Gráficos tridimensionais

Introdução

- Existe uma grande variedade de funções para exibir dados em três dimensões. Podemos utilizar curvas em três dimensões, superfícies em retalhos (patches) ou em malhas (mesh).
  - É possível utilizar cor para representar uma quarta dimensão. É o conteúdo do capítulo 27 do livro.
- Neste capítulo abordaremos os principais conceitos para construção de gráficos tridimensionais.
- A seção 26.12 do livro exibe uma tabela com as funções para gráficos tridimensionais e uma breve descrição de cada uma delas. Veja também *Help –> Matlab Help –> Matlab –> Graphics e 3D-Visualization*
- Acompanhe as transparências deste capítulo observando e executando o arquivo mm26demo.m.

A função $plot3$

- Esta função é um extensão da função $plot$.
- Segue o mesmo formato da função $plot$, mas cada conjunto de dados é uma tripla.
- Usada para traçar uma função tridimensional de uma única variável. Exemplo: $mm2601.m$.
- Todos os recursos básicos dos gráficos bidimensionais existem para os gráficos tridimensionais.

Recursos básicos

- O comando $axis$ se estende para três dimensões.
- Há uma função $zlabel$ para nomear o eixo z.
- Default: $grid\ \ off$ e $box\ \ off$.
- Devemos informar as coordenadas x, y e z à função $text$.
- Podemos usar subgráficos e múltiplas janelas com gráficos tridimensionais.
- A função $hold$ é usada da mesma maneira. É útil para que vários gráficos bidimensionais sejam colocados lado a lado em uma dimensão.
  - Exemplos: $mm2602.m$ e $mm2603.m$.

Funções escalares de duas variáveis

- Muitas vezes queremos visualizar uma função escalar de duas dimensões, uma função do tipo $z = f(x, y)$.
- Um gráfico de z, como função de x e y, é uma superfície em três dimensões.
- Para traçar esta superfície no **MATLAB** é preciso armazenar os valores de z em uma matriz.
- Relembrando, do capítulo sobre interpolação, quando x e y são variáveis independentes e z uma variável dependente, a relação de x e y com z é dada por:

$$z(i,:) = f(x, y(i)) \quad e \quad z(:,j) = f(x(j), y),$$

isto é, a i-ésima linha de z está associada ao i-ésimo elemento de y, e a j-ésima coluna de z associada ao j-ésimo elemento de x.

Função escalar de duas variáveis

- Se $z = f(x, y)$ puder ser expresso de maneira simples, é conveniente usar operações matriciais para calcular todos os valores de z em um único comando.
  - Precisamos criar matrizes com todos os valores de x e y na orientação adequada (*plaid*).
  - O **MATLAB** fornece a função $meshgrid$ para fazer isto.

```
>> x = -3:3; y = 1:3;
>>[X,Y] = meshgrid(x,y)
X =
    -3    -2    -1     0     1     2     3
    -3    -2    -1     0     1     2     3
    -3    -2    -1     0     1     2     3
Y =
     1     1     1     1     1     1     1
     2     2     2     2     2     2     2
     3     3     3     3     3     3     3
>> % calcular agora é simples:
>> Z = (X+Y).^2
```

$meshgrid$ repete x em cada uma das cinco linhas de y e repete y em cada uma das sete colunas de X.

Função escalar de duas variáveis

- Quando uma função não puder ser expressa de forma simples $\Longrightarrow$ comandos de repetição for ou $while$.
- Se for possível calcular os elementos por linha ou por coluna:

```
x = ??? % valores para o eixo x
y = ??? % valores para o eixo y
nx = lenght(x); % número de linhas em Z
ny = lenght(y); % número de colunas em Z
Z = zeros(nx,ny); % iniciando a matriz para melhorar o desempenho
```

Computando Z por linha

```
for r=1:nx
   { comandos preliminares }
   % cálculo da r-ésima linha de Z:
   Z(r,:)= { função de y e x(r) }
end
```

