ЛЕКЦІЯ 7
ВІЗУАЛІЗАЦІЯ, АНАЛІЗ ТА ОБРОБКА ГРАФІЧНИХ ДАНИХ

Через наочність та інформативність велика увага приділяється *візуалізації* графічних даних, а також їх аналізу та обробці. Цілком природно, що постановка й опис розв'язуваної задачі та результати розв'язання повинні бути зрозумілими не лише тим, хто розв'язує задачі, а й тим, хто у подальшому використовуватиме результати їх розв'язання. Велику роль у візуалізації розв'язання задач відіграє графічне подання результатів, причому як остаточних, так і проміжних (графіки різних видів, зображення, анімація тощо).

Як відомо, користувач матричної системи комп'ютерної математики MatLab має ряд можливостей для побудови графіків функцій та візуалізації даних:

- високорівневі графічні функції (plot, surf, mesh тощо);
- інтерактивне середовище Plotting Tools, компонента якої Plot Editor (редактор графіків) також дозволяє змінювати властивості елементів графіку;
- спеціалізовані функції та засоби ToolBox для графічного подання характеристик досліджуваних об'єктів і результатів.

Цього набору виявляється недостатньо, якщо необхідно вивести графічні результати в готовому вигляді, який не потребує їх подальшої правки в редакторі графіків, або ж у процесі своєї роботи керувати елементами графіків: видаляти поверхні, змінювати колір та товщину ліній, додавати стрілки та пояснюючі надписи тощо. В цих випадках використання так званої *дескрипторної графіки* та відповідних низькорівневих графічних функцій дає можливість повного контролю над елементами графіків.

Низькорівнева ***дескрипторна (описова) графіка Handle Graphics*** – це об'єктно-орієнтована графічна система, яка підтримує компоненти, необхідні для створення комп'ютерних графіків та створена за допомогою графічних

об’єктів, що мають властивості наслідування та ієрархію (рис.7.1). *Об’єктами* підсистеми дескрипторної графіки є базові графічні елементи, що використовуються для візуалізації даних. Кожному графічному об’єкту надається унікальний ідентифікатор (описувач) – ***дескриптор***. Використовуючи дескриптор, можна керувати *властивостями* графічного об’єкта. Тобто кожному графічному об’єкту у складі рисунка відповідає деякий дескриптор, на який можна посилатися при звертанні до цього об’єкту.

В системі MatLab використовується наступна ієрархія графічних об’єктів, пов’язаних з візуалізацією даних (рис.7.1).

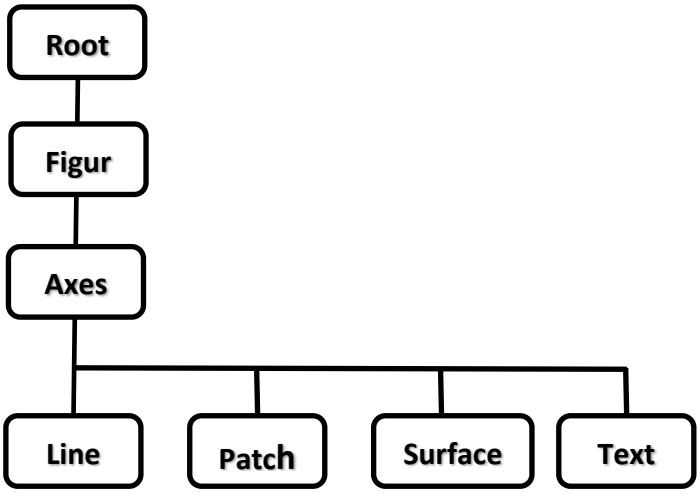


Рисунок 7.1 – Узагальнена ієрархія графічних об’єктів MatLab

Наприклад, задамося питанням, що відбувається при виконанні послідовності команд: `x = 0:0.2:10; y = cos(x); plot(x, y)`. Звичайно, будується графік функції, та при розгляді дескрипторної графіки потрібні інший погляд на цей процес і відповідна термінологія. Якщо не було відкрито графічних вікон, то високорівнева графічна функція plot створить ряд *графічних об’єктів*: спочатку графічне вікно (Figur), потім осі (Axes) і, зрештою, лінію (Line). Всі графічні об’єкти MatLab вишикувані у певну ієрархію, осі є *нащадком* графічного вікна и не можуть існувати самі по собі. В свою чергу, графічне вікно – *предок* для осей. Аналогічно з лінією. Вона є *нащадком* осей, а осі – її *предком*. Одночасно можуть існувати кілька графічних вікон, кожне

з яких може містити й кілька нащадків (осей), а кожна вісь – по кілька нащадків (ліній, поверхонь та інших графічних об’єктів) (рис.7.1).

7.1 Розширена техніка візуалізації

Засоби дескрипторної графіки системи MatLab дозволяють реалізовувати досить ефективну візуалізацію складних об’єктів (перш за все тривимірних) та багатьох фізичних явищ, наприклад таких як струмені газу та рідини, електричні розряди тощо. Розглянемо приклади деяких характерних застосувань дескрипторної графіки MatLab.

