**Вступ.**

**Основні завдання з обробки зображень**

Монохромне зображення можна визначити як двовимірну функцію

F(x,y),

де х і у - координати на площині зображення, а амплітуда F() називається інтенсивністю або яскравістю зображення в точці з цими координатами.

Словосполучення рівень сірого використовується для позначення яскравості монохромного зображення.

Кольорові зображення формуються комбінацією кількох монохромних зображень.

Наприклад, у кольоровій системі RGB кольорове зображення будується з трьох окремих монохромних компонентів (червоної, зеленої та синьої) і може бути представлено вектором з трьох інтенсивностей

R, G, B.

З цієї причини багато методів і прийомів, розроблені для монохромних зображень, можуть бути поширені на кольорові зображення шляхом послідовної обробки трьох монохромний компонент. Первинне зображення має безперервні х - та y - координати, а також безперервну амплітуду f.

Перетворення його на цифрову форму вимагає подання координат та значень амплітуди деякими дискретними відліками. Подання координат кінцевою множиною відліків називається дискретизацією, а подання амплітуди значеннями із кінцевого набору називається квантуванням. Якщо координати х та у, а також величини амплітуди f вибираються з фіксованих кінцевих наборів елементів (дискретних величин), то зображення називається цифровим зображенням.

Цифрова обробка зображень має широкі сфери застосування, включаючи зорові образи. Однак на відміну від людей, які здатні сприймати лише електромагнітне світлове випромінювання видимого діапазону, машинна обробка зображень покриває практично весь спектр електромагнітних та інших хвиль. Наприклад, це ультразвукові зображення або зображення, отримані в електронній мікроскопії. Не існує загальновизнаного кордону, який би поділяв область обробки зображень та інші суміжні дисципліни, такі як аналіз зображень чи машинний зір. Іноді таке розмежування робиться за принципом, що обробка зображень характеризується присутністю зображень на вході та виході системи. Однак таке визначення видається нам невиправдано обмежувальним і штучним.

У всьому різноманітті завдань від обробки зображень до машинного зору немає чітких меж, проте тут можна виділити комп'ютеризовані процеси низького, середнього та високого рівня.

* Процеси низького рівня включають лише примітивні операції над зображеннями типу зменшення шуму,
* підвищення контрастності або
* поліпшення різкості.

Вони характеризуються тим, що на вхід та вихід надходять зображення.

Процеси середнього рівня пов'язані з такими завданнями:

* сегментація (розподіл зображень на області та виділення в них об'єктів),
* опис об'єктів та їх стиск для надання їм зручної форми при подальшій комп'ютерній обробці,
* класифікація (розпізнавання) виділених об'єктів.

У середньорівневих процесах на вході є зображення, а на вихід надходять атрибути та ознаки, вилучені з цих зображень, наприклад, межі, контури та інші відмітні ознаки об'єктів, які є зображеннями.

Нарешті, процеси високого рівня займаються «осмисленням» множини розпізнаних об'єктів, як це робиться в аналізі зображень.

Маючи на увазі перелічені вище зауваження, видно, що природним полем перетину та переходу від обробки зображень до їх аналізу є область розпізнавання окремих фігур або об'єктів на 2 зображення. Таким чином, те, що називається цифровою обробкою зображень, пов'язане з процесами, що мають зображення на вході та на виході, а також з процесами отримання певних зображень.

Найближчими для обробки зображень завданнями з аналізу зображень є проблеми розпізнавання окремих об'єктів. В якості ілюстрації, що прояснює ці концепції, можна навести завдання автоматичного аналізу друкованого чи рукописного тексту. Для розуміння обробки зображень важливо вирішувати практичні завдання, а для цього треба мати готовий комп'ютерний інструмент здатний проілюструвати застосування тих чи інших методів.

В MatLab є середовище для виконання технічних та наукових обчислень. У ньому інтегровані обчислення, візуалізація та програмування в зручній для користувача середовищі, в якому завдання та їх вирішення виражаються за допомогою звичних математичних позначень. Базовим елементом MATLAB є масив елементів (матриця), який не вимагає завдання фіксованого розмірності. Це дозволяє легко формулювати умови та рішення багатьох обчислювальних завдань, яким потрібне матричне подання об'єктів. Система MATLAB має розширення у вигляді наборів спеціалізованих програм, які по-англійськи називаються toolbox (набір інструментів).

Пакет Image Processing Toolbox (IPT) складається з функцій MATLAB (вони називаються М-функції або М-файли), які розширюють можливості стандартного середовища MATLAB для роботи з зображеннями.

Створимо матрицю 5 x 5:

>>A=[1 2 3 0 0; 4 5 6 0 0; 7 8 9 0 0; 0 0 0 0 0; 0 0 0 0 0]

A = 1 2 3 0 0

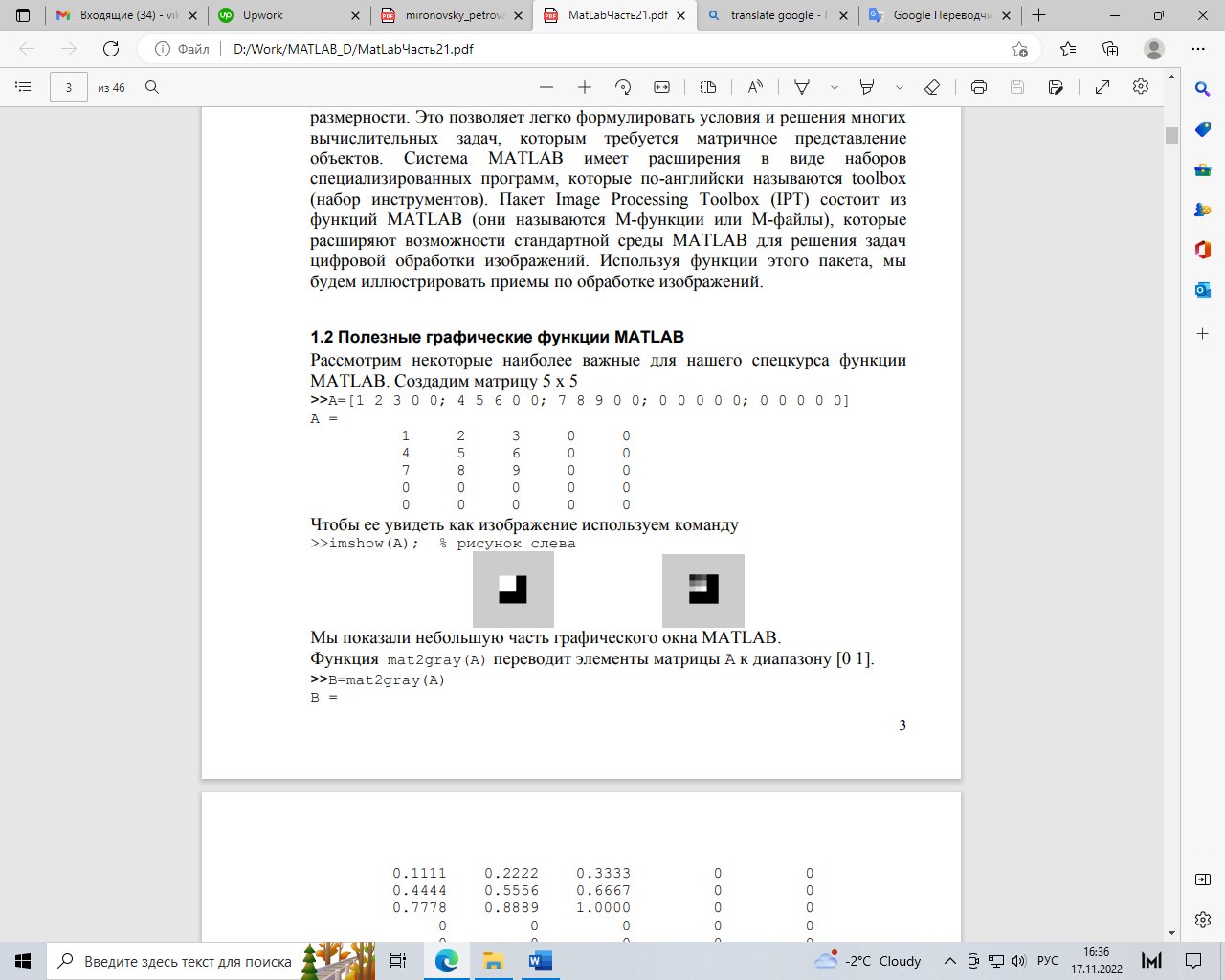
4 5 6 0 0

7 8 9 0 0

0 0 0 0 0

0 0 0 0 0

**Imshow**():



mat2gray(A) приводить A до діапазону [0 1].

>>B=**mat2gray**(A)

B = 3 0.1111

0.2222 0.3333 0 0 0.4444

0.5556 0.6667 0 0 0.7778

0.8889 1.0000 0 0 0

0 0 0 0 0

0 0 0 0 0

А функція **im2uint8**(B) преобраує матрицю-аргумент до діапазону [0 255].

>>C=im2uint8(B)

C = 28 57 85 0 0

113 142 170 0 0

198 227 255 0 0

0 0 0 0 0

0 0 0 0 0

>>imshow(C); % попередній малюнок

В окні Workspace можна подивитись типи матриць:

A – double,

B – double,

C – uint8.

