor—, `fill`, `fill3`, `surface`, `patch`, `hidden`, `surface`.
Vlastnosti `FaceColor` a `EdgeColor` objektu surface a patch.

Funkce Objemová vizualizace.

Syntaxe
```
slice(V,sx,sy,sz,nx)
h=slice(X,Y,Z,V,sx,sy,sz,nx)
```

Popis `slice(V,sx,sy,sz,nx)` kreslí řezy (objekty surface) objemu `V` v místech určených indexovými vektory `sx`, `sy` a `sz`. `nx` je počet řádků objemového pole `V`.

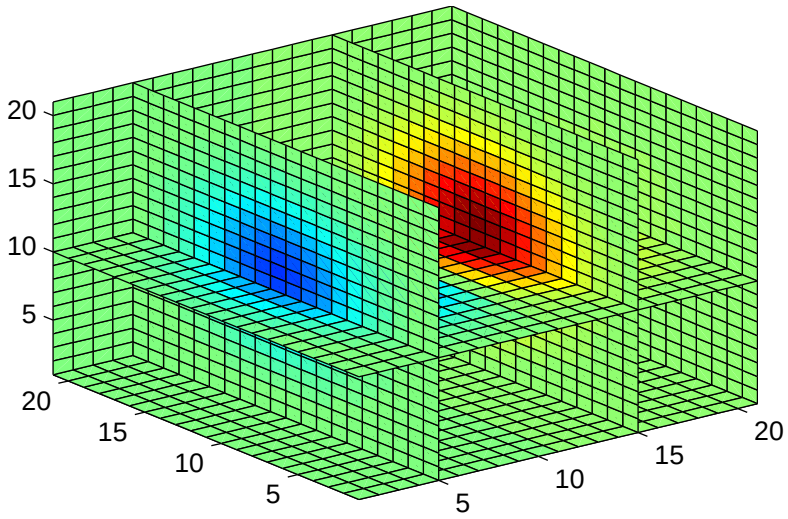
`slice(X,Y,Z,V,sx,sy,sz,nx)` kreslí řezy objemu určeného trojicemi `(X(i), Y(i), Z(i))`.

`slice` vrací vektor identifikátorů objektů surface.

Poznámka Implementováno od verze 4.1.

Příklady Vizualizace funkce $x\,e^{-x^2-y^2-z^2}$ v oblasti $-2 < x < 2$, $-2 < y < 2$, $-2 < z < 2$

```
[x,y,z]=meshgrid(-2:.2:2,-2:.2:2,-2:.2:2);
v=x.*exp(-x.^2-y.^2-z.^2);
slice(v,[5 15 21],21,[1 10],21)
```



Viz též `meshgrid`, `surface`

Funkce Zrcadlový odraz.

Syntaxe `r=specular(Nx,Ny,Nz,S,V,k)`

Popis `specular` vrací koeficienty odrazu zrcadlové plochy určené složkami vektoru normály. Odrazivost je ta část světla, která se odráží od plochy směrem k pozorovateli. Odrazivost se mění od 0 (žádný světelný odraz) do 1 (všechno světlo se odráží).

`specular(Nx,Ny,Nz,S,V,k)` vrací koeficienty odrazu plochy s vektorem normály o složkách `[Nx,Ny,Nz]`. Složky vektoru normály mohou být i matice, potom normála je

```
n(i,j)=[Nx(i,j),Ny(i,j),Nz(i,j)]
```

Tyto složky vektoru normály mohou být vypočteny prostřednictvím funkce `surfnorm`. Zdroj světla `S=[Sx,Sy,Sz]` je vektor o třech složkách, které určují směr, ze kterého je plocha osvětlena. Vektor `V` (délky 3) určuje bod pohledu. Oba vektory mohou mít též délku 2, pak určují azimut a elevaci. Argument `k` udává ostrost zrcadlového odrazu. Mění se v rozsahu $\langle 1,\ \infty)$. Implicitně nastavená hodnota je `k=10`. S rostoucím `k` se odlesky na zobrazovaném tělese stávají menší a ostřejší, dokonalé zrcadlo má `k=`∞.

`specular` je volán ve funkci `surfl`, která vytváří objekt surface včetně stínů.

Algoritmus `specular` provádí běžnou aproximaci pro odrazivost ploch, které mají podobné vlastnosti jako lesklé kovové plochy. Koeficienty odrazu nabývají nejvyšších hodnot, pokud je vektor normály ve směru $(s+\nu)/2$, kde s je směr zdroje a ν směr pohledu.

Viz též `surfl`, `diffuse`, `surfnorm`

Funkce Generování koule.

Syntaxe
```
[X,Y,Z]=sphere(n)
sphere(n)
```

Popis `sphere` generuje souřadnice (x, y, z) jednotkové koule pro následné použití funkcí `surf` a `mesh`.

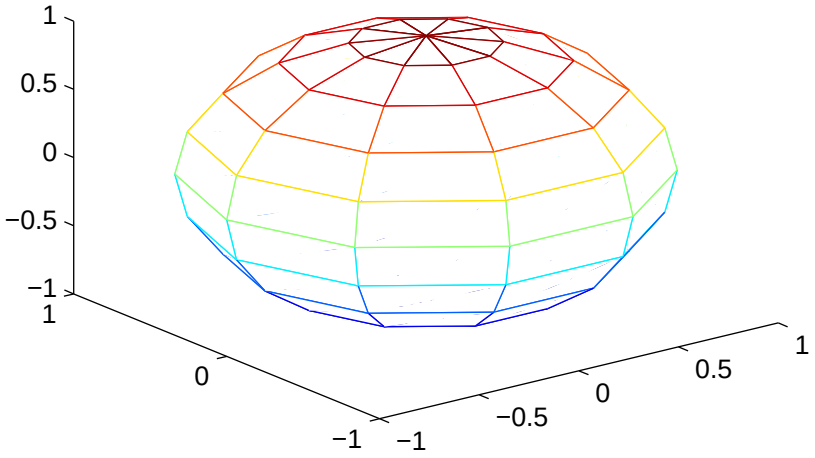
`[X,Y,Z]=sphere(n)` vrací souřadnice koule ve třech maticích typu $(n+1, n+1)$. Graf koule pak můžeme obdržet příkazem

```
surf(X,Y,Z)
```

`sphere(n)`, bez výstupních argumentů, kreslí graf koule na obrazovku (jako příkaz `surf`), argument `n` udává počet dělení po obvodu. Není-li zadán žádný vstupní argument, je `n=20`.

Příklady Generování a graf koule:

```
[X,Y,Z]=sphere(10);
mesh(X,Y,Z)
```



Viz též `cylinder`

Funkce Rotace mapy barev.

Syntaxe

```
spinmap
spinmap(t)
spinmap(t,inc)
spinmap(inf)
```

Popis

`spinmap` nechává cyklicky rotovat zvolenou aktuální mapu barev po dobu asi 5 sekund.

`spinmap(t)` nechává cyklicky rotovat zvolenou aktuální mapu barev po dobu `t` sekund.

`spinmap(t,inc)` používá zadaný časový přírůstek. Implicitně je `inc=2`, takže `inc=1` způsobí pomalejší rotaci, `inc=3` rychlejší rotaci, `inc=-2` rotaci v druhém směru, atd.

`spinmap(inf)` je nekonečná časová smyčka. Pro její přerušení je nutno stisknout **Ctrl-C**.

K zabránění vícenásobného překreslování lze použít příkaz

```
set(gcf,'sharecolors','no')
```

Viz též `colormap`

Funkce Zapisuje formátovaná data do řetězce.

Syntaxe
```
s=sprintf(format,A,...)
[s,errmessage]=sprintf(format,A,...)
```

Popis Funkce `num2str`, `int2str` a `sprintf` konvertují čísla do řetězcové reprezentace MATLABu a jsou mimo jiné vhodné pro tvorbu popisu os a grafů numerickými hodnotami.

`s=sprintf(format,A,...)` formátuje data z matice `A` (popř. dalších matic) podle formátovacího řetězce `format` a zapisuje je do řetězcové proměnné `s`.

`errmessage` je volitelný výstupní parametr, který v případě chyby obsahuje popis chyby, jinak je prázdný.

`sprintf` je stejné jako `fprintf` až na to, že vrací data do řetězcové proměnné namísto do souboru. Viz `fprintf` pro kompletní informaci o použití `sprintf`.

Příklady Příkaz

```
S=sprintf('rho je %6.3f',(1+sqrt(5))/2)
```

vytvoří řetězec

```
S='rho je 1.618'
```

Viz též `num2str`, `int2str`, `fprintf`, `sscanf`, `fwrite`

Funkce Čte formátovaná data z řetězce.

Syntaxe `[A,count,errmsg,nextindex]=sscanf(s,format,size)`

Popis `[A,count]=sscanf(s,format,size)` čte data z řetězcové proměnné `s`, převádí je podle daného formátovacího řetězce `format` a vrací je do matice `A`. `size` je volitelný vstupní parametr určující počet načítaných prvků. `count` je volitelný výstupní parametr, který vrací počet úspěšně načtených prvků.

`errmsg` je volitelný výstupní parametr, který v případě výskytu chyby vrací její popis.

`nextindex` obsahuje index naposled čtené hodnoty z řetězce `s`. Konec řetězce `s` lze testovat parametrem `errmsg`; `errmsg` je na konci řetězce `s` prázdný. Dále lze testovat konec řetězce `s` pomocí parametru `nextindex`; `nextindex` je větší než délka řetězce `s`.

`sscanf` je stejné jako `fscanf` až na to, že čte data z řetězcové proměnné namísto ze souboru. Viz `fscanf` pro kompletní informaci o použití `sscanf`.

Příklady Příkazy

```
S='2.7183 3.1416';
A=sscanf(S,'%f');
```

vytvoří dvojprvkový vektor obsahující hrubou aproximaci e a π.

Viz též `fscanf`, `fread`, `sprintf`, `eval`

Funkce Schodový graf.

Syntaxe

```
stairs(y)
stairs(x,y)
[xb,yb]=stairs(...)
```

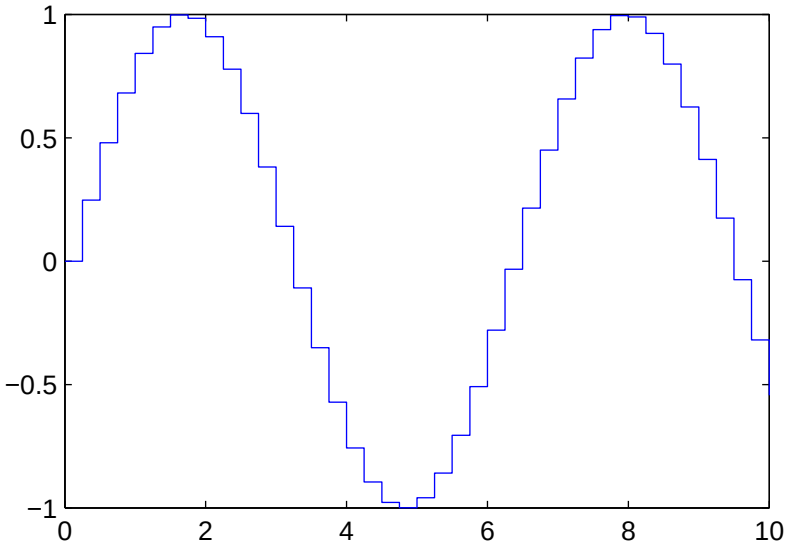
Popis `stairs(y)` kreslí schodový diagram prvků vektoru `y`. Schodový diagram je podobný sloupcovému diagramu generovanému funkcí `bar(y)`, ale bez svislých čar. Schodový diagram se hodí pro zakreslení časových posloupností číslicově vzorkovaných dat.

`stairs(x,y)` kreslí schodový diagram prvků vektoru `y` v místech určených vektorem `x`. Hodnoty ve vektoru `x` musí rovnoměrně vzrůstat.

`[xb,yb]=stairs(y)` a `[xb,yb]=stairs(x,y)` nekreslí grafy, ale vrací vektory `xb` a `yb`, které se použijí pro generování schodového diagramu příkazem `plot(xb,yb)`.

