

# HOLOMAKERS PROJECT

**Motivating secondary school students towards STEM careers through  
hologram making and innovative virtual image processing practices with  
direct links to current research and laboratory practices**

Erasmus+ KA2 2017-1-PL01-KA201-038420

---

## Output 1

### HOLOMAKERS Technical Reference Guide

---

**Lead Partner: WUT**

**Authors: Artur Sobczyk (WUT)**

**Circulation:** *Public*

**Version:** *02*

**Stage:** *Final*

**Date:** *2018*

# Contributions

Karol Kakarenko, Warsaw University of Technology  
Rene Alimisi, EDUMOTIVA

## Declaration

This report has been prepared in the context of the HOLOMAKERS project. Where other published and unpublished source materials have been used, these have been acknowledged.

## Copyright

© Copyright 2017 - 2019 the HOLOMAKERS Consortium  
All rights reserved.



This document is licensed to the public under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License.

## Funding Disclaimer

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Rozdział 1.	Zrozumieć fale .....	6
1.1.	Fale w fizyce .....	6
1.2.	Cechy opisujące fale .....	7
1.2.1.	Amplituda .....	7
1.2.2.	Częstotliwość .....	7
1.2.3.	Okres .....	8
1.2.4.	Długość .....	8
1.2.5.	Faza .....	8
1.2.6.	Inne cechy .....	9
1.3.	Własności fal .....	9
1.3.1.	Dyfrakcja .....	9
1.3.2.	Interferencja .....	10
1.3.1.	Koherencja .....	11
1.4.	Fale w przestrzeni 3D .....	12
1.4.1.	Fala sferyczna i fala płaska .....	12
1.4.2.	Zasada Huygensa .....	13
1.5.	Światło jako fala .....	14
Rozdział 2.	Holografia .....	15
2.1.	Co to jest hologram? .....	15
2.2.	Zasada powstawania hologramu .....	17
2.3.	Rodzaje i własności hologramów .....	19
2.3.1.	Rodzaje hologramów .....	20
2.3.1.	Właściwości hologramów .....	21
2.4.	Podstawowe układy do naświetlania hologramów .....	21
2.4.1.	Hologram Gabora .....	22
2.4.1.	Konfiguracja Leitha-Upatnieksa .....	22
2.4.1.	Hologram tęczowy (Bentona) .....	23
2.4.1.	Hologram objętościowy .....	24
2.4.1.	Hologramy generowane komputerowo .....	25
Rozdział 3.	Podstawy przetwarzania obrazów na w oparciu o program Octave. ....	27
3.1.	Instalacja środowiska Octave .....	27
3.2.	Uruchomienie programu .....	28
3.3.	Okno poleceń .....	29
3.4.	Edytor .....	32
3.5.	Wybrane elementy programowania w środowisku Octave .....	34
3.5.1.	Zmienne .....	34
3.5.2.	Podstawowe operacje wejścia/wyjścia .....	36
3.5.3.	Operatory .....	37
3.5.4.	Instrukcja warunkowa "jeżeli" .....	38
3.5.5.	Pętla "for" .....	41
3.5.6.	Rysowanie wykresów .....	43
3.6.	Przetwarzanie obrazów .....	45
3.6.1.	Tworzenie obrazu .....	45

3.6.2.	Odczyt/zapis obrazu z pliku i wyświetlanie go na ekranie .....	45
3.6.3.	Konwersja kolorów .....	46
3.6.4.	Obrót .....	47
3.6.5.	Dodawanie, odejmowanie, mnożenie, dzielenie. ....	47
3.6.6.	Transformacja Fouriera .....	48
3.7.	Hologramy generowane komputerowo .....	52

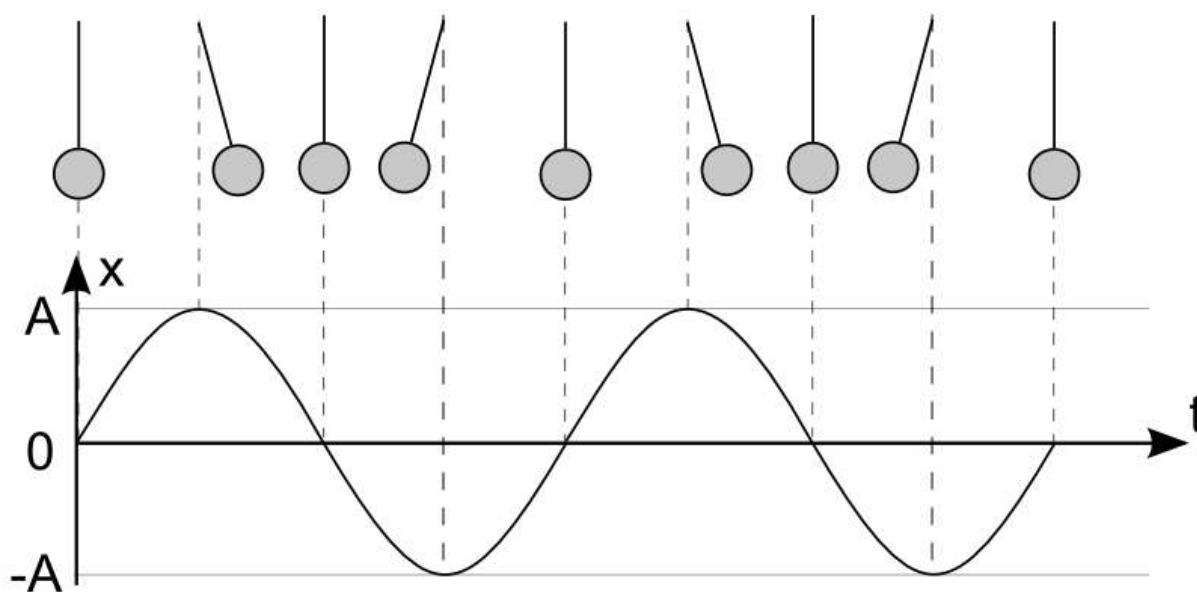


## Rozdział 1. Zrozumieć fale

### 1.1. Fale w fizyce

W fizyce falą nazywamy zaburzenie, które rozchodzi się w ośrodku lub w przestrzeni. Jeśli na przykład wrzucimy kamień do wody, powstanie na powierzchni wody fala rozbiegająca się coraz dalej od miejsca uderzenia kamienia. W tym przypadku energia uderzenia zostanie zamieniona na drgania ośrodka (wody). Innym przykładem może być fala dźwiękowa. W tym przypadku fala również przemieszcza się dzięki drganiom ośrodka (powietrza).

Falą możemy także nazwać drgania ograniczone w przestrzeni. Na przykład poruszające się wahadło wykonuje ruch drgający powracając od czasu do czasu do tego samego punktu. W tym wypadku falą będzie zmieniające się w czasie wychylenie wahadła. Rys. 1 obrazuje zmianę wychylenia wahadła w czasie. Wahadło odchyła się w zakresie  $[-A, A]$ , możemy więc powiedzieć, że  $A$  jest amplitudą wychylenia.



Rys. 1 Wychylenie wahadła w różnych momentach czasu

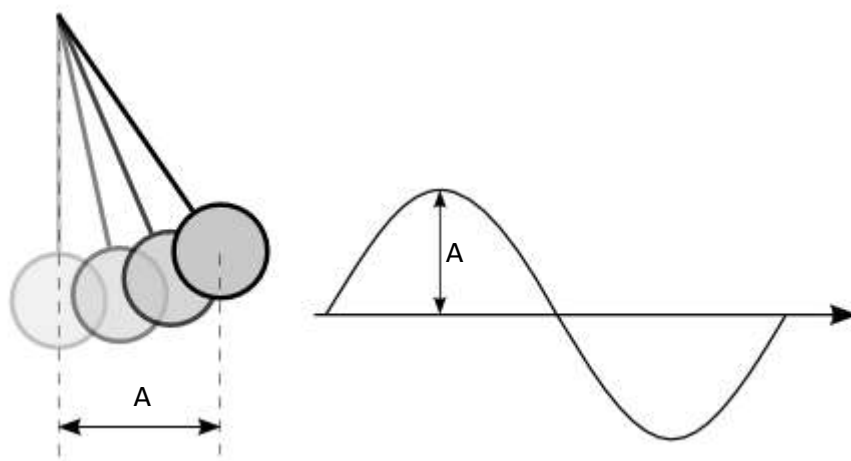
Bardzo często w fizyce drgania można opisać za pomocą funkcji sinus. Ma to miejsce we wspomnianym już wahadle ale także również w przypadku ciężarka zawieszonego na sprężynie, lub w przypadku struny instrumentu muzycznego. Tego typu drgania nazywa się drganiami **harmonicznymi**. Jest to podstawowy rodzaj drgań. Jak się potem dowiemy, każdy inny, bardziej skomplikowany rodzaj drgań składa się właśnie z drgań harmonicznym.

Reasumując, fala harmoniczna to taka, którą możemy opisać za pomocą funkcji sinus. W następnym rozdziale poznamy podstawowe własności służące do opisu fal.

## 1.2. Cechy opisujące fale

### 1.2.1. Amplituda

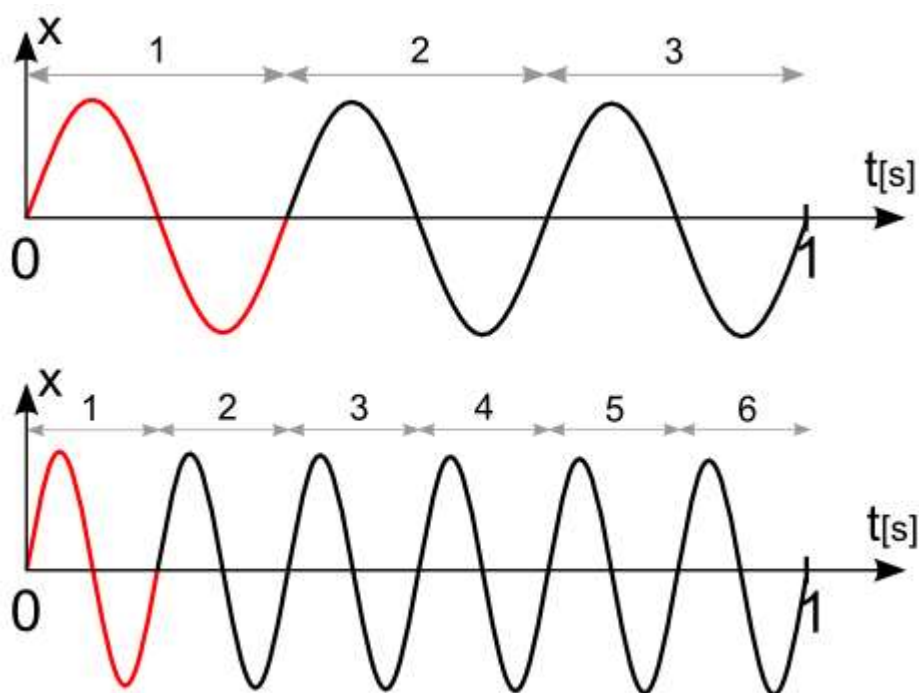
Amplituda fali jest to odległość największego wychylenia od położenia równowagi. Tak więc w przypadku wahadła, amplitudą drgań nazwiemy jego największe wychylenie (patrz Rys. 2). Maksymalna wartość fali nazywa się jej grzbietem a wartość minimalna - doliną.



Rys. 2 Amplituda drgań wahadła

### 1.2.2. Częstotliwość

Częstotliwość mówi nam ile drgań zostało wykonanych w jednostce czasu (w ciągu jednej sekundy). Jednostką częstotliwości jest Hz (Hertz). Np. 10Hz to 10 drgań na sekundę, 15,5Hz to 15,5 drgania w ciągu sekundy. Rys. 3 przedstawia dwie fale o różnych częstotliwościach. Czerwoną linią zaznaczono jedno pełne drganie. Częstotliwość pierwszej fali to 3Hz (3 drgania w ciągu sekundy) a drugiej 6Hz (6 drgań w ciągu sekundy).



Rys. 3 Częstotliwość fal

### 1.2.3. Okres

Okres fali jest bezpośrednio związany z jej częstotliwością. Jest to czas (mierzony w sekundach), w jakim zostanie wykonane jedno pełne drganie (zaznaczone czerwonym kolorem na Rys. 3). Okres fali  $T$  jest związany z częstotliwością  $f$  zależnością  $T = \frac{1}{f}$ . Tak więc okres pierwszej fali na Rys. 3 wynosi  $\frac{1}{3}s$  podczas gdy drugiej fali wynosi  $\frac{1}{6}s$

### 1.2.4. Długość

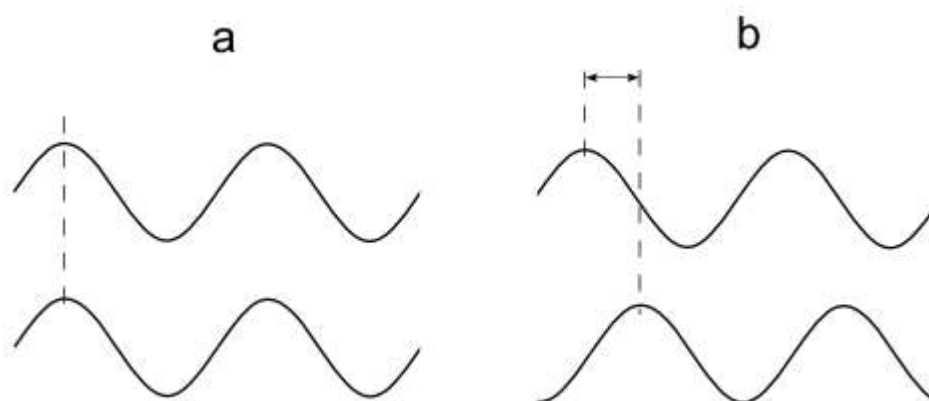
Jeśli fala propaguje się w przestrzeni to można wyznaczyć jej długość. Jest to dystans jaki pokona fala w czasie jednego okresu. Długość fali  $\lambda$  dana jest zależnością  $\lambda = vT$ , gdzie  $v$  jest prędkością propagacji fali a  $T$  jest jej okresem.

### 1.2.5. Faza

Faza określa, w której części okresu fali znajduje się dany punkt fali. W praktyce jednak istotnym elementem jest nie tyle faza pojedynczej fali co **różnica faz** między falami. Mówiąc obrazowo jest to po prostu przesunięcie jednej fali względem drugiej. Rys. 4a



przedstawia sytuację gdzie nie występuje przesunięcie fazowe natomiast Rys. 4b przedstawia fale cechujące się pewną różnicą faz. Różnicę faz można mierzyć między dowolnymi odpowiadającymi sobie punktami fali. Np. na Rys. 4 dla wygody różnicę faz przedstawiono jako odległość między maksimami fali.



**Rys. 4 Przesunięcie fazowe**  
 a) brak przesunięcia fazowego między falami  
 b) fale posiadające różnicę faz

### 1.2.6. Inne cechy

Oprócz wspomnianych powyżej cech możemy wyróżnić także prędkość grupową i prędkość fazową fali. Fale mogą być podłużne lub poprzeczne. W przypadku fal poprzecznych możemy także mówić o polaryzacji. Cechy te nie są konieczne do zrozumienia zagadnień związanych z holografią toteż zostaną one pominięte w niniejszym przewodniku.

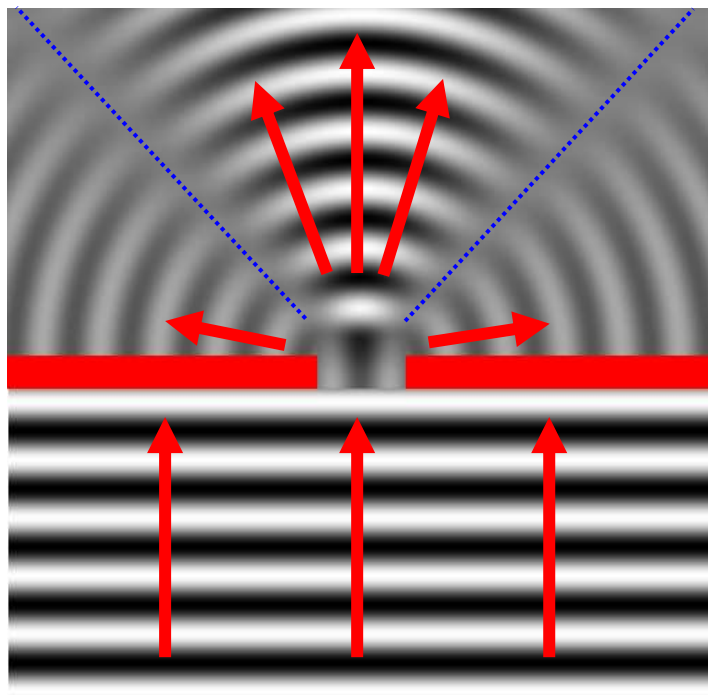
## 1.3. Własności fal

Fale propagujące się w przestrzeni (czyli np. fala na wodzie, fala dźwiękowa, fala elektromagnetyczna) wykazują pewne właściwości takie jak odbicie, załamanie, dyfrakcja czy interferencja.

### 1.3.1. Dyfrakcja

Dyfrakcja polega na zmianie kierunku rozchodzenia się fali na skutek napotkania przeszkody lub szczeliny. Efekt ten jest szczególnie widoczny gdy długość fali jest porównywalna z wielkością szczeliny. Rys. 5 przedstawia sytuację, w której fala napotyka na szczelinę. Przed szczeliną fala porusza się w kierunku prostopadłym do szczeliny (zaznaczonym czerwonymi strzałkami). Po przejściu przez szczelinę, fala propaguje się w różnych kierunkach z różną amplitudą co również symbolicznie zaznaczono czerwonymi

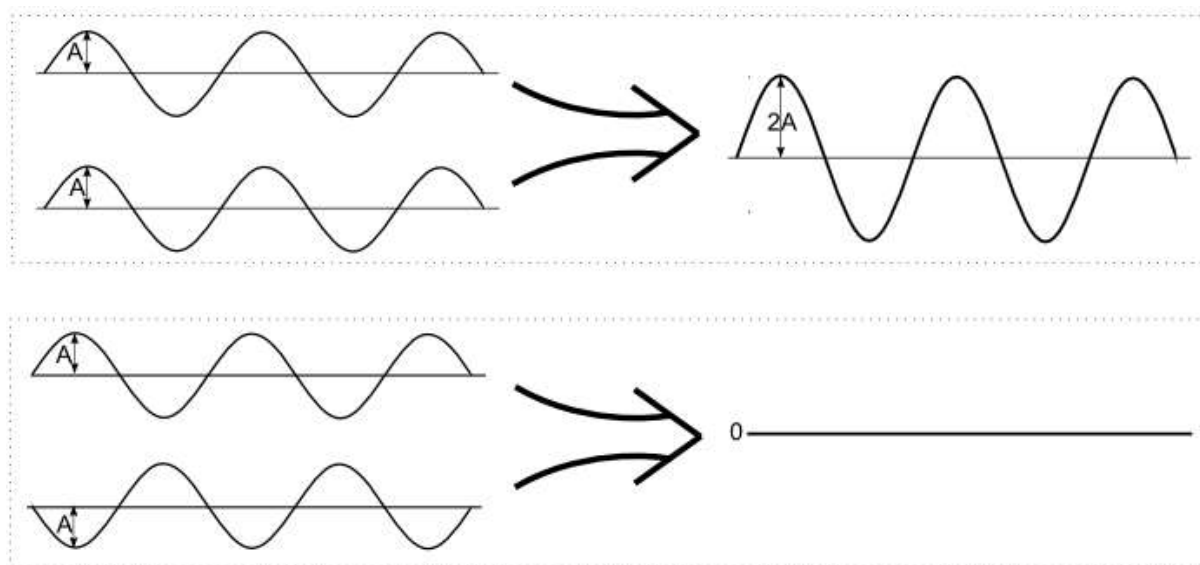
strzałkami. Są także obszary całkowitego wygaszenia fali (zaznaczone symbolicznie niebieską linią). Wzdłuż tych linii amplituda fali wynosi 0.



