

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

Guía de referencia técnica de HOLOMAKERS

Socio Principal: WUT

Autores: Artur Sobczyk (WUT)

Circulation: *Public*

Version: *02*

Stage: *Final*

Date: *2018*

Contributions

Karol Kakarenko, Warsaw University of Technology
Rene Alimisi, EDUMOTIVA
Jose Carlos Sola, AIJU

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 Where other published and unpublished source materials have been used, these have been acknowledged.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.

Chapter 1.	Entendiendo las ondas	7
1.1.	Ondas en la física.....	7
1.2.	Features describing waves	8
1.2.1.	Amplitude	8
1.2.2.	Frequency	8
1.2.3.	Period	9
1.2.4.	Wavelength	Error! Bookmark not defined.
1.2.5.	Phase	10
1.2.6.	Other features	10
1.3.	Properties of waves.....	11
1.3.1.	Diffraction.....	Error! Bookmark not defined.
1.3.2.	Interference.....	11
1.3.1.	Coherence	13
1.4.	Waves in 3D space.....	13
1.4.1.	Spherical wave and plane wave	13
1.4.2.	Huygens–Fresnel principle	14
1.5.	Light as a wave	15
Chapter 2.	17
2.1.	Holography in simple words.....	17
2.2.	The principle of the hologram formation.....	Error! Bookmark not defined.
2.3.	Types and properties of holograms	Error! Bookmark not defined.
2.3.1.	Types of holograms	Error! Bookmark not defined.
2.3.1.	Properties of holograms.....	Error! Bookmark not defined.
2.4.	Basic setups for hologram recording	23
2.4.1.	Gabor Hologram	23
2.4.1.	Leith-Upatnieks setup	23
2.4.1.	Rainbow (Benton) hologram	24
2.4.1.	Volume hologram.....	25
2.4.1.	Computer generated holograms.....	26
Chapter 3.	Image processing with the use of Octave environment	Error! Bookmark not defined.
3.1.	Installation of the Octave	28
3.2.	Starting the program	29
3.3.	Command window	30
3.4.	Editor	32
3.5.	Basic programming issues in the Octave environment.....	34
3.5.1.	Variables.....	34
3.5.2.	Basic input / output operations	36
3.5.3.	Operators	37
3.5.4.	Conditional statement.....	Error! Bookmark not defined.
3.5.5.	"For" loop	41
3.5.6.	Drawing charts	Error! Bookmark not defined.
3.6.	Image processing.....	Error! Bookmark not defined.
3.6.1.	Creating an image.....	Error! Bookmark not defined.

3.6.2.	Reading / writing image and displaying it on the screen...	Error! Bookmark not defined.
3.6.3.	Color conversion.....	Error! Bookmark not defined.
3.6.4.	Rotation	47
3.6.5.	Addition, subtraction, multiplication, division....	Error! Bookmark not defined.
3.6.6.	Fourier transform	48
3.7.	Algorithm for calculating Computer Generated Holograms (CGH) ..	Error! Bookmark not defined.

Capítulo 1. Entendiendo las ondas

1.1. Ondas en la física

En física, la onda es una perturbación que se propaga en el medio o el espacio. Por ejemplo, si arrojas una piedra al agua, una onda se elevará sobre la superficie del agua, extendiéndose más lejos de donde golpea la piedra. En este caso, la energía de impacto se convertirá en oscilaciones del medio (agua). Otro ejemplo puede ser la onda de sonido. En este caso, la onda también viaja gracias a las oscilaciones del medio (aire).

También podemos llamar a una onda una oscilación que está limitada en el espacio. Por ejemplo, mover el péndulo hace que el movimiento oscilatorio regrese de vez en cuando al mismo punto. En este caso, la onda será un ancho de oscilación del péndulo que cambiará con el tiempo. La figura 1 ilustra el cambio en el swing del péndulo a lo largo del tiempo. El péndulo se desvía en el rango $[-A, A]$, por lo que podemos decir que A es la amplitud del swing.