Příklady Vytvoření schodového diagramu sinové vlny:

```
x=0:0.25:10;
stairs(x,sin(x))
```



Viz též `bar`, `hist`

Funkce Vykreslení diskrétní posloupnosti - graf ve tvaru stonků.

Syntaxe
```
stem(y)
stem(x,y)
stem(...,linetype)
```

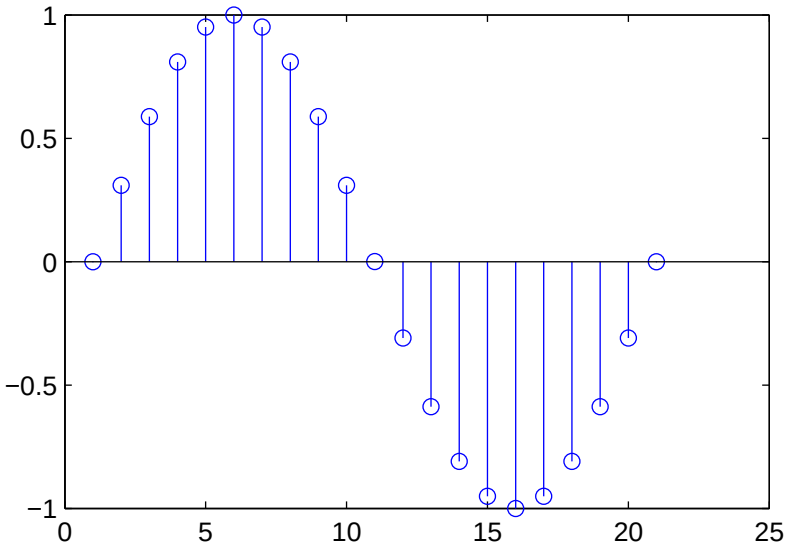
Popis `stem(y)` vykreslí posloupnost danou vektorem `y` jako stonky rostoucí z x-ové osy a zakončené kroužky v místech svých datových hodnot.

`stem(x,y)` vykreslí posloupnost danou vektorem `y` v místech určených vektorem `x`.

`stem(...,linetype)` vykreslí posloupnost dat typem čáry určeným parametrem `linetype`, např. `stem(x,y,'-.')` nebo `stem(y,':')`.

Poznámka Implementováno od verze 4.1.

Příklady
```
y=sin(0:pi/10:2*pi);
stem(y)
```



Viz též `plot`, `bar`, `stairs`

Funkce Vytváří řetězcovou matici z jednotlivých řetězců.

Syntaxe `S=str2mat(t1,t2,t3,...)`

Popis `S=str2mat(t1,t2,t3,...)` vytvoří matici `S` obsahující textové řetězce `t1`, `t2`, `t3`, ... jako řádky. Každý řetězec je automaticky doplněn příslušným počtem mezer, aby se vytvořila platná matice. K vytvoření matice `S` může být použito až 11 řetězců. Navíc každý textový parametr může být sám textovou maticí, což umožňuje vytvoření libovolně velké textové matice.

Příklady Tvorba řetězcové matice z řetězců `'Jeden'`, `'Dvacet'` a `'Třicet šest'`:

```
s=str2mat('Jeden','Dvacet','Třicet šest')

s=
Jeden
Dvacet
Třicet šest
```

Viz též `isstr`, `int2str`, `num2str`, `setstr`

Funkce Transformace řetězce na číslo.

Syntaxe `x=str2num(s)`

Popis `x=str2num(s)` transformuje řetězec `s`, což je ASCII reprezentace číselné hodnoty, na numerickou reprezentaci MATLABu. Řetězec smí obsahovat číslice, desetinnou tečku, úvodní znaménko + nebo -, označení exponentu **e** nebo **E** a **i** nebo **j** pro komplexní jednotku.

Pokud řetězec s nereprezentuje platné číslo, vrací `str2num(s)` prázdnou matici.

Algoritmus M-soubor pro `x=str2num(s)` je jednoduchý řádek:

```
eval(['x=' s ';']);
```

Příklady `str2num('3.14159e0')` je přibližně π.

Viz též `num2str`, hex2num—, `sscanf`

Funkce Porovnává řetězce.

Syntaxe `l=strcmp(s1,s2)`

Popis `strcmp(s1,s2)` porovnává řetězce `s1` a `s2` a vrací 1, jestliže jsou řetězce shodné, jinak vrací 0. Pozor, hodnota vrácená funkcí `strcmp` není stejná jako u jejího jmenovce v jazyce C.

`strcmp` je citlivá na velikost písmen; úvodní a závěrečné mezery v obou řetězcích se berou při porovnávání do úvahy.

Příklady

```
strcmp('Yes','No')=
0

strcmp('Yes','Yes')=
1
```

Viz též `isstr`, `setstr`, `lower`, `upper`, `findstr`

Funkce Přístup MATLABu k řetězcům.

Popis Příkaz `t='Hello,World.'` vytvoří vektor, jehož složky jsou ASCII kódy znaků řetězce. Velikost vektoru `t` odpovídá počtu znaků. Tento vektor se nijak neliší od ostatních vektorů MATLABu, až na to, že je-li zobrazen, je zobrazen text místo dekadických ASCII kódů.

```
t=
Hello, World.
```

Každé proměnné MATLABu je přidružen příznak, který, pokud je nastaven, říká výstupním funkcím MATLABu, aby zobrazily proměnnou jako text.

`t=abs(t)` je jedním ze způsobů jak vymazat tento příznak.

`t=setstr(t)` nastaví příznak zpátky k zobrazení textu.

`isstr(t)` vrací 1, pokud je `t` řetězec, jinak vrací 0.

Dva apostrofy za sebou označují jeden apostrof uvnitř řetězce.

Příklady Řetězec

```
s=['It is o''clock',7,13,'It is 2']
```

používá dva apostrofy k označení jednoho apostrofu a zahrnuje kód zvonku (ASCII 7), není použitelný na Macintoshi a Windows, a kód návratu vozíku (ASCII 13).

Viz též `abs`, `isstr`, `setstr`, `strcmp`

Funkce Zaměňuje řetězce.

Syntaxe `s=strrep(s1,s2,s3)`

Popis `s=strrep(s1,s2,s3)` nahrazuje všechny výskyty řetězce `s2` v řetězci `s1` řetězcem `s3`.

Poznámka Implementováno od verze 4.2.

Příklady Když

```
s1='This is a good example';
```

potom

```
strrep(s1, 'good','great')
'This is a great example'

strrep(s1,'bad','great')
'This is a good example'

strrep(s1,'','great')
'This is a good example'
```

Viz též `findstr`

Funkce Hledá výskyt prvního rámce v řetězci.

Syntaxe
```
[token]=strtok(str)
[token]=strtok(str,dlm)
[token,remainder]=strtok(str)
[token,remainder]=strtok(str,dlm)
```

Popis `[token,remainder]=strtok(str,dlm)` vrací první rámec (`token`) oddělený zvolenými oddělovači `dlm` ve zdrojovém řetězci str. Volitelně vrací také zbytek původního řetězce. dlm je vektor oddělovacích znaků. Pokud není tento argument zadán, uvažují se jako oddělovače *bílé* mezery.

Poznámka Implementováno od verze 4.2.

Příklady Když

```
s='1.25,2.55;3.75:4.85';
```

potom

```
strtok(s,',')
'1.25'

strtok(s,';')
'1.25,2.55'

strtok(s,':')
'1.25,2.55;3.75'
```

Viz též `isspace`

Funkce Vytváří a řídí objekty axes rozmístěné vedle sebe (dlaždičky).

Syntaxe
```
h=subplot(m,n,p)
subplot(m,n,p)
subplot(h)
```

Popis `h=subplot(m,n,p)` rozdělí grafické okno na malá obdélníková pole do tvaru matice typu (m, n); v `p`-tém poli vytvoří objekt axes, zaktualizuje jej a vrátí identifikátor tohoto nového objektu axes. Objekty axes se počítají po řádcích, nejprve horní řádek grafického okna, pak druhý, atd.

`subplot(m,n,p)` zaktualizuje objekt axes, pokud již objekt axes v uvedené pozici existuje.

`subplot(h)`, kde `h` je identifikátor objektu axes, je jen jiný způsob zaktuálnění konkrétního objektu axes pro následné grafické příkazy.

Způsobí-li specifikace `subplot` překrytí existujícího objektu axes novým objektem axes, je původní objekt axes zrušen. Příkaz `subplot(1,1,1)` tedy vymaže v grafickém okně všechny existující menší objekty axes a vytvoří jeden nový objekt axes, který pokrývá celé grafické okno.

Ke smazání všech objektů axes a nastavení implicitní konfigurace `subplot(1,1,1)` můžeme použít buď příkaz `clf` (clear figure) nebo `subplot(1,1,1)`.

Příklady Vykreslení grafu funkce sinus v horní polovině obrazovky a grafu funkce kosinus v dolní polovině obrazovky.

```
subplot(2,1,1), plot(sin(0:pi/10:2*pi))
subplot(2,1,2), plot(cos(0:pi/10:2*pi))
```

Viz též `axes`, `figure`, `gca`, `clf`, `cla`

Funkce Stínovaný model plochy ve 3D.

Syntaxe

```
surf(X,Y,Z,C)
surf(X,Y,Z)
surf(x,y,Z,C)
surf(x,y,Z)
surf(Z,C)
surf(Z)
surf(Z,C,PropertyName,PropertyValue)
surf(Z,PropertyName,PropertyValue)
h=surf(...)
surfc(...)
```

Popis V nejobecnější podobě má `surf` čtyři vstupní argumenty.

`surf(X,Y,Z,C)` kreslí barevnou parametrickou plochu určenou maticemi `X`, `Y` a `Z` barvou definovanou maticí `C`. v jednodušším případě mohou `X` a `Y` být vektory, popř. je můžeme vynechat. Matici `C` lze též vynechat.

Bod pohledu je určen funkcí `view`. Popisy os jsou buď určeny rozsahem polí `X`, `Y` a `Z` nebo aktuálním nastavením `axis`. Měřítko barev je dáno rozsahem matice `C` nebo aktuálním nastavením `caxis`. Transformované barevné hodnoty jsou použity jako indexy aktuální mapy barev.

`surf(X,Y,Z)` používá `C=Z`, takže barva je úměrná výšce plochy.

`surf(x,y,Z,C)` a `surf(x,y,Z)` s dvěma vektorovými argumenty, které nahrazují maticové argumenty, musí mít `length(x)=n` a `length(y)=m`, kde `[m,n]=size(Z)`. V tomto případě jsou vrcholy jednotlivých plošek objektu surface trojice `(x(j), y(i), Z(i,j))`. Vidíme, že `x` odpovídá sloupcům a `y` řádkům matice `Z`.

`surf(Z,C)` a `surf(Z)` používá `x=1:n` a `y=1:m`. V tomto případě, kdy je určena pouze výška `Z`, je plocha definována nad geometricky pravoúhlou sítí.

`surf` s jedním nebo dvěma vstupními argumenty může být doplněno jednou dvojicí PropertyName/PropertyValue.

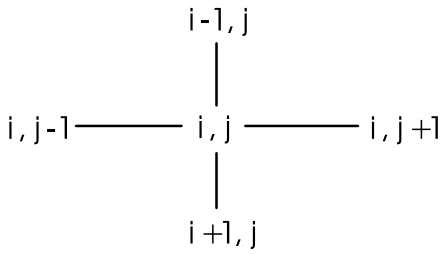
`h=surf(...)` vrací navíc identifikátor objektu surface. Objekty surface jsou dětmi objektů axes.

`surfc` navíc kreslí pod `surf` vrstevnice.

Algoritmus Teoreticky je parametrická plocha parametrizována dvojicí nezávislých proměnných i a j, které se spojitě mění, např. $1 \leq i \leq m$, $1 \leq j \leq n$. Plochy jsou určeny třemi funkcemi $x(i,j)$, $y(i,j)$ a $z(i,j)$. Pokud i a j jsou omezeny pouze na celá čísla,

definují pravoúhlou síť, ve které jsou průsečíky síťových čar tato celá čísla. Funkce $x(i,j)$, $y(i,j)$ a $z(i,j)$ se stávají maticemi `X`, `Y`, a `Z` typu (m,n). Čtvrtou funkcí je barva plochy $c(i,j)$, která se stává čtvrtou maticí `C`.