Функция im2uint16, переводить величини до діапазону [0 65535].

>>A=[1 2 3 0 0; 4 5 6 0 0; 7 8 9 0 0; 0 0 0 0 0; 0 0 0 0 0];

>>image(A); створює зображення (с громадними пікселями) в графічесному вікні.

Для завантаження (читання) зображень у робочий простір MATLAB використовується функція imread з наступним синтаксисом:

>**imread**('filename')

Тут filename - це рядок символів, що утворюють повне ім'я файлу зображення, що завантажується (включаючи будь-яке розширення). Наприклад, командний рядок

>> f=imread('football.jpg');

надає зображення формату JPEG з ім'ям «football.jpg» матричної змінної f.

Зауважимо, що символ '(апостроф) використовується в як обмежувач символьного рядка.

Якщо в ім'я файлу не включена інформація про шлях до файлу, то файл filename шукається у поточній папці. А якщо його там немає, то виконується пошук цього файлу у всіх папках, шляхи до яких вказані в шляхах пошуку MATLAB.

Функція size(f) повертає розмір зображення, тобто число рядків та стовпців масиву, що представляє зображення:

**size**(f) % f містить зображення football.jpg

ans = 256 320 3

Наприклад,

[M,N,K] = size(f);

За такого запису змінної М буде присвоєно число рядків зображення, а змінної N – число стовпців, K – число колірних площин.

Функція whos повідомляє додаткову інформацію про масив. Наприклад, рядок:

**whos** f

отримує:

Name Size Bytes Class f

256x320x3 245760 uint8 array

Grand total is 245760 elements using 245760 bytes

Функція **numel**(f) кількість елементів масиву f, тобто пікселів изображения

>>numel(f)

ans = 245760

Як ми говорили вище, завантажене зображення можна вивести на дисплей комп'ютера за допомогою функції imshow, яка має наступний синтаксис:

>>imshow(f);

або

>>imshow(f, G),

де f - це матриця зображення, a G - це кількість рівнів яскравості, використовується при відображенні цього зображення.

Якщо аргумент G опущено, то за умовчанням приймається 256 рівнів яскравості (то яке отримано з графічного файлу).

Команда imshow(f,[low high]) означає, що всі пікселі зі значенням не більше за число low треба показувати чорними, а всі пікселі зі значеннями не менше числа high білими. Всі значення розташовані між low та high показуються з проміжною яскравістю з використанням числа рівнів прийнятого за замовчуванням.

Нарешті, запис у командному рядку imshow(f,[ ] ) задає для змінної low мінімальне значення масиву f, а змінної high надається його максимальне значення. Така форма функції imshow буває корисною при показі зображень, що мають вузький динамічний діапазон значень пікселів, або коли серед них є позитивні та негативні значення. У цьому посібнику ми вивчатимемо методи обробки зображень. Але після обробки воно має бути збережене у графічний файл. Це можна виконати командою

**imwrite**(A,filename,fmt),

яка зберігає образ A у файл, заданий ім'ям filename в графічний формат, визначений аргументом fmt. Наприклад, для зображення футбольного м'яча, що зберігається у файлі 'football.jpg', наступні команди створять графічний файл того ж зображення, але в форматі bmp:

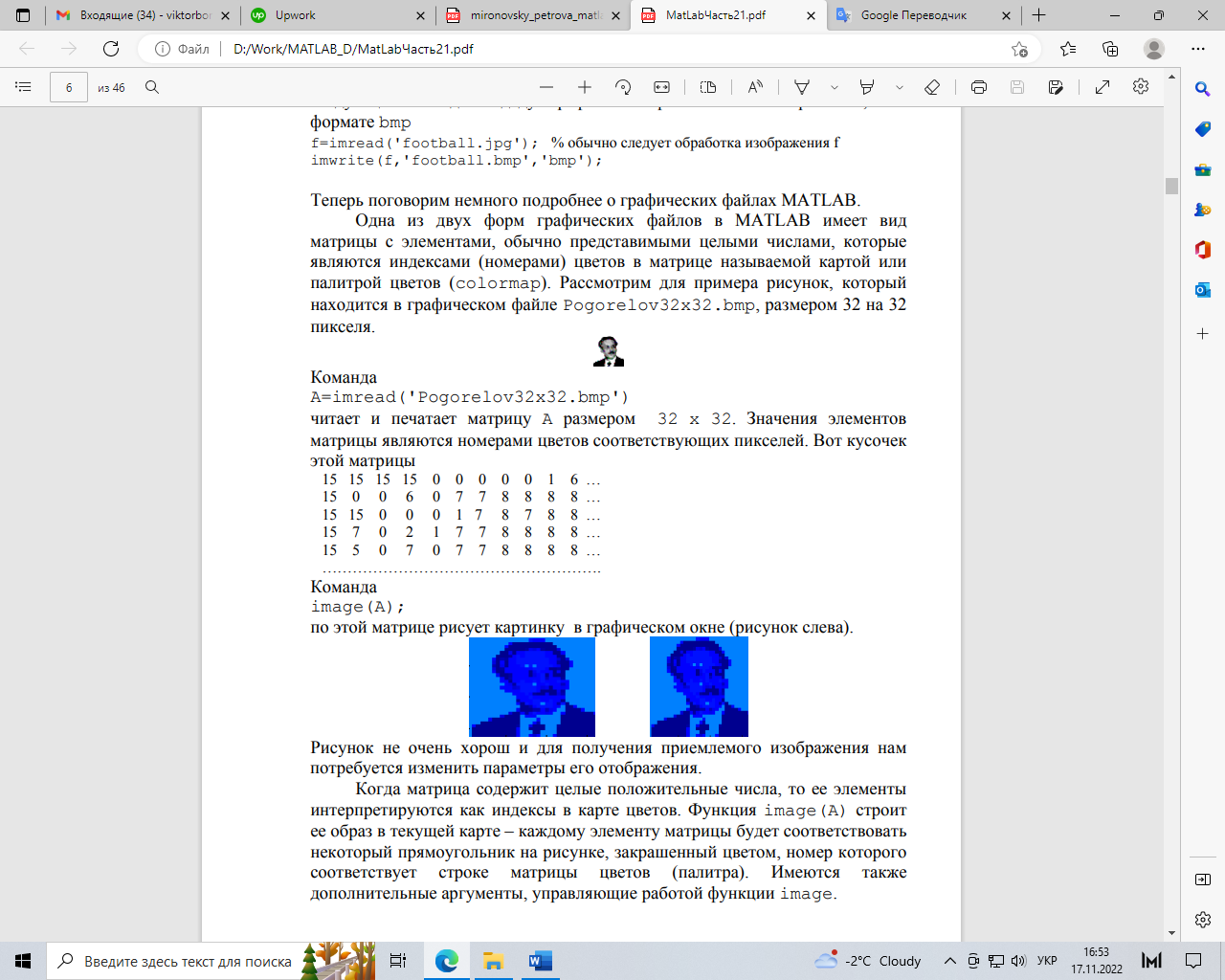
>>f=imread('football.jpg'); % обробка зображення f

>>imwrite(f,'football.bmp','bmp');

Обробка файлів:

A=imread('Pogorelov32x32.bmp')

image(A);



Малюнок не дуже добрий і для отримання прийнятного зображення нам потрібно змінити параметри його відображення. Коли матриця містить цілі позитивні числа, її елементи інтерпретуються як індекси у карті кольорів. Функція image(A) будує її образ у поточній карті – кожному елементу матриці буде відповідати деякий прямокутник на малюнку, зафарбований кольором, номер якого відповідає рядку матриці кольорів (палітра). Є також додаткові аргументи, які керують роботою функції image.

Отриманим чином у графічному вікні, можна керувати, використовуючи Прості функції MatLab, управляючі графікою. Зокрема, у нашому На прикладі малюнок витягнутий по горизонталі. Команда axis equal; вирівнює горизонтальні та вертикальні розміри пікселів у графічне вікно. Вона встановлює коефіцієнт стиснення зображення однаковим по всіх осях (див. попередній малюнок праворуч). Той самий результат дає команда

**axis** image;

але охоплює зображення прямокутник у графічному вікні деформується під розмір зображення. Кольори точок беруться із поточної карти кольорів. Карта кольорів (палітра) - це матриця, яка містить 3 стовпці і кілька (може бути багато) рядків.

Наприклад, команда **gray(5)** створює наступну матрицю, яка може бути використана як палітра кольорів.

0 0 0

0.2500 0.2500 0.2500

0.5000 0.5000 0.5000

0.7500 0.7500 0.7500

1.0000 1.0000 1.0000

min(min(A))

ans = 0

max(max(A))

ans = 15

Команди max та min визначають найбільше та найменше значення матриці по одній із її розмірностей. Наприклад, одноразове застосування команди max(A) повертає вектор з 32 чисел, що є максимальними значеннями за відповідними стовпцями матриці A. визначення максимального значення всієї матриці потрібно ще раз застосувати функцію max до результату першого застосування. Оскільки матриця A містить числа від 0 до 15, розумно створити чорно-білу палітру з 16 відтінками сірого.