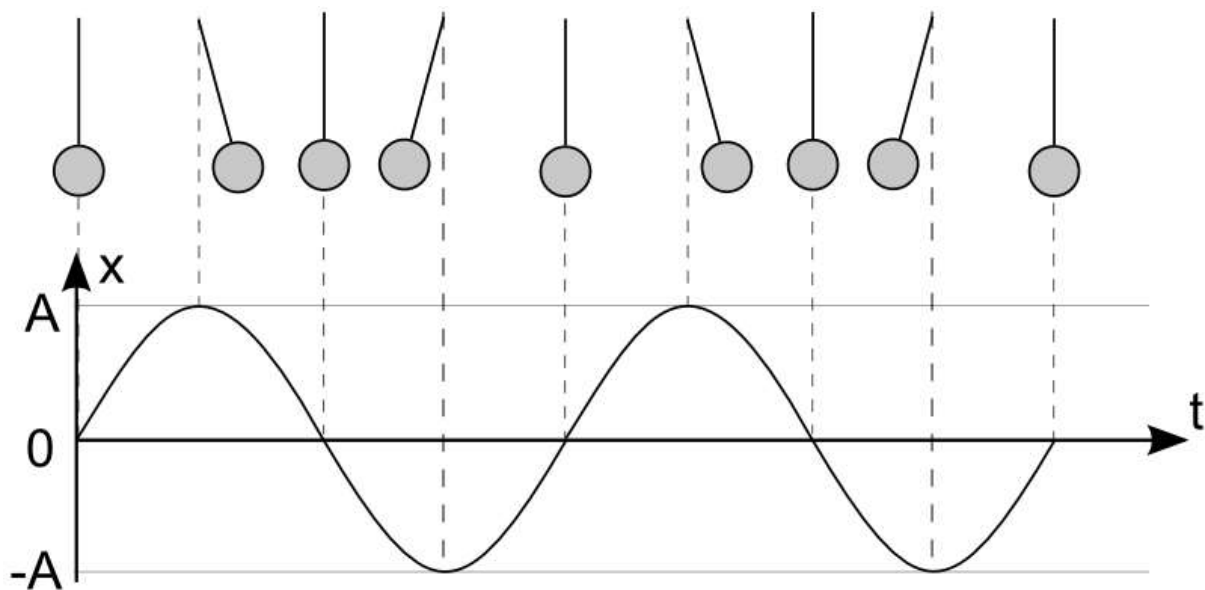


Fig. 1 - Desplazamiento del péndulo en diferentes momentos.

Muy a menudo las oscilaciones se pueden describir usando la función seno. Esto tiene lugar en el péndulo ya mencionado, pero también en el caso de un peso suspendido en un resorte, o en el caso de una cuerda de instrumentos musicales. Este es el tipo básico de oscilación. Este tipo de oscilación se denomina **oscilaciones armónica**. Como aprenderemos más adelante, cualquier otro tipo de oscilación más complicada consiste en oscilaciones armónicas. En conclusión, la onda armónica es una onda que puede describirse utilizando la función seno. En el siguiente capítulo aprenderemos las propiedades básicas que se usan para describir las olas.

1.2. Características que describen ondas

1.2.1. Amplitud

La amplitud de la onda es la distancia de la mayor desviación desde la posición de equilibrio. En el caso de un péndulo, la amplitud de la oscilación es su mayor desviación (ver Fig. 2). El valor máximo de onda se llama cresta y el valor mínimo es el canal.

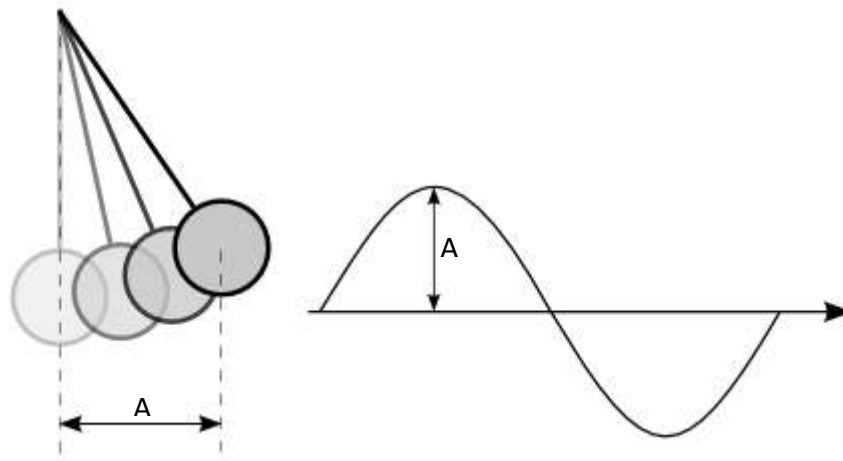


Fig. 2 - Amplitud de las oscilaciones del péndulo.

1.2.2. Frecuencia

La frecuencia nos dice cuántas oscilaciones se han realizado por unidad de tiempo (en un segundo). La unidad de frecuencia es el Hz (Hercio). Por ejemplo, 10Hz son 10 oscilaciones por segundo, 15.5Hz son 15.5 oscilaciones por segundo. La figura 3 presenta dos ondas con diferentes frecuencias. La línea roja es una oscilación completa. La frecuencia de la primera onda es de 3 Hz (3 oscilaciones por segundo) y la segunda de 6 Hz (6 oscilaciones por segundo).