Приклад 7.1. Задання Path-об’єктів. Об’єкт Path може використовуватись для побудови зафарбованих багатокутників (полігонів). Для цього служить команда path. Розглянемо побудову зафарбованого десятикутника з інтерполяцією зафарбовування, що дає плавні переходи кольору (рис.7.2):

```
t=0:pi/5:2*pi;
a=t(1:length(t)-1);
patch(sin(a),cos(a),1:length(a),'FaceColor','interp');
colormap jet;
axis equal;
```

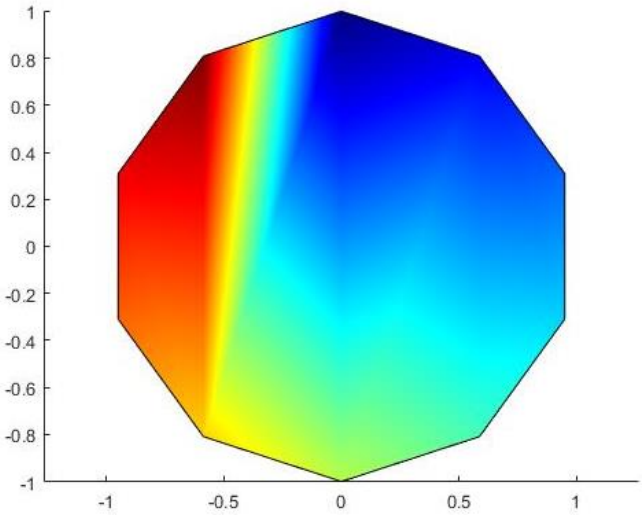


Рисунок 7.2 – Зафарбований багатокутник, побудований командою path

Функція встановлює значення поточної палітри colormap, а axis встановлює масштабування осей та їх виведення.

Приклад 7.2. Побудова зрізу черепної коробки людини. Як відомо, в MatLab за допомогою масивів можуть бути задані досить складні об'єкти. За приклад розглянемо файл mri.mat (від Magnetic Resonance Imaging), який містить масив графічних даних черепної коробки людини разом з її «вмістом», отриманий за допомогою томографу. Завантажимо цей масив та виділимо один шар з розрізаної горизонтальними площинами черепної коробки (рис.7.3):

```
load mri
D=squeeze(D);
image_num=8;
image(D(:,:,image_num));
axis image
colormap(map);
```

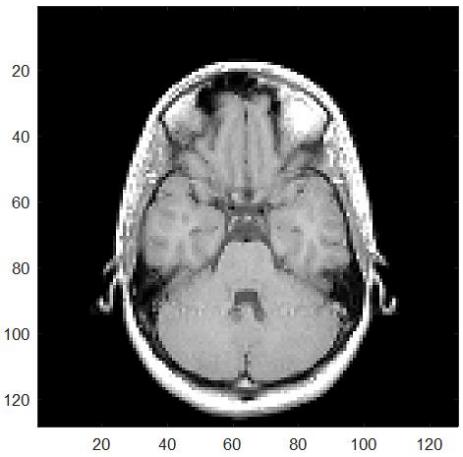


Рисунок 7.3 – Зріз черепної коробки людини

Функція load зчитує змінні з файлу mri.mat, squeeze видаляє всі одиничні розмірності масиву, image створює зображення, отримане з певної частини масиву.

Далі побудуємо кілька зрізів черепної коробки. Розглянемо два способи візуалізації. Перший спосіб. Будуються та виводяться всі 27 зрізів в окремі графічні вікна (на рис.7.4 наведені лише 1-й, 12-й, 19-й та 27-й зрізи, відповідно):

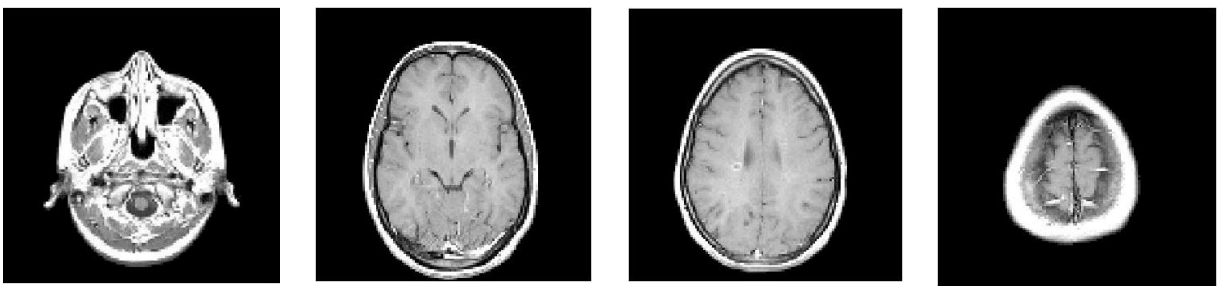


Рисунок 7.4 – Зрізи черепної коробки людини

```
load mri
D=squeeze(D);
[N M K]=size(D);
for i=1:K
    figure
    image(D(:,:,i));
    axis image
    colormap(map);
    set(gca,'xtick',[],'ytick',[]);
    set(gca,'XColor','w','YColor','w');
end
```

Другий спосіб. Будуються та виводяться лише певні зрізи (для відповідності до першого способу, 1-й, 12-й, 19-й та 27-й) (рис.7.5):

```
phandles=contourslice(D,[],[],[1,12,19,28],8);
view(3);
axis tight
daspect([1,1,0.4]);
set(phandles,'LineWidth',2);
box on;
colormap(map);
```

