MATLABの使い方

第10回：3次元プロット

まとめページ：
http://www.nuee.nagoya-u.ac.jp/labs/plaene/koukai/purakaku85/tsukaikata/

☆1

3次元プロットの流れ

プロットするデータ
- 離散点
- 数列
- x,y平面に分布する数値データ(2自由度)
- x,y平面に分布するベクトルデータ(4自由度)
- x,y,z空間に分布する数値データ(3自由度)
- x,y,z空間に分布するベクトルデータ(6自由度)

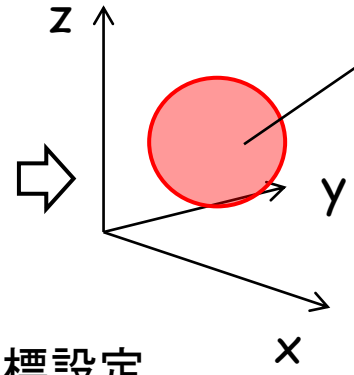
データに適した
プロット方法を選択

- 基本はデカルト座標(x,y,z)上にプロット

プロットするオブジェクト
- 点
- 線
- 平面、曲面
- 矢印、円錐、流線

オブジェクト
設定

Axesのプロパティ
- 視点の角度
- グリッド
- ボックス　etc…



座標設定

オブジェクトプロパティ
- マーカー種類、大きさ、色、塗りつぶし
- 線種、線色
- 格子点色、補間方法、透明度、光源位置 etc…

2次元プロットに比べて設定項目が多い
⇒自分好みのカッコいい図が作れる

☆1

3次元プロット関数の種類

①点を3次元空間上にプロット　・・・scatter3, stem3

②線　〃　・・・plot3, ezplot3

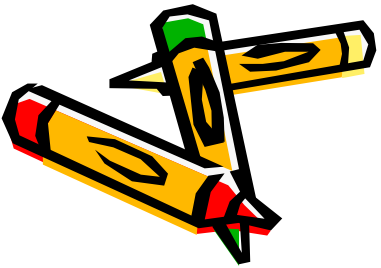
③面　〃　・・・patch, fill3, surf

④2変数スカラーデータ(f(x,y))を表面として描画
・・・mesh, meshc, meshz, surf, surfc, surface, waterfall, ribbon, contour3

⑤スカラーボリュームデータ(f(x,y,z))を可視化
・・・ slice, contourslice, isosurface, endcaps

⑥ベクトルボリュームデータ($v_x(x,y,z)$,$v_y(x,y,z)$,$v_z(x,y,z)$)を可視化
・・・quiver3, coneplot, streamline, streamparticles, streamribbon, streamtube