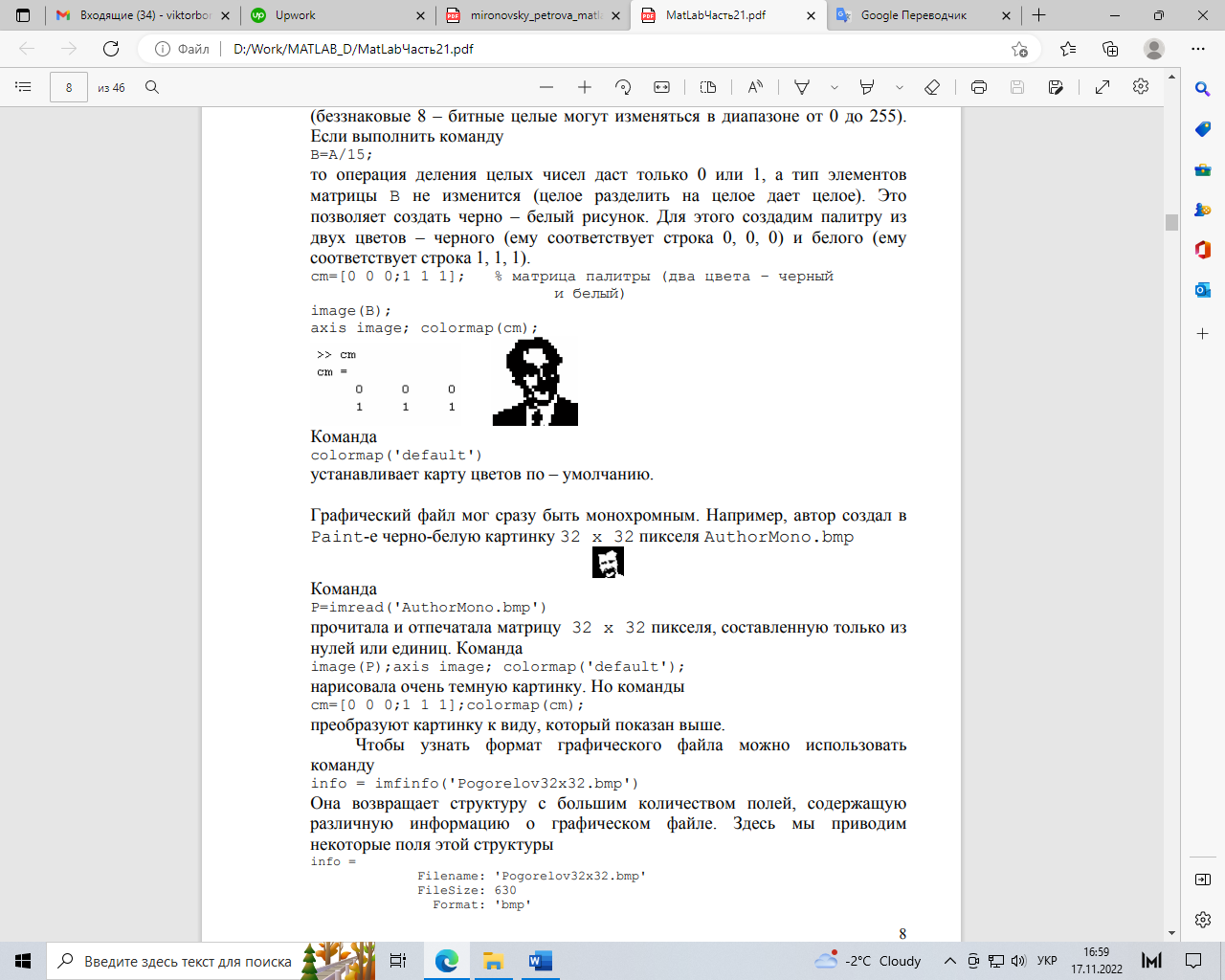
Команда

**colormap**(gray(16));

перетворить малюнок до кольорового представлення

Зверніть увагу на тип елементів матриці A у вікні Workspace – uint8 (беззнакові 8 - бітові цілі можуть змінюватися в діапазоні від 0 до 255). Якщо виконати команду B=A/15; то операція поділу цілих чисел дасть лише 0 чи 1, а тип елементів матриці B не зміниться (ціле поділити на дає ціле). Це дозволяє створити чорно-білий малюнок.

Для цього створимо палітру з двох кольорів – чорного (йому відповідає рядок 0, 0, 0) та білого (йому відповідає рядок 1, 1, 1)



info = **imfinfo**('Pogorelov32x32.bmp') – формат файлу

info =

Filename: 'Pogorelov32x32.bmp'

FileSize: 630

Format: 'bmp' 8

FormatVersion: 'Version 3 (Microsoft Windows 3.x)'

Width: 32

Height: 32

BitDepth: 4

ColorType: 'indexed'

NumColormapEntries: 16

Colormap: [16x3 double]

NumPlanes: 1

[X,map] = imread(filename)

[A,m] = imread('Pogorelov32x32.bmp');

image(A);

colormap(m);

[A,map,alpha] = imread(...)

Зробимо фон білим:

B2=B; % копируем палитру в B2

B2(256,:)=[1 1 1]; % последний цвет палитры сделать белым цветом

D = ones(size(A)) \* (length(B2)-1); %создать матрицу с элементами 255

D(C == 0) = A(C == 0);

image(uint8(D)), colormap(B2); %правый рисунок

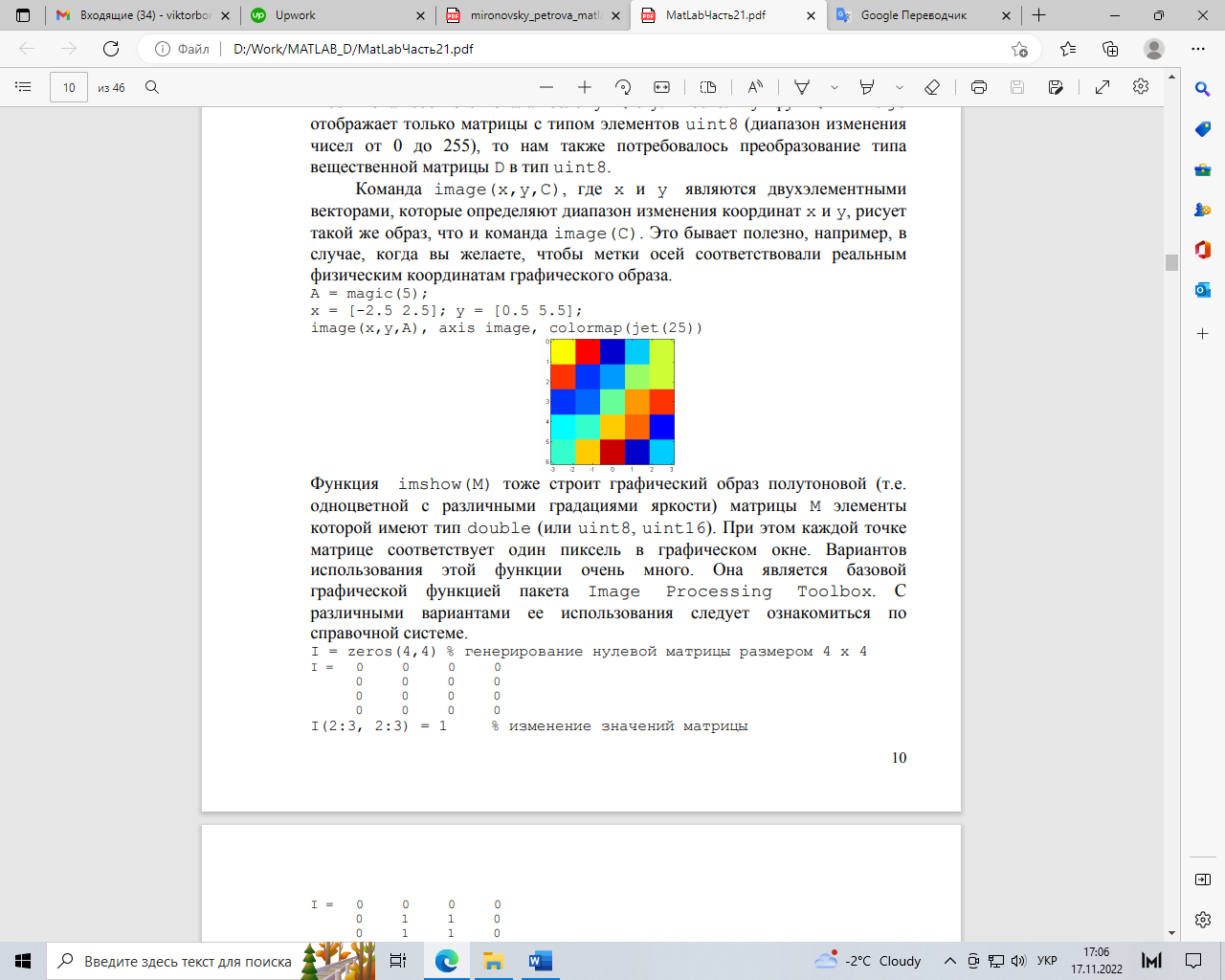
axis image;

Якщо подивитися у вікні Workspace, то видно, що масив C є логічним. І при цьому він має такий самий розмір, що і масив A. Тут використовується логічне індексування у формі ім'я\_матриці (логічний масив) де логічний масив повинен мати той самий розмір, як і матриця. Результатом цієї операції логічного індексування є вектор, складений з елементів вихідної матриці, для яких у логічному 9 масиві відповідні елементи дорівнюють логічній одиниці. Цей вектор може стояти як у правій частині оператора присвоювання, і у лівої.