Computando Z por coluna

```
for r=1:ny
   { comandos preliminares }
   % cálculo da c-ésima coluna de Z:
   Z(:,c)= { função de y(c) e x }
end
```

Função escalar de duas variáveis

- Quando é necessário calcular elemento a elemento, há necessidade de dois for aninhados.

```
x = ??? % valores para o eixo x
y = ??? % valores para o eixo y
nx = lenght(x); % número de linhas em Z
ny = lenght(y); % número de colunas em Z
Z = zeros(nx,ny); % iniciando a matriz para melhorar o desempenho
for r=1:nx
    for c=1:ny
    { comandos preliminares }
    % cálculo do elemento (r,c) de Z
    Z(r,c)= { função de y(c) e x(r) }
    end
end
```

A função $mesh$

- Uma superfície em malha (*mesh surface*) é definida pelas coordenadas z dos pontos, sobre uma malha retangular no plano xy.
  - Resultado: gráfico em que pontos adjacentes são ligados por linhas retas, com os pontos correspondentes aos dados nas interseções destas retas.
- Exemplo: $mm2604.m$.
  - Note a relação das cores com a altura da malha. A alteração das cores está ligada com *mapa de cores*, capítulo 27 do livro.
- $mesh$ possui argumentos opcionais para controlar as cores no gráfico.
- O padrão para $mesh$ e para as outras funções para gráficos tridimensionais, a menos de $plot3$, é $grid\ \ on$.
- $mesh(Z)$: constrói o gráfico da matriz Z *versus* seus índices de linha e coluna.

$mesh$ e variantes

- $hidden\ \ on/off$: Controla se as áreas dos retângulas da malha serão opacas ou transparentes. Sem argumentos alterna o estado ativo. Exemplo: $mm2605.m$.

- Formas alternativas de $mesh$:
  - $meshc$: constrói um gráfico com curvas de nível no plano inferior. Exemplo: $mm2606.m$
  - $meshz$: constrói um gráfico com um plano zero. Exemplo: $mm2607.m$.

- $waterfall$: é uma função semelhante à função $mesh$, mas as linhas da malha aparecem apenas na direção paralela ao eixo x. Exemplo: $mm2608.m$.

A função $surf$

- Usada para contruir gráficos que podem ser vistos como superfícies.
- Parecem-se com os gerados por $mesh$, mas os retângulos (*retalhos*) estão preenchidos.
- Exemplo: $mm2609.m$
  - Note que, em contraste com os anteriores, este gráfico possui a gradação de cores nos retalhos, sendo constante em um mesmo retalho, e uma cor sólida (preta) nas linhas.
  - A cor varia ao longo do eixo z.

Tipos de gradação de cores

- Quando se trabalha com superfícies, pensa-se na gradação de cores como um meio de evidenciar suas propriedades.
- O **MATLAB** fornece dois tipos de gradação de cores, lisa ou interpolada, que podem ser selecionadas pela função $shading$.
  - Lisa: As linhas pretas são removidas e cada retalho mantém sua cor única. Exemplo $mm2610.m$.
  - Interpolada: As linhas pretas são removidas, mas cada retalho recebe uma gradação interpolada. A cor de cada retalho é interpolada sobre a sua área com base nos valores das cores atribuídos a cada um de seus vértices. Exemplo $mm2611.m$.

Tipos de gradação de cores

- A impressão de gráficos com gradação interpolada pode se tornar muito lenta devido aos cálculos necessários para gerar as gradações de cores, ou mesmo gerar erros de impressão.

- A gradação também é aplicável aos gráficos de malhas, embora o impacto visual seja menor, uma vez que as cores estão nas linhas.

Superfícies com “buracos”

- Pode ser útil gerar superfícies que possuem buracos, isto é, regiões retangulares, fechadas, sem valores atribuídos para o eixo z.
- Isto é feito atribuindo-se o valor especial NaN aos pontos da região que será o buraco.
  - Como NaN não possui valor, todos estes pontos são ignorados pelas funções gráficas.
- Exemplo: $mm2612.m$

Variantes de $surf$

- $surfc$: Desenha um gráfico de curvas de nível no plano inferior. Exemplo: $mm2613.m$.