⑦その他特殊なプロット　・・・bar3, bar3h, pie3, comet3



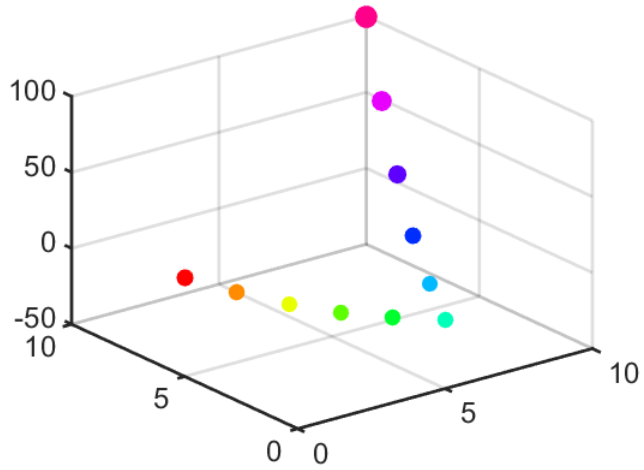
扱うデータの種類によって、関数を使い分けよう！

http://jp.mathworks.com/help/matlab/2-and-3d-plots.html

☆1

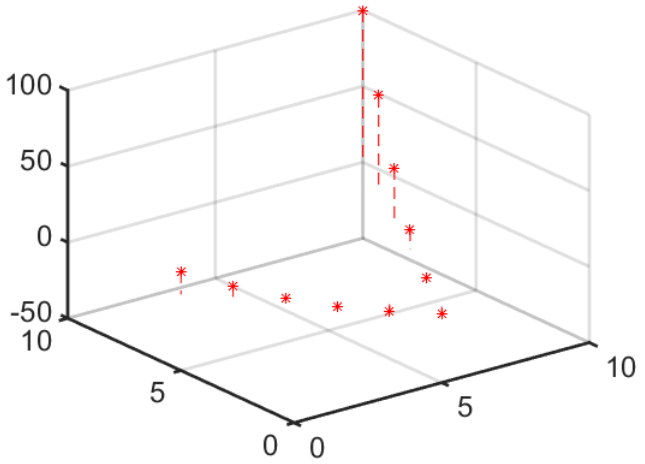
①離散点のプロット

```
figure;
x=[0:10]; y=[5:-1:0,2:2:10];
 z=x+y.^2-10;
s=abs(z)+100; % サイズ
c=hsv(length(z)); % 色
scatter3(x,y,z,s,c,'fill');
```



scatter3では異なる色や
サイズのマーカーを一度に
プロットできる

```
figure;
stem3(x,y,z,'--*r');
```



・stem3ではxy平面から
伸びる線も同時にプロット

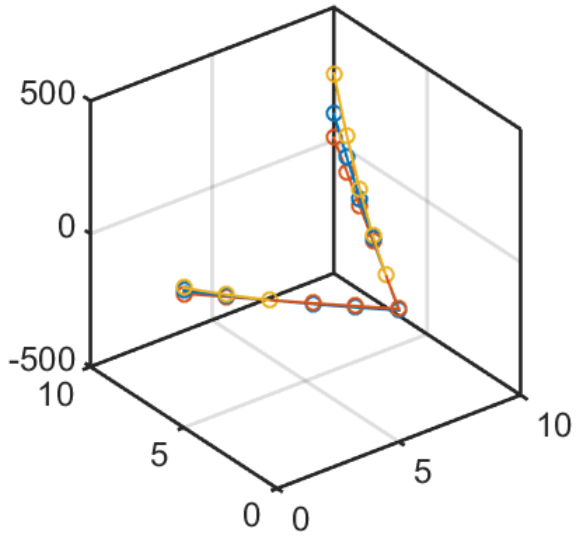
scatter3: 3次元散布図

stem3: 3次元離散データ列のプロット

☆1

②線のプロット

```
figure;
plot3(x,y,[z;sqrt(z);z.^1.2],'-o');
grid on;
box on;
daspect([1,1,100]);
```



plot3はplot関数の3次元拡張版

```
>> help daspect
daspect - Axes データの縦横比の設定とクエリ

    この MATLAB 関数 自身では、現在の Axes のデータの縦横比を返します。

    daspect
    daspect([aspect_ratio])
    daspect('mode')
    daspect('auto')
    daspect('manual')
    daspect(axes_handle,...)
```