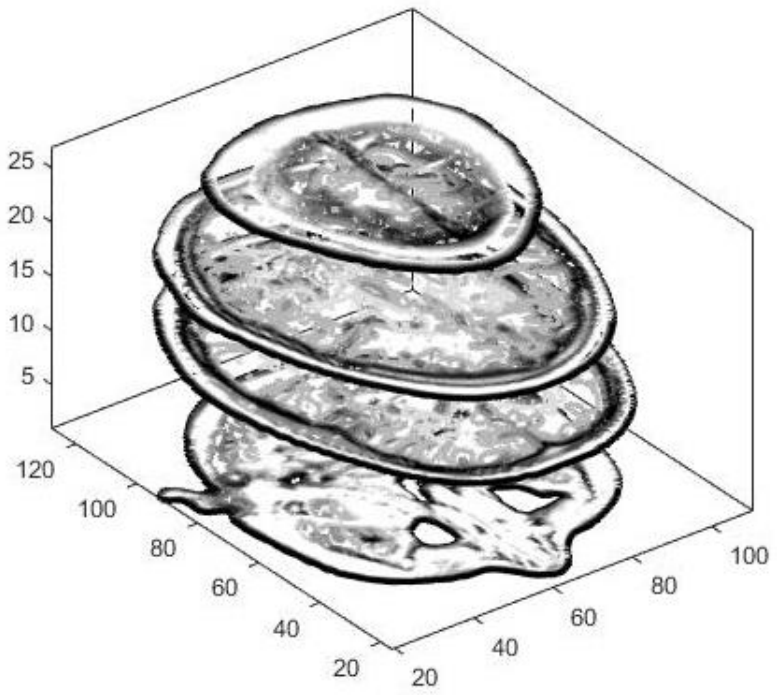


Рисунок 7.5 – Чотири зрізи черепної коробки людини

Функція contourslice рисує контури у площинах об'ємних зрізів, view керує положенням точки перегляду (3 – для тривимірної графіки), daspect повертає значення властивості, що визначає коефіцієнти масштабування даних по осях, set – встановлює властивості графічного вікна, box on – рисує контур паралелепіпеду, в якому розміщується тривимірний об'єкт.

Приклад 7.3. Розширена візуалізація тривимірних об'єктів. Розглянуті вище прийоми можна розповсюдити на візуалізацію тривимірних об'єктів. Наведений нижче фрагмент програми забезпечує згладжування даних масиву черепної коробки командою smooth3 та побудову реалістичного рисунку черепної коробки з видаленим зверху її фрагментом (рис.7.6):

```
load mri
D=squeeze(D);
Ds=smooth3(D);
hiso=patch(isosurface(Ds,5),'FaceColor',[1,0.75,0.65],'EdgeColor','none');
hcap=patch(isocaps(D,5),'FaceColor', 'interp','EdgeColor','none');
colormap(gray);
view(45,30);
axis tight
daspect([1,2,0.4]);
lightangle(45,30);
set(gcf,'Renderer','zbuffer');
lighting phong;
isonormals(Ds,hiso);
set(hcap,'AmbientStrength',0.6);
set(hiso,'SpecularColorReflectance',0,'SpecularExponent',50);
```

Як бачимо, засоби дескрипторної графіки MatLab можуть ефективно застосовуватися для дослідження внутрішньої будови органів.

Приклад 7.4. Виділення частини об'єму. За допомогою функцій subvolume й isonormals можна виділити частину об'єму та наочно її подати.

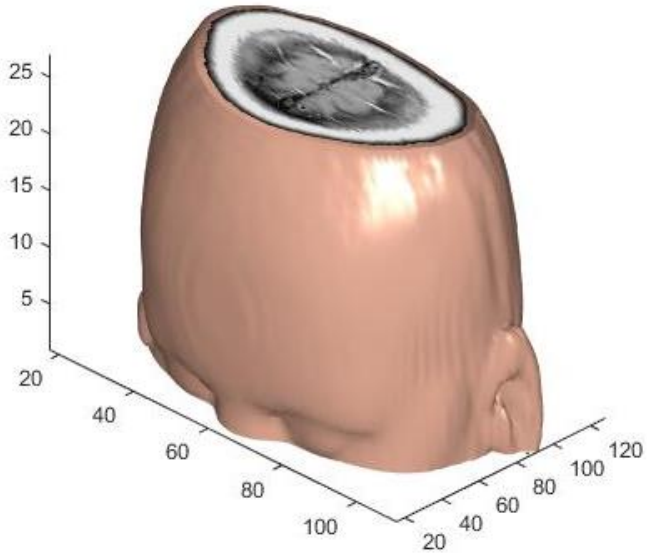


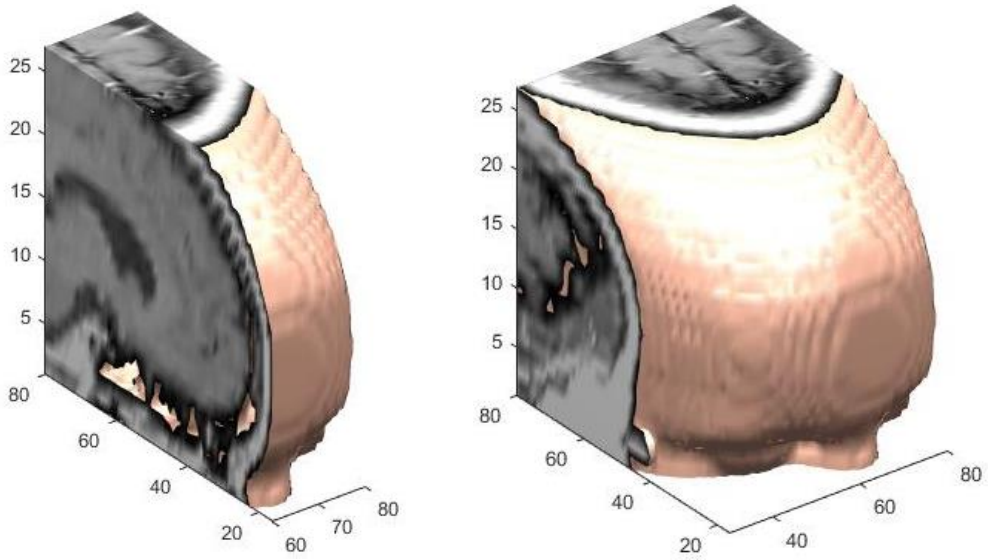
Рисунок 7.6 – Черепна коробка людини зі зрізом