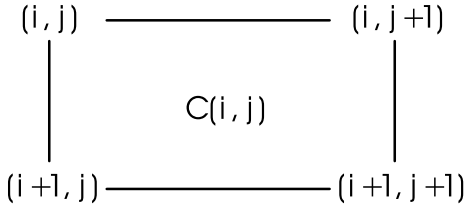
Každý bod v pravoúhlé síti je spojen se svými nejbližšími sousedy:



Tato základní pravoúhlá síť vytváří na ploše políčka o čtyřech stranách. Nebo jinak, `[X(:),Y(:),Z(:)]` vrací seznam trojic, které definují body v prostoru. Každý vnitřní bod je spojen se svými čtyřmi sousedy získanými z indexů matice. Body na hranách plochy mají jen tři sousedy a body v rozích pouze dva sousedy. Tímto je definován drátový model čtyřúhelníků.

Barva plochy může být určena dvěma různými způsoby: ve vrcholech nebo ve středech každého políčka. V obecném nastavení nemusí být plocha jednoznačnou funkcí x a y. Navíc plochy políček (se čtyřmi stranami) nemusí být rovinné. Mohou být např. znázorněny plochy definované v polárním, cylindrickém nebo sférickém souřadném systému.

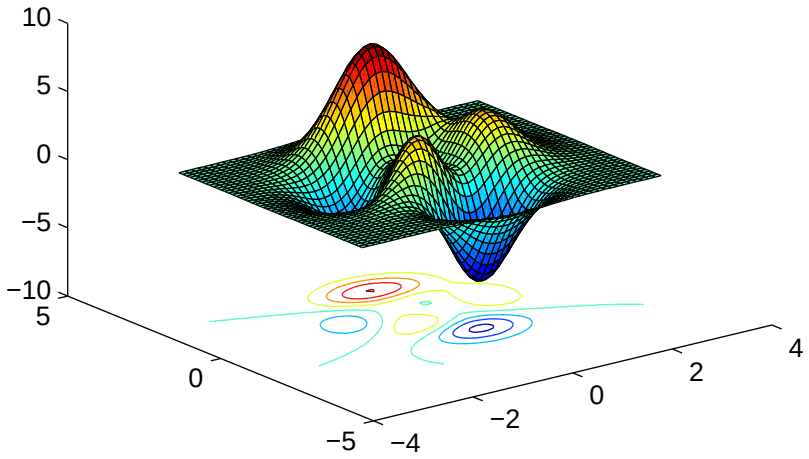
Funkce `shading` nastavuje odstíny barev. Má-li `shading` hodnotu `'interp'`, pak matice `C` musí být stejného typu jako matice `X`, `Y` a `Z`; toto nastavení určuje barvy ve vrcholech a barva uvnitř políčka je pak určena bilineární funkcí lokálních souřadnic. Je-li `shading 'faceted'` (implicitní hodnota) nebo `'flat'`, pak matice `C(i,j)` určuje konstantní barvu políčka:



V tomto případě může být matice `C` stejného typu jako matice `X`, `Y`, `Z` a její poslední řádek a sloupec jsou ignorovány, nebo je počet jejích řádků a sloupců o jednu menší než počet řádků a sloupců matic `X`, `Y` a `Z`.

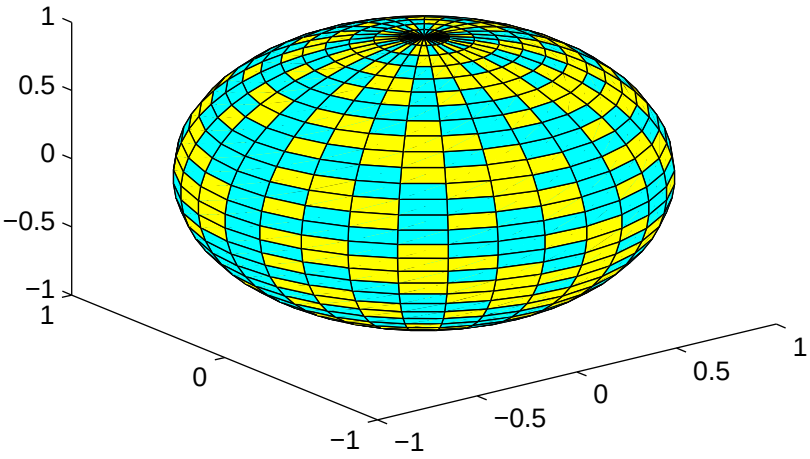
Příklady Vytvoření kombinovaného modelu plochy `peaks` a jejích vrstevnic.

```
[X,Y]=meshgrid(-3:0.125:3);
Z=peaks(X,Y);
surfc(X,Y,Z)
```



Složitější je příklad barevné koule s hodnotami +1 a -1 v Hadamardově matici.

```
k=5;
n=2^k-1;
[x,y,z]=sphere(n);
c=hadamard(2^k);
surf(x,y,z,c);
colormap([1 1 0; 0 1 1])
```



Viz též `axis`, `caxis`, `colormap`, `shading`, `view`, `mesh`, pcolor—, `contour`
vlastnosti objektů figure, axes a surface

Funkce Vytváření objektu surface.

Syntaxe
```
h=surface(X,Y,Z,C)
h=surface(X,Y,Z)
h=surface(Z,C)
h=surface(Z)
h=surface(X,Y,Z,PropertyName,PropertyValue,...)
```

Popis `surface` je grafická funkce nižší úrovně, která vytváří objekty surface. Objekty surface jsou dětmi objektů axes. Objekt surface je zobrazení matice dat s indexy prvků reprezentující souřadnice x a y a hodnotou každého prvku reprezentující buď výšku nad rovinou nebo index do mapy barev.

Funkce surface dovoluje použít jako vstupní parametry dvojice parametrů PropertyName/PropertyValue. Tyto vlastnosti, které řídí různý vzhled objektu surface, jsou popsány v oddíle *Vlastnosti objektu*. Hodnoty těchto vlastností lze též nastavit příkazem `set`. Příkazem `get` se můžeme dotázat na jejich aktuálně nastavenou hodnotu.

Na rozdíl od funkcí vyšší úrovně vytvářející plochy, jako je `surf` nebo `mesh`, nevymazává `surface` osy, nenastavuje parametry pro prohlížení ani neprovádí jiné akce, než je generování objektu surface v aktuálním objektu axes. To je vhodné zvláště v případě, chceme-li do existujícího objektu axes přidat objekt surface nebo jsou-li objekty surface kresleny prostřednictvím jejich vlastností v době vytváření objektu. Příkazy `axis`, `caxis`, `colormap`, `hold`, `shading` a `view` nastavují grafické vlastnosti, které ovlivňují `surface`.

Při nastavení některých vlastností dovoluje funkce `surface` vynechat jméno vlastnosti. Např. následující příkazy jsou ekvivalentní:

```
surface('XData',X,'YData',Y,'ZData',Z,'CData',C)

surface(X,Y,Z,C)
```

`surface(X,Y,Z,C)` kreslí barvou specifikovanou v matici `C` parametrickou plochu určenou maticemi `X`, `Y` a `Z`. v jednodušším případě mohou `X` a `Y` být vektory, popř. je můžeme vynechat. Matici `C` lze též vynechat.

`surface(X,Y,Z,C)` kreslí parametrickou barevnou plochu definovanou čtyřmi maticovými argumenty. Bod pohledu je určen funkcí `view`. Popisy os jsou buď určeny rozsahem polí `X`, `Y` a `Z`, nebo aktuálním nastavením `axis`. Měřítko barev je dáno rozsahem matice `C` nebo aktuálním nastavením `caxis`. Transformované barevné hodnoty

jsou použity jako indexy aktuální mapy barev. Příkaz `shading` nastavuje stínování modelu plochy.

`surface(X,Y,Z)` používá `C=Z`, což znamená, že je barva úměrná výšce plochy.

`surface(x,y,Z,C)` a `surface(x,y,Z)` s dvěma vektorovými argumenty namísto maticových argumentů musí mít `length(x)=n` a `length(y)=m`, kde `[m,n]=size(Z)`. V tomto případě trojice `(x(j),y(i),Z(i,j))` definují vrcholy jednotlivých políček plochy. Vidíme, že `x` odpovídá sloupcům a `y` řádkům matice `Z`.

`surface(Z,C)` a `surface(Z)` používá `x=1:n` a `y=1:m`. v tomto případě je výška `Z` jednoznačná funkce, která je definovaná nad geometricky pravoúhlou sítí.

Za maticovými argumenty mohou následovat dvojice parametrů PropertyName-/PropertyValue, které určují další vlastnosti objektu surface. Maticové argumenty můžeme úplně vynechat a vlastnosti specifikovat dvojicemi PropertyName/PropertyValue.

`h=surface(...)` vrací navíc identifikátor objektu surface. Objekty surface jsou dětmi objektů axes.

Úplná diskuse parametrických ploch viz popis funkce `surf`.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu dvojicemi PropertyName/PropertyValue, které jsou argumenty funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy je kurzor na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu.

Cdata *matice*

Barevná data. Tato vlastnost je matice hodnot, které specifikují barvu každého bodu v `ZData`. Matice `CData` ale nemusí nutně mít stejnou velikost jako `ZData`. Pokud tomu tak není, zachází s ní MATLAB jako s mapou `texture`. V tomto případě je obraz obsažený v `CData` přizpůsoben objektu surface, který je definován maticí `ZData`. Obsahují-li data plochy hodnoty `NaN`, nejsou tyto elementy kresleny (viz uživatelská část této příručky).

Children *vektor identifikátorů*

Děti objektu surface. Objekty surface nemají nikdy žádné děti, a proto je tato vlastnost vždy prázdná matice.

Clipping {`on`} | `off`

Režim ořezávání. Je-li `Clipping on`, nejsou žádné části objektu surface, které leží mimo obdélník os zobrazeny.

EdgeColor *ColorSpec* | `none` | `flat` | `interp`

Barva hrany objektu surface. Tato vlastnost nám umožňuje určit barvu hran samostatných políček, která tvoří objekt surface. `ColorSpec` definuje jednu barvu pro všechny hrany. Zvolíme-li hodnotu `none`, nejsou hrany kresleny. Hodnota `flat` použije pro hrany jedinou barvu na základě prvního záznamu `CData` pro toto políčko. Při hodnotě `interp` je barva hran získána lineární interpolací z hodnot barev ve vrcholech.

EraseMode {`normal`} | `none` | `xor` | `background`

Režim mazání. Tato vlastnost řídí techniku MATLABu používanou pro kreslení a mazání objektů surface. Tato vlastnost je užitečná při vytváření animovaných posloupností, protože řídí překreslování jednotlivých grafických objektů pro získání požadovaných efektů.

Režim `normal` překresluje danou oblast obrazovky tím, že provádí trojrozměrnou analýzu, která je nezbytná pro zabezpečení správného vynesení všech objektů. Tento režim přináší nejpřesnější obrázek, ale je také časově nejnáročnější. Ostatní režimy neprovádějí úplné překreslování, a proto jsou značně rychlejší, ale vytvářejí méně přesný obraz.

Je-li nastaven režim `none`, není objekt při přesunu nebo zničení vymazán.

Při režimu `xor` je objekt kreslen a mazán pomocí funkce `xor` aplikovanou na původní barvu. Je-li objekt vymazán, nepoškodí objekty pod ním. Ale je-li objekt kreslen v režimu `xor`, závisí jeho barva na barvě obrazovky pod ním. Objekt je správně vybarven pouze, je-li nad barvou pozadí objektu figure.

Režim `background` vytváří správně vybarvené objekty. Avšak objekt je mazán svým vykreslováním na barvě pozadí objektu figure. Tím se poškodí objekty, které jsou za mazaným objektem.