・daspect([1,1,1])は axis equalと等価

plot3: 線形3次元プロット

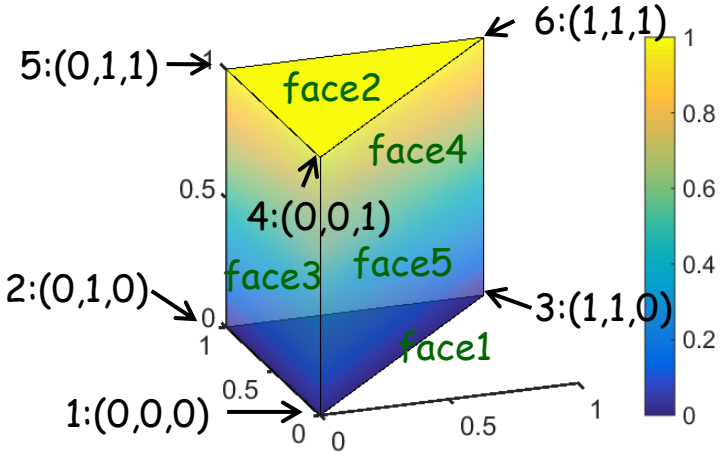
box: Axesの境界

daspect: Axesの縦横比設定

☆1

③面（多角形）のプロット1

```
xyz=[0,0,0;0,1,0;1,1,0];
xyz=[xyz;xyz+repmat([0,0,1],3,1)];
c=[0;0;0;1;1;1]; % 各点の色
face1=[1:3]; % 底面
face2=[4:6]; % 上面
face3=[1,2,5,4]; % 左側面
face4=face3+1; % 背面
face5=[1,3,6,4]; % 右側面
```




```
figure;          ↓x座標      ↓y座標      ↓z座標      ↓色
ob1=patch(xyz(face1,1),xyz(face1,2),xyz(face1,3),c(face1));  ←face1
ob2=patch(xyz(face2,1),xyz(face2,2),xyz(face2,3),c(face2));  ←face2
ob3=patch(xyz(face3,1),xyz(face3,2),xyz(face3,3),c(face3));  ←face3
ob4=patch(xyz(face4,1),xyz(face4,2),xyz(face4,3),c(face4));  ←face4
ob5=patch(xyz(face5,1),xyz(face5,2),xyz(face5,3),c(face5));  ←face5
alpha([ob3,ob4,ob5],0.5); % 透明度の指定
view([-20,20]); axis equal; % 視点、アスペクト比
colorbar('vert'); % カラーバー
```

patch関数を使うことで各点に色の情報を持った多角形の面を配置できる

patch: 多角形の作成

alpha: 透明度の設定

view: 視点の設定

☆1

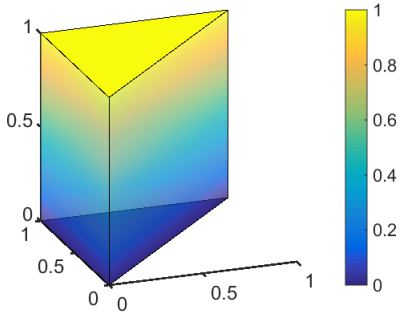
③面（多角形）のプロット2

・patch関数では複数の面を同時に貼ることも可能

```
figure;
ob12=patch('Vertices',xyz,'Faces',[face1;face2],...
 'FaceVertexCData',c,'FaceColor','interp');
ob345=patch('Vertices',xyz,'Faces',[face3;face4;face5],...
    'FaceVertexCData',c,'FaceColor','interp');
alpha(ob345,0.5);
view([-20,20]); axis equal;
colorbar('vert');
```

頂点のxyz座標　頂点の組み合わせ

色　←面の色を補間



前頁と同じ出力結果
（ただしオブジェクト数は5⇒2）

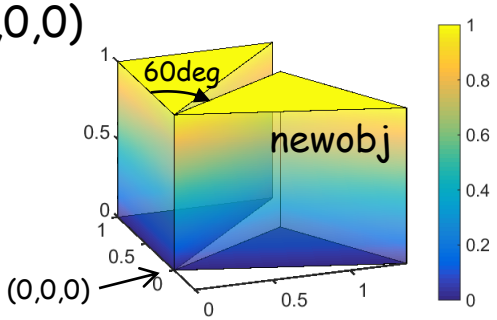
複数の面を一つのオブジェクトとして取り扱うのでpatchを繰り返すより軽快

追加して

```
newobj=copyobj([ob12,ob345],gca);
rotate(newobj,[0,0,1],-60,[0,0,0]);
```