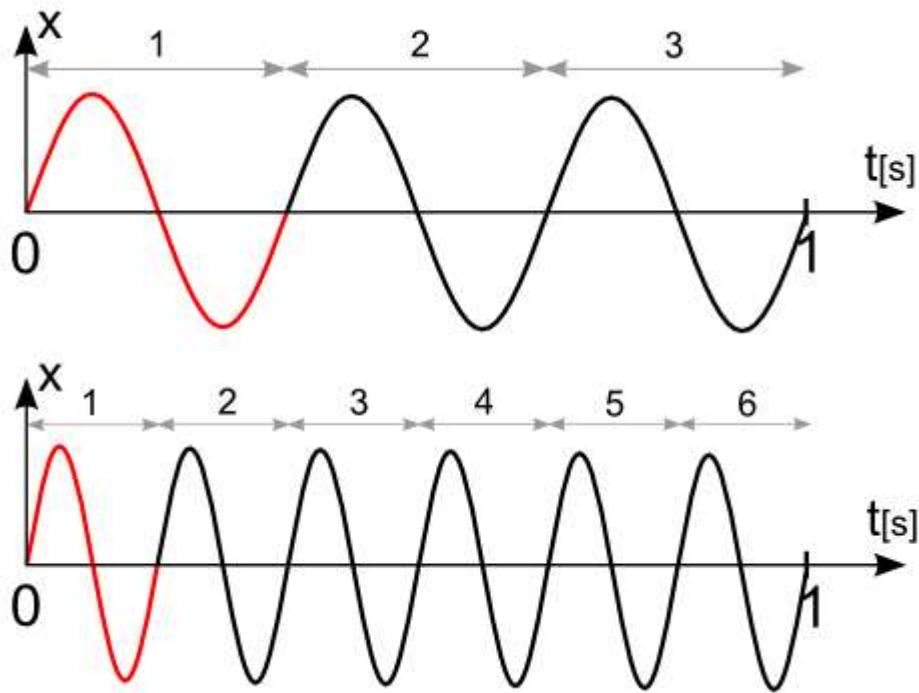


Fig. 3 Frecuencia de las ondas

1.2.3. Periodo

El período de onda está directamente relacionado con su frecuencia. Es el tiempo (medido en segundos) en el que se realizará una oscilación completa (marcada en rojo en la Fig. 3). El período de onda (T) está relacionado con su frecuencia (f) con dependencia $T = \frac{1}{f}$.

Así, el período de la primera onda en la Fig. 3 es $\frac{1}{3}s$ mientras que el periodo de la segunda onda es igual a $\frac{1}{6}s$

1.2.4. Longitud de onda

Si la onda se propaga en el espacio, uno puede determinar su longitud de onda. Esta es la distancia que la onda propagará durante un período. La longitud de onda (λ) es proporcional a la ecuación $\lambda = vT$, donde v es la velocidad de propagación de la onda y T es el periodo de onda.

1.2.5. Fase

La fase determina en qué parte del período de onda se ubica el punto de onda dado. Sin embargo, en la práctica, un elemento importante no es tanto la fase de onda única como el desplazamiento de fase entre ondas. En otras palabras, es simplemente un cambio de una onda a la otra. La Fig. 4a muestra una situación donde no hay cambio de fase, mientras que la Fig. 4b muestra ondas con un cierto cambio de fase. El cambio de fase se puede medir entre cualquier punto de onda correspondiente. Por ejemplo, en la Fig. 4, por conveniencia, el cambio de fase se muestra como la distancia entre el máximo de las ondas.

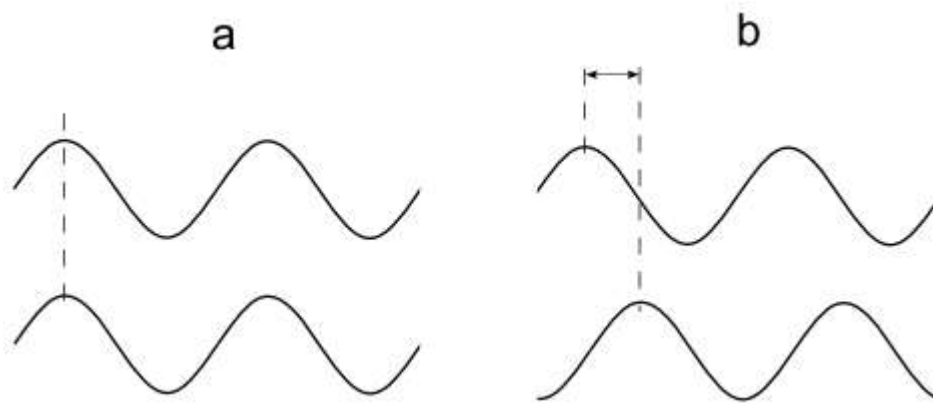


Fig. 4 Cambio de fase
a) no hay cambio de fase entre las ondas
b) ondas que tienen un cambio de fase

1.2.6. Otras características

Además de las características mencionadas anteriormente, también podemos distinguir la velocidad de grupo y la velocidad de fase de la onda. Las ondas pueden ser longitudinales o transversales. En el caso de las ondas transversales, también podemos hablar de polarización. Estas características no son necesarias para comprender los problemas holográficos, por lo que se omitirán en esta guía de referencia.

1.3. Propiedades de las ondas

Las ondas que se propagan en el espacio (por ejemplo, una onda en el agua, una onda de sonido, una onda electromagnética) muestran ciertas propiedades como la reflexión, la refracción, la difracción o la interferencia.

1.3.1. Difracción

La difracción se refiere a cambiar la dirección de la propagación de la onda como resultado de encontrar un obstáculo o una rendija. Este efecto es bien visible cuando la longitud de onda es comparable al tamaño de la ranura. La Fig. 5 presenta la situación en la que la onda se encuentra con una rendija. Antes de la ranura, la onda se mueve en una dirección perpendicular a la ranura (marcada con flechas rojas). También hay áreas de extinción total de las olas (marcadas con una línea azul). En este sentido, la amplitud de la onda es 0.