- $surfl$: Desenha um gráfico com iluminação. Esta função modifica a cor da superfície para dar aparência de iluminação.

  - Exemplo: $mm2614.m$. A função $colormap$ aplica um conjunto diferente de cores à figura. Está detalhada no capítulo 27 do livro.

- $surfnorm(X, Y, Z)$: Calcula vetores normais à superfície definida por X, Y e Z, traça o gráfico e desenha os vetores normais (normalizados) nos pontos dados. Exemplo: $mm2615.m$.

  - A forma $[Nx, Ny, Nz] = surfnorm(X, Y, Z)]$ calcula os vetores normais, mas não constrói o gráfico.

Gráficos com dados irregulares

- Dados irregulares ou não igulamente espaçados podem ser visualizados por meio das funções:
  - $trimesh$: Exemplo: $mm2517.m$.
  - $trisurf$: Exemplo: $mm2518.m$.
  - $voronoi$: Exemplo: $mm2519.m$.
- O capítulo 18 traz mais detalhes sobre triangulação de Delaunay, relacionada com as duas primeiras funções e diagramas de Voronoi, relacionado com a última função.

Ponto de vista

- Considere uma reta traçada de seus olhos à origem do gráfico. Esta reta define o seu ponto de vista.
- O ponto de vista pode ser descrito em termos de dois ângulos.
  - *Elevação:* ângulo formado entre esta reta e o plano $z = 0$.
  - *Azimute:* ângulo formado entre esta reta e o plano $x = 0$.
- Ponto de vista padrão para gráficos 3D: 30 graus de elevação e -37.5 graus de azimute.
- Ponto de vista padrão para gráficos 2D: 90 graus de elevação e 0 graus de azimute.

Alterando o ponto de vista

- $view(az, el)$ ou $view([az\ el])$: modifica o ponto de vista para todos os gráficos bidimensionais e tridimensionais para o azimute especificado em az e a elevação especificada em el.
  - Exemplo: $mm2619.m$.
- $view(2)$: atribui azimute e elevação padrão de gráficos bidimensionais.
- $view(3)$: atribui azimute e elevação padrão de gráficos tridimensionais.
- $rotate3d\ \ on/off$: é usado para mudar o ponto de vista interativamente com o mouse. Sem argumentos alterna o estado ativo.
  - Este recurso pode ser encontrado na janela *Figure* tanto como botão, como item do menu **Tools**

Câmera

- O **MATLAB** oferece recursos semelhantes aos de uma câmera para permitir um controle mais refinado da cena tridimensional.
  - Há dois sistemas de coordenadas: um na câmera e outro no objeto. As funções de câmera do **MATLAB** controlam e manipulam estes sistemas e as lentes da câmera.
- A maioria das funções de câmera está acessível através do menu **Tools** da janela *Figure*, ou da barra de ferramentas *Câmera* (ativada no menu **View** da janela *Figure*.
  - Devido à facilidade de uso das ferramentas interativas, contrastando com a complexidade na utilização de descrição das funções de câmera, tais funções não serão tratadas aqui.
  - O manual *Using* **MATLAB** *Graphics*, que acompanha o **MATLAB**, contém uma descrição detalhada destas funções e de seu uso. A tabela da seção 26.12 descreve brevemente estas funções.

Gráficos de curvas de nível

- Este gráficos exibem curvas onde a altura, ou elevação, é constante.
- $countour$ e $contour3$: exibem gráficos de curvas de nível em duas dimensões e três dimensões, respectivamente.
  - Exemplos: $mm2620.m$ ($countour$) e $mm2621.m$ ($countour3$)
- $pcolor$: relaciona a altura a um conjunto de cores e representa as mesmas informações do gráfico de curvas de vível por meio de cores. Exemplo: $mm2622.m$
- A função $countourf$: gráfico de faixas de nível. Combinação da função $countour$ com a função $pcolor$, gráfico de curvas de nível preenchidas. Exemplo: $mm2623.m$.
- $clabel$: adiciona legendas às curvas de nível. Exemplo: $mm2624.m$ e $mm2625.m$. Veja ajuda *on line* para maiores detalhes sobre o uso.