FaceColor *ColorSpec* | `none` | `flat` | `interp` | `texturemap`

Barva políček objektu surface. Tato vlastnost nám umožňuje určit barvu políček objektu surface (pro všechny políčka objektu surface můžeme definovat pouze jednu hodnotu). Zvolíme-li hodnotu `none`, nejsou políčka kreslena (ale můžeme ještě vykreslit hrany). Hodnota `flat` použije pro každé políčko jednu barvu na základě barvy prvního záznamu v matici CData pro toto políčko. Při hodnotě `interp` je barva políčka získána lineární interpolací z hodnot definovaných v každém bodu sítě na

objektu surface. Hodnota `texturemap` umístí 2D graf obsažený v matici `CData` na objekt surface.

Interruptible `yes | {no}`

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

LineStyle *typ čáry (implicitně '-')*

Typ čáry. Tato vlastnost určuje typ čáry použité pro vykreslení hran. Můžeme si vybrat z následujících typů čar. Požadovaný typ čáry určíme řetězcem, který obsahuje uvedené symboly z následujícího seznamu. (např. `'-'` pro plnou čáru). Plus, bod, hvězdička, kroužek a značka x jsou měřítkovatelné značky.

plná	`-`	kroužek	`o`
čárkovaná	`--`	plus	`+`
tečkovaná	`:`	bod	`.`
čerchovaná	`-.`	hvězdička	`*`
		značka x	`x`

LineWidth *šířka*

Šířka čáry. Nová šířka je určena v bodech (1 bod = 1/72 palce). Implicitní hodnota je 0.5 bodu.

MarkerSize *skalár (implicitně 6 bodů)*

Měřítkovací faktor velikosti značek. Je-li vlastnost `LineStyle` nastavena na značky, měřítkuje tato vlastnost jejich velikost. Používá se pouze pro značky typu bod, plus, hvězdička, kroužek a značka x.

MeshStyle `both | row | column`

Řádkové a sloupcové čáry. Jsou-li kresleny hrany, pak tato vlastnost určuje, zda se budou kreslit všechny hrany nebo pouze řádkové či sloupcové hrany.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu surface. Tato vlastnost je identifikátor rodiče objektu surface, kterým vždy je objekt axes.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
surface(peaks(30),'Tag','hory');
```

vytvoří objekt surface a označí ho visačkou `'hory'`. Je-li potom potřeba se někdy na tento objekt odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','hory')
```

Type *řetězec (pouze pro čtení)*

Typ objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekty surface je `Type` vždy řetězec `'surface'`.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice. kterou chceme přiřadit objektu. Objekt tato data nepoužije, ale my je můžeme získat příkazem `get`.

Visible {`on`} | `off`

Viditelnost objektu. Vlastnost `Visible` určuje, zda je či není objekt zobrazen na obrazovce.

XData *vektor* | *matice*

Souřadnicová data X. Tato vlastnost určuje x-ovou souřadnici bodů objektu surface. Má-li `XData` tvar řádkového vektoru, pak funkce `surface` vytvoří opakováním tohoto řádku matici `XData` se shodným počtem řádků, jako má matice `ZData`.

YData *vektor* | *matice*

Souřadnicová data Y. Tato vlastnost určuje y-ovou souřadnici bodů objektu surface. Má-li `YData` tvar sloupcového vektoru, pak funkce `surface` vytvoří opakováním tohoto sloupce matici `YData` se shodným počtem sloupců, jako má matice `ZData`.

ZData *matice*

Souřadnicová data Z. Tato vlastnost určuje z-ovou souřadnici bodů objektu surface. (Více informací viz část *Popis*)

Viz též `surf`, `mesh`, `contour`

Funkce Stínovaný model plochy ve 3D včetně osvětlení.

Syntaxe

```
surfl(Z)
surfl(Z,s)
surfl(x,y,Z)
surfl(x,y,Z,s)
surfl(x,y,Z,s,k)
```

Popis `surfl` vytváří stínovaný osvětlený model plochy, který je založen na kombinaci složek difúze, zrcadlení a složce příslušné okolnímu světlu.

Nejlepšího efektu osvětleného modelu plochy je dosaženo při šedé mapě barev (`gray`) nebo podobných barevných mapách (např. `copper`, `bone`, `pink`) a s interpolovanými odstíny.

`surfl(Z)` a `surfl(x,y,Z)` se používají shodným způsobem jako `surf`, mohou ale mít i další dva volitelné argumenty `s` a `k`. Argument `s` je vektor o třech složkách `s=[Sx,Sy,Sz]`, které určují směr, z něhož je plocha osvětlena. Vektor `s` může být též zadán ve sférických souřadnicích `s=[az,el]` azimutem a elevací. Argument `k` udává ostrost zrcadlového odrazu. Mění se v rozsahu $\langle 1, \infty)$. Implicitně nastavená hodnota je `k=10`. S rostoucím `k` se odlesky na zobrazovaném tělese stávají menší a ostřejší, dokonalé zrcadlo má `k=`∞.

Implicitní hodnota pro vektor `s` je 45 stupňů ve směru proti směru hodinových ručiček od aktuálního směru pohledu. Vektor `s` směřuje od objektu ke zdroji světla.

Příklady Pohled na model plochy `peaks` se světelným zdrojem:

```
[x,y]=meshgrid(-3:1/8:3);
z=peaks(x,y);
surfl(x,y,z);
shading interp
colormap(gray)
axis([-3 3 -3 3 -8 8])
surflc(X,Y,Z)
```

Viz též `diffuse`, `specular`, `colormap`, `shading`

Funkce Výpočet a zobrazení normál plochy ve 3D.

Syntaxe
```
surfnorm(Z)
surfnorm(X,Y,Z)
[Nx,Ny,Nz]=surfnorm(...)
```

Popis `surfnorm(Z)` a `surfnorm(X,Y,Z)` zobrazují objekt surface včetně normál, které směřují ven z plochy. Normály těch prvků plochy, které nejsou viditelné z místa pohledu, nejsou zobrazeny.

`[Nx,Ny,Nz]=surfnorm(X,Y,Z)` vrací složky normál plochy ve 3D pro objekt surface definovaný maticemi `X`, `Y` a `Z`. Normály plochy jsou nenormalizované a jsou vypočteny v každém vrcholu. Funkce `surfnorm` je volána funkcí `surfl` k výpočtu normál plochy.

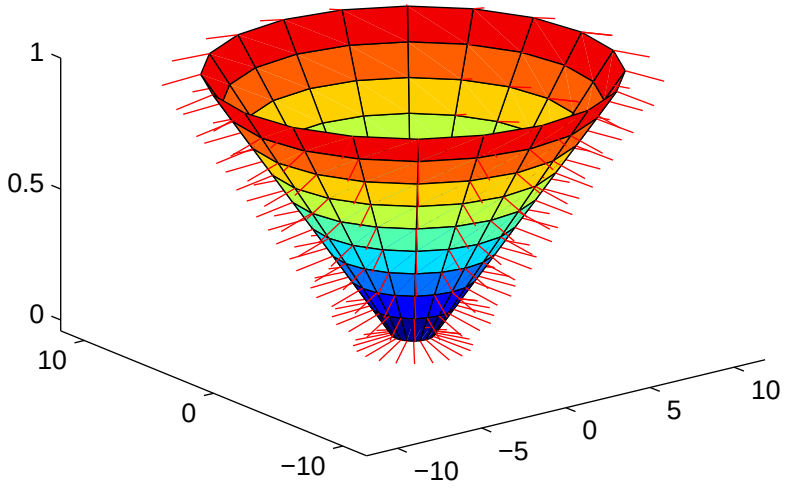
`[Nx,Ny,Nz]=surfnorm(Z)` vrací složky vektoru normály plochy ve 3D pro objekt surface definovaný maticí `Z`.

Směr normál lze otočit vyvoláním funkce `surfnorm` s transponovanými argumenty.

Algoritmus Výpočet normál plochy je založen na proložení dat `X`, `Y` a `Z` bikubickou plochou.

Příklady Vykreslení vektorů normál komolého kužele.

```
[x,y,z]=cylinder(1:10);
surfnorm(x,y,z)
```



Viz též `surfl`, `specular`, `diffuse`

Funkce Vrací jméno dočasného adresáře, pokud existuje.

Syntaxe `d=tempdir`

Popis `d=tempdir` vrací v řetězci `d` název dočasného adresáře, pokud existuje.

Poznámka Implementováno od verze 4.1.

Viz též `tempname`

Funkce Vrací jedinečné jméno souboru, které je vhodné pro využití jako dočasný soubor.

Syntaxe `n=tempname`

Popis `n=tempname` vrací v řetězci `n` jedinečné jméno souboru, které je vhodné pro využití jako dočasný soubor.

Poznámka Implementováno od verze 4.1.

Viz též `tempdir`

Funkce Přidání textu do aktuálního grafu vytvořením objektu text.

Syntaxe
```
h=text(x,y,string)
h=text(x,y,z,string)
h=text(x,y,z,PropertyName,PropertyValue,...)
```

Popis `text` je jak grafický příkaz vyšší úrovně pro přidání textu do grafu, tak i grafická funkce nižší úrovně, která vytváří objekty text.

Objekty text jsou dětmi objektů axes. `text` je funkce vytvářející objekt, která dovoluje použít jako vstupní argumenty dvojice parametrů PropertyName/PropertyValue. Tyto vlastnosti, které řídí různý vzhled objektu text, jsou popsány v oddíle *Vlastnosti objektu*. Hodnoty těchto vlastností lze též nastavit příkazem `set`. Příkazem `get` se můžeme dotázat na jejich aktuálně nastavenou hodnotu.

`text(x,y,string)` přidá do aktuálního objektu axes na pozici určenou bodem `(x,y)` řetězec daný řetězcovým parametrem `string`. Bod `(x,y)` je specifikován v jednotkách dat grafu aktuálního objektu axes. Jsou-li `x` a `y` vektory, zapíše funkce `text` uvedený řetězec na všechny pozice definované seznamem bodů.

Pokud je textový řetězec pole téže délky jako `x` a `y`, pak funkce `text` zapíše odpovídající řádek textového pole do příslušného bodu specifikovaného hodnotami `x` a `y`.

`text(x,y,z,string)` přidá text do grafu ve 3D.

`h=text(...)` vrací sloupcový vektor identifikátorů objektů text, pro každý objekt text jeden identifikátor.

Za dvojicemi `x`, `y` (resp. trojicemi `x`, `y`, `z`) mohou následovat dvojice parametrů PropertyName/PropertyValue, které určují další vlastnosti objektu text. Dvojice `x`, `y` (resp. trojice `x`, `y`, `z`) můžeme úplně vynechat a vlastnosti určit dvojicemi PropertyName/PropertyValue.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu dvojicemi parametrů PropertyName/PropertyValue, které jsou argumenty funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy je kurzor na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu.

Children *vektor identifikátorů*

Děti objektu text. Objekty text nemají nikdy žádné děti, a proto je tato vlastnost vždy prázdná matice.

Clipping `on | {off}`

Režim ořezávání. Je-li `Clipping on`, nejsou žádné části objektu text, které leží mimo obdélník os zobrazeny.

Color *ColorSpec*

Barva objektu text. Tato vlastnost určuje barvu objektu text. Implicitní barva je bílá. Více informací o určování barev viz popis příkazu `ColorSpec`.

EraseMode `{normal} | none | xor | background`

Režim mazání. Tato vlastnost řídí techniku MATLABu používanou pro kreslení a mazání objektů text. Tato vlastnost je užitečná při vytváření animovaných posloupností, protože řídí překreslování jednotlivých grafických objektů pro získání požadovaných efektů.

Režim `normal` překresluje danou oblast displeje tím, že provádí trojrozměrnou analýzu, která je nezbytná pro zabezpečení správného vynesení všech objektů. Tento režim přináší nejpřesnější obrázek, ale je také časově nejnáročnější. Ostatní režimy neprovádějí úplné překreslování, a proto jsou značně rychlejší, ale vytvářejí méně přesný obraz.

Je-li nastaven režim `none`, není objekt při přesunu nebo zničení vymazán.

Při režimu `xor` je objekt kreslen a mazán pomocí funkce `xor` aplikovanou na původní barvu. Je-li objekt vymazán, nepoškodí objekty pod ním. Ale je-li objekt kreslen v režimu `xor`, závisí jeho barva na barvě obrazovky pod ním. Je správně vybarven pouze je-li nad barvou pozadí objektu figure.