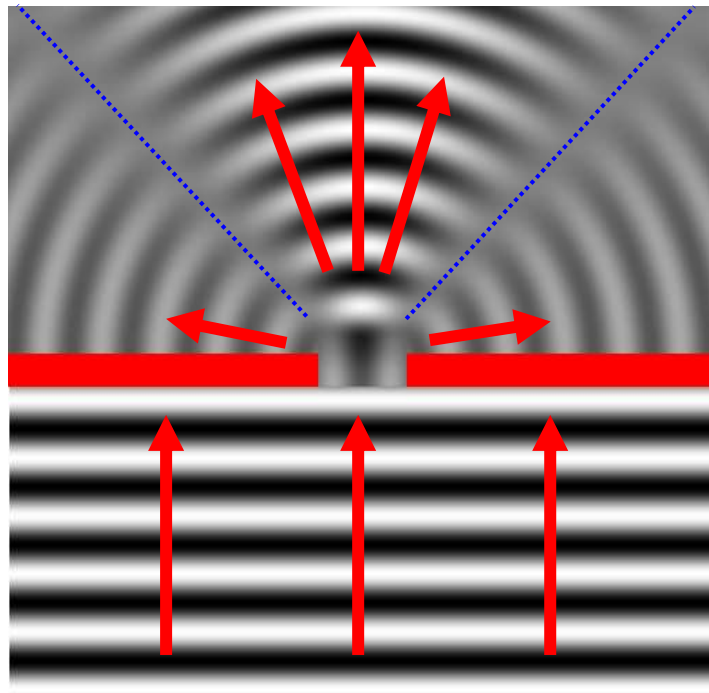


Fig. 5 Difracción en una sola ranura

1.3.2. Interferencia

La interferencia de ondas es el fenómeno en el que se basa la holografía. La interferencia es superposición (suma) de ondas. Las ondas propagantes interactúan entre sí, pueden fortalecerse o debilitarse. La Fig. 6 muestra la amplificación y extinción de las ondas. Si las ondas están "en fase" (la diferencia de fase es 0), se produce la mayor amplificación de onda posible. Si, por otro lado, las ondas están "fuera de fase" en la mitad del período (la diferencia de fase se encuentra en la mitad del período de onda), las ondas se desactivan por

completo. Cuando las ondas se fortalecen, es interferencia constructiva. Cuando las ondas se debilitan es interferencia destructiva.

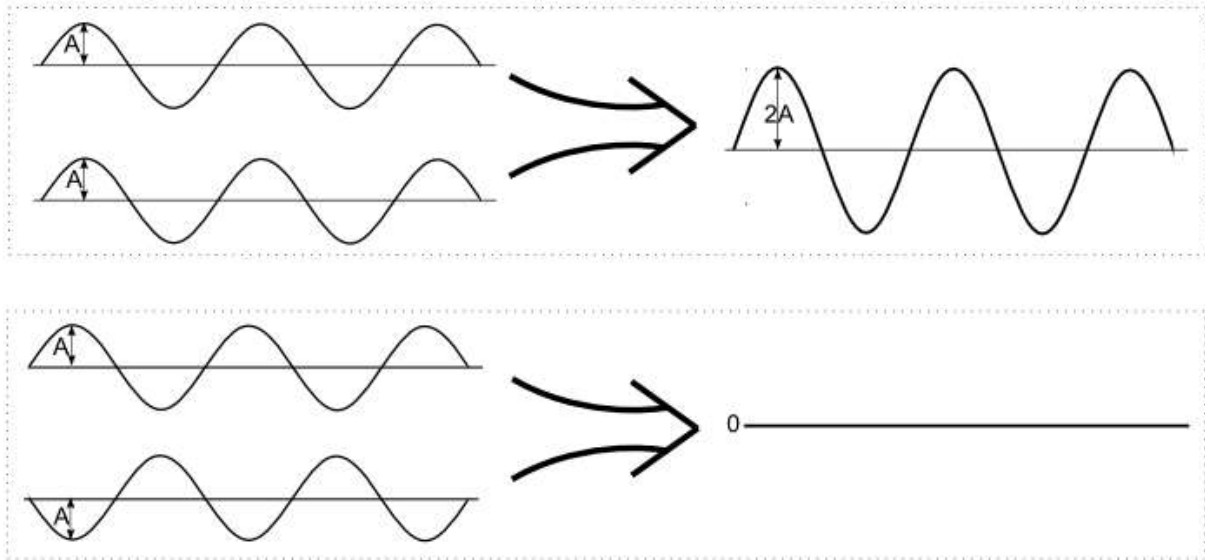


Fig. 6 Interferencia constructiva y destructiva.

La interferencia se puede ilustrar con el ejemplo de las ondas en el agua. Si estimulamos la superficie del agua en un punto con una frecuencia constante, obtendremos una onda que se propaga radialmente. Esto se muestra en la Fig. 7a. Las áreas negras representan los canales de la onda y las blancas representan las crestas. La figura 7b presenta dos ondas una al lado de la otra e interactuando entre sí. Las áreas grises son la amplificación de las áreas de onda.