Gráficos especializados

- $ribbon(x, y, width)$: Desenha fitas bidimensionais em três dimensões. É o mesmo que $plot(x, y)$, mas as colunas de y são desenhadas como fitas separadas em três dimensões.
  - $width$: especifica a largura das fitas. Se omitido o padrão, 0.75, é assumido.
  - $ribbon(y)$: assume $x = 1 : size(y, 1)$.
  - Exemplo: $mm2626.m$.

- $quiver(x, y, dx, dy)$: gráfico de setas bidimensionais. Desenha vetores de direção, ou de velocidade, (dx, dy) nos pontos (x, y). Exemplo: $mm2627.m$.

- $quiver3(x, y, z, Nx, Ny, Nz)$: gráfico de setas tridimensionais. Exibe os vetores (Nx, Ny, Nz) nos pontos (x, y, z). Exemplo: $mm2628.m$.

Gráficos especializados

- $fill3(x, y, z, c)$: Versão tridimensional de $fill$. Desenha polígonos preenchidos no espaço tridimensional.
  - Utliza os vetores x, y e z como vértices do polígono e c define a cor do preenchimento.
  - Exemplo: $mm2629.m$

- $stem3(x, y, z, c, 'filled')$: Equivalente tridimensional da função $stem$. Cria gráficos de seqüências discretas no espaço tridimensional.
  - Faz os gráficos dos pontos em (x, y, z) com linhas com origem no plano xy.
  - O argumento opcional c define estilo de marcador e a cor.
  - o argumento opcional $'filled'$ define o marcador como preenchido.
  - Exemplo: $mm2630.m$

Visualização de volume

- O **MATLAB** fornece funções para visualização de volumes e vetores.
- Estas funções constroem gráficos de quantidades escalares e vetoriais no espaço tridimensional.
- Seus argumentos de entrada são vetores tridimensionais, um para cada eixo x, y e z (constroem volumes e não superfícies).
  - Os elementos em cada vetor tridimensional definem as coordenadas dos pontos ou os dados nas coordenadas.
- Para as funções escalares são necessários 4 vetores: um para cada eixo e um para os dados escalares nos pontos. São denominados: X, Y, Z, V.
- Para as funções vetoriais, são necessários 6 vetores. Um para cada um dos três eixos de coordenadas e uma para cada componente axial do vetor nos pontos. São denominados:X, Y, Z, U, V e W.

Visualização de volumes

- Para um bom uso destas funções é necessário conhecimento da terminologia de volumes e vetores.
- Concentrar-nos-emos na exemplificação da estrutura dos vetores de dados e o uso das funções que trabalham com volumes e vetores.
- Existe uma GUI, denominada $volvec$ para a aprendizagem interativa da maioria das funções de visualização de volume do **MATLAB**.

Visualização de volumes

- Vamos construir uma função escalar sobre um volume ($mm2631.m$).
  - Precisamos definir os eixos das coordenadas:

```
>> x=linspace(-3,3,13);  % valores para a coordenada x
>> y=1:20; % valores para a coordenada y
>> z=-5:5; % valores para a coordenada z
>> [X,Y,Z]=meshgrid(x,y,z); % meshgrid funciona aqui também!
>> disp(size(X)), disp(size(Y)), disp(size(Z))
 20    13    11          20    13    11          20    13    11
```

  - Note que X,Y e Z são vetores tridimensionais que definem a malha de pontos.
  - X contém x repetido em length(y) linhas e length(z) páginas;
  - Y contém y^t em length(x) colunas e length(z) páginas;
  - Z: a i-ésima página contém z(i) repetido em length(y) linhas e length(x) colunas.