Režim `background` vytváří správně barvené objekty. Avšak objekt je mazán svým vykreslováním na barvě pozadí objektu figure. Tím se poškodí objekty, které jsou za mazaným objektem.

Extent *4-prvkový vektor (pouze pro čtení)*

Obdélník určující rozsah textu. Tato vlastnost je obdélník, který definuje velikost a polohu textového řetězce. Je určen vektorem `rect` tvaru

```
rect=[zleva, zdola, šířka, výška]
```

kde prvky `zleva` a `zdola` specifikují pozici levého dolního rohu obdélníku a prvky `šířka` a `výška` jeho rozměry. Aktuální hodnota vlastnosti `Units` určuje, jaké jednotky jsou použity k této specifikaci.

FontAngle {`normal`} | `italic` | `oblique`

Sklon písma. Tato vlastnost určuje sklon písma.

FontName *rodina fontů*

Rodina fontů. Tato vlastnost určuje rodinu fontů (např. Helvetica)

FontSize *velikost v bodech*

Velikost fontu. Tato vlastnost specifikuje velikost fontu v bodech (1 bod = 1/72 palce)

FontStrikeThrough `on` | {`off`}

Přeškrtnutí písma. Tato vlastnost určuje, zda je písmo přeškrtnuto.

FontUnderline `on` | {`off`}

Podtržení písma. Tato vlastnost určuje, zda je písmo podtrženo.

FontWeight `light` | {`normal`} | `demi` | `bold`

Světlost písma. Tato vlastnost určuje charakter váhy (světlost písma).

Font je kromě jména definován též dalšími charakteristikami, které se přidávají za jméno fontu. Ne všechny kombinace jsou však povoleny. MATLAB běžně podporuje čtyři rodiny fontů (Times, Helvetica, Courier a Symbol) a následujících 13 fontů:

Times-Roman	Helvetica	Courier
Times-Italic	*Helvetica-Oblique*	*Courier-Oblique*
Times-Bold	**Helvetica-Bold**	**Courier-Bold**
Times-BoldItalic	***Helvetica-BoldOblique***	***Courier-BoldOblique***
	Symbol	

Například, chceme-li používat 10-ti bodovou Helveticu (BoldOblique), musíme nastavit vlastnosti fontu následovně:

`FontName`	Helvetica
`FontSize`	10
`FontWeight`	bold
`FontAngle`	oblique

Pokud množina současně specifikovaných parametrů neodpovídá žádnému využitelnému fontu, používá MATLAB pro určení aktuálního fontu následující algoritmus:

1. MATLAB akceptuje oblique namísto italic.
2. Pokud není shoda stále nalezena, MATLAB ignoruje `FontAngle`.
3. Pokud není shoda stále nalezena, MATLAB ignoruje `FontWeght`.
4. Pokud není shoda stále nalezena, MATLAB ignoruje `FontSize`.
5. Pokud není shoda stále nalezena, MATLAB font nezmění.

Když MATLAB generuje tiskový výstup, nepokouší se před odesláním tohoto výstupu na dané tiskové zařízení zjistit, jaké fonty jsou na zařízení využitelné.

Implicitním fontem pro textové objekty a popis os je 12-ti bodová Helvetica. Pokud se používají TrueType fonty a Times i Helvetica jsou nevyužitelné, Times je nahrazen fontem NewTimesRoman, Helvetica fontem Arial a Courier fontem NewCourier.

HorizontalAlignment `left | center | right`

Horizontální zarovnání textu. Tato vlastnost určuje horizontální zarovnání textového řetězce vzhledem k poloze zadaného bodu. Např. nastavení této vlastnosti na `right` způsobí zarovnání řetězce tak, že hodnota vlastnosti `Position` (tj. poloha bodu) je vpravo od textového řetězce.

Interruptible `yes | {no}`

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

Parent *identifikátor (pouze pro čtení)*

Rodič objektu text. Tato vlastnost je identifikátor rodiče objektu text, kterým vždy je objekt axes.

Position `[x,y,[z]]`

Poloha textu. Tento vektor o dvou nebo třech prvcích definuje polohu textu ve 3D prostoru. Chybí-li hodnota `z`, je nahrazena hodnotou 0.
Jednotky, které jsou použity ke specifikaci této vlastnosti, jsou určeny aktuálním nastavením vlastnosti `Units`.

Rotation *skalár (default=0)*

Rotace textu. Tato vlastnost nastavuje jednu ze sedmi předdefinovaných orientací textu: 0, ±90, ±180, ±270. (Těchto sedm možností však definuje pouze čtyři různé orientace. Např. −90 dává stejný výsledek jako +270). Na některých systémech podporuje MATLAB rotaci s libovolným úhlem.

String *řetězec*

Textový řetězec. Tato vlastnost je textový řetězec, který je zobrazen.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
text(4,1,'Ahoj','Tag','Pokusný text');
```

vytvoří objekt text a označí ho visačkou `'Pokusný text'`. Je-li potom potřeba se někdy na tento objekt odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','Pokusný text')
```

Type *řetězec (pouze pro čtení)*

Typ objektu. Tato vlastnost identifikuje druh grafického objektu. Pro objekty text je `Type` vždy řetězec `'text'`.

Units `pixels | normalized | inches | centimeters | points | data`

Použité jednotky. Tato vlastnost definuje jednotky, které MATLAB používá pro interpretaci velikosti a umístění dat. Všechny jednotky jsou měřeny od levého dolního rohu okna. Normalizované jednotky transformují levý dolní roh stránky na $(0, 0)$ a horní pravý roh na $(1, 1)$. Pixly, palce, centimetry a body jsou absolutní jednotky (1 bod = 1/72 palce). `data` se odkazují na jednotky dat rodičů axes.

Tato vlastnost má vliv na vlastnosti `Extent` a `Position`. Změníme-li hodnotu vlastnosti `Units`, je dobrým zvykem ji po ukončení našich výpočtů vrátit na její implicitní hodnotu, aby neovlivnila ostatní funkce, které předpokládají implicitní nastavení této vlastnosti.

Definujeme-li jednotky dvojicemi NázevVlastnost/PropertyValue během vytváření objektu, musíme nastavit vlastnost `Units` před určením těch vlastností, u nichž chceme, aby tyto jednotky používaly.

UserData *matice*

Uživatelem určená data. `UserData` může být libovolná matice. kterou chceme přiřadit objektu. Objekt tato data nepoužije, ale my je můžeme získat příkazem `get`.

VerticalAlignment `top | cap | middle | baseline | bottom`

Vertikální zarovnání textu. Tato vlastnost určuje vertikální zarovnání textového řetězce vzhledem k poloze zadaného bodu. Nastavení této vlastnosti na `top` umístí text tak, že zadaný bod, který definuje polohu textu, je nyní na horním okraji textového řetězce, hodnota `middle` vycentruje text kolem zadaného bodu ve směru vertikálním a hodnota `bottom` umístí bod na spodní kraj textového řetězce.

Visible {`on`} `| off`

Viditelnost objektu. Vlastnost `Visible` určuje, zda je či není objekt zobrazen na obrazovce.

Příklad Příkazy

```
plot([1 5 10],[1 10 20],'x')
text(5,10,'Pracovní bod')
```

označí bod $(5, 10)$ řetězcem `Pracovní bod`.

Příkazy

```
plot(x1,y1,x2,y2)
text(x1,y1,'1'),text(x2,y2,'2')
```

popíší dvě křivky tak, že je lze snadno rozeznat.

Viz též `title`, `xlabel`, `gtext`, `plot`, `num2str`, `int2str`

Funkce Nadpis grafu.

Syntaxe `title(string)`

Popis `title(string)` zapíše text daný řetězcovou proměnnou `string` jako nadpis na horní okraj aktuálního grafu.

POZOR! Nelze změnit polohu nadpisu. Proto, chceme-li tímto textovým řetězcem pohybovat, je nutné jej vytvořit příkazem `gtext` nebo `text`.

Algoritmus `title` nastavuje vlastnost `Title` aktuálního objektu axes na nový objekt text.

Příklady Příkaz

```
title(date)
```

umístí dnešní datum na horní okraj aktuálního grafu.

V nadpisu lze též použít proměnných, např.

```
f=70; c=(f-32)/1.8;
title(['Teplota je ',num2str(c),'stupňů C'])
n=3
title(['Příklad číslo #',int2str(n)])
```

Viz též `xlabel`, `text`, `gtext`, `plot` , `num2str`, `int2str`

Funkce Vytváří objekt uživatelského rozhraní.

Syntaxe `h=uicontrol(PropertyName,PropertyValue,...)`

Popis `uicontrol` vytváří nástroj uživatelského rozhraní v aktuálním grafickém okně a vrátí jeho identifikátor `h`. MATLAB podporuje šest druhů nástrojů uživatelského rozhraní:

- Tlačítka (push buttons),
- Kontrolní boxy (check boxes),
- Menu (popup menus),
- Radio tlačítka (radio buttons),
- Táhla (sliders),
- Vstupní řádky (editable text).

Tlačítka odpovídají tlačítkům na telefonu – vyvolají akci při každém stisknutí, ale nezůstávají ve stisknuté poloze. Abychom zaktivovali tlačítko, je třeba najet myší na daný objekt tlačítka a stisknout a uvolnit tlačítko myši.

Kontrolní boxy vyvolají při stisknutí také akci, ale zůstávají ve stisknuté poloze, dokud nejsou stisknuty podruhé. Abychom zaktivovali kontrolní box, je třeba najet myší na daný objekt kontrolního boxu a stisknout a uvolnit tlačítko myši. Stav objektu je zobrazen na obrazovce.

Menu zobrazí seznam voleb, je-li stisknuto. Pokud není aktivováno, zobrazuje aktuální volbu.

Radio tlačítka jsou podobná kontrolním boxům, avšak vzájemně se vylučují (tj. v daném čase může být stisknuto pouze jedno radio tlačítko). Abychom zaktivovali radio tlačítko, je třeba najet myší na daný objekt radio tlačítka a stisknout a uvolnit tlačítko myši. Stav objektu je zobrazen na obrazovce. Poznamenejme, že vzájemná výlučnost radio tlačítek musí být ošetřena v našem kódu.

Táhla slouží pro vstup číselné hodnoty pomocí pohybu posuvného tlačítka uvnitř obdélníku (je třeba najet myší na posuvné tlačítko táhla, stisknout tlačítko myši a pohybem myši přesunout posuvné tlačítko do nové polohy a tlačítko myši uvolnit). Pozice posuvného tlačítka určuje číselnou hodnotu. Lze nastavit minimum, maximum a počáteční hodnotu táhla.

Vstupní řádky jsou boxy obsahující editovatelný text. Po zapsání požadovaného textu stiskněte **Ctrl-Return** nebo přesuňte textový kurzor mimo objekt, aby se spustila funkce Callback tohoto objektu.

Všechny tyto objekty jsou dětmi objektu figure, a jsou tedy nezávislé na osách.

`uicontrol` je funkce, která slouží pro vytvoření těchto objektů. Jejími vstupními parametry jsou dvojice PropertyName/PropertyValue. Tyto vlastnosti jsou popsány v části *Vlastnosti objektu*. Dále lze samozřejmě pro práci s vlastnostmi objektu využít funkce `get` a `set`.

Příklady Následující příkaz vytvoří tlačítko, které, je-li stisknuto, vyčistí aktuální osy:

```
h=uicontrol('Style','Pushbutton','Position',[20 150 100 70],...
            'Callback','cla','String','Clear');
```

Můžete vytvořit objekt, který mění mapu barev podle volby v menu

```
hpop=uicontrol('Style','Popup',...
               'String',' hsv | hot | cool | gray ',...
               'Position',[20 320 100 50],'Callback','setmap');
```

Tento objekt definuje menu s následujícími položkami: `hsv`, `hot`, `cool` a `gray`. Tyto volby jsou určeny vlastností `String`, každá volba je oddělena znakem `'|'`.

V tomto případě je vlastnost `Callback` jméno m-souboru, `setmap`:

```
val=get(hpop,'Value');
if val==1
  colormap(hsv)
elseif val==2
  colormap(hot)
elseif val==3
  colormap(cool)
elseif val==4
  colormap(gray)
end
```

Vlastnost `Value` obsahuje číslo, které udává volbu zvolenou uživatelem. Položky v menu jsou číslovány postupně od jedné do čtyř.