Наступний фрагмент програми робить це для масиву з файлу mri.mat, що представляє черепну коробку людини (рис.7.7,а):

```
load mri
D=squeeze(D);
[x,y,z,D]=subvolume(D,[60,80,nan,80,nan,nan]);
p1=patch(isosurface(x,y,z,D,5),'FaceColor',[1,0.75,0.65],'EdgeColor','none');
isonormals(x,y,z,D,p1);
p2=patch(isocaps(x,y,z,D,5),'FaceColor','interp','EdgeColor','none');
view(3);
axis tight;
daspect([1,1,0.4]);
colormap(gray(100));
camlight right;
camlight left;
lighting gouraud;
```

Змінюючи параметри функції subvolume, можна візуалізувати довільну частину черепної коробки (рис.7.7,б):

```
⋮
[x,y,z,D]=subvolume(D,[30,80,nan,80,nan,nan]);
⋮
```



а) б)

Рисунок 7.7 – Об’ємний зріз черепної коробки людини зі зрізами з різних боків

Приклад 7.5. Візуалізація струменя у просторі. Дескрипторна графіка забезпечує ефективну візуалізацію у просторі струменів рідин або газів. Зокрема, такі задачі часто зустрічаються у гідро- та аеродинаміці. Наочний приклад цього демонструє функція flow, що створює масиви представлення струменя:

```
[x,y,z,v]=flow;
```

Виконаємо нормалізацію масштабів по осях координат та допоміжні операції у підготовці представлення струменя:

```
xmin=min(x(:));    ymin=min(y(:));    zmin=min(z(:));
xmax=max(x(:));    ymax=max(y(:));    zmax=max(z(:));
hslice=surf(linspace(xmin,xmax,100),linspace(ymin,ymax,100),zeros(100));
rotate(hslice,[-1,0,0],-45);
xd=get(hslice,'XData');
yd=get(hslice,'YData');
zd=get(hslice,'ZData');
delete(hslice);
```

Слід звергнути увагу на те, що об'ємний масив hslice *у цьому відіграє допоміжну роль. Після отримання з нього масивів* xd, yd *та* zd *проекцій масив* hslice *видаляється, щоб уникнути перевантаження пам'яті комп'ютера.*

Далі побудуємо проекції струменя на площини координатного ящика та діагональну площину:

```
h=slice(x,y,z,v,xd,yd,zd);
set(h,'FaceColor','interp','EdgeColor','none','DiffuseStrength',0.8);
hold on;
hx=slice(x,y,z,v,xmax,[],[]);
set(hx,'FaceColor','interp','EdgeColor','none');
hy=slice(x,y,z,v,[],ymax,[]);
set(hy,'FaceColor','interp','EdgeColor','none');
hz=slice(x,y,z,v,[],[],zmin);
set(hz,'FaceColor','interp','EdgeColor','none');
daspect([1,1,1]);
axis tight;
box on
view(-38.5,16);
camzoom(1.4);
camproj perspective
lightangle(-45,45);
colormap(jet(24));
set(gcf,'Renderer','zbuffer');
colormap(flipud(jet(24)));
caxis([-5,2.4832]);
colorbar('horiz');
```

Зібравши разом наведені вище фрагменти та виконавши їх, отримаємо наочну картину будови струменя (рис.7.8).

Образ струменя у просторі має високу наочність (особливо в оригіналі – у вигляді кольорового зображення на екрані) та добре виділяє різні особливості струменя, наприклад його турбулентність. *Проте слід зазначити, що наведений фрагмент програми не моделює струмінь, а лише відображає*

готовий результат моделювання, наведений у тестовій функції flow *у деякий момент часу.*