Visualização de volumes

- Vamos agora definir uma função escalar $v = f(x, y, z)$ e utilizar a função $slice$ para visualizá-la em fatias.

```
>> V = sqrt(X.^2+cos(Y).^2+Z.^2);
>> slice(X,Y,Z,V,[0 3],[5 15],[-3 5])
>> xlabel('X-axis'),  ylabel('Y-axis'), zlabel('Z-axis')
>> title('Figure 26.31: Slice Plot Through a Volume')
```

  - Os três últimos argumentos da função $slice$ definiram os planos que exibiram as fatias. A cor do gráfico está associada aos valores de V nas fatias.

- As fatias da função $slice$ não precisam ser planos. Exemplo: $mm2632.m$.
  - Neste exemplo xs, ys e zs definem a superfície que atravessa o plano.

Visualização de volumes

- Resumindo a função $slice$:
  - $slice(X, Y, Z, V, Sx, Sy, Sz)$: Desenha fatias ao longo das direções x, y e z nos pontos dos vetores Sx, Sy e Sz. Os vetores X, Y e Z definem as coordenadas para V. As cores de cada ponto são determinadas por uma interpolação em 3D no volume V.

- $contourslice(X, Y, Z, V, Sx, Sy, Sz)$: Acrescenta curvas de nível a planos selecionados.
  - Exemplo: $mm2633.m$. Neste exemplo são acrescentadas curvas de nível aos planos $x = 3$, $y = 5$, $y = 15$. Usando recursos de *handle graphics*, define-se o preto como a cor para as curvas de nível e especifica-se sua largura em 1.5 pontos.

Visualização de volumes

- Podemos traçar superfícies nas quais os dados de V possuem um valor específico. Para isso usamos:
  - $FV = isosurface(X, Y, Z, V, ISOVALUE)$,: calcula a geometria da superfície isométrica para os dados em V, no valor de superfície ISOVALUE. Arrays (X,Y,Z) especificam os pontos nos quais os dados em V são dados. A “struct” FV contém as faces e vértices da superfície e pode ser passada diretamente ao comando PATCH..
    - retorna os vértices de triângulos, de maneira similar à da triangulação de Delaunay, de modo que o resultado tem a forma requerida pela função $patch$.
  - $patch$: cria gráficos de triângulos.
- $shrinkfaces(P, sf)$: Reduz o tamanho das faces de P (*patch*) em $(sf * 100)\%$.
- O arquivo $mm2634.m$ exemplifica estas funções.

Visualização de volumes

- O **MATLAB** fornece duas funções que permitem um ajuste quando os dados possuem pontos demais para uma exibição satisfatória:
  - $reducevolume$: Elimina dados antes que a isosuperfície seja formada.
  - $reducepatch$: Elimina retalhos, enquanto minimiza a distorção na superfície subjacente.
  - O exemplo $mm2635.m$ exibe o gráfico original, o gráfico com $reducevolume$, o gráfico com $reducepatch$ e o gráfico usando ambas as funções.

Visualização de volumes

- $smooth3(V,'filter')$: Função que suaviza os dados de entrada. O parâmetro $'filter'$ pode ser *gaussian* ou *box* e determina o "*convolution kernel*". Se 'filter' for omitido é assumido o padrão (*Box*). Veja help *on line* para mais detalhes.

- $isocaps$: cria as faces nas superfícies externas do bloco.

- $isonormals$: modifica as propriedades dos retalhos desenhados de modo que a iluminação funcione corretamente.

- Exemplo: $mm2636.m$.

Automatização da entrada de dados

- As funções $ezcountour$, $ezcountour3$, $ezmesh$, $ezmeshc$, $ezplot3$ são usadas para evitar especificação efetiva dos pontos de um gráfico tridimensional.
  - Constroem gráficos como suas equivalentes sem o prefixo ez, usando como argumentos de entrada string, ou expressão simbólica representando uma função matemática de duas variáveis.
  - Opcionalmente podem usar como argumentos de entrada os intervalos dos eixos dos gráficos (domínio padrão:[-2*pi,2*pi,-2*pi,2*pi]).
  - As funções calculam os dados e posteriormente geram os gráficos.
- Exemplo: $mm2637.m$.