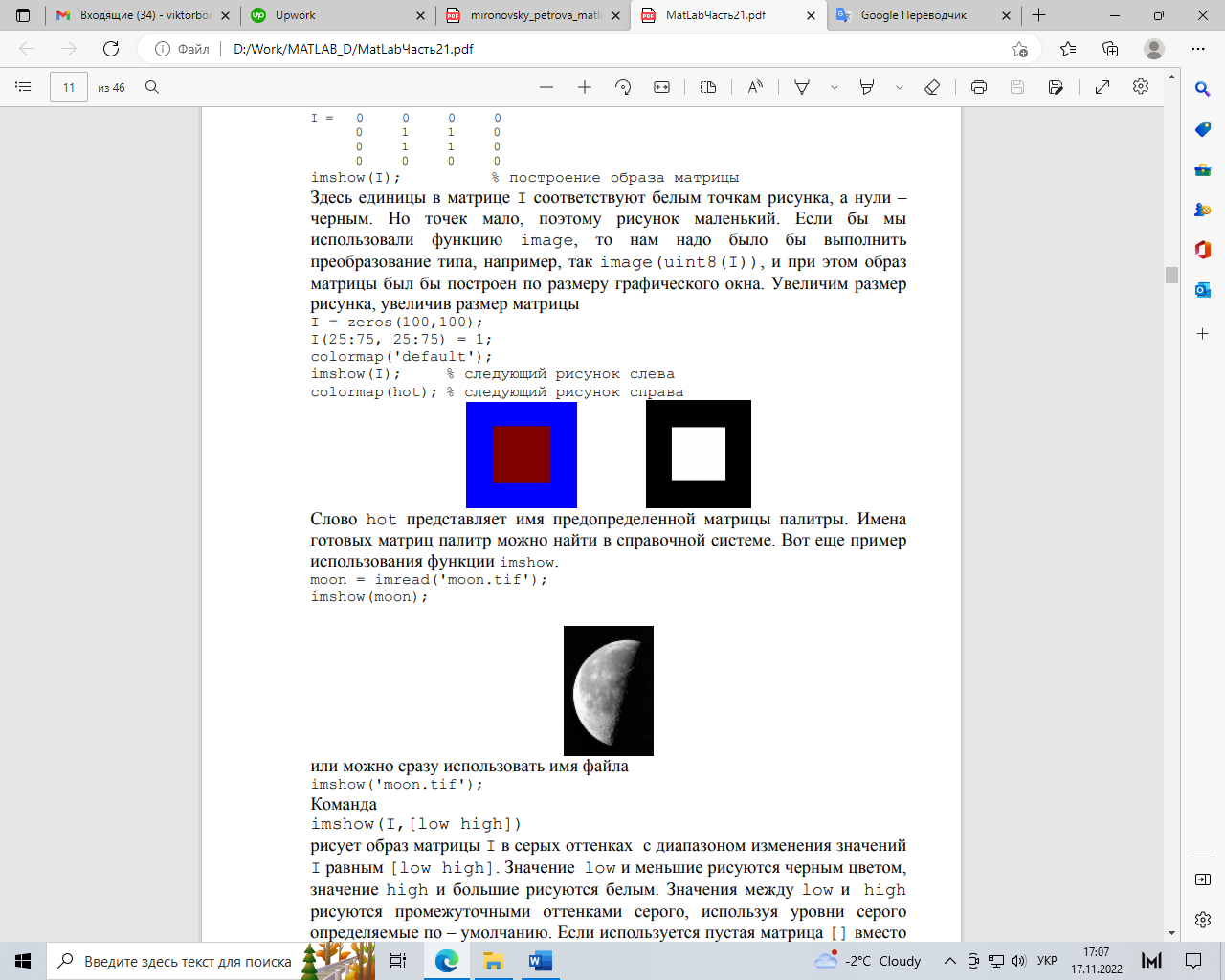
Спочатку ми створили double масив D з чисел 255, а потім створили вектор однакової довжини D(C==0) та A(C==0). Другий вектор містить лише елементи матриці A, котрим у матриці C елементи дорівнюють нулю. Це означає, що відповідні точки образу мають бути намальовані.

Значення вектора A(C==0) присвоюються елементам вектора D(C==0), тобто. відповідним елементам матриці D. Якщо у матриці C елемент 1, то відповідний елемент матриці D не змінюється, тобто. залишається рівним 255. В результаті елементи матриці A відповідають точках, що відображаються, скопіювалися в матрицю D, а відповідальні точкам, що не відображаються, залишилися в матриці D рівними 255. Крім того, нам знадобилося замінити останній (256 – й) рядок матриці палітри, щоб вона відповідала білому кольору.

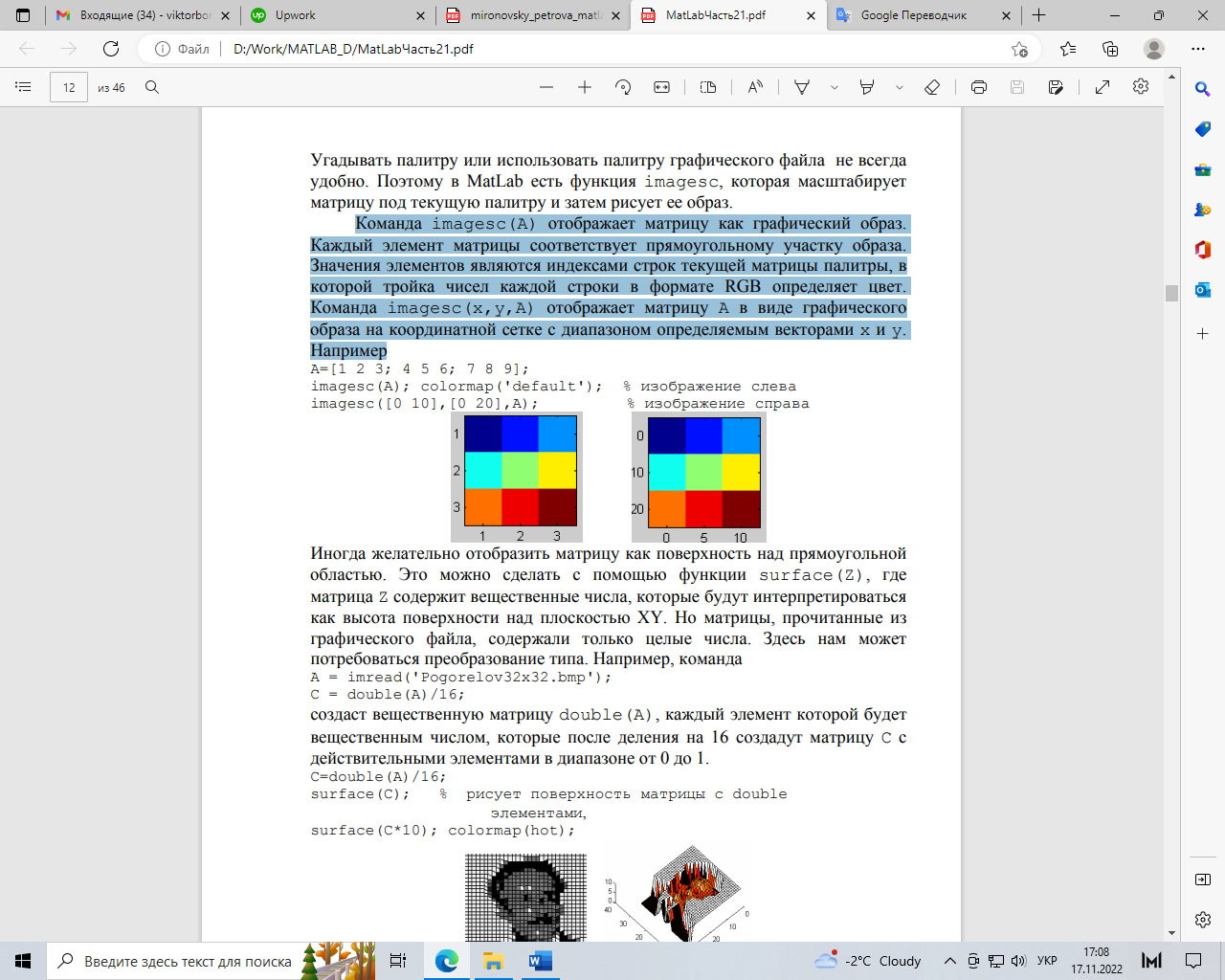
Оскільки функція image відображає лише матриці з типом елементів uint8 (діапазон зміни чисел від 0 до 255), то нам також знадобилося перетворення типу речової матриці D тип uint8. Команда image(x,y,C), де x та y є двоелементними векторами, які визначають діапазон зміни координат x та y, малює такий самий образ, як і команда image(C). Це буває корисно, наприклад, випадку, коли ви бажаєте, щоб мітки осей відповідали реальним фізичним координатам графічного образу