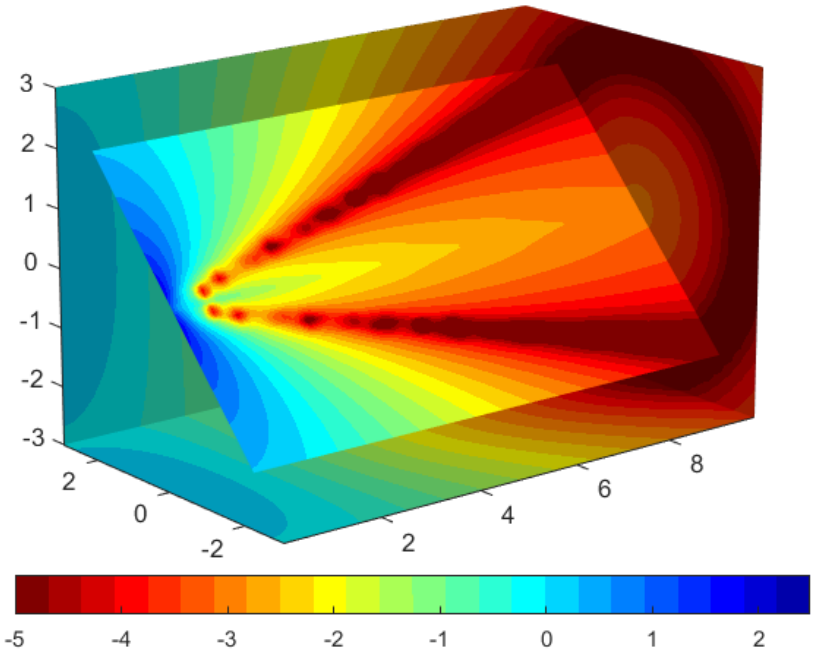


Рисунок 7.8 – Візуальне представлення струменя у просторі

Приклад 7.6. Візуалізація електричних розрядів. Файл wind зберігає масив, що представляє розвиток електричних розрядів в атмосфері. Наведений нижче фрагмент програми ілюструє техніку візуалізації цих явищ (рис.7.9):

```
load wind
xmin=min(x(:)); xmax=max(x(:)); ymax=max(y(:)); zmin=min(z(:));
wind_speed=sqrt(u.^2+v.^2+w.^2);
hsurfaces=slice(x,y,z,wind_speed,[xmin,100,xmax],ymax,zmin);
set(hsurfaces,'FaceColor','interp','EdgeColor','none');
hcont=contourslice(x,y,z,wind_speed,[xmin,100,xmax],ymax,zmin);
set(hcont,'FaceColor',[0.7,0.7,0.7],'LineWidth',1);
[sx,sy,sz]=meshgrid(80,20:10:50,0:5:15);
hlines=streamline(x,y,z,u,v,w,sx,sy,sz);
set(hlines,'LineWidth',1,'Color','y');
view(3);
daspect([2,2,1]);
axis tight
colormap(jet(16));
```

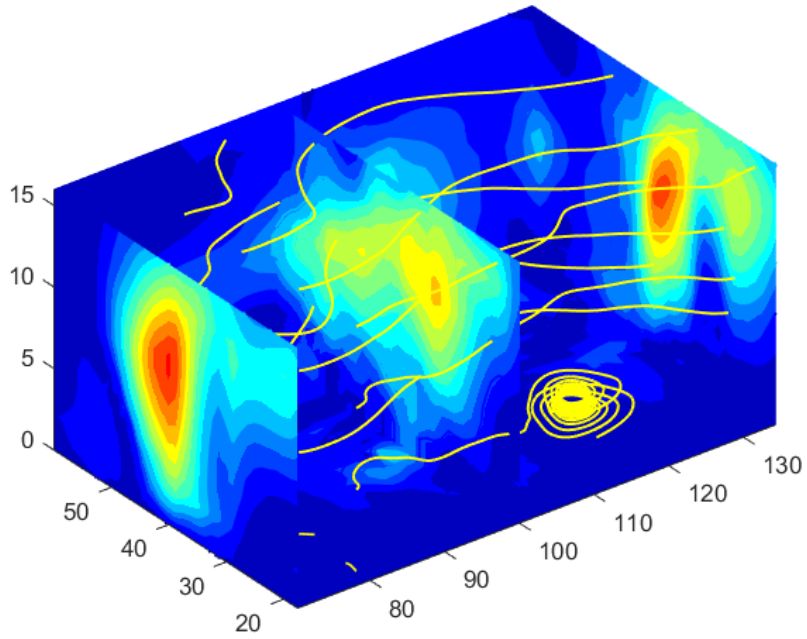


Рисунок 7.9 – Візуальне представлення електричних розрядів

Приклад 7.7. Анімація явища підйому предметів вихорами. Доволі часто доводиться спостерігати складні атмосферні явища, такі як вихори, торнадо та смерчі. Вони піднімають не лише тучі пилу, а й різні предмети – від каменів до автомобілів та будиночків. Вже згаданий файл wind зберігає дані не лише про розвиток атмосферних розрядів електрики, а й про супроводжуючі їх атмосферні вихори (типу смерчу).

Наведений нижче фрагмент програми візуалізує ці ефекти (стоп-кадр наведений на рис.7.10):

```
load wind
[sx sy sz]=meshgrid(100,20:2:50,5);
verts=stream3(x,y,z,u,v,w,sx,sy,sz);
s1=streamline(verts);
view(-10.5,18);
daspect([2,2,0.125]);
axis tight;
box on;
iverts=interpstreamspeed(x,y,z,u,v,w,verts,0.05);
streamparticles(iverts,15,'Animate',10,'ParticleAlignment','on',...
                'MarkerEdgeColor','none','MarkerFaceColor','red','Marker','o');
```

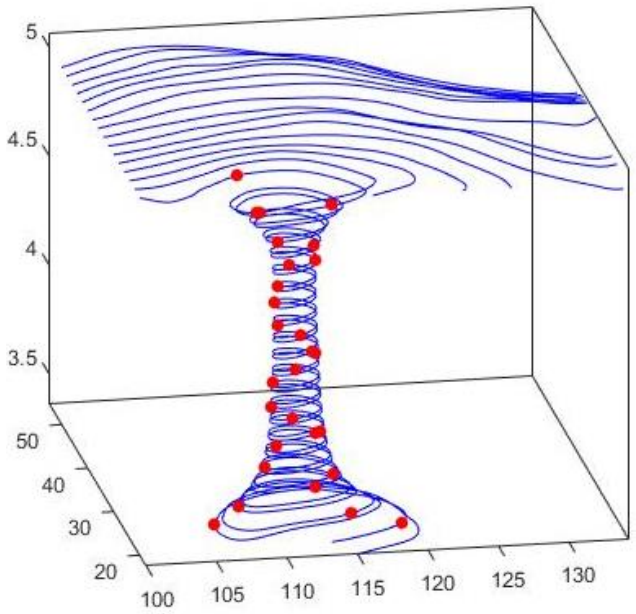


Рисунок 7.10 – Стоп-кадр анімації ефекту підйому предметів смерчем

Виконавши наведений вище код, можна протягом кількох секунд спостерігати анімаційну картину підйому предметів, зображених кружечками (рис.7.10).