Rys. 5 Dyfrakcja fali na pojedynczej szczelinie.

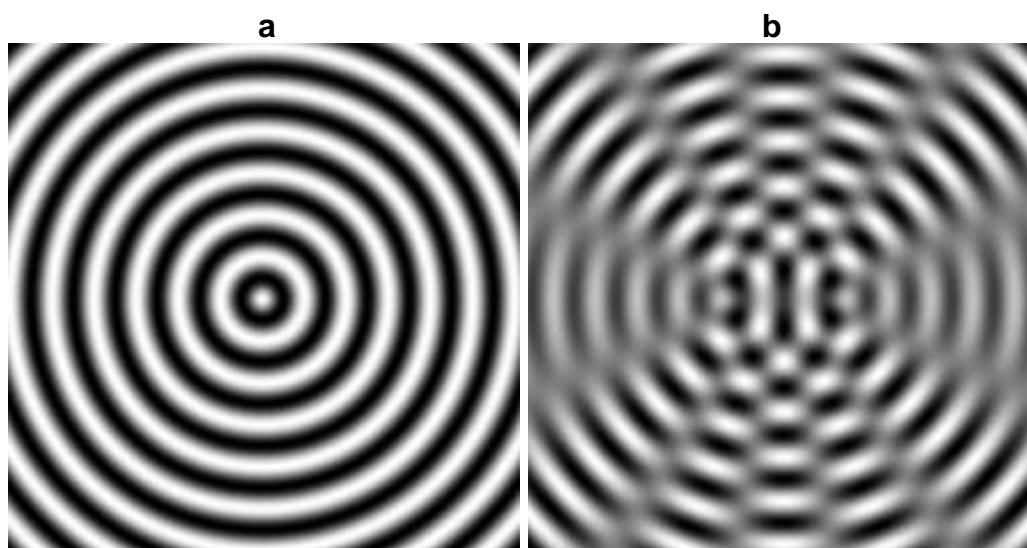
### 1.3.2. Interferencja

Interferencja fal jest właśnie zjawiskiem, na którym opiera się holografia. Interferencja to nic innego jak nakładanie (sumowanie) się fal. Kiedy biegnące fale spotkają się ze sobą zaczynają ze sobą oddziaływać, mogą się wzajemnie wzmacniać lub osłabiać. Rys. 6 przedstawia wzmocnienie i wygaszenie się fal. Jeśli fale są zgodne w fazie (różnica faz wynosi 0) to następuje największe możliwe wzmocnienie fal. Jeśli natomiast fale są przeciwne w fazie (różnica faz równa jest połowie okresu fali) to następuje całkowite wygaszenie fal. Kiedy fale się wzmacniają to mówimy o konstruktywnej interferencji, w przeciwnym wypadku jest to interferencja destruktywna.



**Rys. 6 Interferencja fal, wzmocnienie i wygaszenie**

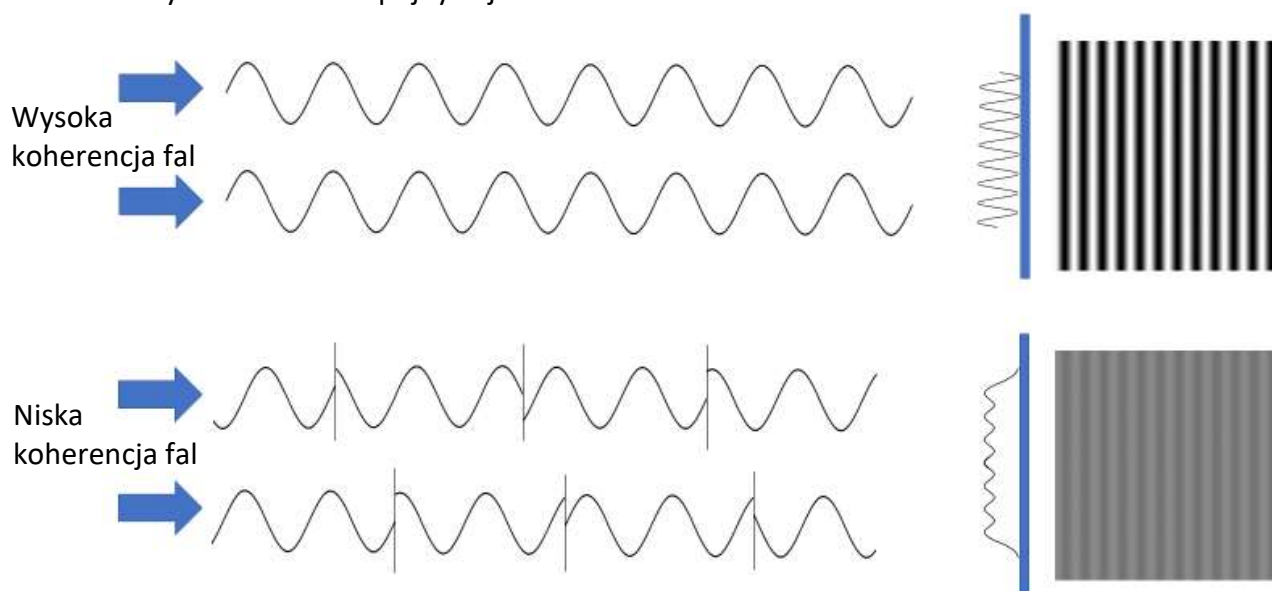
Interferencję fal można zilustrować na przykładzie fal na wodzie. Jeśli będziemy pobudzać taflę wody w jednym punkcie ze stałą częstotliwością to otrzymamy falę rozbiegającą się promieniście. Jest to pokazane na Rys. 7a. Czarne obszary reprezentują doliny fali a białe jej grzbiety. Rys. 7b przedstawia dwie fale powstające obok siebie i oddziałujące wzajemnie. Obszary szare to wygaszenia fali obszary nie rozmazane to wzmocnienia fali.



**Rys. 7 Przykład fal na wodzie**

### 1.3.1. Koherencja

Koherencja (spójność) fal jest konieczna aby otrzymać stały w czasie obraz interferencyjny. Dwie fale są ze sobą spójne jeśli posiadają stałą różnicę faz. Na Rys. 8. pokazano jak w uproszczony sposób można sobie wyobrazić różnicę pomiędzy falami spójnymi oraz niespójnymi. Górna część rysunku przedstawia fale wysoce koherentne (spójne). W wyniku nakładania się tych fal powstają stałe w czasie prążki interferencyjne. Widzimy zatem obraz interferencyjny o dobrym kontraście (prążki są dobrze widoczne). W przypadku gdy fale posiadają niski stopień koherencji, powstające prążki w każdej chwili czasu są nieco przesunięte względem siebie. Nasze oczy rejestrują to jako rozmazany obraz prążków (o obniżonym kontraście). Im mniejsza koherencja fal tym obraz interferencyjny jest bardziej rozmazany. W przypadku fal całkowicie niespójnych, nie zaobserwujemy prążków interferencyjnych. Większość fal świetlnych w przyrodzie to fale niekoherentne. Natomiast dobrym źródłem fal spójnych jest laser.

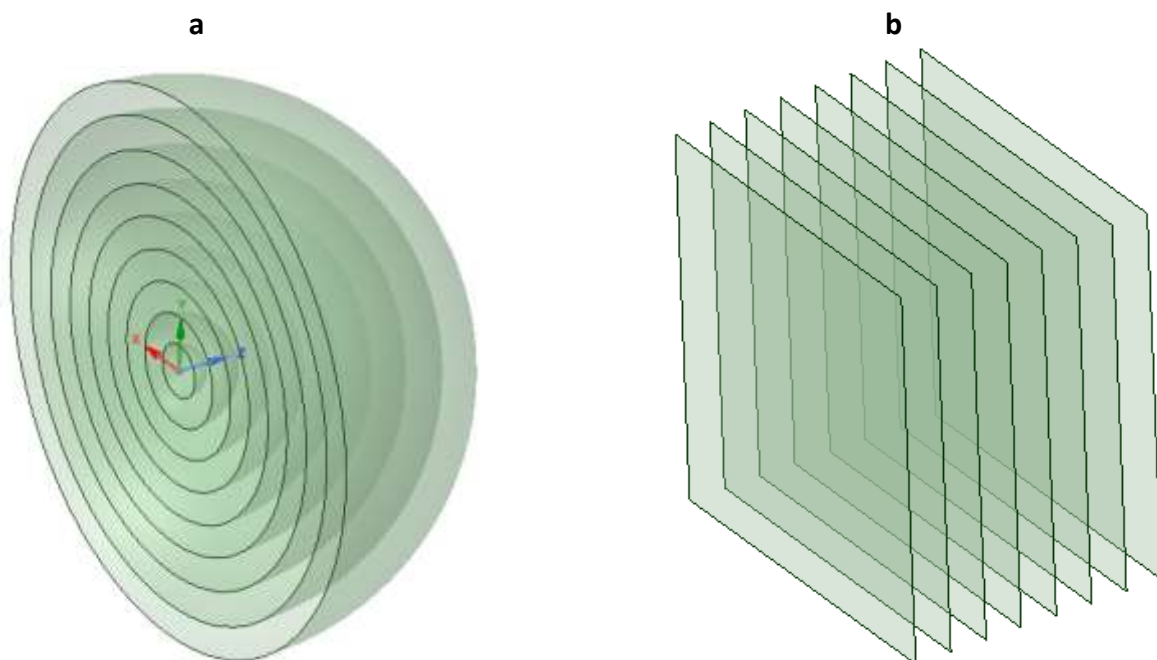


Rys. 8 Wysoka koherencja fal jest warunkiem uzyskania obrazu interferencyjnego o wysokim kontraście.

## 1.4. Fale w przestrzeni 3D

### 1.4.1. Fala sferyczna i fala płaska

Fala w przestrzeni 3D może być przedstawiona w postaci powierzchni stałej fazy. Najczęściej spotykanym przykładem jest fala sferyczna lub tzw. fala płaska (Rys. 9). W pierwszym przypadku (Rys. 9a) powierzchnie stałej fazy (czyli powierzchnie, w których fala ma tę samą wartość np. osiąga wartość maksymalną) mają kształt sfer (stąd nazwa fala sferyczna). W drugim przypadku (Rys. 9b) powierzchnie stałej fazy mają kształt płaszczyzn (są płaskie i stąd nazwa fala płaska).

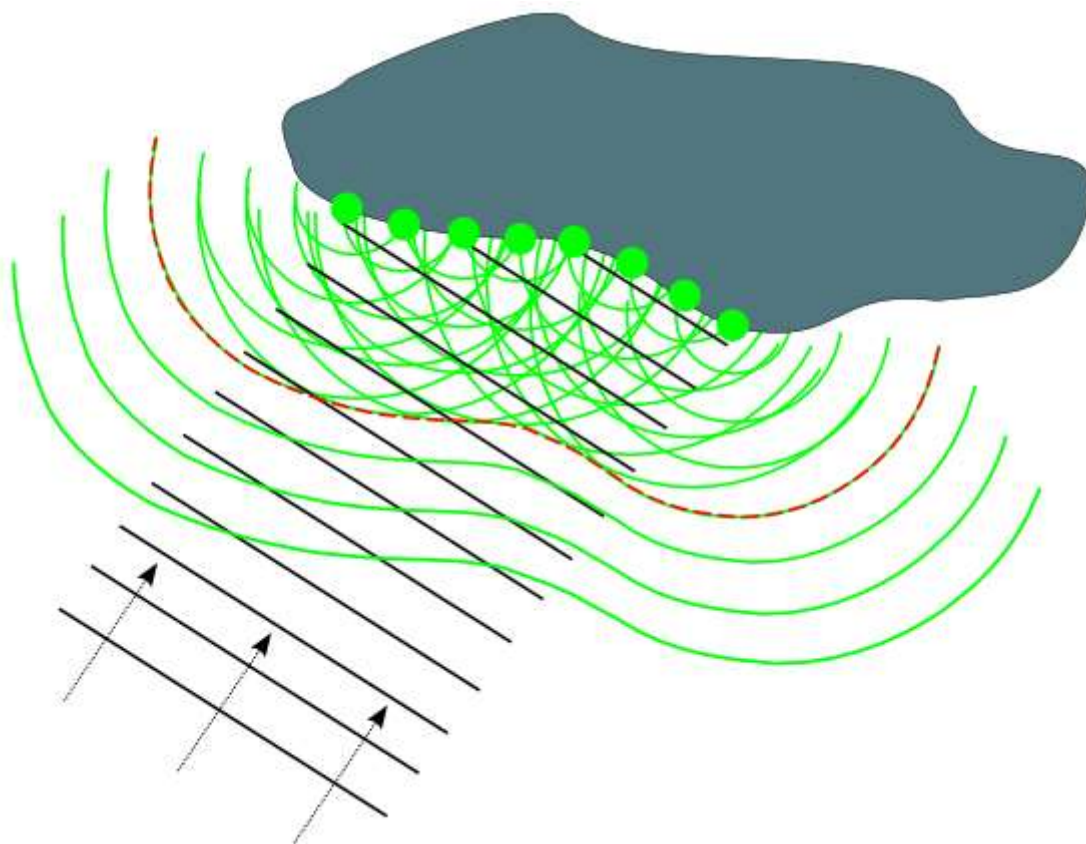


Rys. 9 Fala sferyczna (a) oraz fala płaska (b)

### 1.4.2. Zasada Huygensa

Rozważmy przypadek, gdy fala dociera do jakiegoś obiektu (Rys. 10). Można przyjąć, że każdy punkt, do którego dotarła fala staje się źródłem nowej fali sferycznej. Suma wszystkich tych fal sferycznych (czyli interferencja) wyznacza czoło nowej fali. Na Rys. 10 czarnym kolorem zaznaczono falę padającą. Obiekt, do którego dociera fala można przedstawić jako zbiór nieskończenie wielu punktów przedstawionych na rysunku jako zielone kropki. Każdy z tych punktów jest źródłem wtórnej fali sferycznej. Po zsumowaniu wszystkich fal wtórnych otrzymujemy nową falę (czerwona przerywana linia na rysunku) która następnie propaguje się w przestrzeni. Jest to właśnie zasada Huygensa, która brzmi następująco:

*Każdy punkt ośrodka, do którego dotarło czoło fali, można uważać za źródło nowej fali sferycznej. Wypadkową powierzchnię falową tworzy powierzchnia styczna do wszystkich powierzchni fal cząstkowych i ją właśnie obserwuje się w ośrodku.*

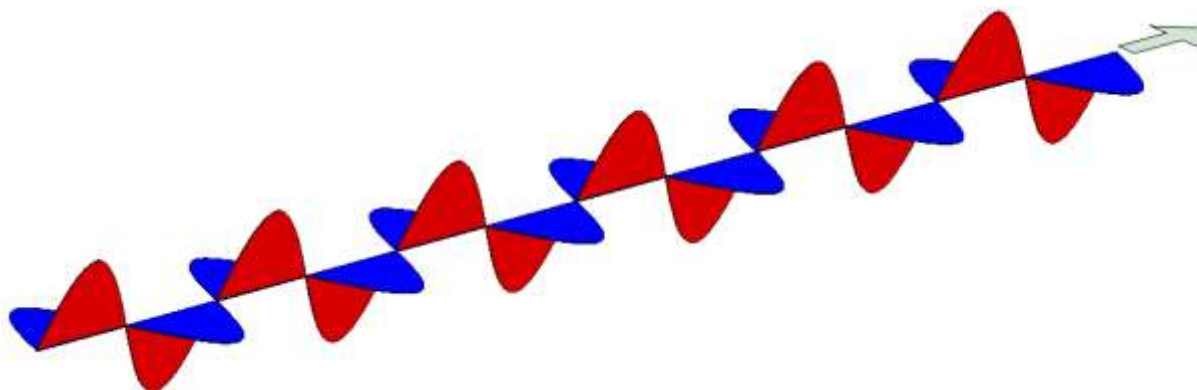


Rys. 10 Ilustracja zasady Huygensa

## 1.5. Światło jako fala

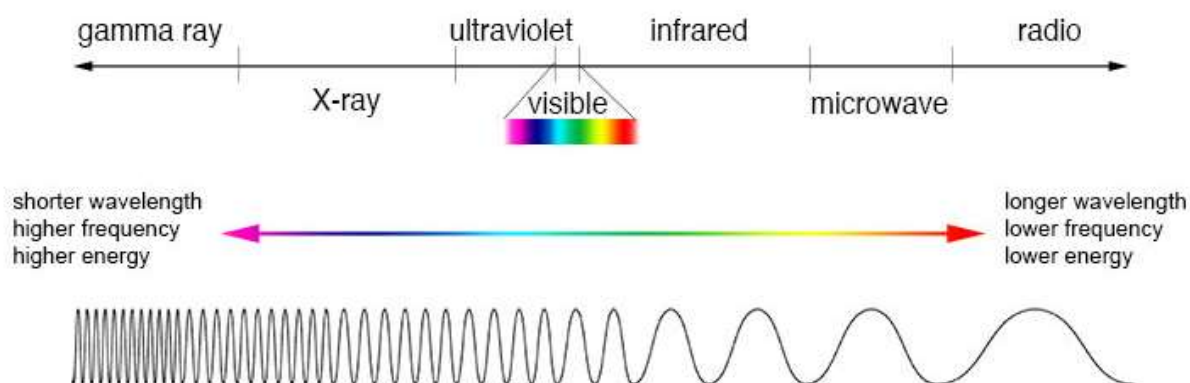
Okazuje się, że światło również jest falą. Jeśli tak, to co tak naprawdę tam faliuje? Tym co faliuje jest pole elektryczne i pole magnetyczne. Stąd mówi się, że światło jest falą elektromagnetyczną (EM). Falę taką można sobie wyobrazić jako zwiększanie się oraz zmniejszanie pola elektrycznego i magnetycznego, które propaguje się w przestrzeni (Rys. 11). Kolorem czerwonym umownie zaznaczono pole elektryczne a kolorem niebieskim - pole magnetyczne.