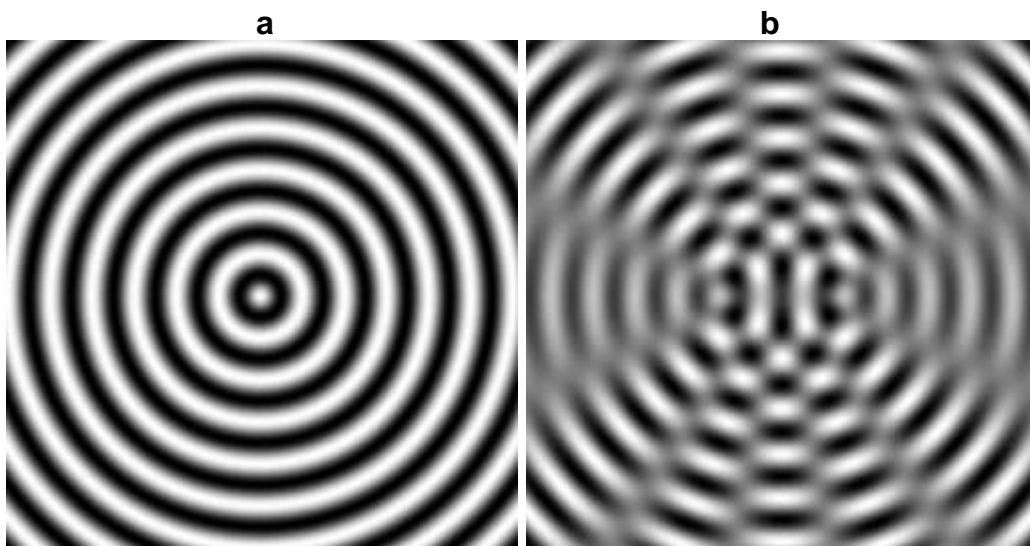


Fig. 7 Un ejemplo de olas en el agua.

1.3.1. Coherencia

La coherencia de las ondas es necesaria para obtener un patrón constante de interferencia en el tiempo. Dos ondas son coherentes si tienen una diferencia de fase constante. En la Fig. 8 se muestra cómo, de manera simplificada, se puede imaginar la diferencia entre ondas coherentes e incoherentes. La parte superior de la figura muestra ondas altamente coherentes. Como resultado de la superposición de estas ondas, las franjas de interferencia formadas son constantes en el tiempo. Vemos, por lo tanto, un patrón de interferencia con buen contraste (las franjas son claramente visibles). En el caso de que las ondas tengan un bajo grado de coherencia, las franjas resultantes en cualquier momento del tiempo se desplazan entre sí de alguna manera. Nuestros ojos ven esto como una imagen borrosa del patrón de interferencia (con contraste reducido). El menor grado de coherencia, la imagen de interferencia es más borrosa. En el caso de ondas totalmente incoherentes, no observaremos un patrón de interferencia. La mayor parte de la luz en la naturaleza son olas incoherentes. Sin embargo, una buena fuente de ondas coherentes es un láser.

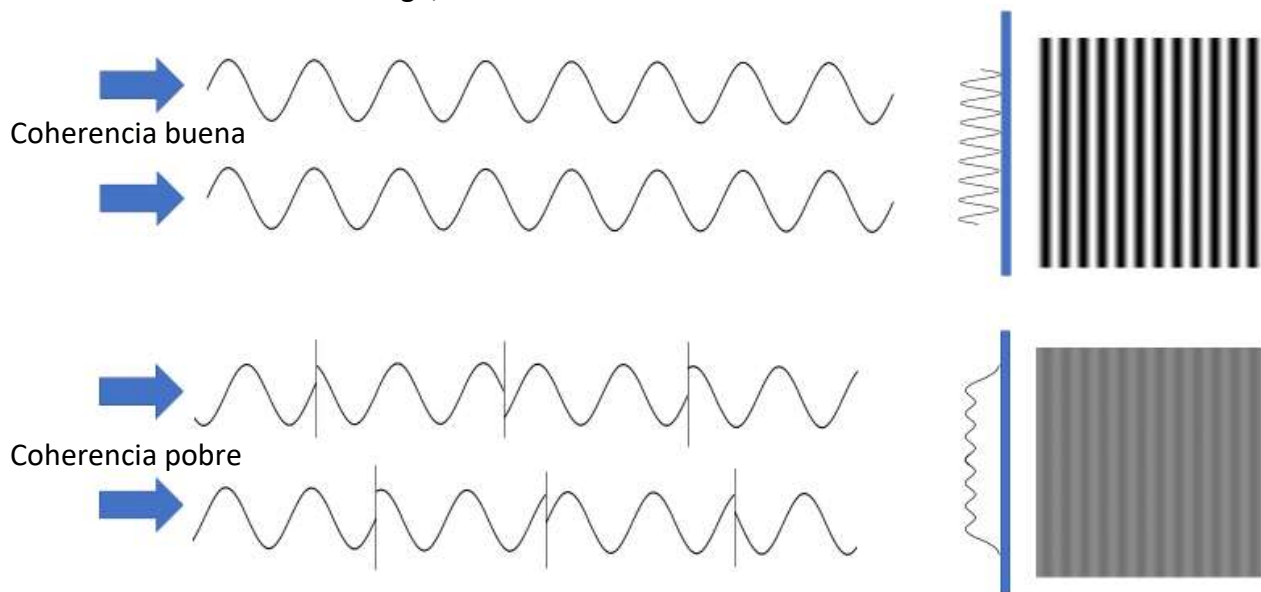


Fig. 8 Es necesaria una buena coherencia para obtener un patrón de interferencia de alto contraste.