Приклад:



Команда imagesc(A) відображає матрицю як графічний образ. Кожен елемент матриці відповідає прямокутній ділянці образу. Значення елементів є індексами рядків поточної матриці палітри, якою трійка чисел кожного рядка у форматі RGB визначає колір. Команда imagesc(x,y,A) відображає матрицю A у вигляді графічного образу на координатній сітці з діапазоном, що визначається векторами x і y. Наприклад,



Іноді бажано відобразити матрицю як поверхню над прямокутною. областю. Це можна зробити за допомогою функції surface(Z), де матриця Z містить речові числа, які інтерпретуватимуться як висота поверхні над площиною XY. Але матриці, прочитані з графічного файлу, містили лише цілі числа. Тут нам може потрібно перетворення типу. Наприклад, команда

A = imread('Pogorelov32x32.bmp');

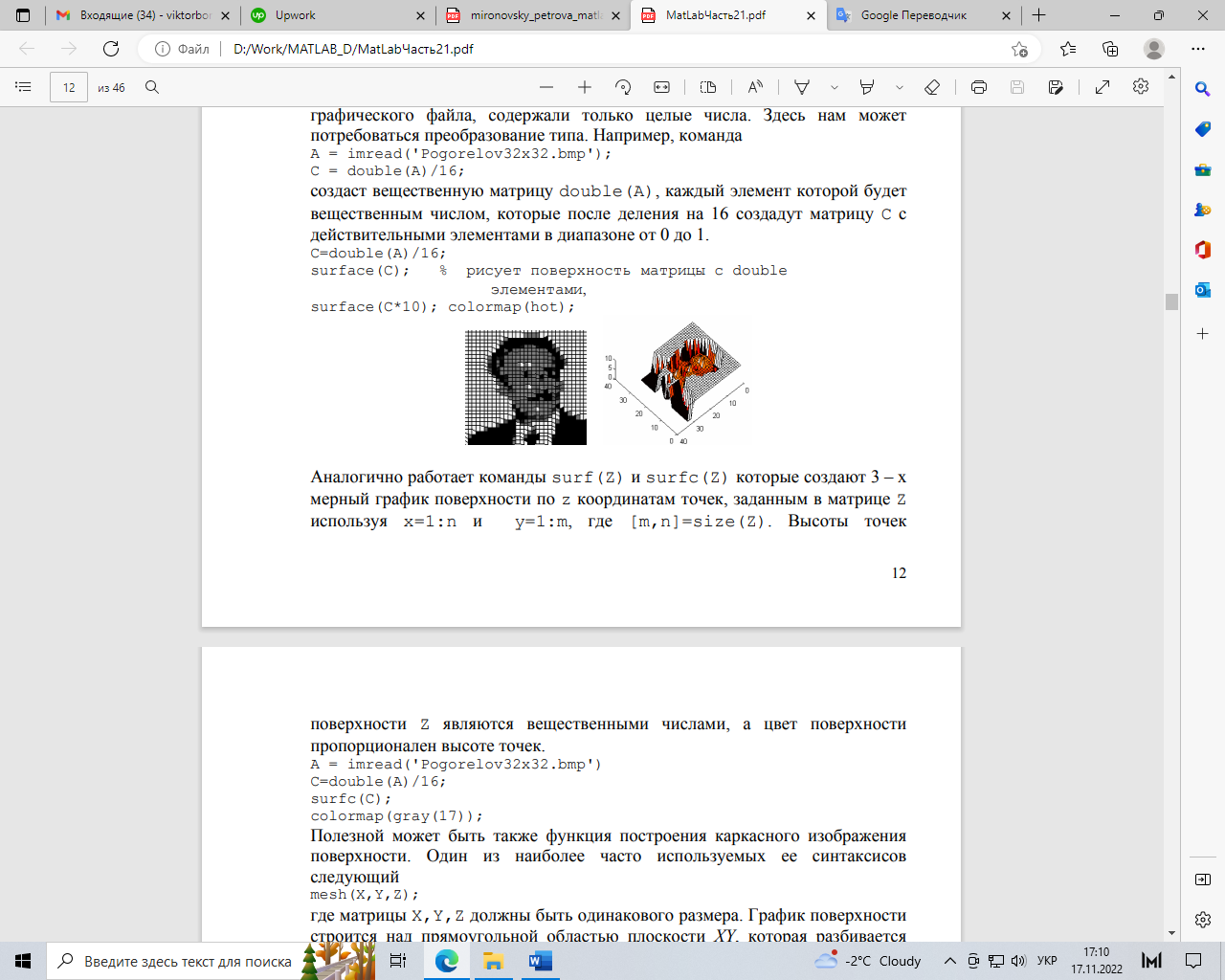
C = double(A)/16;

створить дійсну матрицю double(A), кожен елемент якої буде речовим числом, які після розподілу на 16 створять матрицю C з дійсними елементами у діапазоні від 0 до 1.

C=double(A)/16; surface(C); % малює поверхню матриці з double елементами,

surface(C\*10);

colormap(hot);



Аналогічно працють команди

**surf**(Z)

та

**surfс**(Z)

які створюють 3–х вимірний графік поверхні по z координат точок, заданих у матриці Z використовуючи

x=1:n та y=1:m, де [m,n]=size(Z).

Висоти точок поверхні Z є дійсними числами, а колір поверхні пропорційний висоті точок.

A = imread('Pogorelov32x32.bmp')

C=double(A)/16;

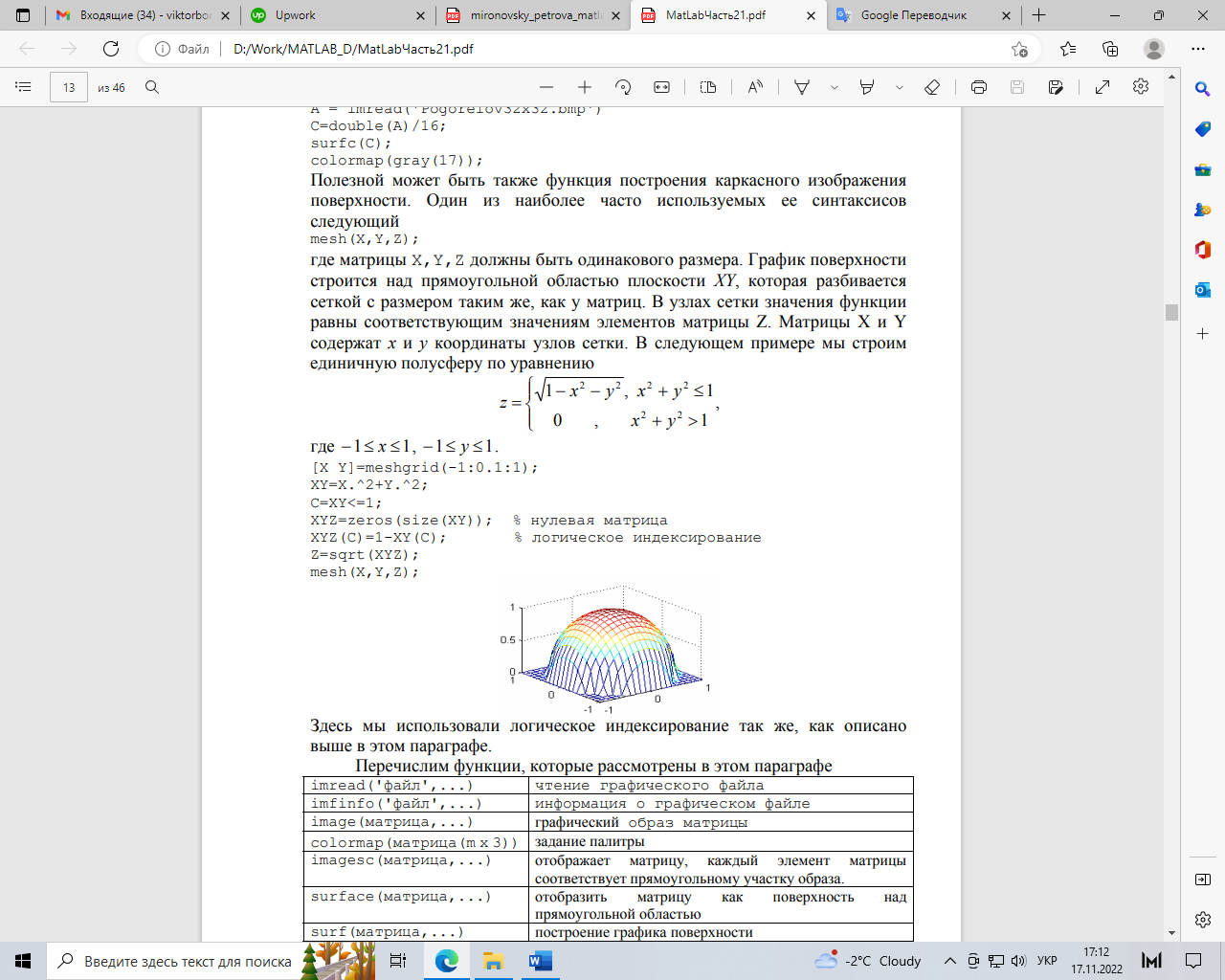
surfc(C);

colormap(gray(17));