Ще один варіант анімації підйому предметів смерчем представлений наступним фрагментом програми (рис.7.11):

```
load wind
[sx sy sz]=meshgrid(80,20:1:55,5);
verts=stream3(x,y,z,u,v,w,sx,sy,sz);
s1=streamline(verts);
iverts=interpstreamspeed(x,y,z,u,v,w,verts,0.025);
axis tight;
view(30,30);
daspect([1,1,0.125]);
camproj perspective;
camva(8);
box on;
streamparticles(iverts,35,'Animate',10,'ParticleAlignment','on');
```

Як вже зазначалося вище, приклад 7.7 ілюструє в динаміці результати моделювання.

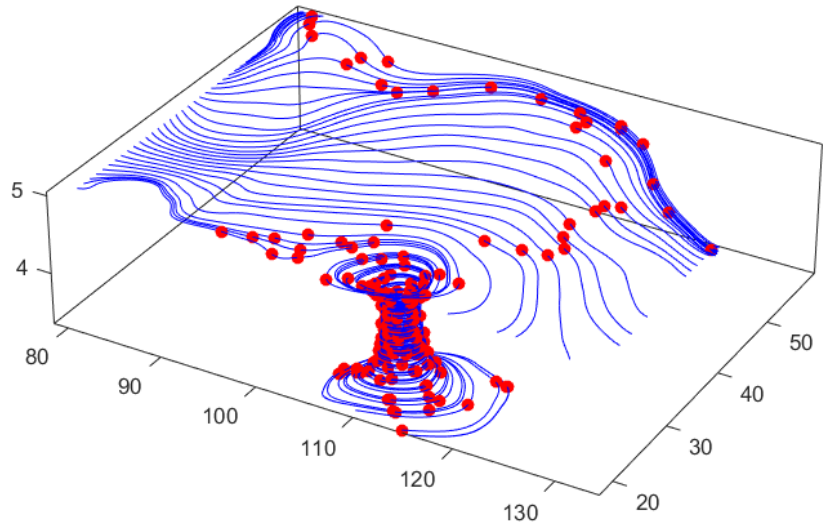


Рисунок 7.11 – Стоп-кадр анімації ефекту підйому предметів смерчем в обмеженому об’ємі

Приклад 7.8. Застосування «конусної» графіки для візуалізації струменів. Діти досліджують течію струмочків води, пускаючи паперові кораблики. Фізики, що вийшли з дитячого віку, теж використовують подібний підхід – тільки замість корабликів вони запускають у струмені повітря маленькі конуси – їх рух дозволяє досліджувати рух повітряних (газових, рідинних) струменів. Наведений нижче фрагмент програми дає уяву про застосування цієї техніки візуалізації (рис.7.12):

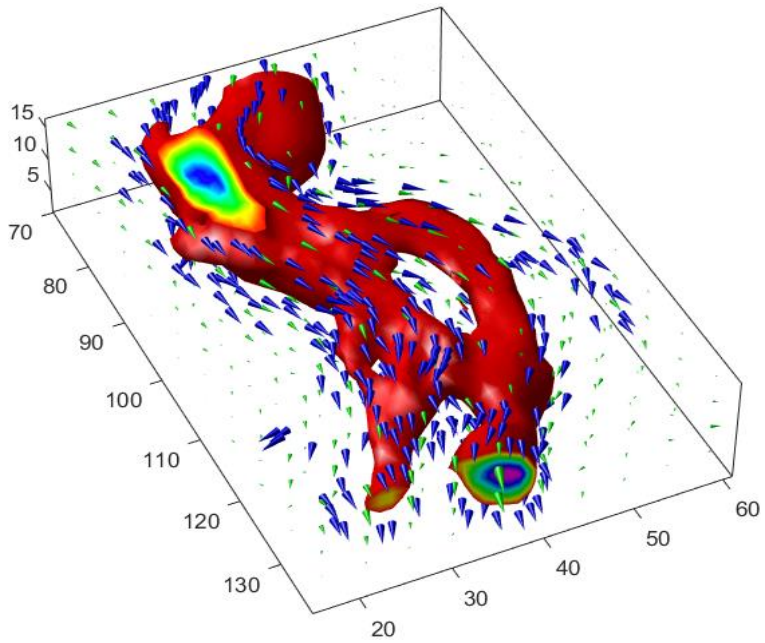


Рисунок 7.12 – Візуалізація потоків та струменів за допомогою конусів

```
load wind
wind_speed=sqrt(u.^2+v.^2+w.^2);
hiso=patch(isosurface(x,y,z,wind_speed,40));
isonormals(x,y,z,wind_speed,hiso);
set(hiso,'FaceColor','red','EdgeColor','none');
hcap=patch(isocaps(x,y,z,wind_speed,40),'FaceColor','interp','EdgeColor','none');
colormap(hsv);
daspect([1,1,1]);
[f verts]=reducepatch(isosurface(x,y,z,wind_speed,30),0.07);
h1=coneplot(x,y,z,u,v,w,verts(:,1),verts(:,2),verts(:,3),3);
set(h1,'FaceColor','blue','EdgeColor','none');
xrange=linspace(min(x(:)),max(x(:)),10);
yrange=linspace(min(y(:)),max(y(:)),10);
zrange=3:4:15;
[cx,cy,cz]=meshgrid(xrange,yrange,zrange);
h2=coneplot(x,y,z,u,v,w,cx,cy,cz,2);
set(h2,'FaceColor','green','EdgeColor','none');
axis tight;
box on;
camproj perspective;
camzoom(1.25);
view(65,45);
camlight(-45,45);
set(gcf,'Renderer','zbuffer');
lighting phong;
set(hcap,'AmbientStrength',0.6);
```