1.4. Ondas en el espacio 3D

1.4.1. Onda esférica y onda plana

La onda en el espacio 3D puede representarse como una superficie donde la fase es constante. El ejemplo más común es una onda esférica o la llamada onda plana (Fig. 9). En el primer caso (Fig. 9a), las superficies de la fase constante (es decir, las superficies en las que la onda tiene el mismo valor, por ejemplo, alcanza el valor máximo) tienen la forma de

esferas (de ahí el nombre de la esfera esférica). En el segundo caso (Fig. 9b), las superficies de la fase constante tienen la forma de planos (de ahí el nombre de la onda plana).

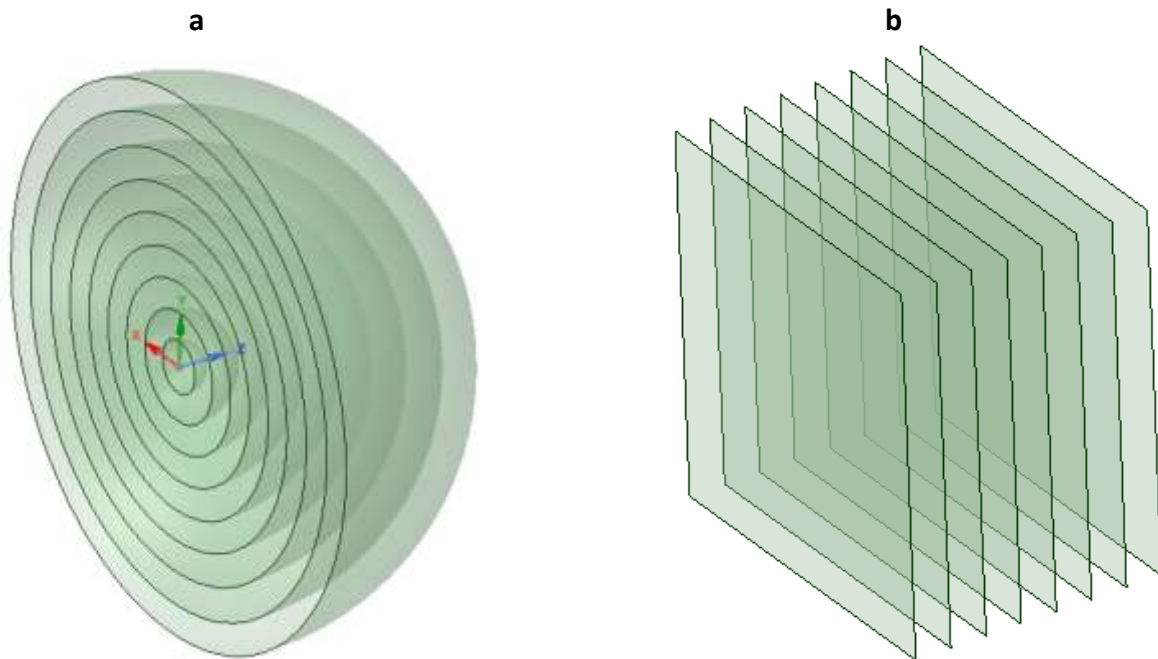


Fig. 9 Onda esférica (a) y onda plana (b)

1.4.2. Principio de Huygens–Fresnel

Considere el caso cuando la onda llega a un objeto (Fig. 10). Se puede suponer que cada punto al que llega la onda se convierte en la fuente de una nueva onda esférica. La suma de todas estas ondas esféricas (es decir, la interferencia) determina la forma de la nueva onda. En la figura 10, la onda incidente está marcada en negro. El objeto al que llega la onda se puede representar como un conjunto de puntos infinitos que se muestran en la figura como puntos verdes. Cada uno de estos puntos es la fuente de la onda esférica secundaria. Después de sumar todas las ondas secundarias, obtenemos una nueva onda (línea roja punteada en la figura) que luego se propaga en el espacio. Este es el principio de Huygens-Fresnel, que dice lo siguiente:

Cada punto al que llega una onda se convierte en una fuente de una onda esférica; la suma de estas ondas secundarias determina la forma de la onda en cualquier momento posterior.