Rys. 11 Fala elektromagnetyczna

W przyrodzie istnieje wiele rodzajów promieniowania elektromagnetycznego. Klasyfikuje się je ze względu na szybkość zmian pola elektromagnetycznego w przestrzeni (czyli ze względu na częstotliwość fal). Fale EM o największej częstotliwości to promieniowanie gamma, następnie mamy promieniowanie X, ultrafiolet, światło widzialne, podczerwień, mikrofae oraz fale radiowe jako promieniowanie EM o najmniejszej częstotliwości (Rys. 12). Cały zakres fal od promieniowania gamma do fal radiowych nazywa się widmem fal elektromagnetycznych. Jak widać na Rys. 12, światło zajmuje tylko wąski fragment całego widma fal EM.

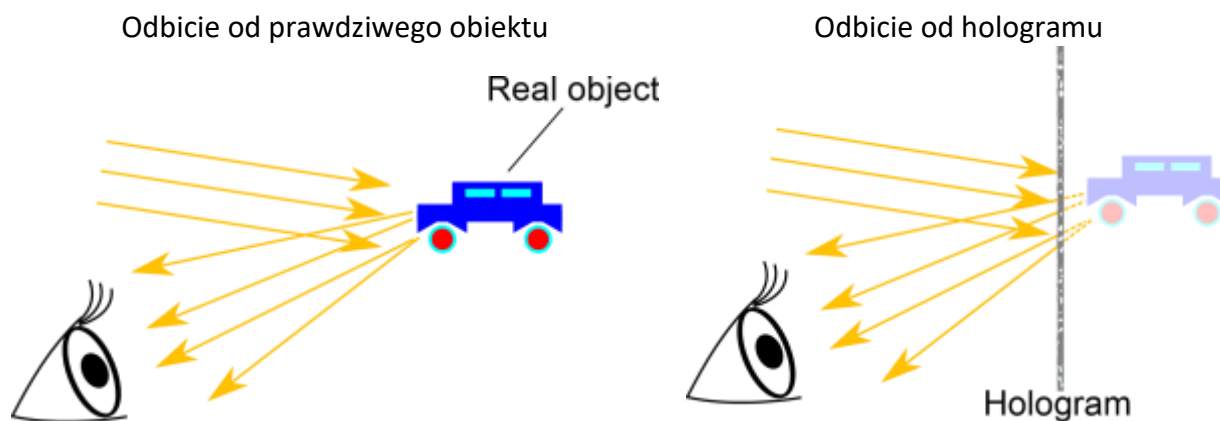


Rys. 12 Widmo fal elektromagnetycznych. Źródło: NASA's Imagine the Universe  
(<https://imagine.gsfc.nasa.gov/science/toolbox/emspectrum1.html>)

## Rozdział 2. Holografia

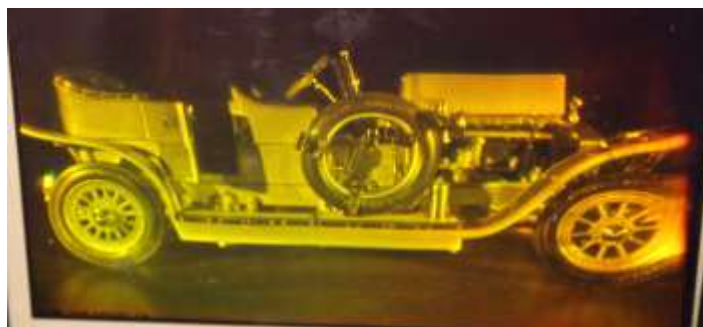
### 2.1. Co to jest hologram?

Hologram jest to klisza lub szklana płytka, na której powierzchni lub w jej objętości zapisany jest wzór interferencyjny. Światło przechodzące lub odbite od hologramu tworzy obraz identyczny jak światło odbite od rzeczywistego obiektu (Rys. 13).



Rys. 13 Światło odbite od hologramu zachowuje się identycznie jakby pochodziło od prawdziwego obiektu

Cechą charakterystyczną hologramu jest to, że obserwowany obiekt jest w pełni trójwymiarowy. Oznacza to, że patrząc na hologram pod pewnym kątem, możemy dostrzec szczegóły niewidoczne podczas patrzenia na ten sam hologram pod innym kątem (Rys. 14). Takiej właściwości nie posiadają zwykłe zdjęcia, gdyż wyglądają one tak samo niezależnie od kąta obserwacji.



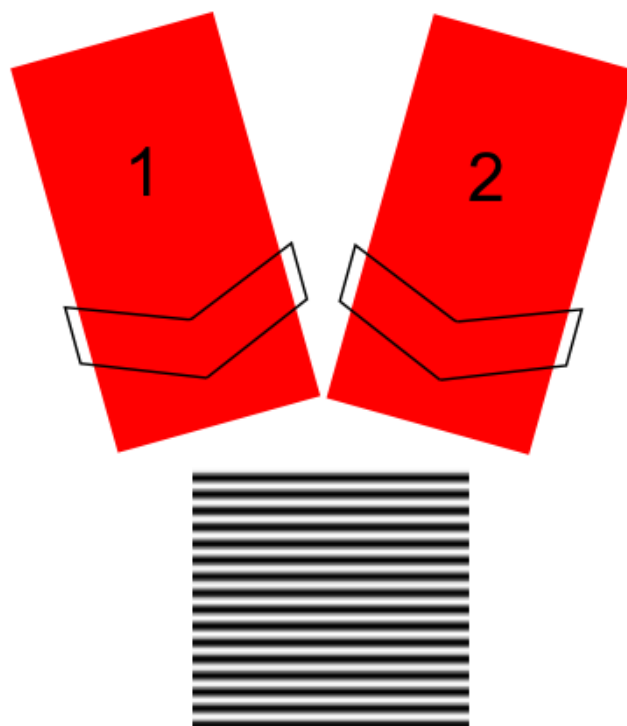




Rys. 14 Obraz widoczny na hologramie zmienia się wraz z kątem obserwacji

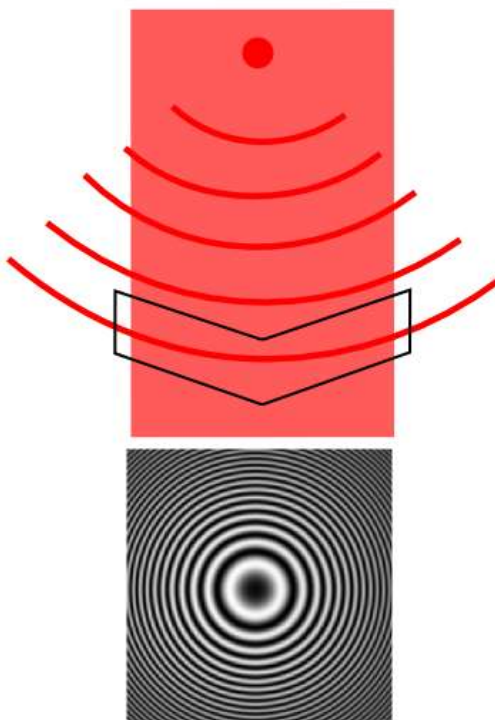
## 2.2. Zasada powstawania hologramu

Zjawisko fizyczne, dzięki któremu powstaje hologram to interferencja. Wiemy już, że dwie koherentne fale interferują ze sobą tworząc prążki interferencyjne. Jako prosty przykład można podać dwie fale płaskie, które interferując ze sobą tworzą prostoliniowe prążki (Rys. 15).



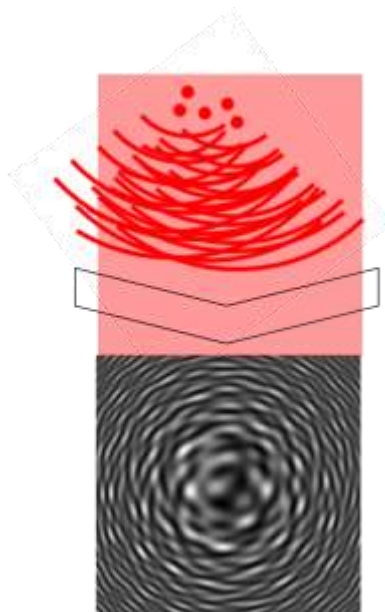
Rys. 15 Interferencja dwóch fal płaskich.

Nieco inaczej wygląda interferencja fali płaskiej z punktowym źródłem światła (Rys. 16). Prążki interferencyjne mają kształt okręgów. Możemy powiedzieć, że taki rozkład interferencyjny zawiera w sobie informację o punkcie w przestrzeni. Jeśli uda się nam utrwalić ten rozkład np. na kliszy holograficznej to po oświetleniu jej falą płaską część przechodzącego światła uformuje w przestrzeni zakodowany punkt.



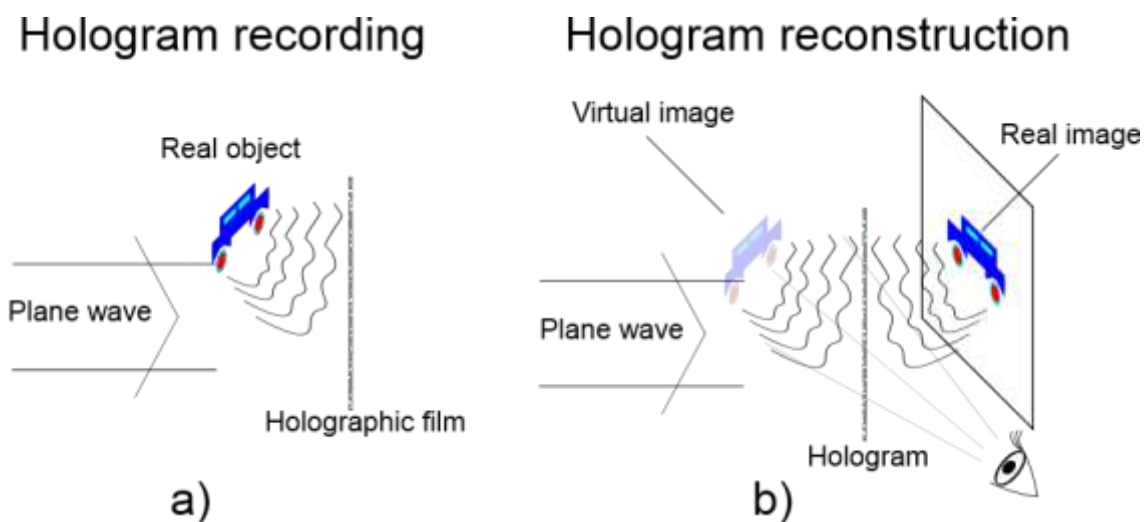
**Rys. 16 Interferencja źródła punkowego z falą płaską**

Dowolny oświetlony obiekt można traktować jako zbiór wielu wtórnych źródeł punktowych (zasada Huygensa). Obraz interferencyjny powstały w wyniku interferencji wielu źródeł punktowych z falą płaską jest w ogólności skomplikowany (patrz Rys. 17). Możemy powiedzieć, że obiekt został zakodowany w postaci obrazu interferencyjnego.



Rys. 17 Interferencja wielu źródeł punktowych z falą płaską

Taki zakodowany obiekt to właśnie hologram. Hologram ten można odtworzyć oświetlając go falą płaską. Rys. 18 ilustruje przykład rejestracji oraz rekonstrukcji hologramu. W tym przypadku podczas odtworzenia hologramu uzyskujemy obraz rzeczywisty (widoczny na ekranie) oraz obraz pozorny (widoczny gdy patrzymy bezpośrednio na oświetlony hologram).

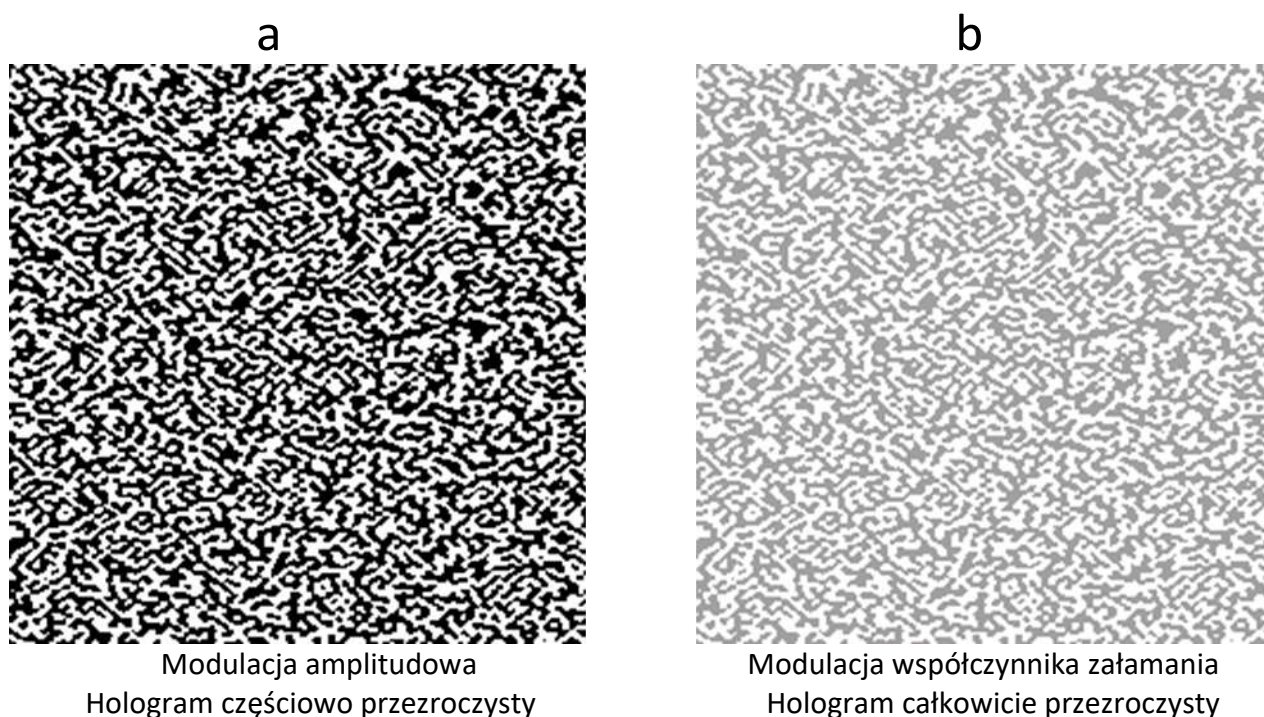


Rys. 18 Rejestracja hologramu (a) oraz jego rekonstrukcja (b)

### 2.3. Rodzaje i własności hologramów

### 2.3.1. Rodzaje hologramów

Hologramy można sklasyfikować w różny sposób w oparciu o ich właściwości. Jednym z podziałów może sposób, w jaki światło przechodzi przez hologram. Hologram może być modulowany amplitudowo lub fazowo. Modulacja amplitudowa polega na tym, że wzór interferencyjny zapisany jest w postaci mniej lub bardziej przezroczystych obszarów (Rys. 19a). Hologram modulowany fazowo jest całkowicie przezroczysty a wzór interferencyjny zapisany jest w postaci obszarów o różnym współczynniku załamania światła (Rys. 19b).



Rys. 19 Hologram modulowany amplitudowo (a) oraz fazowo (b)

Hologramy można także podzielić na hologramy tzw. cienkie i grube. Hologramy cienkie to takie, które zachowują się tak jakby wzór interferencyjny został zapisany tylko na powierzchni kliszy (Rys. 20a). Hologramy grube to takie, w których obraz interferencyjny zapisany jest w całej objętości hologramu (Rys. 20b).



Rys. 20 Hologram „cienki” (a) oraz „gruby” (b)

Kolejny sposób podziału hologramów to hologramy transmisyjne i odbiciowe (Rys. 21). Jak nazwa wskazuje, hologram transmisyjny możemy zaobserwować przepuszczając światło przez hologram natomiast hologram odbiciowy obserwujemy odbijając światło od hologramu.



Rys. 21 Hologram transmisyjny (a) oraz odbiciowy (b)

### 2.3.1. Właściwości hologramów

Holografia jest metodą fotograficzną – na kliszy holograficznej rejestrowana jest informacja o obrazie. Pomimo to różnice pomiędzy hologramem a zwykłym zdjęciem są zasadnicze:

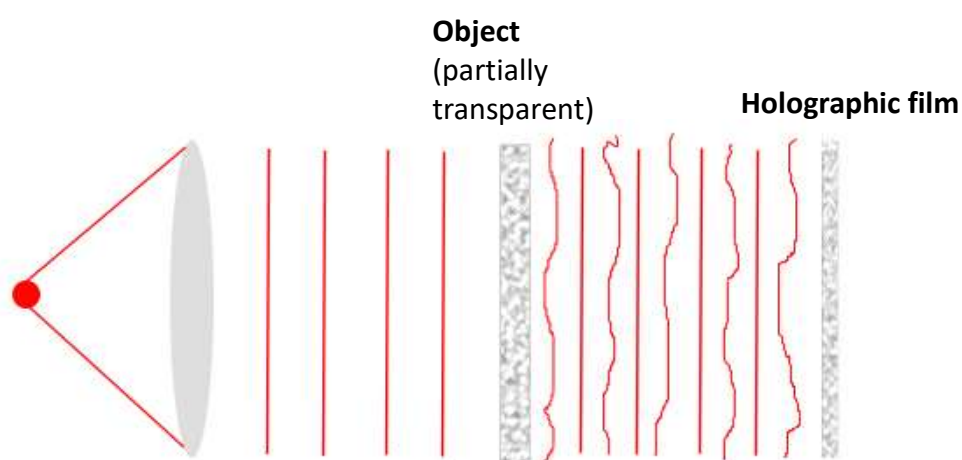
- W przypadku holografii, klisza musi posiadać o wiele większą rozdzielczość (mierzoną w liniach na milimetr).
- Negatyw fotograficzny zawiera informacje tylko o natężeniu światła natomiast w hologramie pod postacią wzoru interferencyjnego zakodowana jest informacja zarówno o natężeniu światła jak i o fazie (czyli odległości punktów obiektu od hologramu)
- Każdy fragment negatywu fotograficznego zawiera informację o innym fragmencie obrazu, podczas gdy w przypadku hologramu, każdy jego kawałek zawiera informację o całym obrazie. Oznacza to, że jeśli przetniemy hologram na dwie części to każda z nich odtworzy cały obraz. Będzie natomiast różnica w jasności obrazu oraz zmniejszy się zakres kątowy, pod jakim możemy obserwować hologram.