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu dvojicemi parametrů PropertyName/PropertyValue , které jsou argumenty funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

BackGroundColor *ColorSpec*

Barva objektu. Tato vlastnost určuje barvu pozadí objektu. Barvu lze určit RGB vektorem nebo předdefinovanými jmény MATLABu (více informací o barvách viz `ColorSpec`).

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy je kurzor na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu. Všimněme si, že funkce `CallBack` pro objekt uimenu nahrazuje `ButtonDownFcn`, ale objekty uicontrol mají funkci `CallBack` i funkci `ButtonDownFcn`.

CallBack *řetězec*

Akce objektu. Tato vlastnost určuje libovolný legální výraz MATLABu, včetně jména m-souboru nebo funkce. Pokud je objekt uicontrol aktivován, je tento řetězec předán funkci `eval` k vyhodnocení.

Children *vektor identifikátorů*

Děti objektu uicontrol. Objekty uicontrol nemají žádné děti, proto je tato vlastnost vždy prázdný vektor.

Clipping {`on`} | `off`

Režim ořezávání. V objektu uicontrol má vždy hodnotu `on`.

ForeGroundColor *ColorSpec*

Barva textu. Tato vlastnost určuje barvu textu zobrazeného na objektu. Barvu lze určit RGB vektorem nebo předdefinovanými jmény (více informací o barvách viz `ColorSpec`).

HorizontalAlignment `left` | `center` | {`right`}

Horizontální umístění popisu objektu. Tato vlastnost určuje, jak je umístěn popis objektu (vlastnost `String`).

Interruptible `yes` | {`no`}

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

U objektů uimenu a uicontrol tato vlastnost také rozhoduje o tom, zda mohou být funkce zpětného volání v průběhu své činnosti přerušeny či nikoliv.

Implicitní hodnota je opět hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím zpětného volání pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

Max *skalár*

Maximální hodnota. Tato vlastnost určuje největší hodnotu přípustnou pro vlastnost `Value`. Pro radio tlačítka a kontrolní boxy, které mohou nabývat pouze dvou stavů (`on` a `off`), tato vlastnost představuje hodnotu, na kterou je nastavena vlastnost `Value` při stavu `on`. Pro menu představuje tato vlastnost maximální počet položek v menu. Pro táhla představuje tato vlastnost největší hodnotu, kterou můžeme určit. Implicitní hodnota je 1.

Min *skalár*

Minimální hodnota. Tato vlastnost určuje nejmenší hodnotu přípustnou pro vlastnost `Value`. Pro radio tlačítka a kontrolní boxy, které mohou nabývat pouze dvou stavů (`on` a `off`), tato vlastnost představuje hodnotu, na kterou je nastavena vlastnost `Value` při stavu `off`. Pro táhla představuje tato vlastnost nejmenší hodnotu, kterou můžete určit. Implicitní hodnota je 0.

Parent *identifikátor*

Rodič objektu uicontrol. Tato vlastnost je identifikátor rodičovského objektu. Rodičem objektu uicontrol je objekt figure, ve kterém se uicontrol zobrazí. Tato vlastnost se může určit prvním parametrem funkce `uicontrol`, který bude udávat identifikátor rodičovského objektu (bez slova `Parent`). Např.

```
uicontrol(1,'Style','Slider',...)
```

Funkce `gcf` vrací identifikátor aktuálního objektu figure.

Position *4-prvkový vektor*

Umístění objektu. Tato vlastnost je obdélník, který specifikuje velikost a umístění objektu uicontrol uvnitř objektu figure. Obdélník je určen vektorem

```
rect=[zleva, zdola, šířka, výška]
```

kde `zleva` a `zdola` definují vzdálenost levého dolního rohu obdélníku od levého dolního rohu objektu figure, `šířka` a `výška` určují velikost obdélníku. U popisu vlastnosti `Units` jsou pro tuto specifikaci uvedeny informace o jednotkách.

String *řetězec*

Popis objektu. Tato vlastnost definuje popis na tlačítku, radio tlačítku, kontrolním boxu a menu. U víceřádkových položek (menu a vstupní řádky) jsou jednotlivé řádky odděleny znakem `'|'`.

Style `pushbutton | radiobutton | checkbox | slider | edit | popmenu`

Typ objektu uicontrol. Tato vlastnost definuje typ objektu, který chcete vytvořit. Viz část *Popis*.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
uicontrol('Style','Pushbutton','Position',[20 150 100 70],...
          'Tag','Pokusné tlačítko','Callback','cla');
```

vytvoří tlačítko a označí ho visačkou `'Pokusná osa'`. Je-li potom potřeba se někdy na toto tlačítko odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např.

```
h=findobj('Tag','Pokusné tlačítko')
```

Type *řetězec (pouze pro čtení)*

Typ grafického objektu. Tato vlastnost určuje druh grafického objektu. Pro objekty uicontrol je vlastnost `Type` vždy řetězec `'uicontrol'`.

Units {`pixels`} | `normalized` | `inches` | `centimeters` | `points`

Použité jednotky. Tato vlastnost určuje jednotky použité pro interpretaci vlastnosti `Position`. Všechny jednotky jsou počítány z levého dolního rohu grafického okna. U normalizovaných jednotek odpovídá levý dolní roh hodnotě $(0, 0)$ a horní pravý roh hodnotě $(1, 1)$. Palce, centimetry a body jsou absolutními jednotkami (1 bod = 1/72 palce).

UserData *matice*

Data specifikovaná uživatelem. `UserData` může být libovolná matice, kterou chceme objektu přiřadit. Objekt tato data nepoužije, ale my je můžeme získat funkcí `get`.

Value *skalár*

Aktuální hodnota objektu. Možné hodnoty závisí na stylu objektu:

- Radio tlačítko a kontrolní box nastaví `Value` na `Max` (obvykle 1), jsou-li ve stavu `on` (tlačítko je stisknuto), nebo na `Min` (obvykle 0), jsou-li ve stavu `off`.
- Táhla nastaví `Value` na číslo odpovídající pozici posuvného tlačítka vzhledem k rozsahu určenému vlastnostmi `Min` a `Max`.
- Menu nastaví `Value` na pořadové číslo vybrané položky menu.
- Tlačítka a vstupní řádky tuto vlastnost nenastavují.

Vlastnost `Value` lze nastavit interaktivně myší nebo použitím funkce `set`.

Visible {`on`} | `off`

Viditelnost objektu. Tato vlastnost určuje, zda je či není objekt zobrazen na obrazovce.

Viz též `set`, `get`, `uimenu`

Funkce Interaktivní volba souboru pro načtení.

Syntaxe `[filename,pathname]=uigetfile(filterSpec,dialogTitle,X,Y)`

Popis `[filename,pathname]=uigetfile(filterSpec,dialogTitle,X,Y)` zobrazí dialogový box, ve kterém si uživatel vybere požadovaný soubor, a potom vrátí jméno souboru `filename` a cestu k tomuto souboru `pathname`. Pokud uživatel zadá soubor, který neexistuje, zobrazí se chybové hlášení a řízení je vráceno dialogovému boxu. Uživatel potom může zadat jiné jméno souboru nebo stisknout tlačítko **Cancel**.

Všechny parametry jsou volitelné, ale pokud je použit jeden parametr, musí být zadány všechny předchozí parametry.

Řetězcový parametr `filterSpec` určuje masku pro počáteční seznam souborů v dialogovém boxu. Např. `'*.m'` vypíše všechny m-soubory MATLABu.

Parametr `dialogTitle` je řetězec obsahující nadpis dialogového boxu.

Parametry `X` a `Y` definují počáteční pozici dialogového boxu v pixlech. Některé systémy nemusí tuto volbu podporovat.

Výstupní parametr `filename` je řetězec obsahující jméno souboru vybraného v dialogovém boxu. Pokud uživatel stiskne tlačítko **Cancel** nebo se vyskytne chyba, je tato proměnná nastavena na nulu.

Výstupní parametr `pathname` je řetězec obsahující cestu k souboru vybranému v dialogovém boxu. Pokud uživatel stiskne tlačítko **Cancel** nebo se vyskytne chyba, je tato proměnná nastavena na nulu.

Poznámka Implementováno od verze 4.1.

Viz též `uiputfile`

Funkce Vytvoří uživatelské menu.

Syntaxe
```
h=uimenu(PropertyName,PropertyValue,...)
hsub=uimenu(h, PropertyName, PropertyValue,...)
```

Popis `uimenu` vytvoří menu, která mohou vykonávat předdefinované akce. Tato menu se zobrazí na horním okraji aktuálního grafického okna. Položkou menu může být další menu (podmenu).

Podmenu můžeme vytvořit zadáním identifikátoru rodičovského menu do funkce `uimenu`. Pokud nezadáme identifikátor existujícího menu nebo grafického okna, stane se rodičem aktuální grafické okno.

Vlastnosti menu mohou být nastaveny v době vytvoření objektu použitím parametrů PropertyName/PropertyValue ve funkci uimenu nebo mohou být vlastnosti změněny později použitím funkce `set`.

Příklady Tento příklad vytvoří menu pojmenované `'Workspace'`, jehož položky umožní uživateli vytvořit nové grafické okno, uložit proměnné a opustit MATLAB.

```
f=uimenu('Label','Workspace');
uimenu(f,'Label','New Figure','Callback','figure');
uimenu(f,'Label','Save','Callback','save');
uimenu(f,'Label','Quit','Callback','exit',...
         'Separator','on','Accelerator','Q');
```

Vlastnosti objektu

Vlastnosti objektu můžeme určit buď v době vytváření objektu dvojicemi parametrů PropertyName/PropertyValue, které jsou argumenty funkce vytvářející objekt, nebo je můžeme specifikovat až po vytvoření objektu identifikací objektu a funkcemi `get` a `set`.

V této kapitole je uveden seznam názvů vlastností spolu s typem jejich možných hodnot. Implicitně nastavené hodnoty jsou uvnitř složených závorek.

Accelerator *znak*

Odpovídající vstup z klávesnice. Tato vlastnost určuje pro položku menu odpovídající vstup z klávesnice. To umožňuje uživateli určit nějakou položku menu stisknutím určité posloupnosti kláves, ne myší.

Posloupnost kláves je závislá na systému:

znak-**Ctrl**	X-Windows
znak-**Command**	Macintosh
znak-**Ctrl**	PC

BackGroundColor *ColorSpec*

Barva pozadí. Tato vlastnost určuje barvu pozadí objektu uimenu. Barvu lze určit RGB vektorem nebo předdefinovanými jmény MATLABu (více informací o barvách viz `ColorSpec`). Implicitní barva je světle šedá; RGB vektor `[0.7020 0.7020 0.7020]`.

ButtonDownFcn *řetězec*

Funkce zpětného volání. Vlastnost `ButtonDownFcn` nám dovoluje definovat funkci, která bude vykonána, stiskneme-li tlačítko myši v době, kdy je kurzor na odpovídajícím objektu. Funkci zpětného volání definujeme řetězcem, který je vyhodnocen příkazem `eval`. Řetězcem může proto být libovolný platný výraz MATLABu nebo jméno m-souboru. Řetězec je vykonán v pracovním prostoru MATLABu. Všimněme si, že funkce `CallBack` pro objekt uimenu nahrazuje `ButtonDownFcn`, ale objekty uicontrol mají funkci `CallBack` i funkci `ButtonDownFcn`.

CallBack *řetězec*

Akce menu. Tato vlastnost určuje libovolný legální výraz MATLABu, včetně jména m-souboru nebo funkce. Pokud je menu aktivováno, je tento řetězec předán funkci `eval` k vyhodnocení.

Checked `on | {off}`

Zatrhávací značka. Pokud se tato vlastnost nastaví na `on`, objeví se vedle dané položky menu zatrhávací značka.

Children *vektor identifikátorů*

Děti objektu uimenu. Děti objektu uimenu jsou jiné objekty uimenu (podmenu).

Clipping `{on} | off`

Režim ořezávání. V objektu uimenu má vždy hodnotu `on`.

Enable `{on} | off`

Režim přístupu k položce. Tato vlastnost určuje, zda lze či nelze danou položku menu vybrat. Zakázaná položka se vykreslí tlumeně, čímž oznamuje, že ji nelze vybrat.

ForeGroundColor *ColorSpec*

Barva textu. Tato vlastnost určuje barvu textu zobrazeného na objektu uimenu. Barvu lze určit RGB vektorem nebo předdefinovanými jmény MATLABu (více informací o barvách viz `ColorSpec`). Implicitní barva je černá.

Interruptible `yes | {no}`

Režim přerušení. Tato vlastnost rozhoduje o tom, zda může být akce definovaná pomocí `ButtonDownFcn` během své činnosti přerušena či nikoliv. Implicitní hodnota je hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím pracovat, dokud není akce ukončena.

U objektů uimenu a uicontrol tato vlastnost také rozhoduje o tom, zda mohou být funkce zpětného volání v průběhu své činnosti přerušeny či nikoliv.

Implicitní hodnota je opět hodnota `no`, což znamená, že MATLAB nepovoluje ostatním funkcím zpětného volání pracovat, dokud není akce ukončena.

Má-li vlastnost objektu `Interruptible` hodnotu `yes`, musí se o obnovení (nebo alespoň zaznamenání) podmínek, které existovaly v okamžiku přerušení funkce zpětného volání, postarat sama funkce zpětného volání.

Label *řetězec*

Popis objektu. Popis položky menu.

Parent *identifikátor*

Rodiče objektu uimenu. Tato vlastnost je identifikátor rodičovského objektu. Rodičem objektu uimenu je objekt figure nebo objekt uimenu. Pokud chceme vytvořit podmenu, musíme tuto vlastnost nastavit.

Position *skalár*

Relativní pozice menu. Tato vlastnost řídí umístění položek menu. Základní menu se umisťují zleva doprava na horní řádku grafického okna podle hodnoty své vlastnosti `Position`. Individuální položky podmenu se umisťují shora dolů podle hodnoty své vlastnosti `Position`.

Separator `on | {off}`

Oddělovací linka. Nastavení této vlastnosti na `on` způsobí vykreslení oddělovací linky nad položkou menu.

Tag *řetězec*

Označení objektu. Tato vlastnost definuje uživatelské jméno objektu. Např.