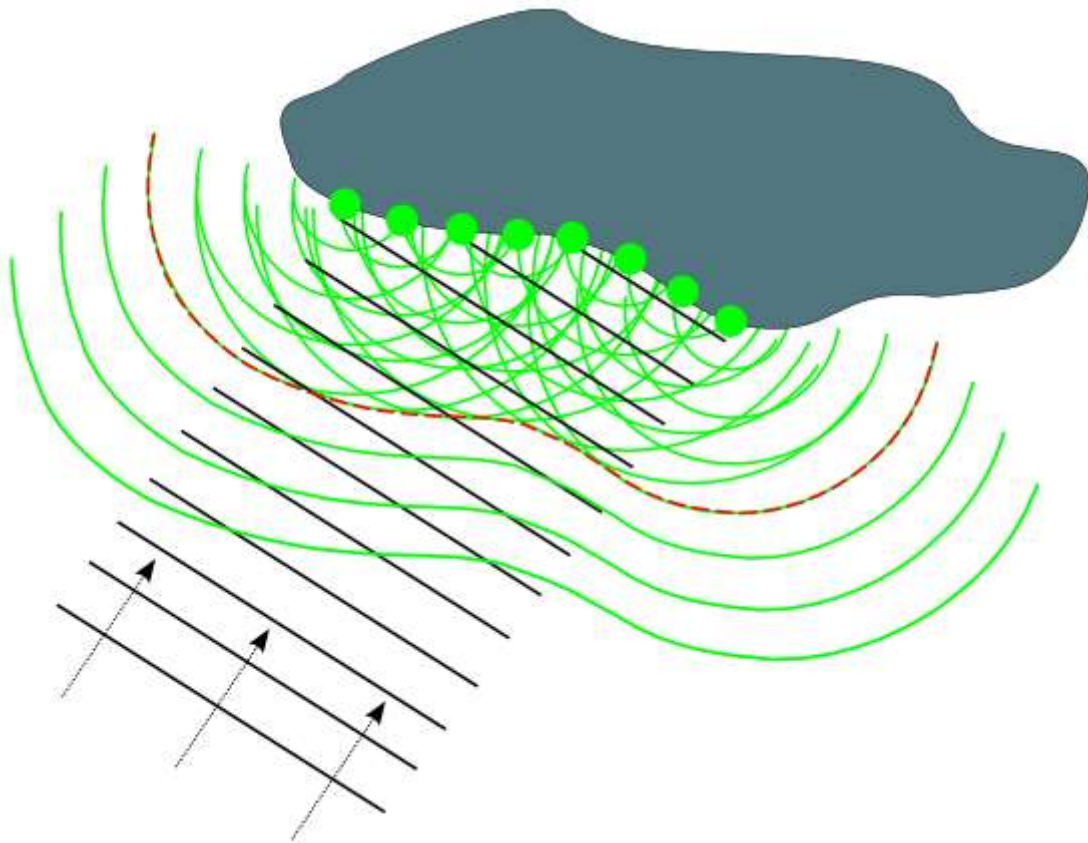


Fig. 10

1.5. Luz como una onda

La luz es también una onda. Si es así, entonces, ¿qué es lo que realmente se mueve allí? Lo que se agita es el campo eléctrico y el campo magnético. Por eso se dice que la luz es una onda electromagnética (EM). Se puede imaginar que tal onda aumenta y disminuye los campos eléctricos y magnéticos que se propagan en el espacio (Fig. 11). El campo eléctrico está marcado en rojo y el campo magnético en azul.

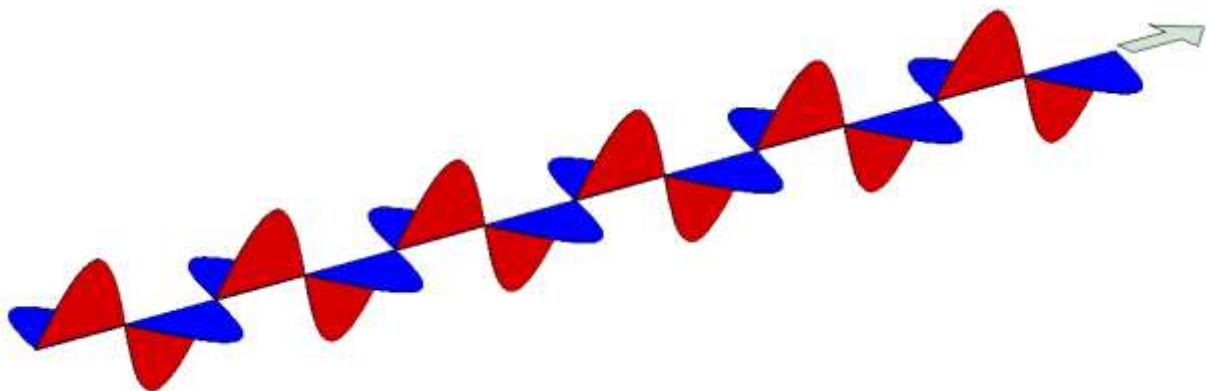


Fig. 11 Onda electromagnética

En la naturaleza, hay muchos tipos de radiación electromagnética. Se clasifican debido a la velocidad de los cambios del campo electromagnético en el espacio (es decir, debido a la frecuencia de las ondas). Las ondas EM con la frecuencia más alta son rayos *gamma*, entonces tenemos rayos X, rayos ultravioleta, luz visible, infrarrojos, microondas y ondas de radio como la frecuencia de radiación EM más baja (Fig. 12).