・同じ形状を回転させながら追加して貼るときはcopyobj+rotateが便利

z軸[0,0,1]に対して(0,0,0)を中心に-60度回転



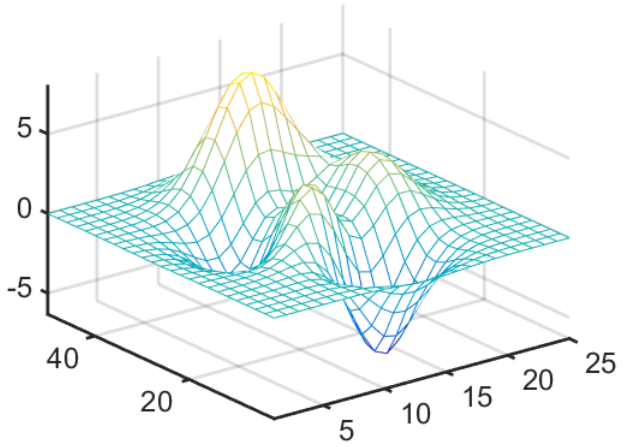
copyobj: オブジェクトのコピー

rotate: オブジェクトを回転

☆1

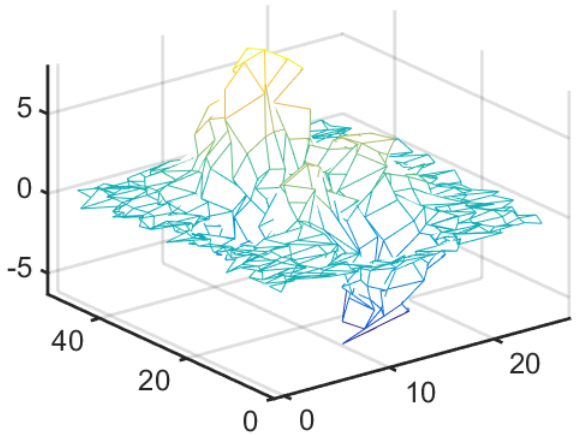
④2変数スカラーデータ1

```
x=1:25;
y=x*2;
z=peaks(25);
figure;
mesh(x,y,z);
axis tight;

[xm,ym]=meshgrid(x,y);
figure;
mesh(xm+randn(25),ym+randn(25),z);
axis tight;
```



・x,yがどちらも1次元配列の場合、
長方形のメッシュでのプロットのみ可能

・x,yにzと同一サイズの2次元配列を
入力することで、長方形以外の四角形
メッシュを使用可能



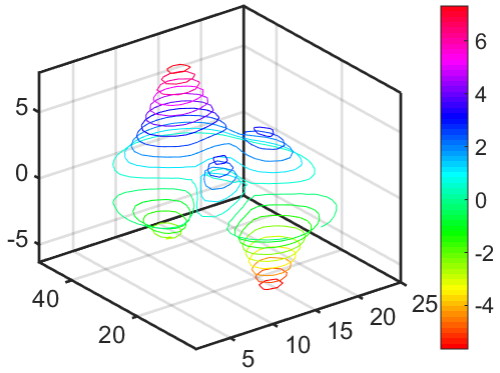
mesh: メッシュプロット

meshgrid: 四角形グリッド

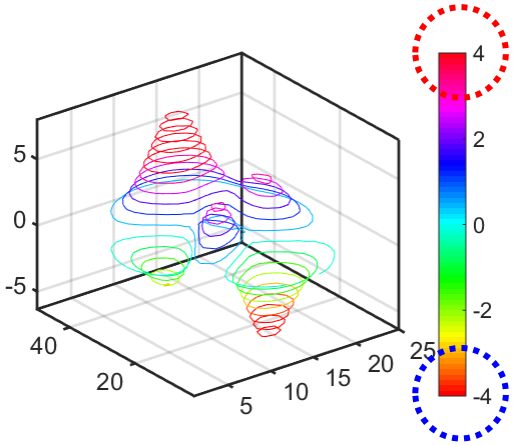
☆1

④2変数スカラーデータ2

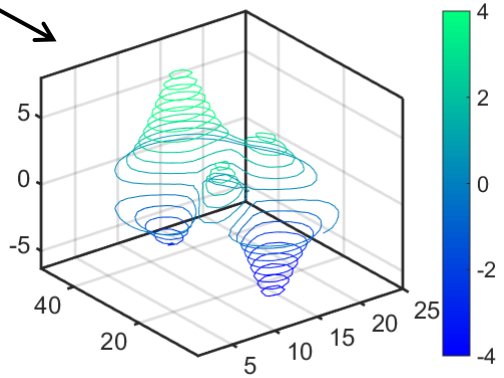
```
figure;
contour3(x,y,z,20);
colorbar('vert');
colormap('hsv');

caxis([-4,4]);

colormap('winter');
```