## 2.4. Podstawowe układy do naświetlania hologramów



### 2.4.1. Hologram Gabora

Historycznie, pierwszy układ do naświetlania hologramów został zaproponowany przez Denisa Gabora w 1948r. W tamtym czasie nie wynaleziono jeszcze laserów. Układ ten wykorzystuje punktowe źródło światła dzięki czemu jest ono w pewnym stopniu koherentne i możliwy jest zapis hologramu. Obiektem jest częściowo przezroczysta klisza (np. napis lub prosty obrazek). Światło częściowo przechodzi przez kliszę niezakłócone a częściowo ulega rozproszeniu. Obie te wiązki ze sobą interferują zapisując się na kliszy w postaci rozkładu interferencyjnego (Rys. 22)

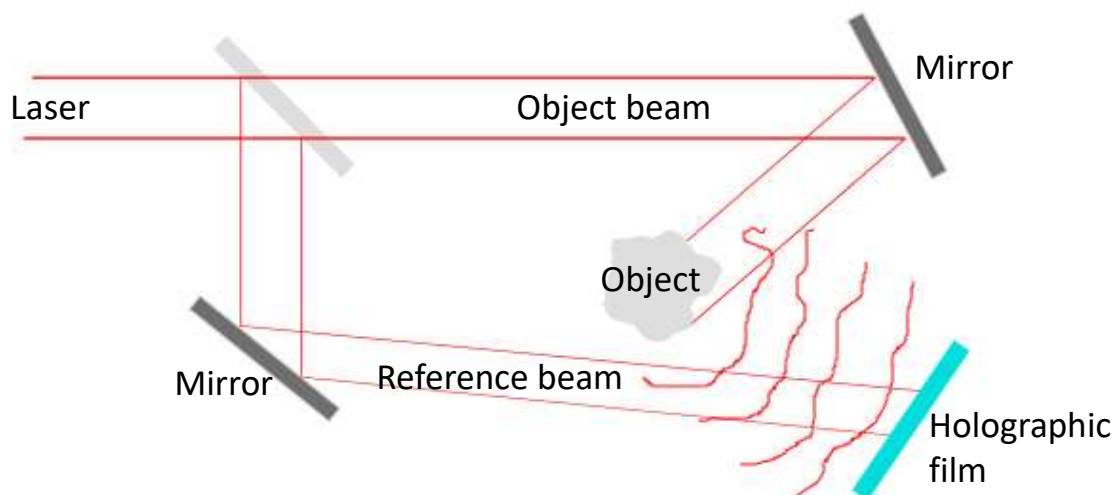


Rys. 22 Hologram Gabora

### 2.4.1. Konfiguracja Leitha-Upatnieksa

W tej konfiguracji można zapisać tzw. Hologram Fresnela, który następnie może zostać odtworzony w świetle lasera. Wiązka lasera dzielona jest na dwie (na tzw. wiązkę odniesienia oraz wiązkę obiektową). Światło rozproszone od obiektu interferuje z wiązką odniesienia tworząc zapis interferencyjny na kliszy holograficznej (Rys. 23). Przykład hologramu Fresnela znajduje się na Rys. 24.

Beam  
Splitter



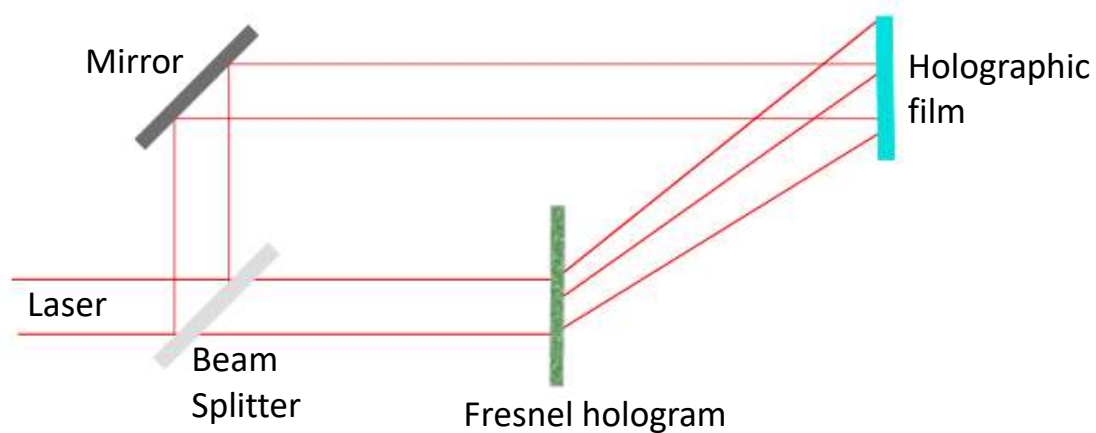
Rys. 23 Układ w konfiguracji Leitha-Upatnieksa



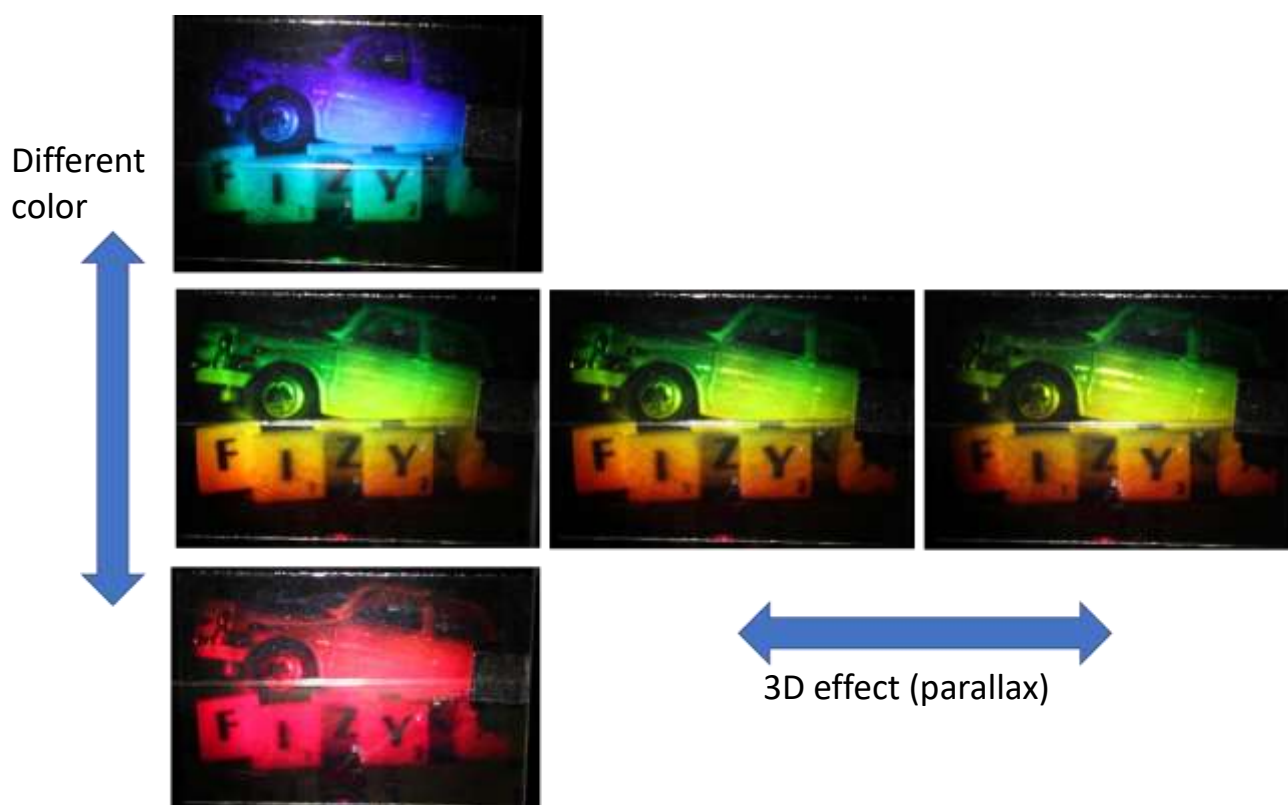
Rys. 24 Przykład hologramu Fresnela

### 2.4.1. Hologram tęczowy (Bentona)

W tym układzie (Rys. 25) jako obiekt użyty jest hologram Fresnela (np. naświetlony wcześniej w układzie Leitha-Upatnieksa). Otrzymany hologram ma tę zaletę, że może być oglądany w świetle białym. Obracając nieco hologram w jednym kierunku widoczny jest efekt 3D (czyli tzw. Paralaksa) a w drugim kierunku – zmiana kolorów. Jest to zilustrowane na Rys. 26.



Rys. 25 Układ do naświetlania hologramu Bentona

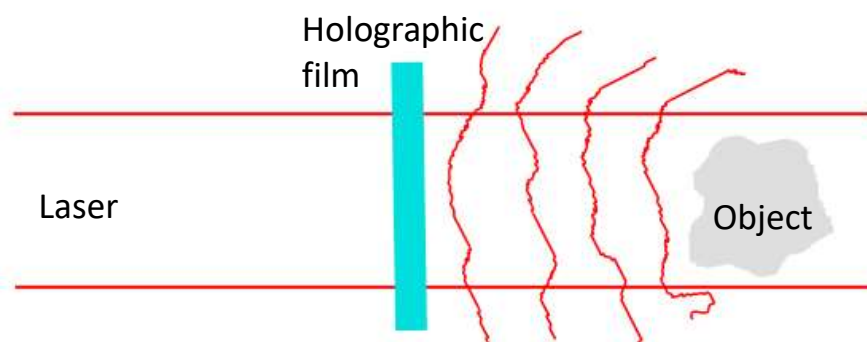


Rys. 26 Przykład hologramu Bentona

### 2.4.1. Hologram objętościowy



Hologram ten może być naświetlony w układzie pokazanym na Rys. 27. Hologram objętościowy może być oglądany w odbiciu w świetle białym. Przykład hologramu objętościowego znajduje się na Rys. 28.



Rys. 27 Układ do naświetlania hologramu objętościowego

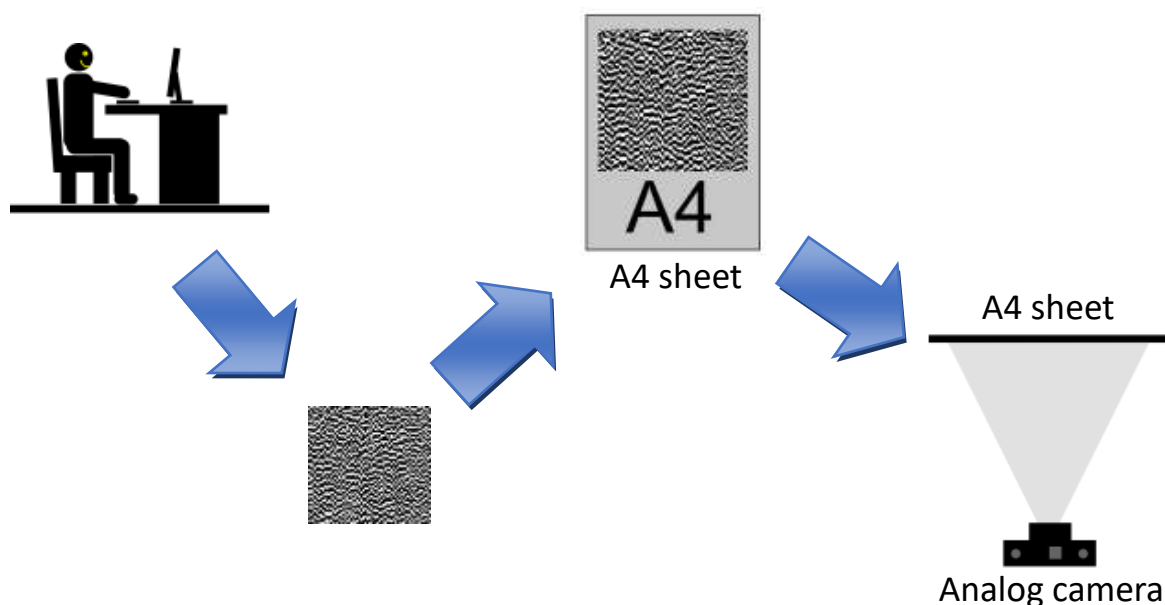


Rys. 28 Przykład hologramu objętościowego

### 2.4.1. Hologramy generowane komputerowo

Jak sama nazwa wskazuje, hologramy te nie powstają w wyniku oświetlania fizycznego obiektu światłem lasera. W najprostszym przypadku rolę obiektu pełni tu plik graficzny z napisem lub dwuwymiarowym kształtem. Na podstawie takiego pliku obliczany jest za pomocą komputera wzór interferencyjny (również plik graficzny). Do obliczeń stosuje się algorytm iteracyjny (najbardziej popularny jest algorytm Gerchberga – Saxtona). Obliczony wzór interferencyjny należy następnie wykonać w postaci elementu optycznego (przezroczca). Jeśli oświetlimy przezroczce wiązką lasera (np. wskaźnikiem laserowym) to przechodzące światło uformuje się w zaprojektowany wcześniej kształt. Jako przykład takiego hologramu można podać popularne niegdyś wymienne końcówki do wskaźników laserowych. Każda z nich generowała inny kształt. Wykonanie wysokiej jakości hologramu wymaga

specjalistycznego sprzętu. Nie mniej jednak w domowych warunkach można wykonać hologram generowany komputerowo nadający się do celów edukacyjnych. W tym celu należy obliczony wzór interferencyjny wydrukować np. na kartce A4 a następnie wykonać jego zdjęcie za pomocą aparatu analogowego. Potem wystarczy już tylko wywołać kliszę otrzymując w ten sposób hologram generowany komputerowo. Proces otrzymywania takich hologramów jest przedstawiony na Rys. 29.



**Rys. 29 Proces otrzymywania hologramów generowanych komputerowo**

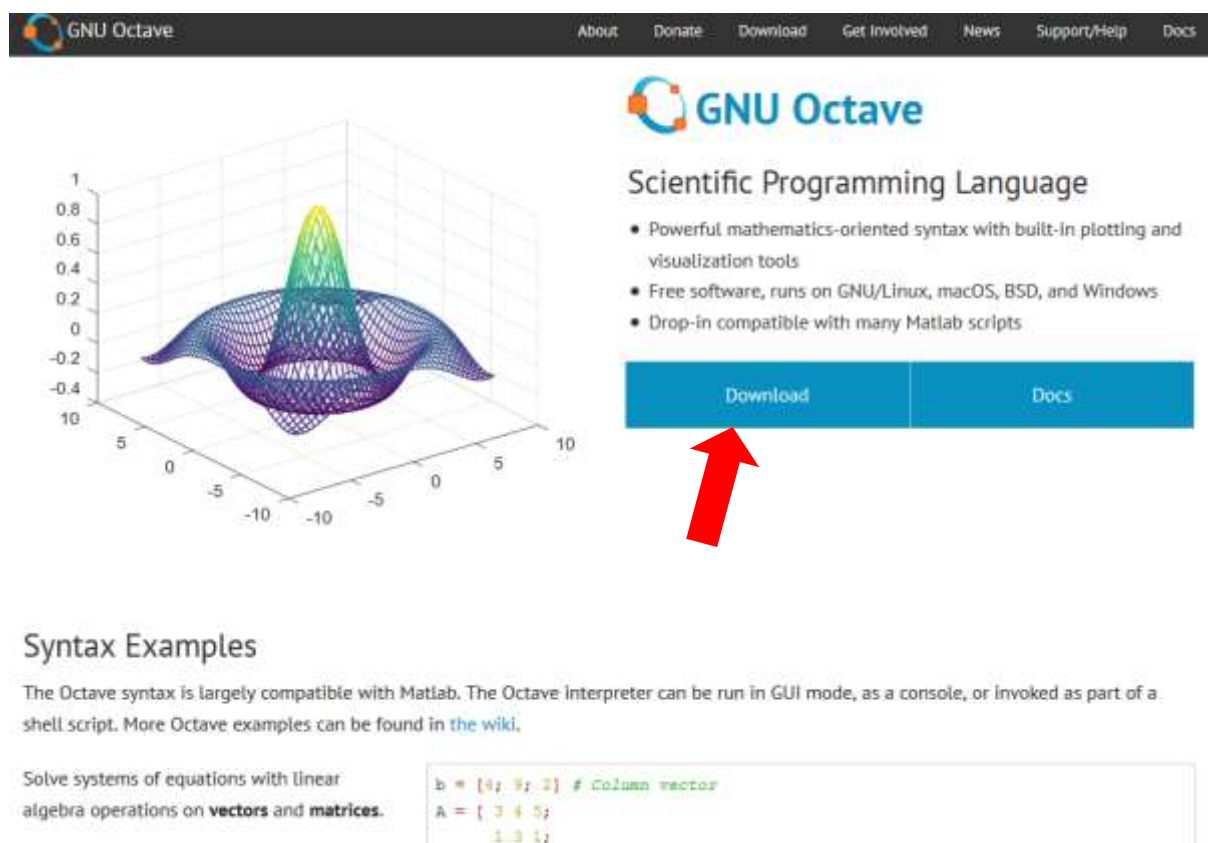
## Rozdział 3. Podstawy przetwarzania obrazów na w oparciu o program Octave.

### 3.1. Instalacja środowiska Octave

Program Octave należy pobrać ze strony:

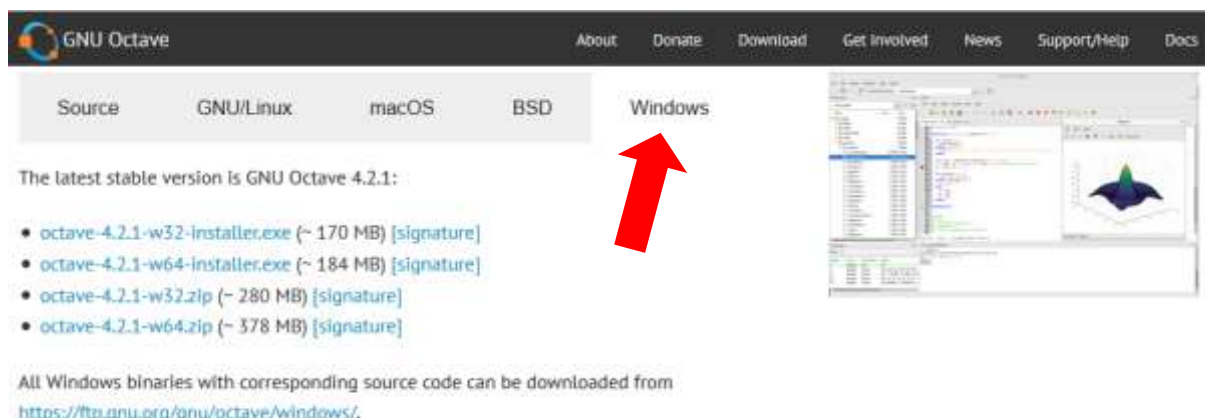
<https://www.gnu.org/software/octave/>

W tym celu należy kliknąć przycisk download zgodnie z Rys. 30:



Rys. 30 Pobieranie programu Octave

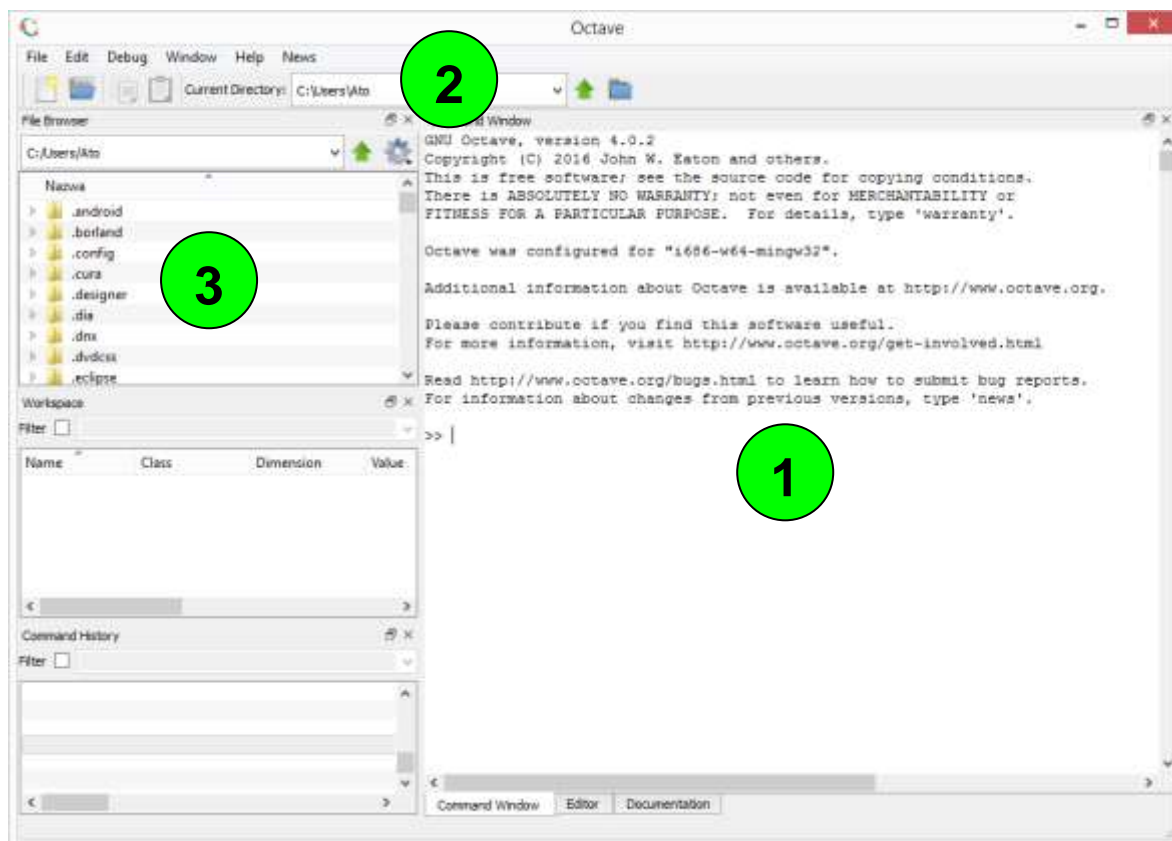
Następnie należy przejść do zakładki Windows i pobrać plik [octave-4.2.1-w32-installer.exe](#) lub [octave-4.2.1-w64-installer.exe](#) w zależności czy ma to być wersja 32 bitowa czy 64 bitowa (Rys. 31).