```
uimenu('Label','Workspace','Tag','Pokusné menu');
```

vytvoří menu a označí ho visačkou `'Pokusné menu'`. Je-li potom potřeba se někdy na toto menu odkázat, lze jeho identifikátor jednoduše najít pomocí funkce `findobj`, např. `h=findobj('Tag','Pokusné menu')`

Type *řetězec (pouze pro čtení)*

Typ grafického objektu. Tato vlastnost určuje druh grafického objektu. Pro objekty uimenu je vlastnost `Type` vždy řetězec `'uimenu'`.

UserData *matice*

Data specifikovaná uživatelem. `UserData` může být libovolná matice, kterou chceme objektu přiřadit. Objekt tato data nepoužije, ale my je můžeme získat funkcí `get`.

Visible `{on} | off`

Viditelnost objektu. Tato vlastnost určuje, zda je či není objekt zobrazen na obrazovce. Pokud je vlastnost `Visible` objektu uimenu nastavena na `off`, menu se nezobrazí.

Viz též `set`, `get`, `uicontrol`

Funkce Interaktivní volba souboru pro uložení.

Syntaxe `[filename,pathname]=uiputfile(initFile,dialogTitle,X,Y)`

Popis `[filename,pathname]=uiputfile(initFile,dialogTitle,X,Y)` zobrazí dialogový box, ve kterém si uživatel vybere požadovaný soubor, a potom vrátí jméno souboru `filename` a cestu k tomuto souboru `pathname`. Úspěšný návrat z funkce nastane pouze v případě, že soubor neexistuje nebo pokud uživatel explicitně souhlasí se smazáním vybraného souboru.

Pokud uživatel určí soubor, který existuje, objeví se dotaz, zda si přeje tento soubor smazat.

Všechny parametry jsou volitelné, ale pokud je použit jeden parametr, musí být zadány všechny předchozí parametry.

Řetězcový parametr `initFile` určuje masku pro počáteční seznam souborů v dialogovém boxu. Je možno zadat jak plné jméno souboru, tak žolíkové znaky (wildcards) `?` a `*`. Např. maska `'*.m'` vypíše všechny existující soubory MATLABu.

Parametr `'dialogTitle'` je řetězec obsahující nadpis dialogového boxu.

Parametry `X` a `Y` definují počáteční pozici dialogového boxu v pixlech. Některé systémy nemusí tuto volbu podporovat.

Výstupní parametr `filename` je řetězec obsahující jméno souboru vybraného v dialogovém boxu. Pokud uživatel stiskne tlačítko **Cancel** nebo se vyskytne chyba, je tato proměnná nastavena na nulu.

Výstupní parametr `pathname` je řetězec obsahující cestu k souboru vybranému v dialogovém boxu. Pokud uživatel stiskne tlačítko **Cancel** nebo se vyskytne chyba, je tato proměnná nastavena na nulu.

Poznámka Implementováno od verze 4.1.

Příklady
```
[newmatfile,newpath]=uiputfile('*.mat','Save As');
```

Viz též `uigetfile`

Funkce Interaktivní nastavení *ColorSpec*.

Syntaxe `C=uisetcolor(arg,dialogTitle)`

Popis `C=uisetcolor(arg,dialogTitle)` zobrazí dialogový box, ve kterém si uživatel vybere barvu, kterou může dále aplikovat na libovolný grafický objekt.

Parametry jsou volitelné a mohou být zadávány v libovolném pořadí.

`arg` může být buď identifikátor grafického objektu nebo RGB vektor (např. `[1 0 0]` pro červenou). Pokud je použit identifikátor, musí specifikovat grafický objekt, který podporuje barvu. V obou případech je specikovaná barva použita v dialogovém boxu jako inicializační barva. Pokud není specifikována žádná barva, je použita v dialogovém boxu jako inicializační barva černá.

Parametr `dialogTitle` je řetězec obsahující nadpis dialogového boxu.

Výstupní parametr `C` je požadovaný RGB vektor. Pokud je vstupní parametr identifikátor grafického objektu, je barva tohoto objektu nastavena podle požadovaného RGB vektoru.

Pokud uživatel stiskne tlačítko **Cancel** nebo se vyskytne nějaká chyba, je výstupní parametr, pokud byl zadán vstupní RGB vektor, nastaven na hodnotu tohoto vstupního RGB vektoru; jinak je vrácena 0.

Příklady

```
C=uisetcolor(hText,'Set Text Color')
```

Funkce Interaktivní nastavení fontu.

Syntaxe `h=uisetfont(hin,dialogTitle)`

Popis `h=uisetfont(hin,dialogTitle)` zobrazí dialogový box, ve kterém uživatel zadá požadovaný font a aplikuje zvolený font na vstupní grafický objekt.

Parametry jsou volitelné a lze je zadat v libovolném pořadí.

Je-li zadán parameter `hin`, musí specifikovat identifikátor textového objektu nebo objektu os. Fontové vlastnosti tohoto objektu jsou použity k inicializaci dialogového boxu.

Parametr `dialogTitle` je řetězec obsahující nadpis dialogového boxu.

Výstupní parametr `h` je identifikátor grafického objektu. Pokud je zadán `hin`, je `h` identické s `hin`. Jestilže `hin` nebylo zadáno, je vytvořen nový textový objekt se zadanými fontovými vlastnostmi a vrácen jeho identifikátor.

Pokud uživatel stiskne tlačítko **Cancel** nebo se vyskytne nějaká chyba, je výstupní parametr, pokud byl zadán vstupní parametr `hin`, nastaven na hodnotu tohoto vstupního parametru; jinak je vrácena 0.

Příklady `uisetfont(hText,'Update Font')`

Funkce Transformuje seznam hran sítě na matice grafu.

Syntaxe `[A,xy]=unmesh(M)`

Popis `[A,xy]=unmesh(M)` transformuje seznam hran sítě popsaných maticí `M` na matice grafu `A` a `xy`. Každý řádek matice `M` představuje souřadnice koncových bodů hran sítě ve 2-D, tj. $[x1\, y1\, x2\, y2]$. Výstupní matice `A` je Laplaceovská matice sítě (symetrická matice, kde pro hranu spojující uzel `i` a `j` platí `A(i,j)=-1` a prvky `A(i,i)` se rovnají počtu hran vycházejících z uzlu `i`). Každý řádek matice `xy` představuje souřadnice $[x\, y]$ bodu sítě.

Příklady

```
[A,xy]=unmesh([1 1 2 1; 2 2 1 2; 1 1 2 2]);

4 vertices:

4/4

A=
(1,1) 2
(3,1) -1
(4,1) -1
(2,2) 1
(4,2) -1
(1,3) -1
(3,3) 1
(1,4) -1
(2,4) -1
(4,4) 2

xy=
1.0000 1.0000
1.0000 2.0000
2.0000 1.0000
2.0000 2.0000
```

Viz též `gplot`, `mesh`

Funkce Transformuje malá písmena v řetězci na velká.

Syntaxe `t=upper(s)`

Popis `upper(s)` vrací řetězec vytvořený z původního řetězce transformací malých písmen na velká a ponecháním ostatních znaků beze změny.

Algoritmus Všechny hodnoty v rozsahu `'a':'z'` jsou zvětšeny o `'A'-'a'`.

Příklady `upper('attention!')` je `'ATTENTION!'`

Viz též `lower`, `isstr`, `strcmp`

Funkce Definování bodu pohledu na graf ve 3D.

Syntaxe

```
view(az,el)
view([az,el])
view([x,y,z])
view(2)
view(3)
view(T) [az,el]=view
T=view
```

Popis `view(az,el)` a `view([az,el])` nastavuje úhel pohledu na graf 3D. Hodnota `az` je azimut, neboli horizontální rotace, a hodnota `el` je vertikální elevace (obě hodnoty jsou zadávány ve stupních). Azimut se otáčí kolem osy z s kladnými hodnotami ve směru proti pohybu hodinových ručiček. Kladné hodnoty elevace odpovídají pohledu shora; záporné hodnoty pohledu zdola.

`view([x,y,z])` nastavuje úhel pohledu v kartézských souřadnicích. Velikosti `x`, `y` a `z` jsou ignorovány.

`view(2)` nastavuje implicitní 2D pohled, tj. `az=0`, `el=90`.

`view(3)` nastavuje implicitní 3D pohled, tj. `az=-37.5`, `el=30`.

`view(T)` nastavuje pohled podle zadané transformační matice `T` řádu 4. Tuto matici můžeme vygenerovat funkcí `viewmtx`, která nám umožní definovat i perspektivní transformaci.

`[az,el]=view` vrací aktuální hodnoty azimutu a elevace.

`T=view` vrací aktuální transformační matici `T` řádu 4.

Příklady

```
el=90             %pohled přímo shora
az=el=0           %pohled přímo zepředu
az=180            %pohled zezadu
az=-37.5, el=30   %implicitní 3D pohled
```

Viz též `viewmtx`
Vlastnosti `View` a `XForm` objektu axes.

Funkce Transformační matice pohledu.

Syntaxe
```
T=viewmtx(az,el)
T=viewmtx(az, el,phi)
T=viewmtx(az,el,phi,tp)
```

Popis `viewmtx` počítá transformační matici řádu 4 rovnoběžného nebo středového (perspektivního) promítání, která popisuje transformaci čtyřrozměrných homogenních vektorů na dvojrozměrnou plochu (např. na obrazovku počítače). Čtyřrozměrný homogenní vektor je tvořen přidáním čísla 1 k odpovídajícímu trojrozměrnému vektoru. Např. čtyřrozměrný vektor odpovídající bodu `[x,y,z]` ve 3D je vektor `[x,y,z,1]`. Složky `x` a `y` výsledného promítaného vektoru jsou požadované složky ve 2D (viz příklad níže).

`T=viewmtx(az,el)` vrací transformační matici rovnoběžného promítání odpovídající azimutu `az` a elevaci `el`. Funkce `viewmtx` používá tutéž definici pro azimut a elevaci jako funkce `view` (hodnoty `az` a `el` jsou určeny ve stupních).

`T=viewmtx(az,el)` vrací tutéž matici jako příkazy