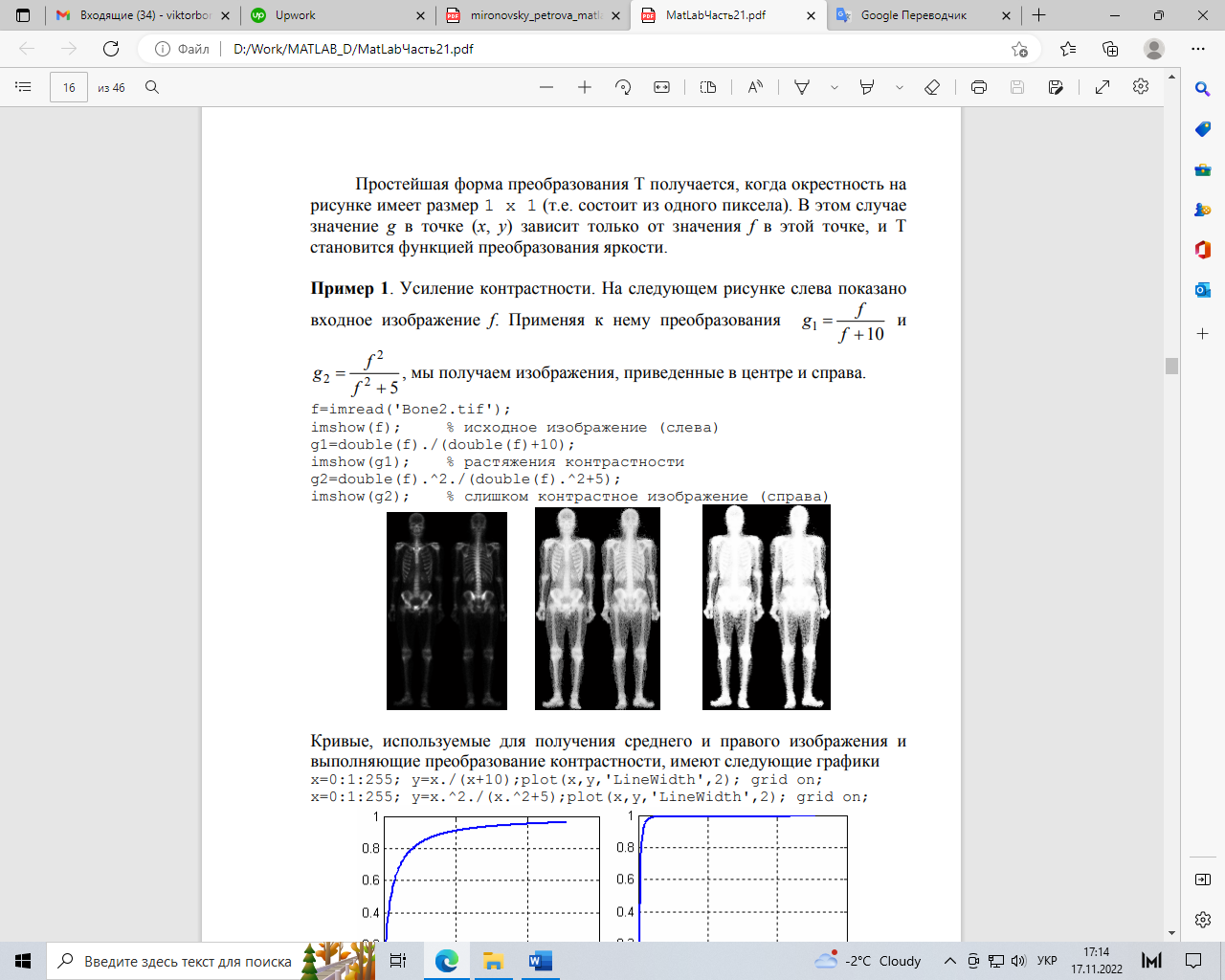
Корисною може бути також функція побудови каркасного зображення поверхні. Один з синтаксисів, що найчастіше використовуються. наступний

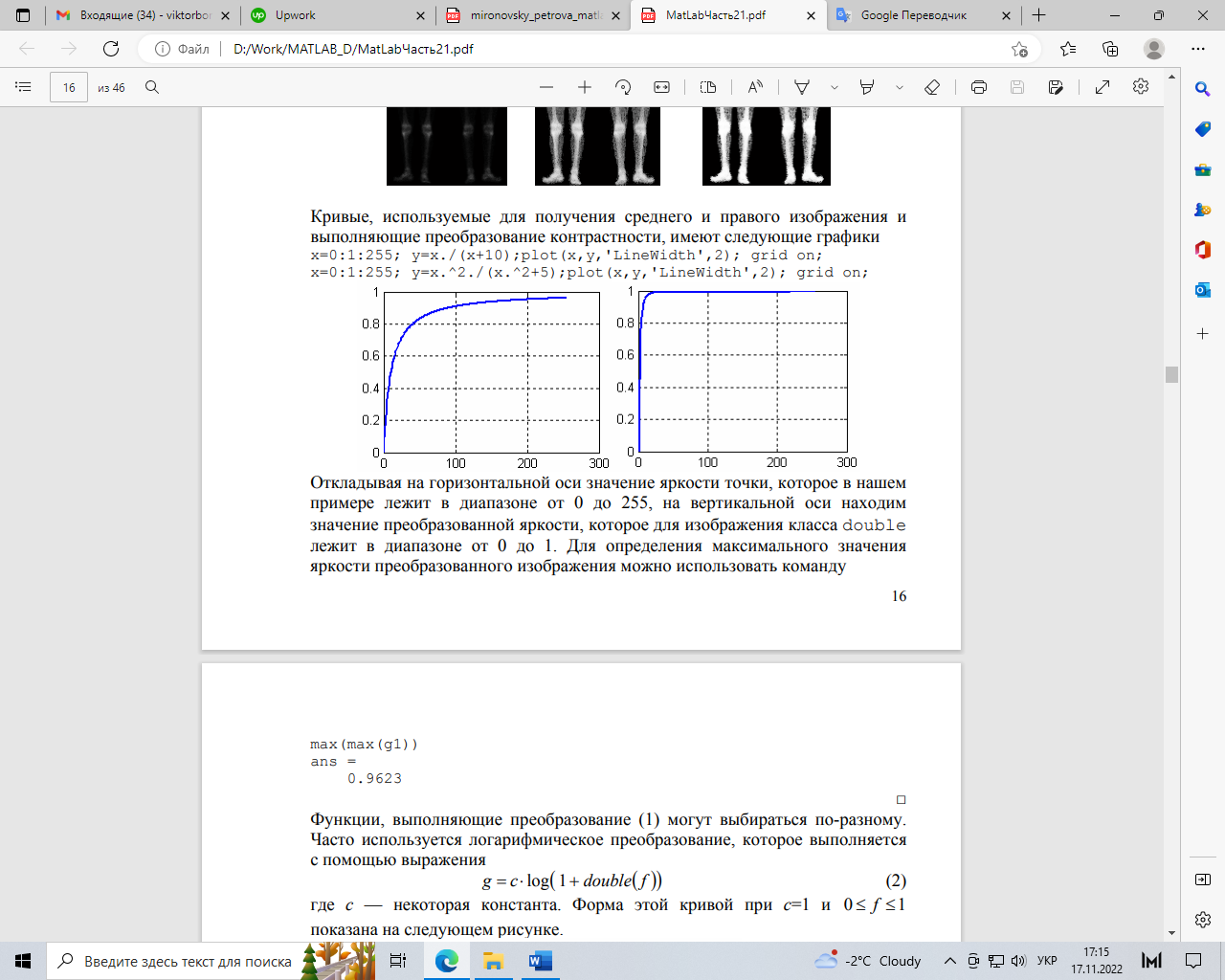
**mesh** (X, Y, Z);

де матриці X, Y, Z мають бути однакового розміру. Графік поверхні будується над прямокутною областю площини XY, що розбивається сіткою з розміром таким самим, як у матриць. У вузлах сітки значення функції рівні відповідним значенням елементів матриці Z. Матриці X та Y містять x та y координати вузлів сітки. У наступному прикладі ми будуємо одиничну півсферу за рівнянням



Приклад 1. Посилення контрастності:





Приклад 2.

f=imread('Spectrum2.tif');

imshow(f); %зліва

size(f)