Rys. 31 Pobieranie programu Octave

## 3.2. Uruchomienie programu

Jeśli pracujesz z systemem Windows, kliknij menu Start i w polu wyszukiwania zacznij pisać "Octave". Następnie otwórz plik "Octave (GUI)". Pojawi się widok główny, w którym znajduje się kilka okienek (Rys. 32). Najistotniejsze z nich to obszar do wpisywania poleceń (zaznaczony na rysunku cyfrą 1), okienko pozwalające wybrać folder, w którym będziemy pracować (zaznaczony cyfrą 2) a także przeglądarkę plików znajdujących się w aktualnym folderze (cyfra 3).

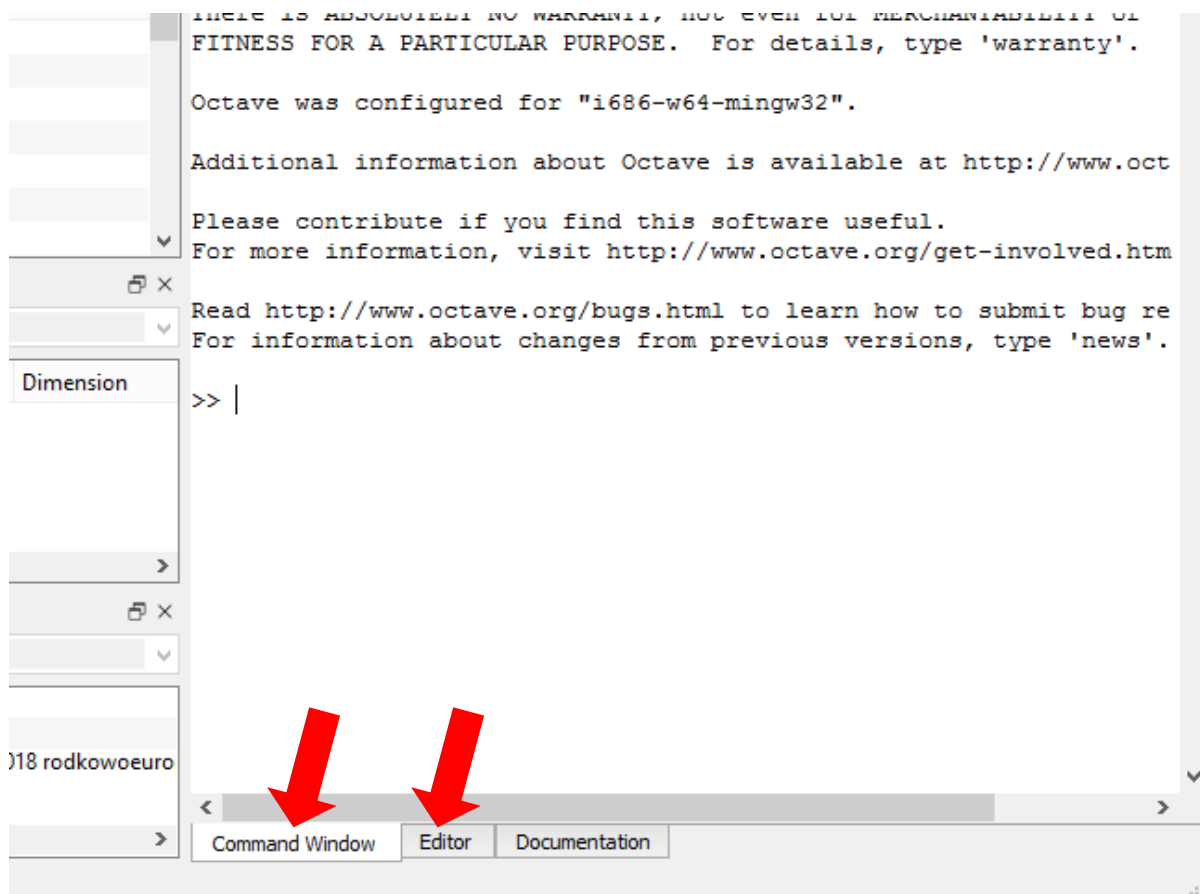


Rys. 32 Uruchomienie programu Octave

### 3.3. Okno poleceń

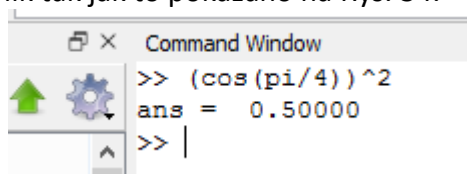
Wpisywanie poleceń może odbywać się na dwa sposoby: przy użyciu okna poleceń "Command Window" oraz przy użyciu edytora "Editor". W pierwszym przypadku polecenie jest natychmiast wykonywane po wciśnięciu klawisza Enter. W drugim przypadku należy napisać tzw. skrypt czyli szereg poleceń, które następnie po uruchomieniu skryptu będą wykonywane linijka po linijce. W tej chwili może to być trochę niezrozumiałe dlatego też poniżej przedstawionych jest kilka przykładów.

Przełączanie pomiędzy oknem poleceń a edytorem pokazane jest na Rys. 33.



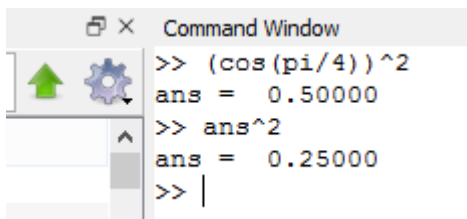
Rys. 33 "Command Window" i "Editor"

Okno poleceń "Command Window" służy do wykonywania krótkich i szybkich obliczeń. Jeśli np. chcemy obliczyć wartość wyrażenia  $\cos^2\left(\frac{\pi}{4}\right)$  to w oknie wystarczy wpisać wyrażenie `(cos(pi/4))^2`. Operator `^` oznacza podnoszenie do potęgi (w naszym przypadku do potęgi 2). Po wpisaniu powyższego wyrażenia i zatwierdzeniu klawiszem Enter otrzymamy natychmiast wynik tak jak to pokazano na Rys. 34.



Rys. 34 Wynik obliczeń

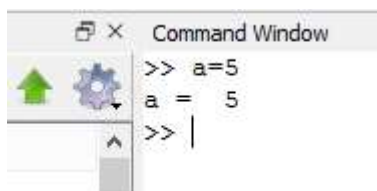
Wynik naszego wyrażenia wpisywany jest automatycznie do zmiennej o nazwie "ans" (od słowa answer - odpowiedź). Możemy tą zmienną w bardzo prosty sposób wykorzystać do dalszych obliczeń. Jeśli na przykład chcemy otrzymany wynik ponownie podnieść do potęgi 2 to wystarczy napisać wyrażenie `ans^2` jak to pokazano na Rys. 35. Wtedy zmienna "ans" przyjmie nową wartość równą w tym przypadku 0,25.



```
>> (cos(pi/4))^2
ans = 0.50000
>> ans^2
ans = 0.25000
>> |
```

Rys. 35 Dalsze obliczenia

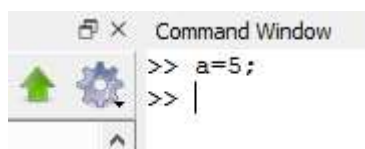
Oczywiście różne obliczenia możemy przechowywać w samodzielnie utworzonych zmiennych. Jeśli chcielibyśmy na przykład zapamiętać liczbę 5 w zmiennej, którą nazwiemy "a" to wystarczy napisać `a=5` (Rys. 36).



```
>> a=5
a = 5
>> |
```

Rys. 36 Deklaracja zmiennej

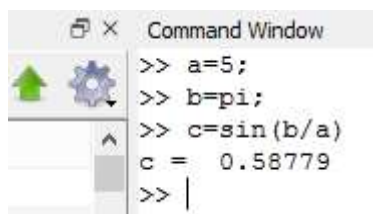
W ten sposób zadeklarowaliśmy nową zmienną o nazwie "a" i jednocześnie przypisaliśmy jej wartość równą 5. Po wciśnięciu klawisza Enter został wypisany wynik naszego polecenia (`a=5`). Jeśli chcemy aby wyniki nie były w jawny sposób wypisywane na ekranie, należy polecenie zakończyć znakiem średnika jak to pokazano na Rys. 37. Zwróć uwagę na różnicę pomiędzy Rys. 36 a Rys. 37.



```
>> a=5;
>> |
```

Rys. 37 Wynik polecenia nie jest wyświetlany w sposób jawny.

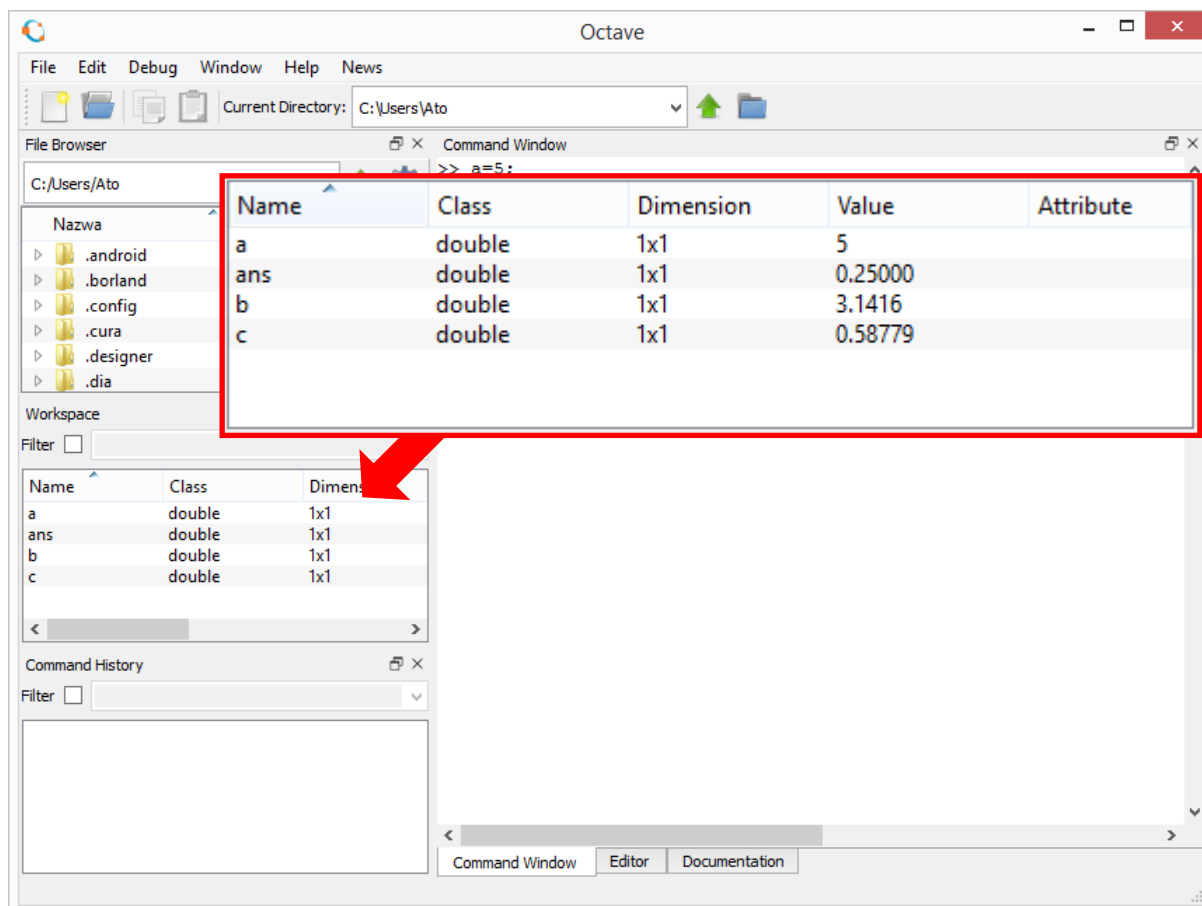
Można w ten sposób utworzyć kilka zmiennych oraz wykonać różne obliczenia (Rys. 38)



```
>> a=5;
>> b=pi;
>> c=sin(b/a)
c = 0.58779
>> |
```

Rys. 38 Kolejne obliczenia

Wszystkie zadeklarowane do tej pory zmienne można zobaczyć w oknie "Workspace" (Rys. 39)

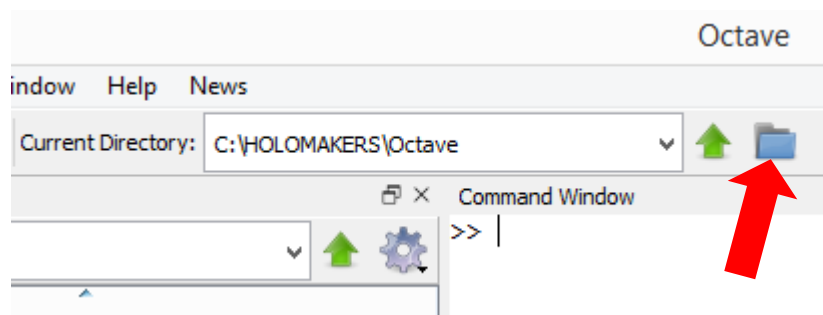


Rys. 39 Okno "Workspace"

### 3.4. Edytor

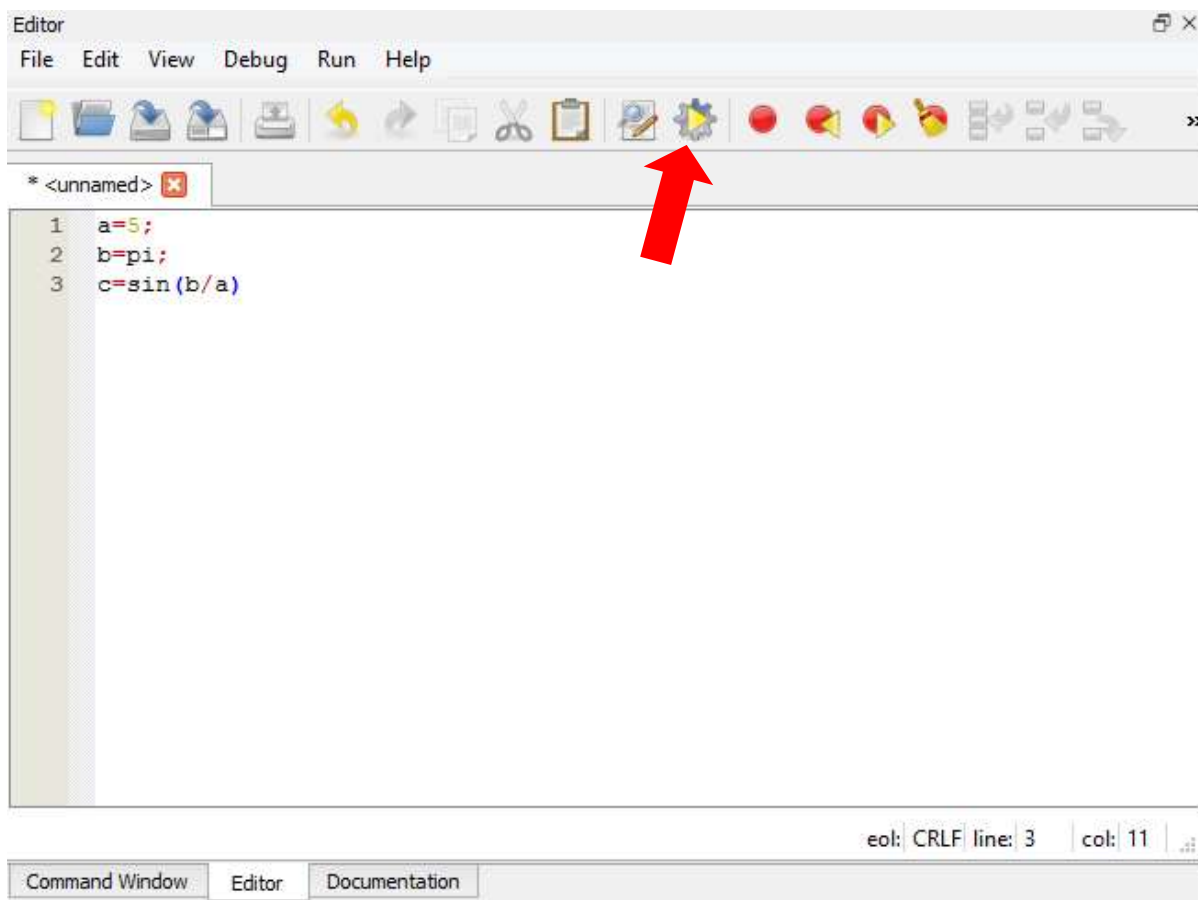
Jak to zostało wcześniej powiedziane, drugą możliwością wpisywania poleceń jest korzystanie z edytora. W tym przypadku należy najpierw napisać wszystkie polecenia i dopiero potem "uruchomić" napisany przez nas skrypt. Należy przejść do zakładki "Editor" (Rys. 33) i wpisać polecenia (tzw. kod źródłowy). Najpierw jednak warto ustawić folder, w którym chcemy pracować. W tym celu należy kliknąć niebieską ikonę zgodnie z Rys. 40 a następnie wybrać odpowiedni folder. (można np. wcześniej utworzyć folder C:\HOLOMAKERS\Octave).