```
view(az,el)
T=view
```

ale nemění aktuální pohled.

`T=viewmtx(az,el,phi)` vrací transformační matici středového (perspektivního) promítání. Hodnota `phi` je úhel, pod kterým je vidět normalizovaná krychle (ve stupních); řídí stupeň perspektivního zkreslení.

`phi=0`	rovnoběžné promítání,
`phi=10`	stupňů teleobjektiv,
`phi=25`	stupňů normální objektiv,
`phi=60`	stupňů širokoúhlý objektiv.

Získaná matice `T` může být použita funkcí `view(T)` pro nastavení pohledu. Transformační matice středového promítání transformuje čtyřrozměrné homogenní vektory na nenormalizované vektory tvaru `(x,y,z,w)`, kde `w` nemusí být rovno 1. První dva prvky normalizovaného vektoru `(x/w,y/w,z/w,1)` jsou pak požadované složky ve 2D (viz příklad).

`T=viewmtx(az,el,phi,tp)` vrací transformační matici středového (perspektivního) promítání s cílovým bodem `tp` (target point) v normalizované krychli (tj. objektiv se zaměří na bod `tp`). Souřadnice bodu jsou určeny trojrozměrným vektorem

`tp=[xc,yc,zc]`, které se mohou měnit v rozsahu $\langle 0, 1\rangle$. Implicitní hodnota cílového bodu pro `phi=0` je `tp=[0,0,0]`.

Algoritmus Hodnoty azimutu a elevace lze zpět získat dotazem na vlastnost `View` aktuálního objektu `axes`

```
[az,el]=get(gca,'View');
```

Hodnotu úhlu `phi` (perspektivní zkreslení) můžeme určit z prvků transformační matice `T`, kterou lze též získat např. dotazem na vlastnost `XForm` aktuálního objektu axes a hodnoty elevace.

```
T=get(gca,'xform');
phi=2*atan(-T(4,3)/sin(el*pi/180)/sqrt(2))*180/pi
```

Souřadnice cílového bodu `tp=[xc,yc,zc]` obdržíme z prvních tří prvků posledního sloupce transformační matice `T` a matice `A` v následujícím tvaru

```
abc=T(1:3,4);
A=[        -cos(az),         -sin(az),        0; ...
    sin(az)*sin(el), -cos(az)*sin(el), -cos(el); ...
   -sin(az)*cos(el),  cos(az)*cos(el), -sin(el)];
tp=A/abc;
```

Příklady Následující příkazy ukazují, jakým způsobem lze pomocí implicitního 3D pohledu určit k bodu `(0.5,0,-3)` výsledný 2D vektor.

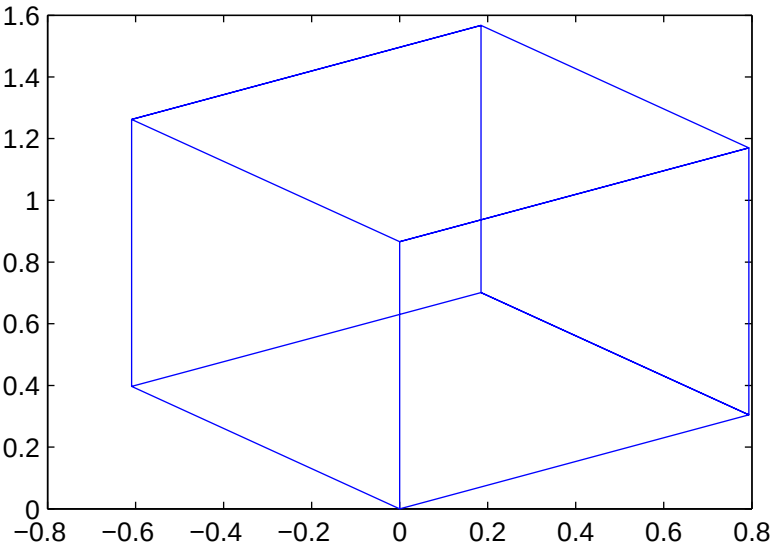
```
A=viewmtx(-37.5,30);

x4d=[0.5 0 -3 1]';
x2d=A*x4d;
x2d=x2d(1:2)

x2d=
0.3967
-2.4459
```

Vektory, které nakreslí hrany jednotkové krychle, jsou

```
x=[0 1 1 0 0 0 1 1 0 0 1 1 1 1 0 0];
y=[0 0 1 1 0 0 0 1 1 0 0 0 1 1 1 1];
z=[0 0 0 0 0 1 1 1 1 1 1 0 0 1 1 0];
```

Transformací bodů vektorů `x`, `y` a `z` na obrazovku získáme vektory `x2` a `y2`, které lze již použít pro vykreslení objektu funkcí `plot`.

```
A=viewmtx(-37.5,30);
[m,n]=size(x);
x4d=[x(:),y(:),z(:),ones(m*n,1)]';
x2d=A*x4d;
x2=zeros(m,n);
y2=zeros(m,n);
x2(:)=x2d(1,:);
y2(:)=x2d(2,:);
plot(x2,y2)
```



Zopakujeme nyní výše uvedený postup, ale použijeme perspektivní transformaci.

```
A=viewmtx(-37.5,30,25);

x4d=[0.5 0 -3 1]';
x2d=A*x4d;
x2d=x2d(1:2)/x2d(4); % Normování

x2d=
0.1777

-1.8858
```

A opět transformujeme vektory krychle na obrazovku a vykreslíme objekt funkcí `plot`.

```
A=viewmtx(-37.5,30,25);
[m,n]=size(x);
```

```
x4d=[x(:),y(:),z(:),ones(m*n,1)]';
x2d=A*x4d;
x2=zeros(m,n);
y2=zeros(m,n);
x2(:)=x2d(1,:)./x2d(4,:);
y2(:)=x2d(2,:)./x2d(4,:);
plot(x2,y2)
```

Viz též `view`

Funkce Vytvoří dialogový box varování.

Syntaxe `h=warndlg(warnstr,dlgname)`

Popis `h=warndlg(warnstr,dlgname)` vytvoří dialogový box varování, který zobrazí řetězec `warnstr` v objektu figure se jménem `dlgname`. Má-li dialogové okno zmizet, musíte stisknout tlačítko **OK**. Pokud dialogový box se jménem `dlgmane` již existuje, přesune se pouze do popředí (nové dialogové okno se nevytvoří). `warndlg` vrací identifikátor `h` objektu figure.

Poznámka Implementováno od verze 4.2.

Příklady `warndlg('WARNING message','WARNING')`

Viz též `dialog`

Funkce Graf ve tvaru *vodopádu*.

Syntaxe `waterfall(...)`

Popis `waterfall` je podobná funkci `mesh`, ale sloupcové čáry nejsou zobrazeny (graf ve tvaru *vodopádu*) a matice jsou transponovány tak, že jsou správně orientovány pro sloupcově orientovanou datovou analýzu.

Poznámka Implementováno od verze 4.1.

Viz též `mesh`

Funkce Mění barvu pozadí objektu figure.

Syntaxe
`whitebg`
`whitebg(fig)`
`whitebg(fig,c)` nebo `whitebg(c)`

Popis `whitebg` přepíná barvu pozadí aktuálního objektu figure mezi černou a bílou a mění ostatní vlastnosti tak, aby graf vypadal dobře. Navíc jsou změněny i implicitní vlastnosti objektu root tak, aby následné grafy v aktuálním objektu figure i nově vytvořených objektech figure používaly novou barvu pozadí.

`whitebg(fig)` mění objekty figure, jejichž identifikátory jsou uloženy ve sloupcovém vektoru `fig`. Navíc dochází ke změně implicitních vlastností i v objektu root (identifikátor=0), aby se změny promítly také do nově vytvořených objektů figure nebo se uplatnily po zadání příkazu `clf reset`.

`whitebg(fig,c)` nebo `whitebg(c)` nastaví implicitní barvu pozadí na `c` a změní ostatní vlastnosti tak ,aby graf vypadal dobře. `c` může být RGB vektor nebo jedno z předdefinovaných jmen. Více informací o zadání barev je uvedeno u popisu `ColorSpec`.

Poznámka Implementováno od verze 4.1.

Viz též `graymon`

Funkce Čte Lotus123 WK1 soubor do matice.

Syntaxe
```
M=wk1read(filename,r,c)
M=wk1read(filename)
M=wk1read(filename,r,c, rng)
```

Popis `M=wk1read(filename,r,c)` načítá soubor s formátem Lotus WK1 do matice `M`. Volitelné argumenty `r` a `c` udávají první řádek a sloupec spreadsheetu (matice), kam se má soubor načíst. `filename` musí být jméno WK1 souboru bez přípony `'.wk1'`.

`M=wk1read(filename)` načítá soubor s formátem Lotus WK1 do matice `M`. Toto je ekvivalentní `r=c=0`, poněvadž horní levá buňka ve spreadsheetu má index $(0, 0)$.

Volitelný čtvrtý argument, `rng`, může být použit ke stanovení požadované části spreadsheetu. Chcete-li určit část spreadsheetu, specifikujte `rng` takto,

`rng=[UpperLeftRow UpperLeftColumn LowerRightRow LowerRightColumn]`.

Pro použití pojmenovaného rozsahu specifikujte `rng` řetězcem reprezentujícím WK1 jméno.

Poznámka Implementováno od verze 4.2.

Příklady
```
M=wk1read('c:/pokus');
```

Viz též `csvread`, `csvwrite`, `wk1write`

Funkce Zapisuje matici do souboru ve formátu Lotus WK1.

Syntaxe
```
wk1write(filename,M,r,c)
wk1write(filename,M)
```

Popis `wk1write(filename,M,r,c)` zapisuje matici `M` do souboru `filename` ve formátu Lotus WK1. Volitelné argumenty `r` a `c` udávají první řádek a sloupec spreadsheetu (souboru), od kterého se má zapisovat.

`wk1write(filename,M)` zapisuje matici `M` do souboru `filename` ve formátu Lotus WK1. Toto je ekvivalentní `r=c=0`, poněvadž horní levá buňka ve spreadsheetu má index $(0, 0)$.

Poznámka Implementováno od verze 4.2.

Příklady
```
wk1write('c:/pokus',M);
```

Viz též `csvread`, `csvwrite`, `wk1read`

Funkce Popis os x, y a z

Syntaxe
```
xlabel(string)
ylabel(string)
zlabel(string)
```

Popis `xlabel(string)` přidá k aktuálnímu grafu pod x-ovou osu text určený řetězcovou proměnnou `string`.

`ylabel(string)` přidá k aktuálnímu grafu vedle y-ové osy text určený řetězcovou proměnnou `string`.

`zlabel(string)` přidá k aktuálnímu 3D grafu vedle z-ové osy text určený řetězcovou proměnnou `string`.

Opětným vyvoláním příkazu `xlabel`, `ylabel` a `zlabel` se nahradí starý text novým textem.

POZOR! Nelze změnit polohu popisu os. Proto chceme-li těmito textovými řetězci pohybovat, je nutné je vytvořit příkazem `gtext` nebo `text`.

Algoritmus Pro 3D grafiku dává MATLAB popisy os dopředu nebo na stranu grafu tak, že nejsou nikdy samotným grafem skryty.

Viz též `title`, `text`

Funkce Transfokace 2-D grafů.

Syntaxe

```
zoom on
zoom off
zoom
zoom out
```

Popis `zoom on` zapíná transfokaci aktuálního objektu figure. Následné kliknutí levým tlačítkem myši způsobí zmenšení úhlu transfokace (zvětšení) a kliknutí pravým tlačítkem myši způsobí zvětšení úhlu transfokace (zmenšení). Při každém kliknutí se meze os zdvojnásobí nebo klesnou na polovinu. Výřez můžete provést kliknutím a tažením myši.

`zoom off` vypíná transfokaci.

`zoom` bez argumentů přepíná status transfokace.

`zoom out` vrací graf do stavu před transfokací.

Poznámka Implementováno od verze 4.2.