・caxisにより色の上限下限を設定



・colormapにより色軸を設定



contour3はcontour関数の3次元拡張版

contour3: 3次元等高線図

colormap: カラーマップの設定

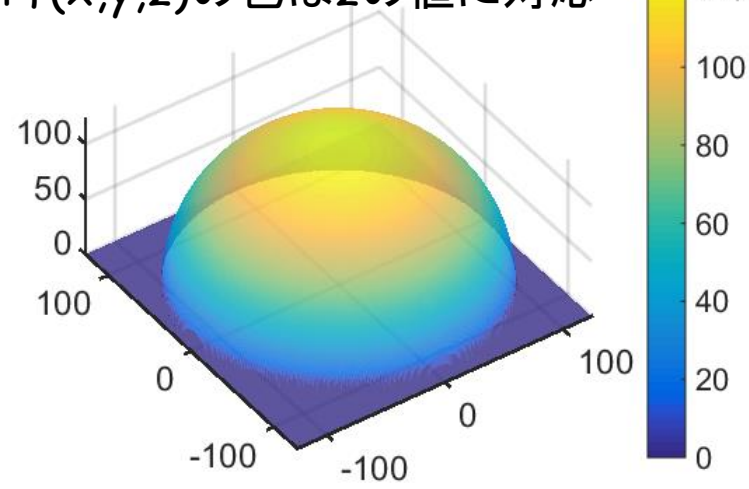
caxis: 色軸のスケーリング

☆1

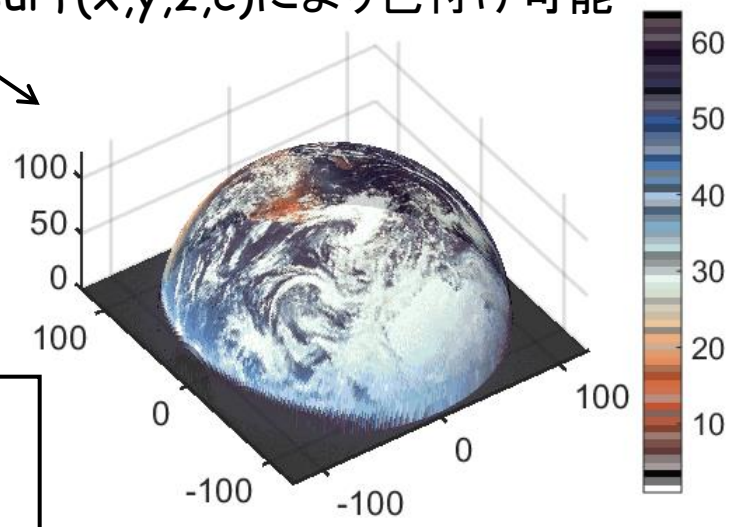
④2変数スカラーデータ3

```
load earth;
c=flipud(X);
x=(0:size(c,2)-1)-size(c,2)/2;
y=(0:size(c,1)-1)-size(c,1)/2;
[xm,ym]=meshgrid(x,y);
zm=sqrt(123^2-xm.^2-ym.^2);
zm(imag(zm)~=0)=0;
figure;
%surf(xm,ym,zm);
surf(xm,ym,zm,c); colormap(map);
shading flat;
axis equal; alpha(0.8);
view([-35,40]);
```