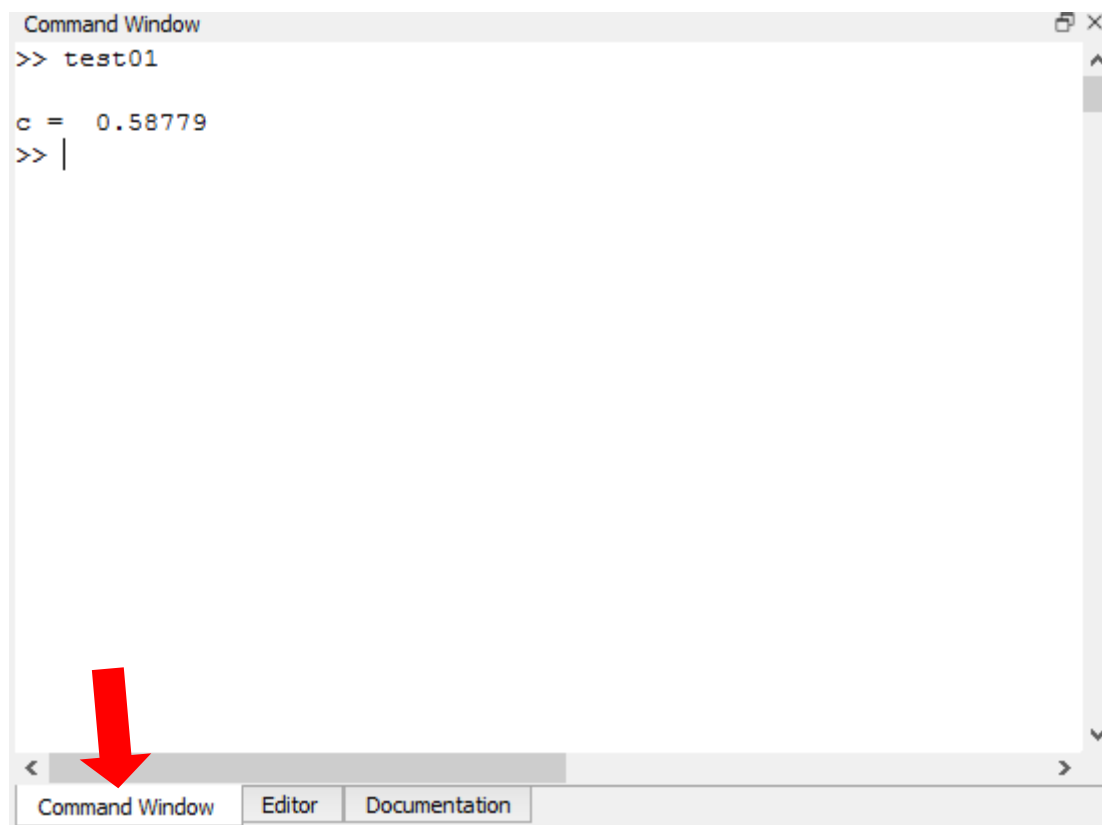
**Rys. 40 Zmiana folderu roboczego**

Teraz możemy napisać własny skrypt. Rys. 41 przedstawia kilka prostych poleceń. Każde z nich znajduje się w oddzielnej linii (pozwala to zachować czytelność kodu oraz uniknąć ewentualnych błędów). Aby obliczenia się wykonały, należy uruchomić nasz skrypt za pomocą przycisku "Zapisz i uruchom" zaznaczonego na Rys. 41 czerwoną strzałką. Podczas pierwszego uruchomienia program poprosi nas o zapisanie pliku. Plik zapisze się we wcześniej ustawionym folderze roboczym. Nadaj plikowi nazwę "test01". Plik zostanie zapisany jako "test01.m", a następnie skrypt zostanie uruchomiony (wykonają się wszystkie polecenia w kolejności od pierwszej do ostatniej linii).



**Rys. 41 Edytor**

Aby zobaczyć wynik naszych działań, należy powrócić do okna "Command Window" (Rys. 42). Wyświetlona została jedynie zmienna "c" ponieważ polecenie  $c = \sin(b/a)$  nie było zakończone średnikiem. Dwa pierwsze polecenia są zakończone średnikiem tak więc się nie wyświetliły.



Rys. 42 Wynik obliczeń

## 3.5. Wybrane elementy programowania w środowisku Octave

### 3.5.1. Zmienne

Zmienne służą do przechowywania potrzebnych nam danych (liczby, znaki, tekst) w pamięci komputera.

Dzięki temu w każdej chwili możemy odczytać wartość potrzebnej zmiennej. W Octave bardzo łatwo deklaruje się zmienną. Wystarczy napisać jej nazwę a następnie przypisać jej jakąś wartość tak jak to wspomniano w ostatnim rozdziale. W Octave jedna zmienna może zapamiętać wiele liczb. Wtedy taka zmienna nazywa się tablicą. Tablice mogą być jedno wymiarowe lub dwuwymiarowe co zostało zobrazowane na

Tablica jednowymiarowa

Tablica dwuwymiarowa

Rys. 43.

Tablica jednowymiarowa

$a(1)$	$a(2)$	$a(3)$	...	$a(n)$
--------	--------	--------	-----	--------

Tablica dwuwymiarowa

$a(1,1)$	$a(1,2)$	$a(1,3)$	...	$a(1,n)$
$a(2,1)$	$a(2,2)$	$a(2,3)$	...	$a(2,n)$
$a(3,1)$	$a(3,2)$	$a(3,3)$	...	$a(3,n)$
...	...	...	...	...
$a(m,1)$	$a(m,2)$	$a(m,3)$	...	$a(m,n)$

**Rys. 43 Zmienna tablicowa**

Jak widać każdy element tablicy jednowymiarowej zdefiniowany jest przez jeden indeks a w przypadku tablicy dwuwymiarowej - dwa indeksy. Wyobraźmy sobie, że chcemy zadeklarować jedną zmienną o nazwie `tab1`, w której będziemy przechowywać 5 liczb. W Octave wystarczy napisać:

```
tab1 = zeros(5,1)
```

Jest to tablica, która posiada 5 wierszy i jedną kolumnę. Każdy z pięciu elementów tablicy posiada wartość 0.

Teraz zadeklarujemy tablicę dwuwymiarową 3x3 również wypełnioną zerami:

```
tab2 = zeros(3,3)
```

Wpisz powyższe instrukcje w Octave w oknie "Command Window" aby zobaczyć jak wyglądają stworzone przez Ciebie tablice.

Poniżej przedstawiony jest ciekawy przykład tworzenia tablicy jednowymiarowej składającej się z równooddalonych liczb z pewnego zakresu. Stwórzmy tablicę i nazwijmy ją `X`, która zawiera 5 elementów z zakresu od 0 do  $\pi$ . Posłużymy się tu funkcją `linspace(początek zakresu, koniec zakresu, liczba elementów)`:

```
X=linspace(0,pi,5)
```

Wpisz powyższą instrukcję w Octave w oknie "Command Window" aby zobaczyć wynik.

### 3.5.2. Podstawowe operacje wejścia/wyjścia

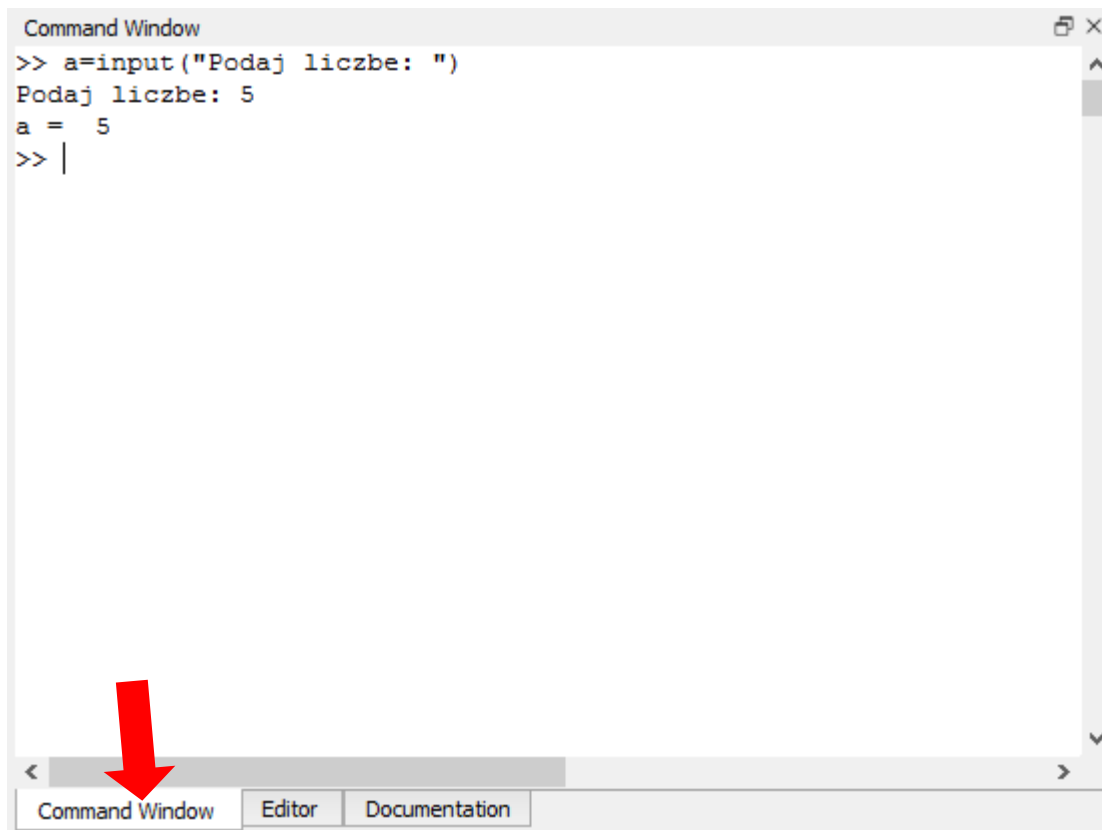
Operacja wejścia, jaką tu omówimy to wczytywanie danych z klawiatury (użytkownik podaje jakąś wartość z klawiatury, która zostanie następnie zapamiętana przez program w odpowiedniej zmiennej. Operacją wyjścia jest na przykład wyświetlenie wyniku obliczeń na ekranie.

Aby wczytać liczbę z klawiatury i zapamiętać ją w zmiennej o nazwie "a", należy użyć następującego polecenia: `a=input("Tekst dla użytkownika")`

Wpisz w Octave w oknie "Command Window" poniższe polecenie a następnie zatwierdź je klawiszem Enter

```
a=input("Podaj liczbę: ")
```

W następnej linijce wyświetli się napis `Podaj liczbę:` i program będzie oczekiwał aż wpiszesz z klawiatury jakąś wartość. Wpisz liczbę 5 i zatwierdź klawiszem Enter. Od tego momentu zmienna "a" zawiera wartość 5. Powinieneś otrzymać wynik taki jak na Rys. 44.



Rys. 44 Wczytywanie danych z klawiatury

Najprostsza operacja wyjścia to wypisanie tekstu za ekranie za pomocą polecenia `disp`. W oknie "Command Window" wpisz następujące polecenie zatwierdzając je klawiszem Enter:

```
disp('Zmienna "a" zawiera wartosc:'), disp(a)
```

Zostanie najpierw wypisany tekst Zmienna "a" zawiera wartosc: a w następnym wierszu wypisana zostanie wartość zmiennej "a". Widać więc, że za pomocą funkcji `disp` możemy wyświetlać na ekranie zarówno tekst jak i wartości zmiennych. Jeśli chcemy wyświetlić tylko tekst to możemy również użyć funkcji `puts`.

### 3.5.3. Operatory

Omówimy trzy klasy operatorów: operatory arytmetyczne, relacyjne oraz logiczne.

1. Operatory arytmetyczne to nic innego jak operator dodawania (+), odejmowania (-), mnożenia(\*), dzielenia (/) oraz operator przypisania wartości (=). Tak więc na przykład wyrażenie `a=c+2*b` zawiera trzy operatory (=, +, \*).
2. Operatory relacyjne służą do porównywania dwóch wartości (np. `a>b`). Te operatory to: operator równości (==), mniejszy niż (<), większy niż (>), mniejszy lub równy (<=), większy lub równy (>=), różny od (!=). Wynikiem porównania dwóch wartości jest tzw. wartość logiczna

czyli prawda lub fałsz. Jeśli np. napiszemy wyrażenie  $2==5$  to będzie to fałsz, natomiast wyrażenie  $2<5$  będzie prawdą.

3. Operatory logiczne jak sama nazwa wskazuje służą do wykonywania podstawowych operacji logicznych takich jak suma logiczna ( $|$ ), iloczyn logiczny ( $\&$ ) oraz operator negacji ( $!$ ). Wynikiem wyrażenia logicznego jest albo prawda albo fałsz. Np. wyrażenie  $(5>2) \& (2<3)$  jest prawdą ponieważ jednocześnie wyrażenia  $5>2$  oraz  $2<3$  są prawdą. Zastosowano tutaj wyrażenie, które zawiera zarówno operatory relacyjne ( $<$ ,  $>$ ) jak i operator logiczny ( $\&$ ).

Dość istotnym pojęciem jest tzw. priorytet operatorów. Pomimo, że brzmi to dość zagadkowo to każdy uczeń spotkał się z tym już od pierwszych lat nauki matematyki. Podaj wynik następującego wyrażenia:  $5+2*3=?$ . Zgadza się, wynik to 11. Zauważ, że intuicyjnie najpierw wykonałeś mnożenie a dopiero potem dodawanie. Dlaczego najpierw nie dodałeś liczb 5 i 2 a potem nie pomnożyłeś wyniku przez 3? Jest tak dlatego, że masz świadomość iż najpierw wykonuje się operację mnożenia a dopiero potem dodawania. To jest właśnie priorytet operatorów. Mówi on nam, które operacje w wyrażeniu wykonywane są najpierw a które w dalszej kolejności. Tabela 1 przedstawia operatory uporządkowane względem ich priorytetu. Najwyższy priorytet ma operator negacji logicznej. Oznacza to, że jeśli taki operator znajduje się gdzieś w wyrażeniu to zostanie on najpierw wykonany. Najniższy priorytet mają operatory relacyjne (wykonywane są one na samym końcu).

Priorytet	Operator
1	!
2	&, *, /
3	, +, -
4	==, <, >, <=, >=, !=

**Tabela 1 Priorytet operatorów**

Pewną oczywistą rzeczą znaną również z lekcji matematyki jest fakt, że jeśli chcemy zmienić kolejność wykonywania operatorów to musimy w odpowiednim miejscu postawić nawiasy. Tak więc wyrażenie  $5+2*3$  daje wartość 11 natomiast wynikiem wyrażenia  $(5+2)*3$  będzie liczba 21. Czasem wstawienie nawiasów jest konieczne aby wyrażenie miało sens. Taki przykład był już podany powyżej:  $(5>2) \& (2<3)$ .

### 3.5.4. Instrukcja warunkowa "jeżeli"

Każdy z nas codziennie podejmuje różne decyzje. Np. jeśli świeci słońce to możemy zdecydować, że pójdziemy na rower a w przeciwnym przypadku zostaniemy w domu. Jest to przykład prostej instrukcji warunkowej. Najpierw sprawdzamy jakiś warunek (np. czy świeci słońce) a potem w zależności od tego jaki jest wynik warunku podejmujemy odpowiednie działanie (np. idziemy na rower lub zostajemy w domu). Warunek przedstawia się za pomocą wyrażenia logicznego, które albo jest prawdą albo fałszem. Wyrażeniem logicznym może być np. zdanie "Teraz jest słoneczna pogoda". To zdanie (wyrażenie logiczne) w zależności od warunków pogodowych jest albo prawdą albo fałszem.

W programowaniu instrukcje warunkowe są tak samo niezbędne jak w codziennym życiu. Rys. 45 przedstawia strukturę prostej instrukcji warunkowej. W zależności od tego czy warunek jest prawdziwy czy nie, wykonują się inne instrukcje (polecenia).