7.2 Image Processing Toolbox

Згадаємо пакет розширення Image Processing Toolbox (IPT) системи MatLab. Слід зазначити, що основним призначенням пакету IPT не є звичайна обробка зображень, наприклад, зміна яскравості та контрастності, підвищення різкості тощо, хоча все це можливе. Для цього створені чудові спеціалізовані програмні засоби, наприклад, CorelDraw, PhotoShop та інші, які мають сучасний інтерфейс користувача та широкі можливості обробки зображень, у

тому числі найвищої фотографічної якості. До того ж з максимально можливою швидкістю такої обробки. *Призначення пакету IPT в іншому: апробація й удосконалення вже відомих і розробка нових математичних методів обробки зображень та створення нових високоефективних, а часом і унікальних, програмних засобів для цього*.

Розглянемо коротко деякі основні функції IPT.

7.2.1 Виведення зображень на екран

Пакет розширення IPT додає до застосованої вище (див. приклад 2) функції виводу зображення image свою функцію **imshow** того ж призначення (рис.7.13): imshow hestain.png

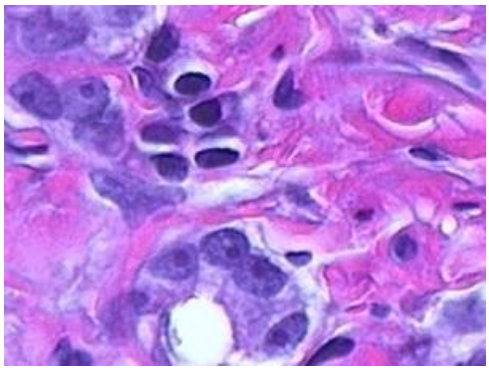


Рисунок 7.13 – Зображення, отримане за допомогою функції imshow

Функція **colorbar** виводить на екран шкалу кольорів даного зображення. При виводі шкали кольорів змінюється масштаб зображення та вивільнюєть місце для виводу кольорової палітри. У наведеному нижче прикладі функція imshow зчитує зображення з файлу spine.tif та будує його разом зі шкалою відтінків сірого кольору (за допомогою функції colorbar) (рис.7.14):

```
imshow spine.tif
colorbar vert
```

Рисунок 7.14 – Зображення, отримане з файлу spine.tif зі шкалою відтінків сірого

В наведеному нижче прикладі функція **imshow** задає показ палітрового зображення з файлу hands1.jpg, потім функцією getimage аналізується та створюється його матриця G, після чого командою whos виводяться дані про масив, що зберігає дане зображення (рис.7.15):

```
imshow hands1.jpg
G=getimage;
whos
```

Name	Size	Bytes	Class	Attributes
G	240x320x3	230400	uint8	

Рисунок 7.15

Слід зазначити, що такий вид виводу командою можливий лише на початку сеансу роботи в або після очищення робочої області.

Функція **immovie** повертає матрицю відеопослідовності з багатокадрового індексованого зображення: mov=immovie(X,map); Багатокадрове зображення X складене з множини індексованих (палітрових) зображень однакового розміру та з однаковою кольоровою палітрою mov. X – багатовимірний масив розміру m×n×1×k, де k – кількість кадрів.

Функція **montage** виводить на екран всі кадри відеопослідовності (анімації зображення). Наведений нижче приклад дає картину всіх кадрів анімації для багатокадрового індексованого зображення (рис.7.16):

```
load mri
mov=montage(D,map);
```