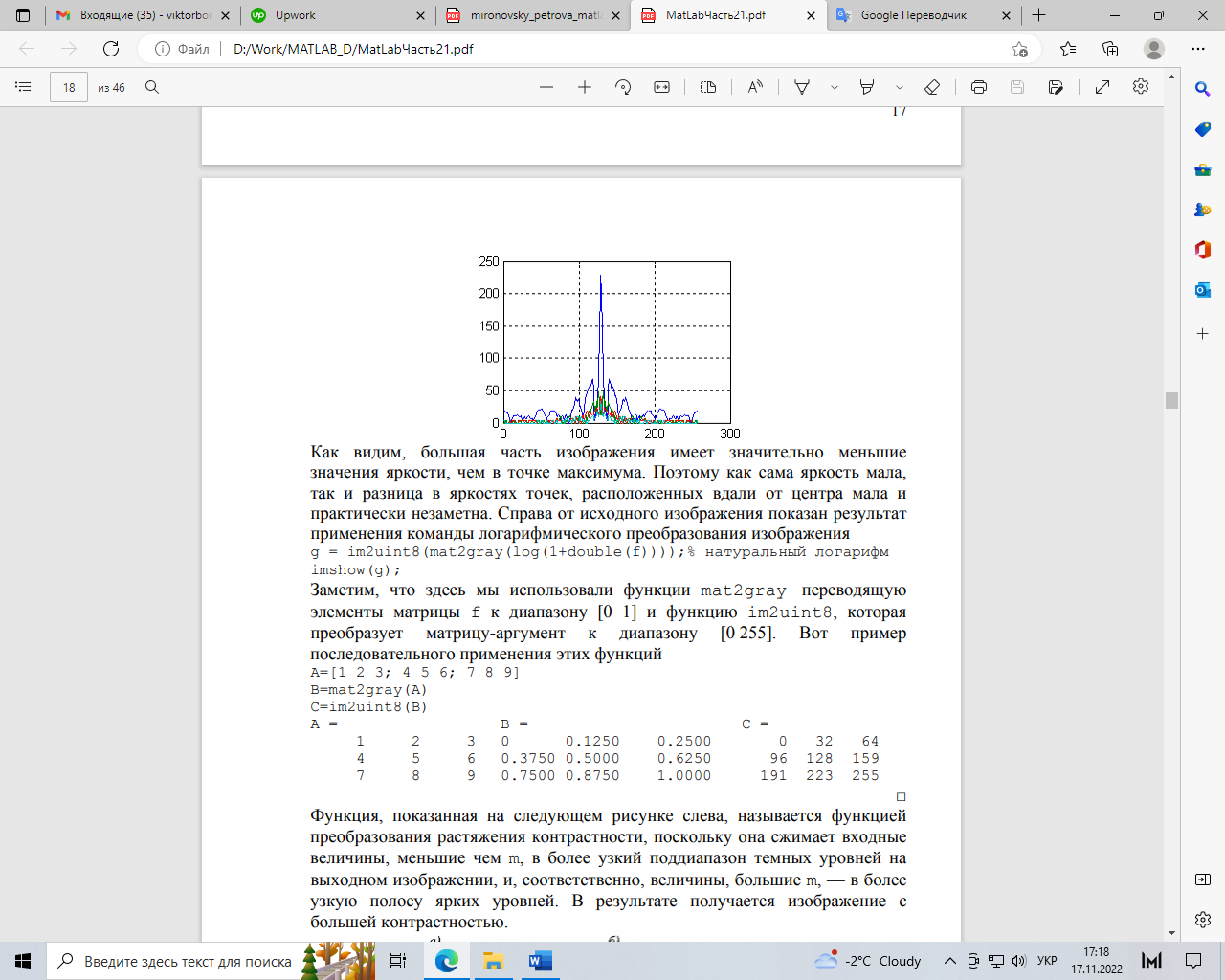
ans = 257 257

max(max(f))

ans = 255

x=[1:1:257];

plot(x,f(128,:),x,f(120,:),x,f(110,:),x,f(100,:));



Як бачимо, більша частина зображення має значно менші значення яскравості, ніж у точці максимуму. Тому як сама яскравість мала, так і різниця в яскравостях точок, розташованих далеко від центру мала і практично непомітна. Праворуч від вихідного зображення показано результат застосування команди логарифмічного перетворення зображення

g = im2uint8(mat2gray(log(1+double(f)))));% натуральний логарифм imshow(g);

Зауважимо, що тут ми використали функції mat2gray, що перекладає елементи матриці f до діапазону [0 1] та функцію im2uint8, яка перетворює матрицю-аргумент до діапазону [0255].

Ось приклад послідовного застосування цих функцій

A=[1 2 3; 4 5 6; 7 8 9]

B=mat2gray(A)

C=im2uint8(B)

A = 1 2 3

4 5 6

7 8 9

B = 0 0.1250 0.2500

0.3750 0.5000 0.6250

0.7500 0.8750 1.0000

C = 0 32 64

96 128 159

191 223 255

Приклад 3:

f=imread('Pollen2.tif');

imshow(f);

g = imcomplement(f);

imshow(g);

function z=polyline2(x,pnt) % рівняння ламаної, з двома зломами в точках (x1,y1) и (x2,y2)

x1=pnt(1);

y1=pnt(2);

x2=pnt(3);

y2=pnt(4); % две точки излома z0=y1/x1.\*x; z2=y2+(255-y2)/(255-x2).\*(x-x2); z=0.5.\*(z0+z2)+0.5.\*(((y2-y1)/(x2-x1)-y1/x1).\*abs(x-x1)+... ((255-y2)/(255-x2)-(y2-y1)/(x2-x1)).\*abs(x-x2));

x=0:1:255;

pnt=[60 30 80 224];

plot(x,polyline2(x,pnt),'LineWidth',2);

grid on;

f=imread('Pollen2.tif');

g=polyline2(f,pnt);

imshow(g);

або

x=0:1:255;

pnt=[30 10 60 224];

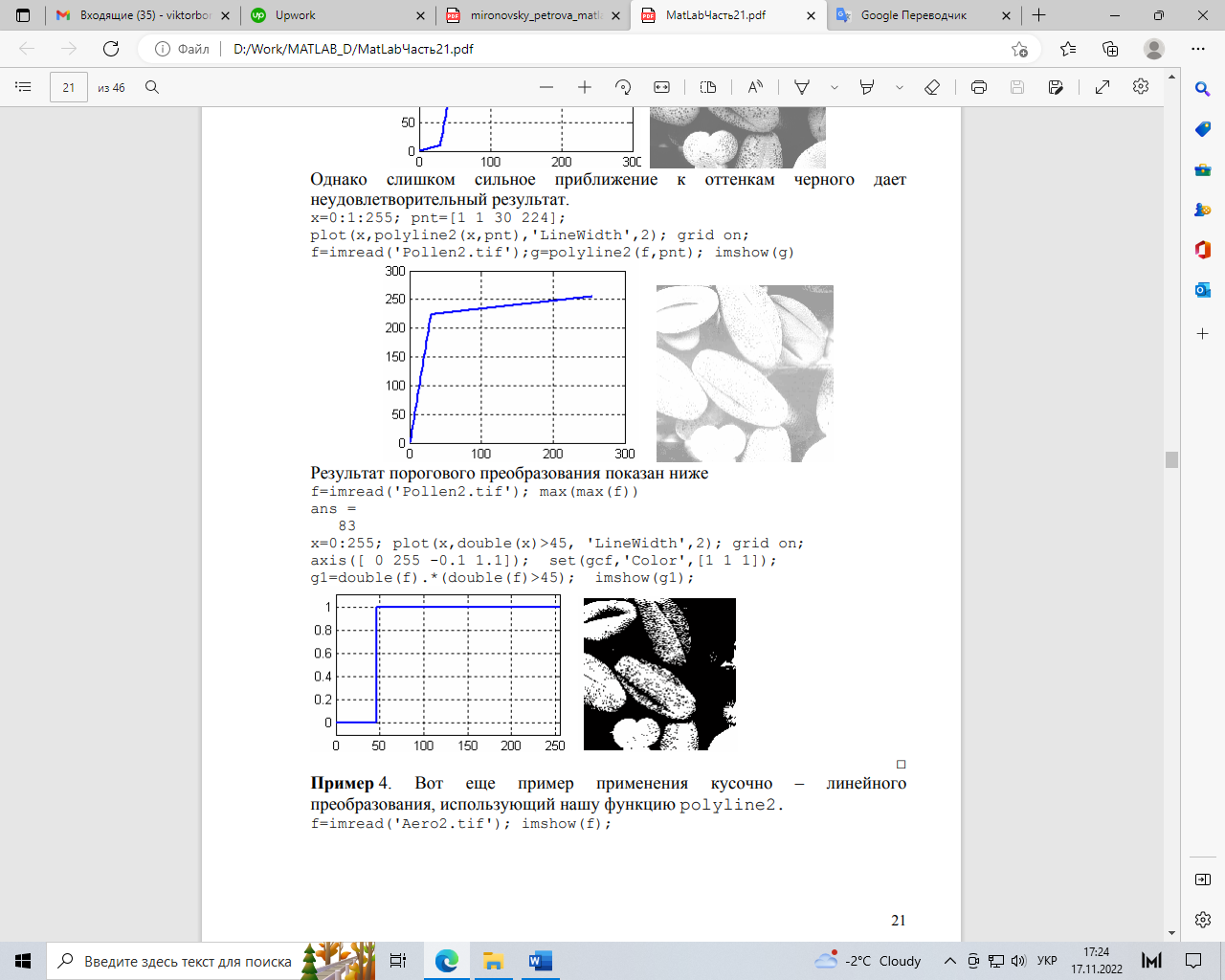
plot(x,polyline2(x,pnt),'LineWidth',2);

grid on;

f=imread('Pollen2.tif');

g=polyline2(f,pnt);