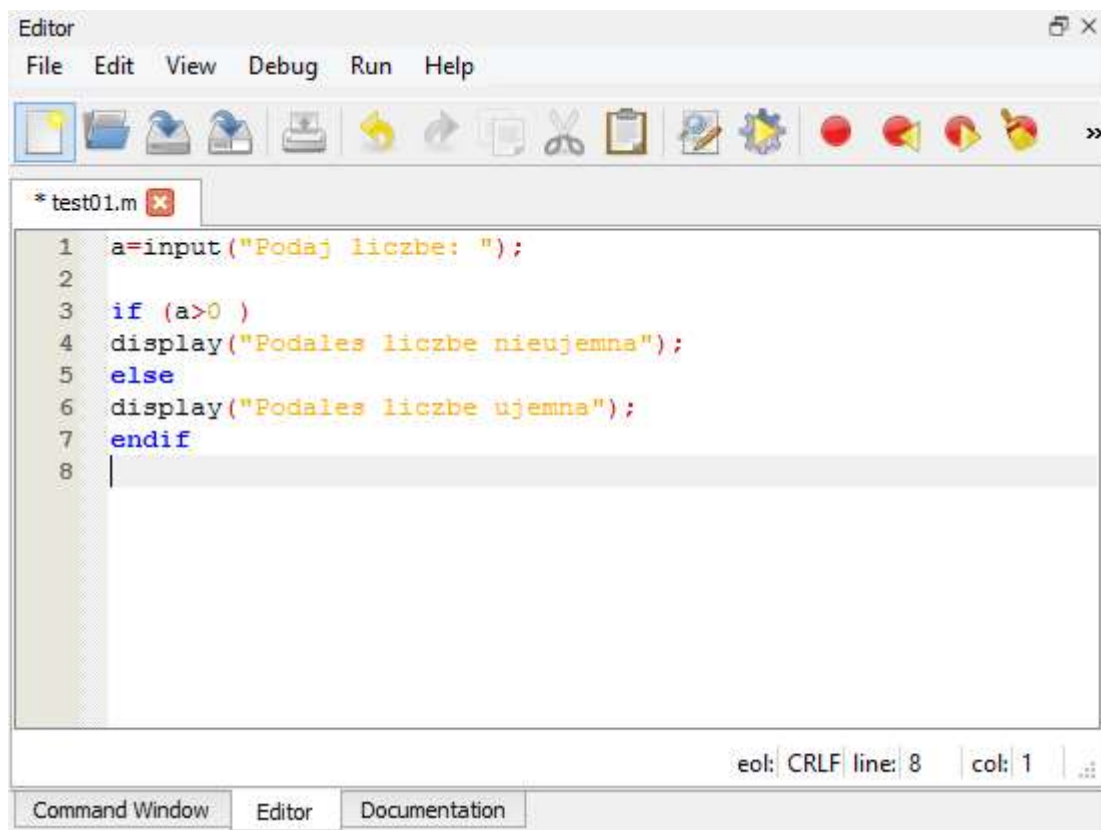
Rys. 45 Struktura prostej instrukcji warunkowej "Jeżeli"

W Octave instrukcja jeżeli wygląda następująco:

```

if (warunek)
Instrukcje, które się wykonają jeśli warunek jest prawdziwy
else
Instrukcje, które się wykonają jeśli warunek jest nieprawdziwy
endif
  
```

Jako przykład użycia instrukcji "Jeżeli" napiszemy krótki skrypt, który będzie sprawdzał czy podana przez użytkownika liczba jest ujemna czy nieujemna. Napisz w oknie Edytora skrypt jak na Rys. 46. Następnie wciśnij przycisk "zapisz plik i uruchom"  i przejdź do okna "Command Window". Po wpisaniu liczby i wciśnięciu klawisza Enter pojawi się odpowiedni tekst.



```

1 a=input("Podaj liczbe: ");
2
3 if (a>0 )
4 display("Podales liczbe nieujemna");
5 else
6 display("Podales liczbe ujemna");
7 endif
8

```

**Rys. 46 Przykład użycia instrukcji warunkowej "Jeżeli"**

Często jednak mamy do czynienia z sytuacją, kiedy po sprawdzeniu jednego warunku chcemy sprawdzić kolejny. W Octave napiszemy wtedy:

```
if (warunek1)
```

Instrukcje, które się wykonają jeśli warunek1 jest prawdziwy

```
elseif (warunek2)
```

Instrukcje, które się wykonają jeśli warunek1 jest nieprawdziwy oraz warunek2 jest prawdziwy

```
else
```

Instrukcje, które się wykonają jeśli zarówno warunek1 jak i warunek2 jest nieprawdziwy

```
endif
```

Aby to lepiej zilustrować, zmodyfikujmy nieco ostatni skrypt tak, aby program wskazywał dodatkowo czy użytkownik podał liczbę zero (Rys. 47).



```

1 a=input('Podaj liczbe: ');
2
3 if (a>0 )
4 display('Podales liczbe dodatnia');
5 elseif (a<0)
6 display('Podales liczbe ujemna');
7 else
8 display('Podales wartosc 0');
9 endif

```

Rys. 47 Przykład użycia instrukcji warunkowej "Jeżeli"

### 3.5.5. Pętla "for"

Często zachodzi potrzeba aby wykonać jakieś polecenie wiele razy. Aby nie wpisywać tych samych poleceń wiele razy, w programowaniu korzysta się z pętli. My omówimy krótko tylko jeden rodzaj pętli - pętlę for. Pozwala ona na wykonanie jednego polecenia określoną ilość razy. Struktura pętli for pokazana jest na Rys. 48. Jest ona bardzo prosta i mówi nam, że instrukcje wykonają się w tym przypadku 10 razy. Skąd wiadomo, że akurat 10 razy? Jest to zdefiniowane na samym początku. Pewna zmienna, którą nazwaliśmy "i" w każdym kolejnym przebiegu pętli zwiększa swoją wartość o 1 poczynając od 1 aż do 10. Gdy zmienna "i" osiągnie wartość końcową (w tym przypadku 10), pętla kończy swoje działanie.


```

1
2 for i=1:10
3
4 instrukcje...
5
6 end
7

```

Rys. 48 Pętla "for"

Bardzo prostym przykładem użycia pętli "for" może być wypisanie pewnej liczby znaków na ekranie. Wyobraźmy sobie, że chcemy mieć możliwość wypisania linii składającej się ze znaków "-". Możemy w tym celu napisać bardzo prosty skrypt, pokazany na Rys. 49. Utwórz

nowy skrypt na pomocą ikonki . Podczas uruchomienia zapiszesz go pod nazwą test02. Widać, że nasza linia będzie miała długość 15 znaków (pętla "for" wykona się 15 razy).

```

1
2 for i=1:15
3
4 puts ("-");
5
6 end

```

Rys. 49 Wypisanie linii o długości 15 znaków

Wynik działania skryptu można zobaczyć w oknie "Command Window" (Rys. 50)

```
Command Window
>> test02

----->> |
```

Rys. 50 Linia o długości 15 znaków.

Możemy nieco zmodyfikować skrypt tak aby pozwalał użytkownikowi zdecydować jakiej długości linię wypisać (Rys. 51). Zwróć uwagę, że pętla wykona się  $n$  razy, gdzie  $n$  jest liczbą podaną przez użytkownika.

```
1 n=input("Podaj dlugosc linii: ");
2
3 for i=1:n
4
5 puts("-");
6
7 end
```

Rys. 51 Wypisanie linii o dowolnej liczbie znaków

Pętla for jest bardzo często używana do różnego rodzaju obliczeń. Czy umiałbyś napisać skrypt, w którym podasz 5 liczb a program wyliczy ich średnią? Jeśli nie to spójrz na Rys. 52.

```
1 for i=1:5
2
3 a(i) = input("Podaj liczbe: ");
4
5 end
6
7 suma = 0;
8 for i=1:5
9
10 suma = suma + a(i);
11
12 end
13
14 srednia = suma/5
```

Rys. 52 Obliczanie średniej arytmetycznej

Przeanalizujmy teraz ten skrypt. Najpierw następuje wczytanie przez użytkownika 5 liczb. Liczby te zapisywane są do tablicy o nazwie "a". Następnie deklarowana jest zmienna o nazwie suma i przypisana zostaje jej wartość wynosząca 0. Kolejna pętla wykonuje sumę 5

liczb wczytanych przez użytkownika. Ostatnia linijka to obliczenie średniej arytmetycznej. Ponieważ nie jest ona zakończona średnikiem to wynik wyświetli się na ekranie.

### 3.5.6. Rysowanie wykresów

Octave pozwala w bardzo prosty sposób narysować wykres funkcji. Aby narysować funkcję  $y = f(x)$ , musimy najpierw stworzyć tablicę argumentów funkcji  $x$  oraz odpowiadającą jej tablicę wartości funkcji  $f(x)$ . Załóżmy, że chcemy narysować funkcję  $y = \sin(x)$  w przedziale  $[0, 2\pi]$ . Octave pozwala na zadeklarowanie tablicy o  $n$  równooddalonych elementach z przedziału  $[a, b]$ . Służy do tego funkcja `linspace(a, b, n)`. Załóżmy, że chcemy nasz przedział  $[0, 2\pi]$  podzielić na 100 równooddalonych wartości. W takim wypadku musimy napisać:

```
x = linspace(0, 2*pi, 100);
```

W ten oto sposób zadeklarowaliśmy tablicę 100 elementów. Uzyskanie tablicy wartości funkcji jest jeszcze prostsze, wystarczy bowiem napisać:

```
y = sin(x);
```

Ponieważ  $x$  jest tablicą 100-elementową to  $y$  automatycznie też stanie się tablicą o tej samej liczbie elementów. Teraz wystarczy wyświetlić wykres. W tym celu należy napisać:

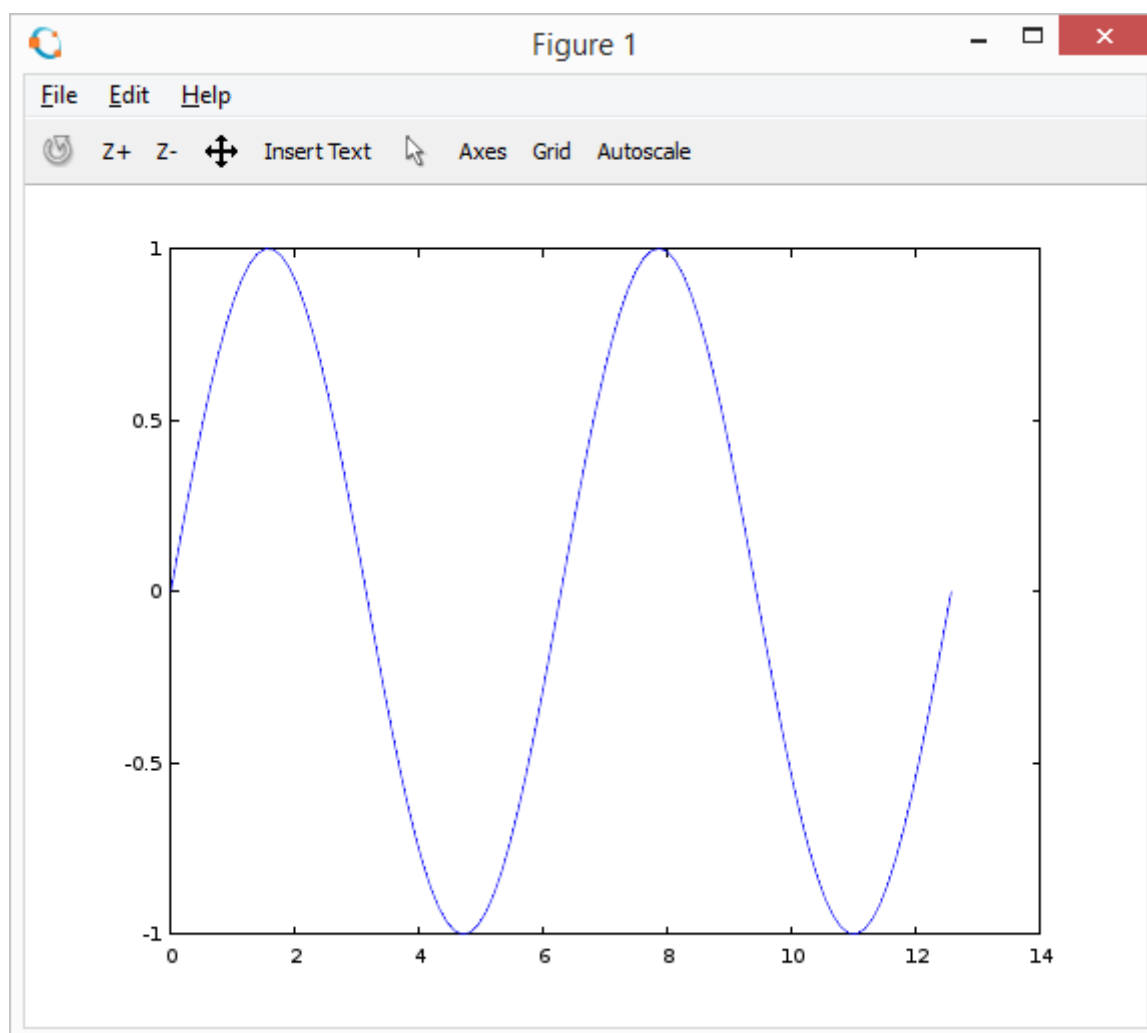
```
plot(x, y)
```

Cały skrypt zaprezentowany jest na Rys. 53.

```
1 x = linspace(0, 4*pi, 100);
2 y = sin(x);
3 plot(x, y)
```

Rys. 53 Przygotowanie danych oraz wyświetlenie wykresu

Po uruchomieniu skryptu pojawi się nowe okno z wykresem (patrz Rys. 54)



Rys. 54 Wykres

## 3.6. Przetwarzanie obrazów

### 3.6.1. Tworzenie obrazu

Szaroodcieniowe obrazy w Octave reprezentowane są w postaci dwuwymiarowej tablicy, której każdy element odpowiada jednemu pikselowi (Rys. 55). Wartość każdego elementu tablicy to odcień szarości od czarnego do białego. Kolorowi czarnemu odpowiada wartość 0 a białemu 255. W przypadku obrazów binarnych (czyli czarno-białych) kolorowi czarnemu odpowiada wartość 0 a białemu 1.



Rys. 55

Utworzenie nowego obrazu to po prostu zadeklarowanie tablicy dwuwymiarowej. Jeśli chcemy utworzyć obraz o rozmiarach 200x200 pikseli to wystarczy napisać:

```
image1 = zeros(200,200);
```

W ten sposób zmiennej `image1` została przyporządkowana tablica 200x200 wypełniona zerami (każdy element tablicy wynosi zero). Skorzystaliśmy tutaj z funkcji `zeros`.

### 3.6.2. Odczyt/zapis obrazu z pliku i wyświetlanie go na ekranie

Często istnieje potrzeba aby wczytać do obróbki istniejący już na dysku plik graficzny. Aby to zrobić najpierw należy skopiować obraz do wcześniej ustawionego aktualnego katalogu. Następnie wystarczy napisać polecenie

```
image1 = imread('nazwa pliku');
```

Plik graficzny o określonej nazwie został zapamiętany w zmiennej `image1`.

Aby wyświetlić obraz ze zmiennej `image1` należy użyć polecenia

```
imshow(image1);
```

Aby wyświetlić kilka obrazów, należy każdemu obrazowi nadać inny numer za pomocą polecenia `figure(number)`. Warto również zatytułować okno poleceniem `title('Tytuł obrazu')`. Poniżej znajduje się przykład wyświetlenia dwóch różnych obrazów w dwóch różnych oknach:

```
figure(1);imshow(image1);title('Obraz nr 1');  
figure(2);imshow(image2);title('Obraz nr 2');
```

Zapis obrazu do pliku odbywa się za pomocą polecenia

```
imwrite(image1, 'nazwa.rozszerzenie');
```

Obraz można zapisywać w różnych formatach np. bmp, png, jpg, tiff, gif.

Jako przykład wgraj do aktualnego katalogu roboczego jakiś plik graficzny z rozszerzeniem bmp a następnie napisz skrypt, który wyświetli ten plik na ekranie oraz zapisze z różnymi rozszerzeniami (Rys. 56).

```
1 image1=imread('test.bmp');  
2 imshow(image1);  
3 imwrite(image1, 'test.gif');  
4 imwrite(image1, 'test.tiff');  
5 imwrite(image1, 'test.png');  
6 imwrite(image1, 'test.jpg');
```

Rys. 56 Konwersja obrazu na różne formaty

### 3.6.3. Konwersja kolorów

Octave zaopatrzony jest wiele różnych funkcji służących do konwersji kolorów. Wśród nich znajduje się funkcja konwertująca obraz kolorowy na szary oraz funkcja konwertująca obraz (szary lub kolorowy) na obraz binarny czyli składający się tylko z dwóch kolorów - czarnego i białego. Funkcje te nie są jednak domyślnie dostępne wraz z uruchomieniem środowiska Octave. Aby móc korzystać z tych funkcji należy załadować tzw. pakiet o nazwie `image`. Należy więc na samym początku skryptu napisać:

```
pkg install image  
pkg load image
```

Od teraz swobodnie możemy korzystać z wielu funkcji znajdujących się w tym pakiecie.

W celu zamiany kolorowego obrazu na szary należy użyć funkcji `rgb2gray`. Tak więc jeśli na przykład zmienna `colorImage` przechowuje obraz kolorowy to możemy go zamienić na szary i zapisać w zmiennej `grayImage` w następujący sposób:

```
grayImage= rgb2gray(colorImage);
```

Podobnie obraz może być przekonwertowany na binarny:

```
bwImage= im2bw(colorImage,0.5);
```

### 3.6.4. Obrót

Obraz w Octave możemy obrócić względem środka o dowolny kąt za pomocą funkcji `imrotate`. Aby użyć tej funkcji musimy wskazać obracany obraz oraz kąt obrotu. Jeśli np. chcemy w zmiennej `Image2` umieścić obrócony o  $45^\circ$  obraz `Image1`, należy napisać:

```
Image2 = imrotate(Image1,45);
```

### 3.6.5. Dodawanie, odejmowanie, mnożenie, dzielenie.

Przypomnijmy, że obraz w Octave reprezentowany jest jako dwuwymiarowa tablica o wymiarach odpowiadających szerokości i wysokości obrazu w pikselach. Mając zatem dwa obrazy o tych samych wymiarach, możemy z łatwością wykonać pewne operacje takie jak dodawanie czy mnożenie obrazów. Przykładowo dodanie do siebie dwóch obrazów będzie po prostu dodaniem do siebie odpowiadających sobie wartości piksela (Rys. 57).

Obraz A		Obraz B		Obraz C
8 3 0 8 14		20 12 2 19 15		28 15 2 27 29
18 9 10 13 0		15 11 5 17 15		33 20 15 30 15
17 9 10 7 6	+	16 6 5 9 11	=	33 15 15 16 17
9 19 17 4 19		8 15 13 19 17		17 34 30 23 36
7 1 16 19 7		8 0 9 5 3		15 1 25 24 10

Rys. 57 Dodawanie obrazów

W podobny sposób wykonuje się odejmowanie a także mnożenie i dzielenie. Tabela 2 przedstawia nazwy poszczególnych funkcji w środowisku Octave.