・surf(x,y,z)の色はzの値に対応



・surf(x,y,z,c)により色付け可能



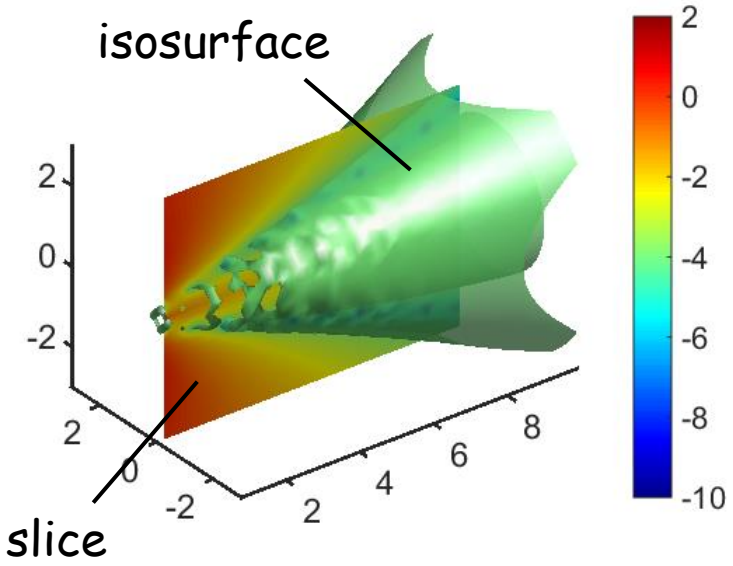
surfはmesh関数の塗りつぶし版

surf: 表面プロット

shading: シェーディングの設定

☆1

⑤スカラーボリュームデータ



平面・曲面と透明度の
組み合わせにより可視化

isosurface: 等値面データの抽出

isonormals: 頂点の法線を計算

slice: スライスプロット

camlight: Lightオブジェクト作成

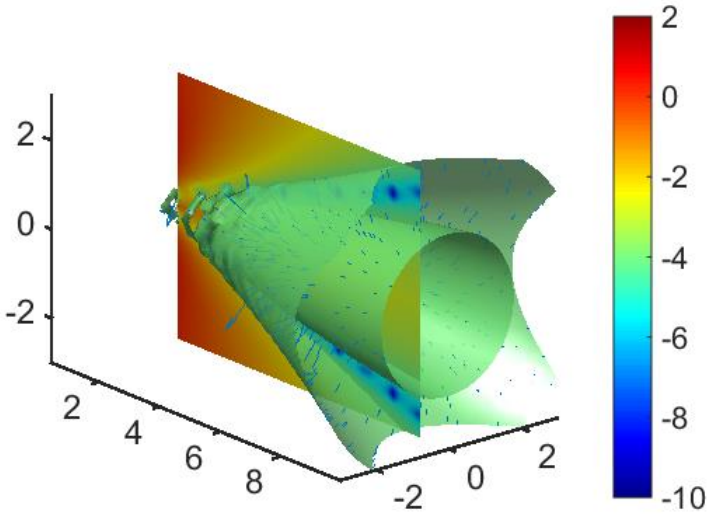
lighting: ライティング方法選択

```
[x,y,z,v]=flow;
cran=[-10,2];
map=jet(diff(cran)*10+1);
clin=linspace(cran(1),cran(2),size(map,1));
val=-4;

figure;
p=patch(isosurface(x,y,z,v,val));
isonormals(x,y,z,v,p);   ←等値面をよりスムーズに表現
set(p,'FaceColor',map(clin==val,:),...
    'EdgeColor','none','FaceAlpha',0.8);
daspect([1,1,1]);
view(3); axis tight;
camlight;                          メッシュの黒線を消去
lighting phong;

hold on;
[sx,sz]=meshgrid(1:0.1:9,-3:0.1:3);
ob=slice(x,y,z,v,sx,0*ones(size(sz)),sz);
set(ob,'FaceColor','interp','EdgeColor','none');
colormap(map);
colorbar('vert');
caxis(cran);
```

☆1

⑥ベクトルボリュームデータ

```
%p=patch(isosurface(x,y,z,v,val));
vv=isonormals(x,y,z,v,p);

quiver3(p.Vertices(1:10:end,1),...
    p.Vertices(1:10:end,2),...
    p.Vertices(1:10:end,3),...
    vv(1:10:end,1),vv(1:10:end,2),...
    vv(1:10:end,3),2);
view([50,20]);
```



quiver3 - 3 次元の矢印プロットまたは速度プロット

この MATLAB 関数 は、成分 (u,v,w) で決定される方向をもつベクトルを (x,y,z) で決定される点にプロットします。行列 x、y、z、u、v、および w は同じサイズで、対応する位置とベクトルの成分を含んでいなければなりません。

```
quiver3(x,y,z,u,v,w)
quiver3(z,u,v,w)
quiver3(...,scale)
quiver3(...,LineSpec)
quiver3(...,LineSpec,'filled')
quiver3(...,'PropertyName',PropertyValue,...)
quiver3(axes_handle,...)
h = quiver3(...)
```