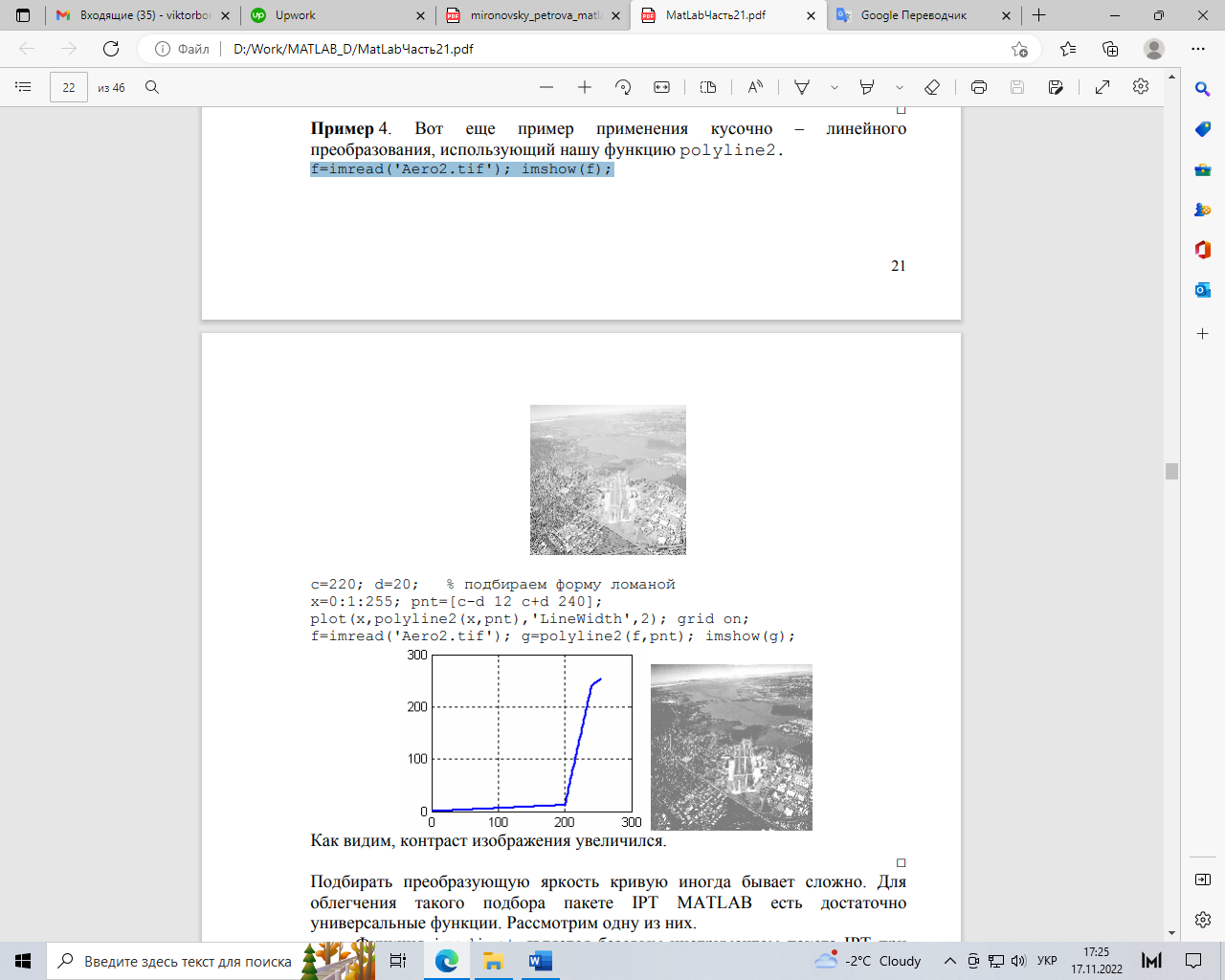
imshow(g);



Приклад 3.

f=imread('Aero2.tif');

imshow(f);



**Гістограми:**

р = **imhist**(f, b)/numel(f),

f=imread('Aero2.tif');

numel(f)

ans = 586752

imhist(f); % гистограма

grid on;

h=imhist(f);

h1 = h(1:10:256);

horz = 1:10:256;

bar (horz, h1);

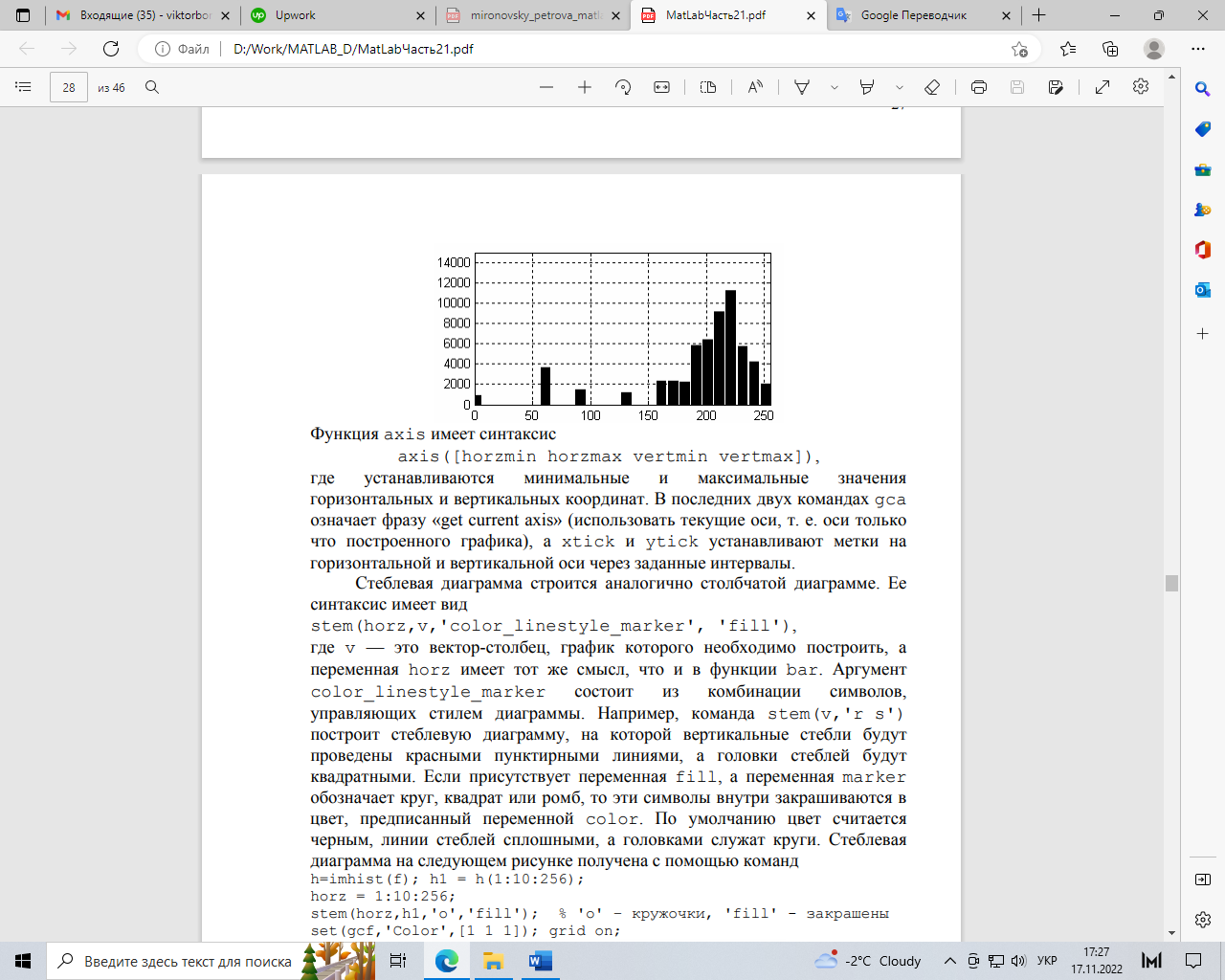
grid on; % построение гистограммы

axis([0 255 0 15000]);

set(gca, 'xtick', 0:50:255)

set(gca, 'ytick', 0:2000:15000)

set(gcf,'Color',[1 1 1]); % білий фон



Стеблова діаграма будується аналогічно стовпчастій діаграмі. Її синтаксис має вигляд

**stem**(horz,v,'color\_linestyle\_marker', 'fill'),

де v - це вектор-стовпець, графік якого необхідно побудувати, а змінна horz має той самий сенс, що і функції bar. Аргумент color\_linestyle\_marker складається з комбінації символів, керуючих стиль діаграми. Наприклад, команда stem(v,'r s') побудує стеблову діаграму, на якій вертикальні стебла будуть проведені червоними пунктирними лініями, а головки стебел будуть квадратними. Якщо є змінна fill, а змінна marker означає коло, квадрат або ромб, то ці символи всередині зафарбовуються в колір, визначений змінною барвисті. За замовчуванням колір вважається чорним, лінії стебел суцільними, а головками служать кола. Стеблова діаграма на наступному малюнку отримана за допомогою команд

h=imhist(f);

h1 = h(1:10:256);

horz = 1:10:256;

stem(horz,h1,'o','fill'); % 'o' – кружочки, 'fill' - закрашены

set(gcf,'Color',[1 1 1]);

grid on;