Operacja	Octave
Dodawanie	<code>imadd</code>
Odejmowanie	<code>imsubtract</code>
Mnożenie	<code>immultiply</code>
Dzielenie	<code>imdivide</code>

Tabela 2 Operacje na obrazach

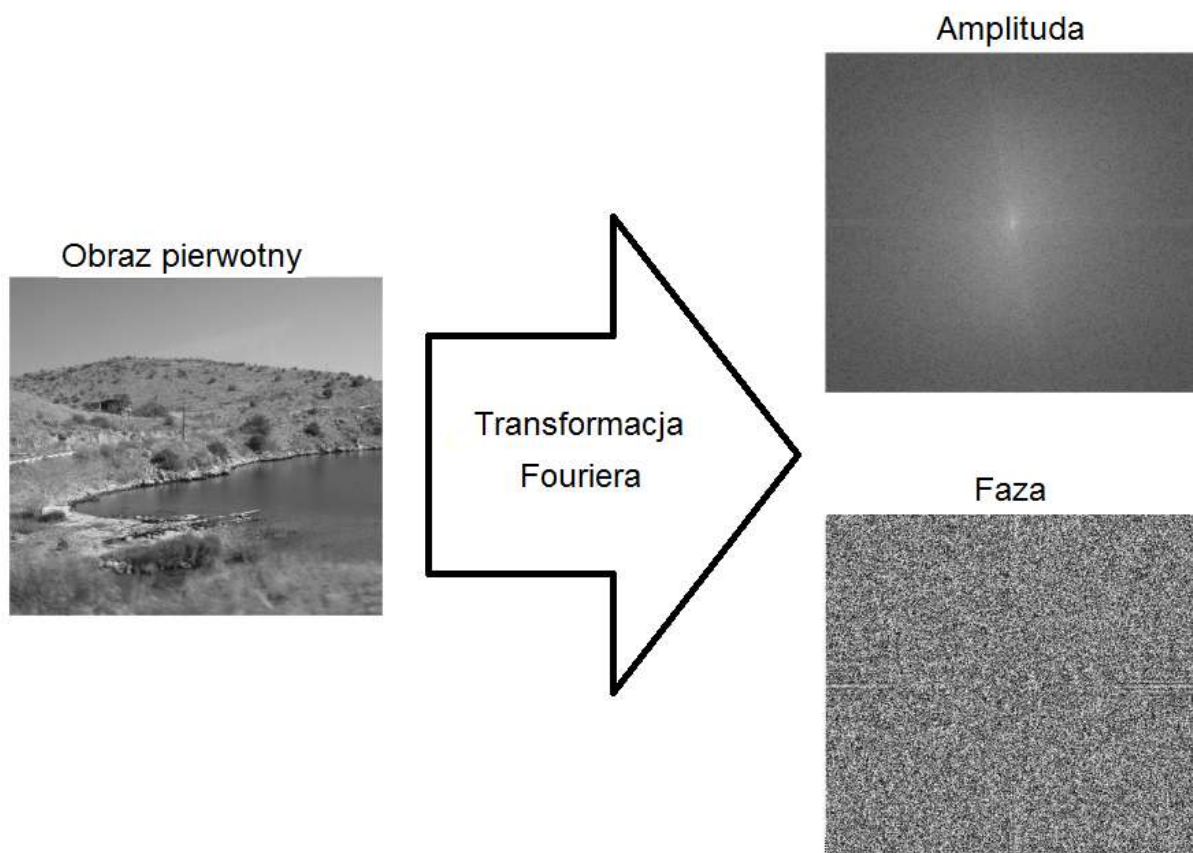
Niech zmienne `imgA` oraz `imgB` reprezentują dwa obrazy o tych samych wymiarach. Wtedy możemy stworzyć obraz `imgC` będący sumą lub iloczynem tych dwóch obrazów:

```
imgC = imadd(imgA,imgB);  
imgC = immultiply(imgA,imgB);
```



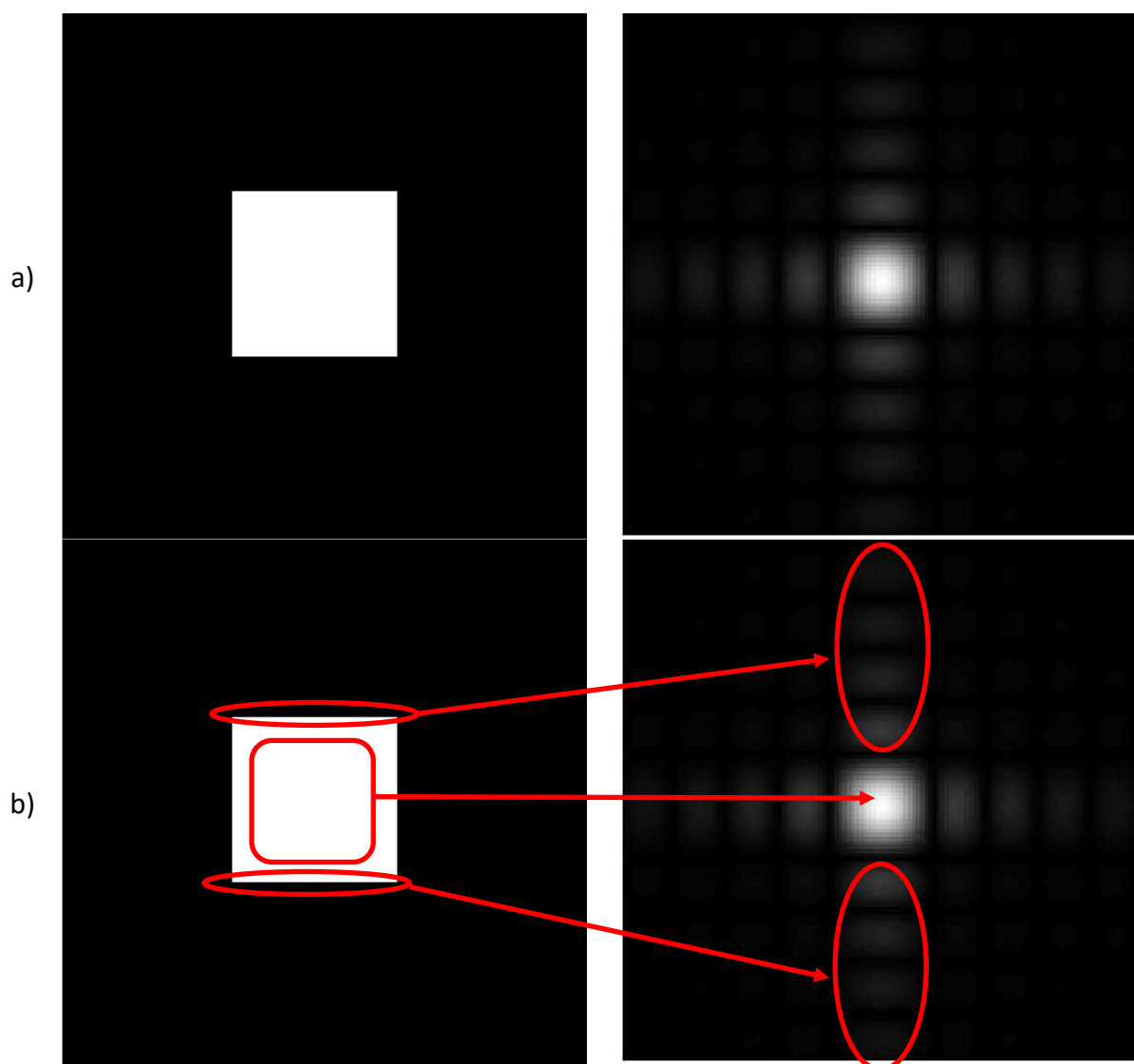
### 3.6.6. Transformacja Fouriera

Transformacja Fouriera jest pewnym przekształceniem szeroko stosowanym w wielu dziedzinach takich jak elektronika, optyka, muzyka. W przypadku przetwarzania obrazów, również można wykonać transformację Fouriera. Obraz poddany tej transformacji często wygląda jak zbiór przypadkowych pikseli jednak zawiera on tą samą informację co obraz pierwotny (Rys. 58).



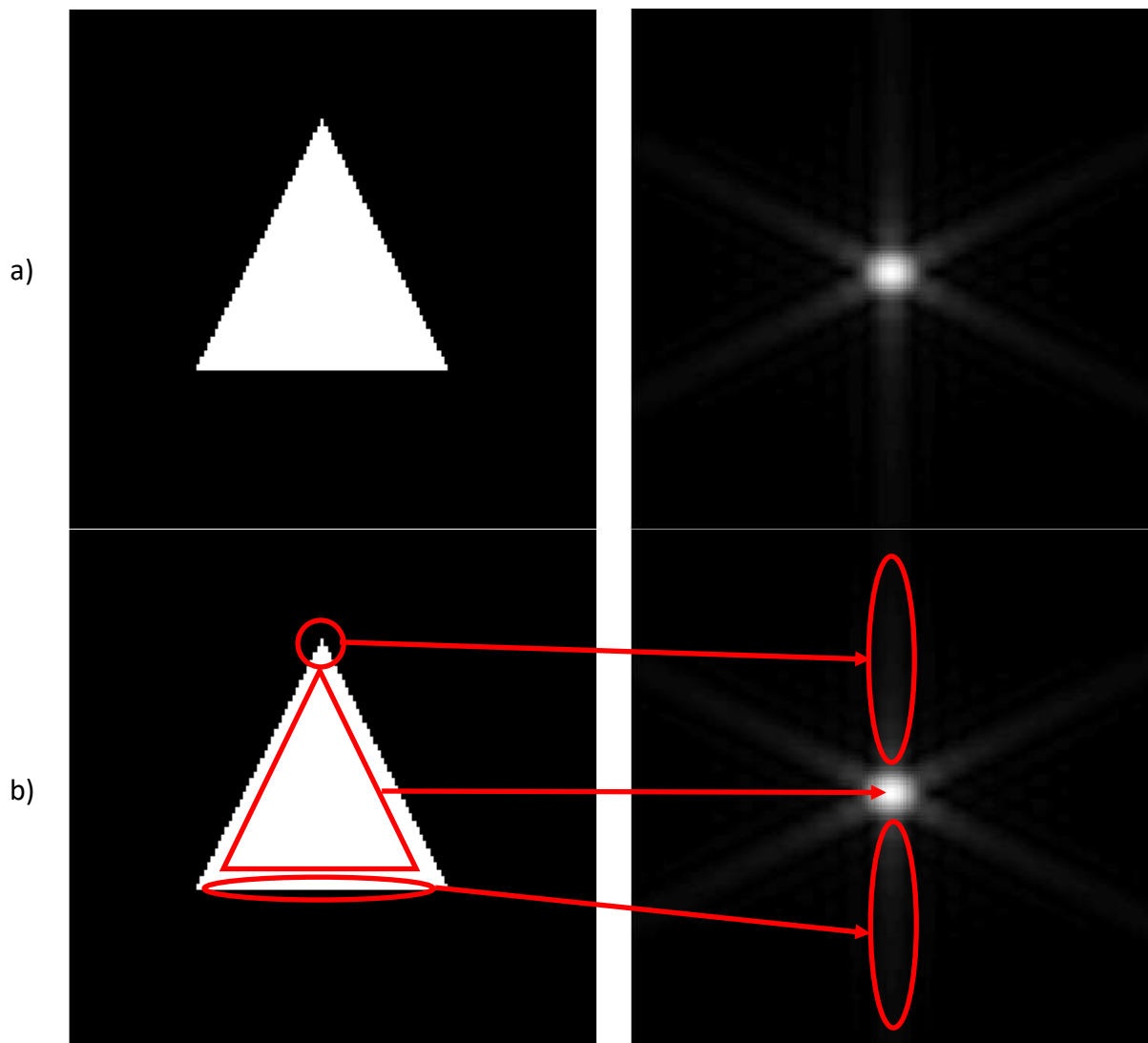
**Rys. 58 Obraz i jego transformata Fouriera**

Każdy punkt transformaty Fouriera posiada nie tylko amplitudę ale także i fazę. Szczegółowe omówienie tego zagadnienia znacznie wykracza poza zakres niniejszego przewodnika. Zapamiętajmy tylko, że faza jest niezbędna do odtworzenia pierwotnego obrazu (poprzez odwrotną transformację Fouriera), natomiast amplituda rozłożona jest w następujący sposób: punkty w środku transformaty odpowiadają gładkim obszarom obrazu, natomiast punkty transformaty oddalone od środka odpowiadają krawędziom obrazu (krawędzią jest np. granica pomiędzy gładkim niebem a górą, pomiędzy wodą a lądem itp.). Można to zilustrować na prostym przykładzie. Rys. 59a przedstawia biały kwadrat oraz jego transformatę Fouriera. Środkowa plamka transformaty odpowiada środkowej części kwadratu (gładki obraz) natomiast każde z czterech "ramion" odpowiada odpowiedniej krawędzi kwadratu (Rys. 59b).



Rys. 59 Transformata Fouriera kwadratu

W przypadku obrazów prostych figur geometrycznych jest to dość wyraźnie widoczne. Rys. 60 przedstawia obraz trójkąta i jego transformatę. Środkowa plamka transformaty odpowiada środkowej części trójkąta. 3 boki trójkąta można rozpoznać w transformacie jako 3 "ramiona". Widoczne są również dodatkowe 3 "ramiona". Reprezentują one wierzchołki trójkąta.



Rys. 60 Transformata Fouriera trójkąta

W Octave funkcja wykonująca transformatę Fouriera nazywa się `fft2`. Funkcja ta działa w taki sposób, że transformata Fouriera jest przesunięta względem środka. Należy więc dodatkowo użyć funkcji `fftshift`. Jeśli na przykład chcemy w zmiennej `img2` umieścić transformatę Fouriera obrazu `img1` należy napisać:

```
img2=fftshift(fft2(img1));
```

Aby wyświetlić amplitudę transformaty należy napisać:

```
imshow(abs(img2),[]);
```

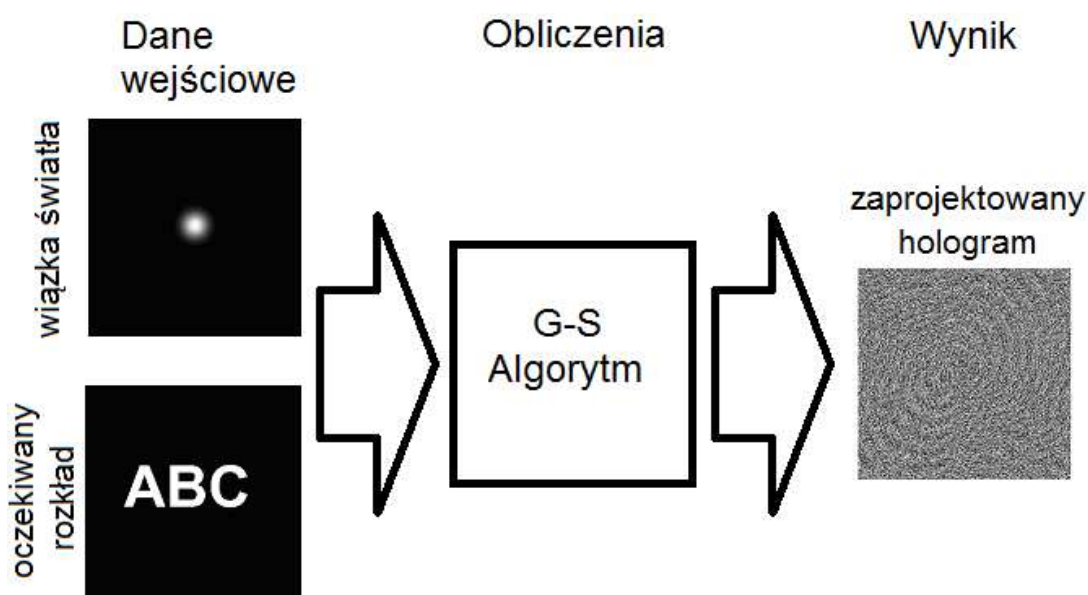
Natomiast w celu wypisania fazy transformaty należy napisać:

```
imshow(angle(img2),[]);
```

### 3.7. Hologramy generowane komputerowo

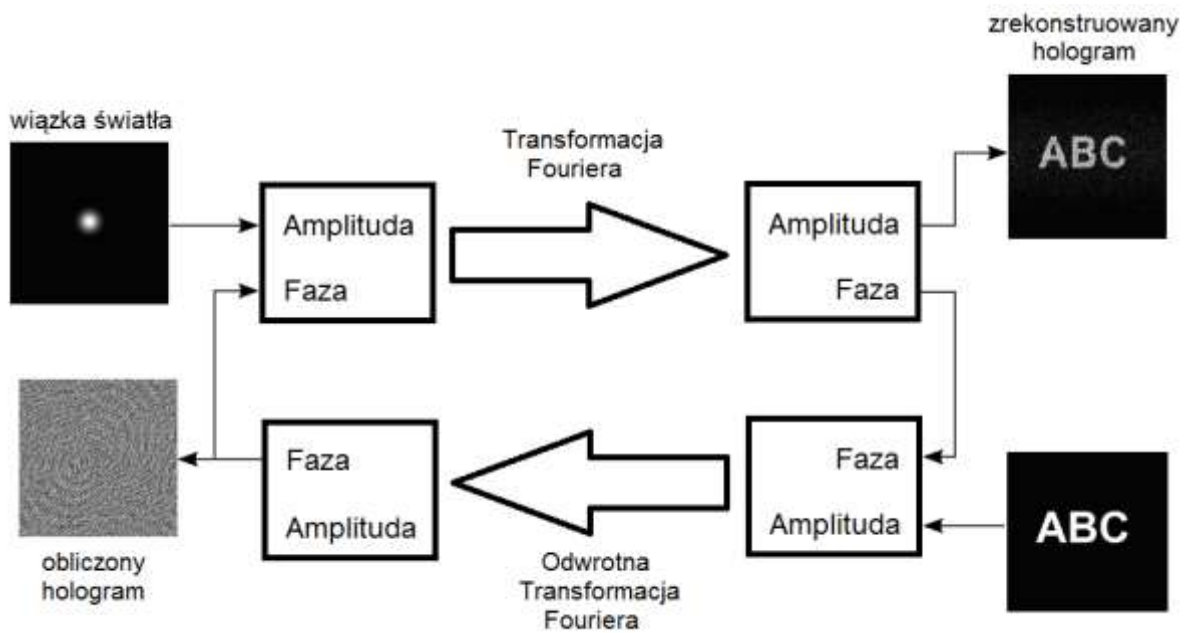
Aby wytworzyć klasyczny hologram, musimy posiadać rzeczywisty przedmiot, który zostanie zapisany w postaci hologramu. W przypadku hologramów generowanych komputerowo, rzeczywisty przedmiot zostaje zastąpiony plikiem graficznym zawierającym dowolny kształt (np. plik graficzny zawierający biały napis na czarnym tle). Wygenerowanie hologramu polega na komputerowym obliczeniu rozkładu interferencyjnego. Rozkład może zostać następnie naświetlony na kliszy lub płytce holograficznej. Światło lasera po przejściu przez taki hologram ugnie się w taki sposób, że utworzy wcześniej zaprojektowany wzór (np. napis).

Komputerowa generacja hologramu odbywa się za pomocą pewnego algorytmu zwanego algorytmem Gerchberga–Saxtona. Algorytm ten wymaga podania wejściowego rozkładu projektowanego natężenia (czyli np. wspomniany wcześniej plik graficzny z napisem) oraz rozkładu natężenia wiązki światła w jakiej hologram ma być odtworzony (również plik graficzny). W wyniku działania algorytmu otrzymuje się plik graficzny z rozkładem fazowym będącym projektowanym hologramem. Ogólna idea przedstawiona jest na Rys. 61.



Rys. 61 Dane wejściowe, na podstawie których powstaje hologram

Algorytm Gerchberga–Saxtona jest algorytmem iteracyjnym co oznacza, że pewne operacje (obliczenia) wykonuje się wiele razy. Każde kolejne wykonanie operacji powoduje, że otrzymujemy coraz lepszej jakości hologram. Schemat algorytmu G-S przedstawiony jest na Rys. 62.



Rys. 62 Algorytm Gerchberga–Saxtona

Poniżej przedstawiony jest skrypt, którym będziemy się posługiwać w celu komputerowego generowania hologramów. Obliczony hologram zostanie zapisany do pliku CGH.bmp. Przykładowe pliki wejściowe można pobrać ze strony holomakers.eu.

```

pkg install image
pkg load image

GaussianBeam = imread('GaussianBeam.png');
GaussianBeam = rgb2gray(GaussianBeam);

StartPhase=zeros(1024,1024);

Source=fft2(iff2(GaussianBeam));
StartPhase=fft2(iff2(StartPhase));
Source = abs(Source).*exp(1i*angle(StartPhase));

TargetImg = imread('target.bmp');
Target=fft2(iff2(TargetImg));

A = fftshift(iff2(fftshift(Target)));
for i=1:20
    B = abs(Source).* exp(1i*angle(A));
    C = fftshift(fft2(fftshift(B)));
    D = abs(Target).* exp(1i*angle(C));
    A = fftshift(iff2(fftshift(D)));
end
    
```

end

```
figure(1);imshow(GaussianBeam);title('Source Intensity');  
figure(2);imshow(Target);title('Expected Intensity');  
figure(3);imshow(angle(A),[]);title('Calculated Hologram');  
figure(4);imshow(abs(C),[]);title('Reconstructed Intensity');
```

```
A=angle(A);  
A=A-min(A(:));  
A=A/max(A(:));
```

```
imwrite(A, 'CGH.bmp');
```