矢印によりベクトルを表現

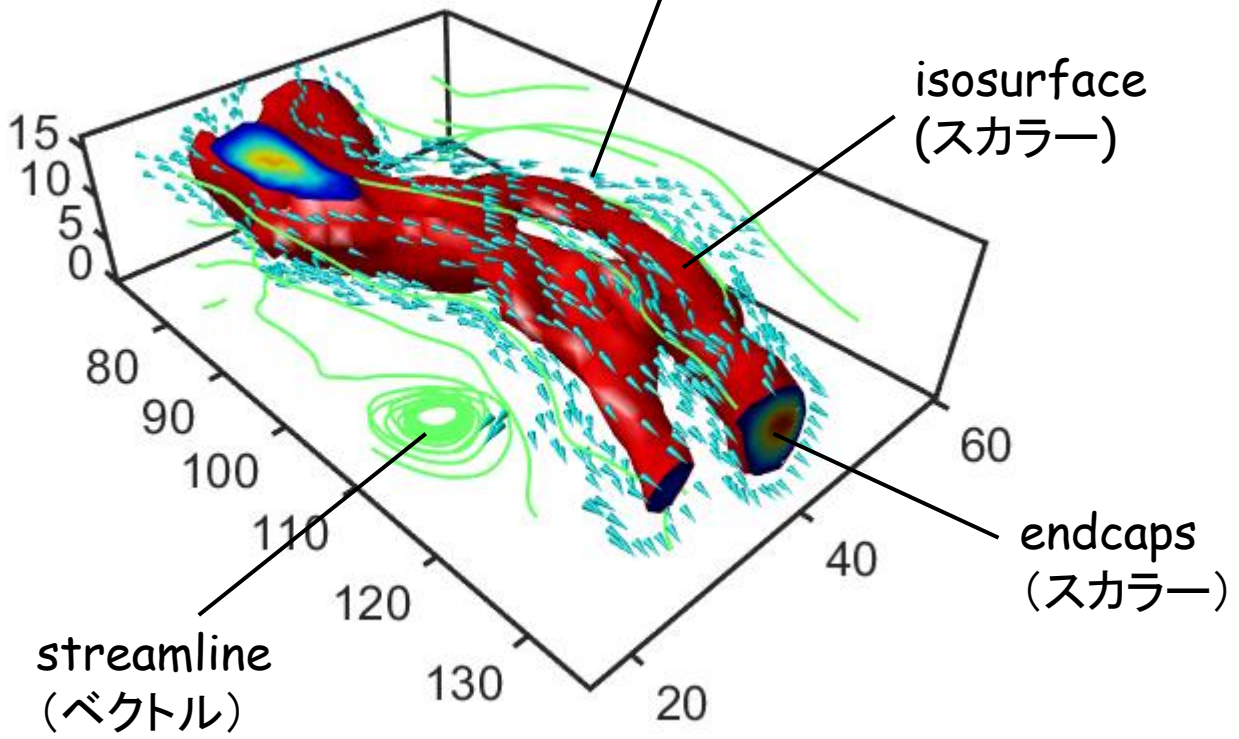
quiver3: 3次元の矢印プロット

☆1

⑤＋⑥の複合プロット

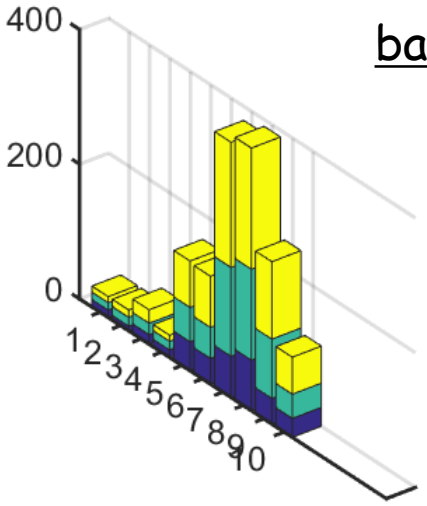
・可視化のデモプログラム

volvec; % ボリュームビジュアライゼーション

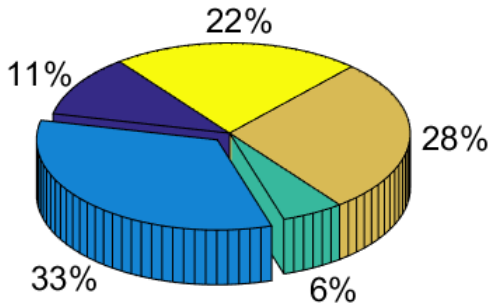


⑦特殊なプロット

bar3



pie3



comet3