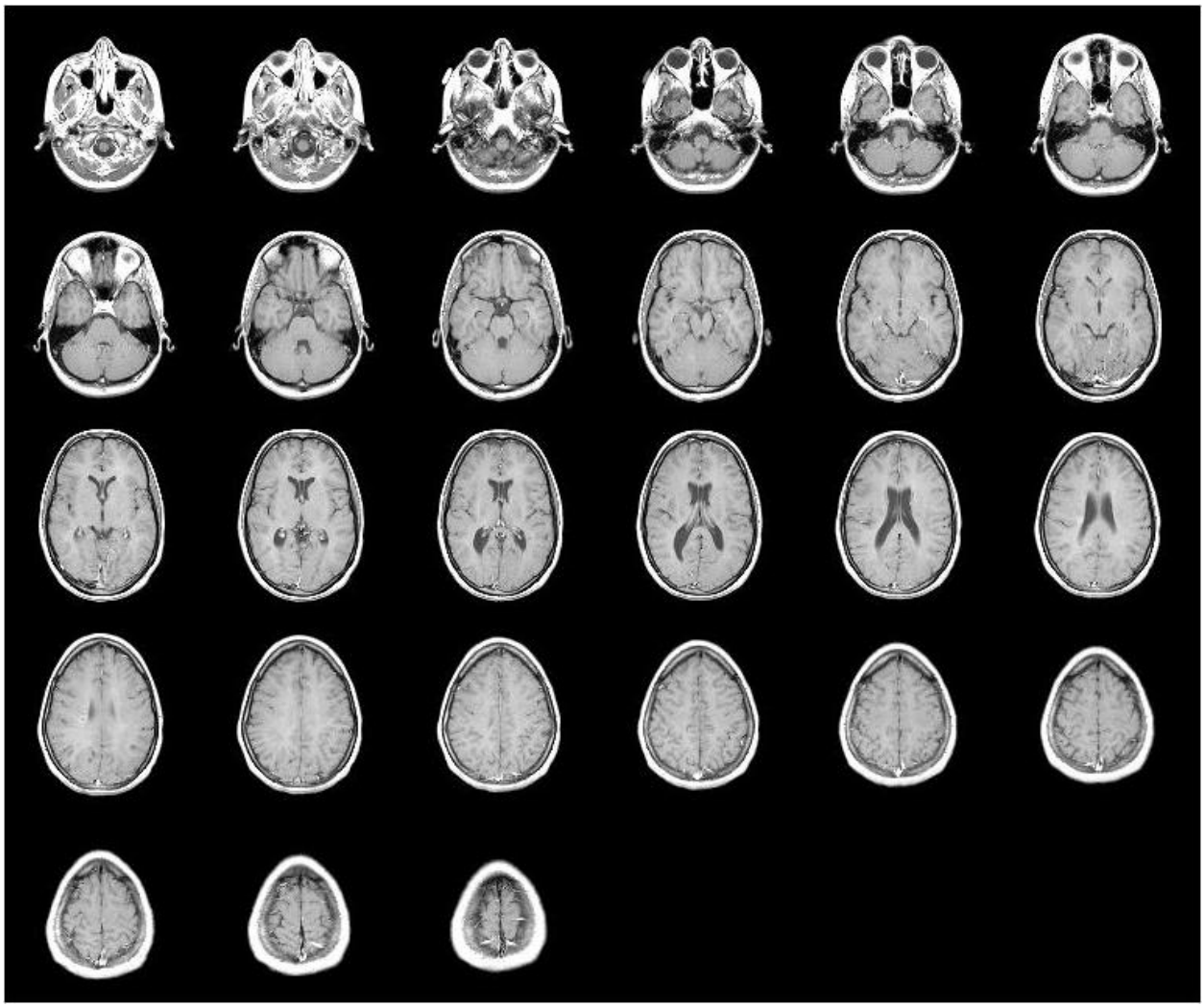


Рисунок 7.16 – Всі кадри анімаційного рисунку, отримані за допомогою функції montage

Функція **subimage** може бути використана для виводу на екран кількох зображень в одному вікні, навіть якщо ці зображення мають різні кольорові карти. Її слід використовувати після застосування функції створення підвікон **subplot**.

7.2.2 Покращання зображень та фільтрація

Одним зі способів покращання зображення, наприклад, шляхом збільшення або зменшення його контрастності, є *вирівнювання* його *гістограми* – розподілу яскравості пікселів зображення. Цей спосіб реалізується функцією **histeq**.

Для зображень низької якості введення *гама-корекції* нерідко суттєво покращую їх якість – помітно вище, ніж просто збільшення або зменшення яскравості зображення. Функція **imadjust** створює нове зображення з вихідного при заданому параметрі гама-корекції.

Функція **adapthisteq** створює нове зображення шляхом копіювання вихідного зображення з *автоматичним вирівнюванням гістограми* розподілу яскравості та контрастності.

Для здійснення *рангової фільтрації* напівтонових зображень служить функція **ordfilt2**. Ця функція має широкий спектр дій – вона може підкреслювати дефекти (а може, навпаки, потрібні особливості) зображення та здатна ефективно фільтрувати дрібні деталі, наприклад, завади.

Окремим випадком рангової фільтрації є *медіанна фільтрація*, що проводиться за допомогою функції **medfilt2**.

Функція **wiener2** реалізує адаптивну фільтрацію зображення методом Вінера.

Для зміни характеристик зображення (зокрема, різкості) або виділення його характерних ознак широко використовується *лінійна фільтрація*, представлена в системі MatLab кількома десятками функцій, такими як, наприклад, функції згортки, двовимірної фільтрації **filter2** тощо.

Функція **fspecial** створює маску двовимірного фільтра заданого типу у вигляді матриці, що зберігає коефіцієнти фільтра, які задають його передаточну характеристику та властивості. Можливі такі типи фільтрів (масок): 'average' (усереднюючий фільтр), 'gaussian' (фільтр Гаусса нижніх

частот), 'laplacian' (фільтр Лапласа високих частот), 'log'(комбінований фільтр Лапласа-Гаусса високих частот, в який послідовно включені фільтри Лапласа та Гаусса), 'sobel' (фільтр Собеля для виділення горизонтальних границь), 'prewitt' (фільтр Превітта для виділення горизонтальних границь) та 'unsharp' (фільтр підвищення різкості).

До інструменту дослідження та маніпуляцій із зображеннями, окрім основного вікна, входять вікна навігації по зображенню, перегляду значень пікселів, даних про зображення та вікно з гістограмами розподілу яскравості. Для підтримання графічного інтерфейсу вказаного інструменту існує ряд функцій, зокрема: imageinfo – вікно індикації даних про зображення, imcontrast – вікно зміни контрасту зображення, imdisplayrange – вікно задання границь зображення, impixelinfo – вікно даних про пікселі зображення, impixelregion – вікно встановлення границь пікселів тощо.

Також інструмент дослідження та маніпуляцій із зображеннями містить функції аналізу текстури зображень, зокрема: rangefilt – обчислення локальних границь зображення, stdfilt – обчислення локального середньоквадратичного відхилення, entropyfilt – обчислення локальної ентропії напівтонового зображення тощо.

Детально про наведені вище та багато інших функцій системи MatLab можна дізнатися з відповідного файлу довідки.

Контрольні питання

1. Дескрипторна графіка системи MatLab та розширена техніка візуалізації.
2. Використання Image Processing Toolbox.