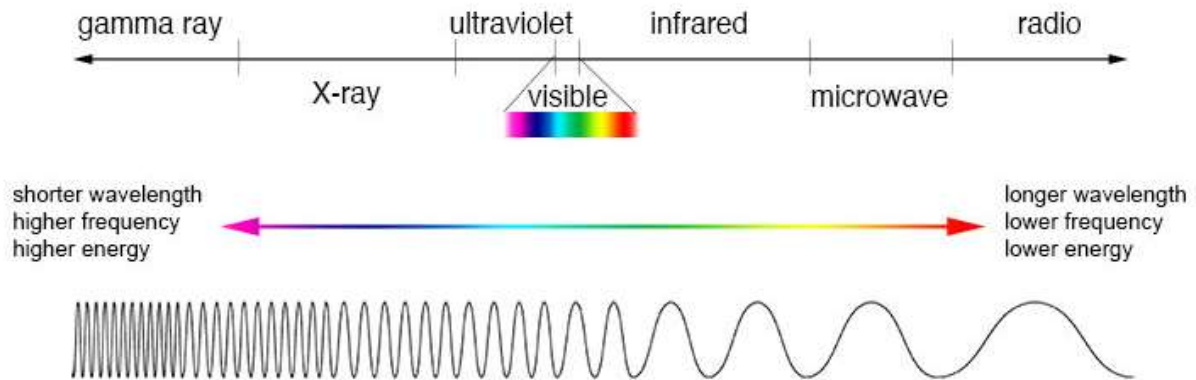


Fig. 12 El espectro de las ondas electromagnéticas. Fuente: NASA's Imagine the Universe
(<https://imagine.gsfc.nasa.gov/science/toolbox/emspectrum1.html>)

Capítulo 2.

2.1. Holografía en simples palabras

Un holograma es una placa o una placa de cristal en cuya superficie o en su volumen se registra un patrón de interferencia. La luz que pasa o se refleja desde el holograma crea una imagen idéntica a la luz reflejada desde el objeto real (vea la Fig. 13).

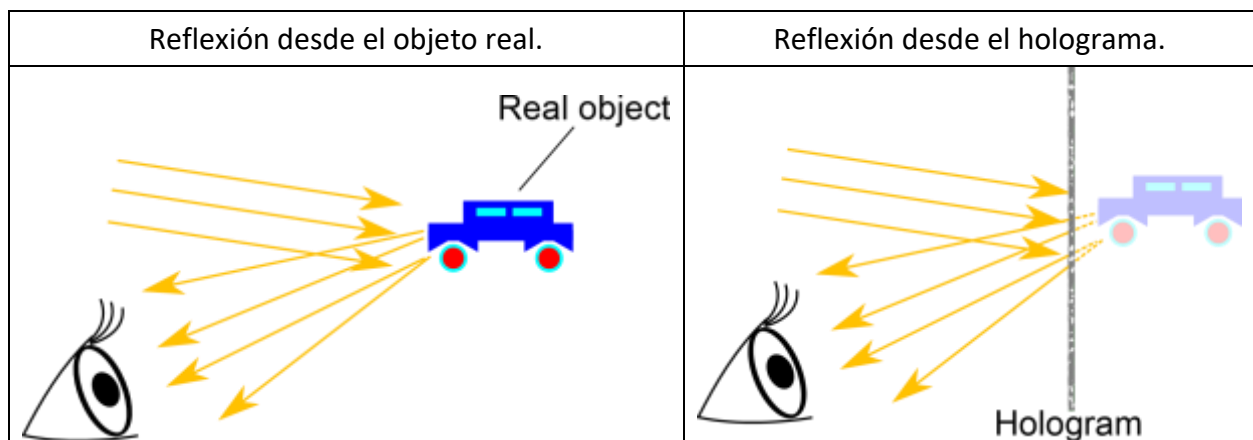


Fig. 13 La luz reflejada desde el holograma se comporta de manera idéntica como si viniera de un objeto real

Un rasgo característico del holograma es que el objeto observado es completamente tridimensional. Esto significa que al mirar el holograma en un cierto ángulo, podemos ver detalles que son invisibles cuando se mira el mismo holograma desde un ángulo diferente (ver Fig. 14). Este tipo de propiedad no tiene fotos ordinarias; Las fotos se ven iguales sin importar el ángulo de observación.

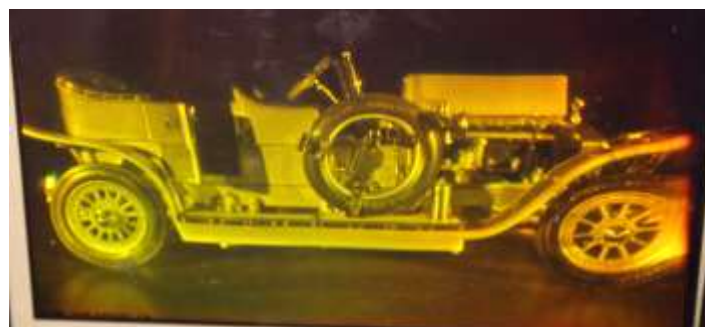




Fig. 14 La imagen visible en el holograma cambia con el ángulo de observación.

2.2. El principio de la formación de hologramas.

El fenómeno físico que crea un holograma es la interferencia. Ya sabemos que dos ondas coherentes interfieren entre sí para crear franjas de interferencia. Como un simple ejemplo, la Fig. 15 muestra dos ondas planas que interfieren entre sí para formar franjas rectas.

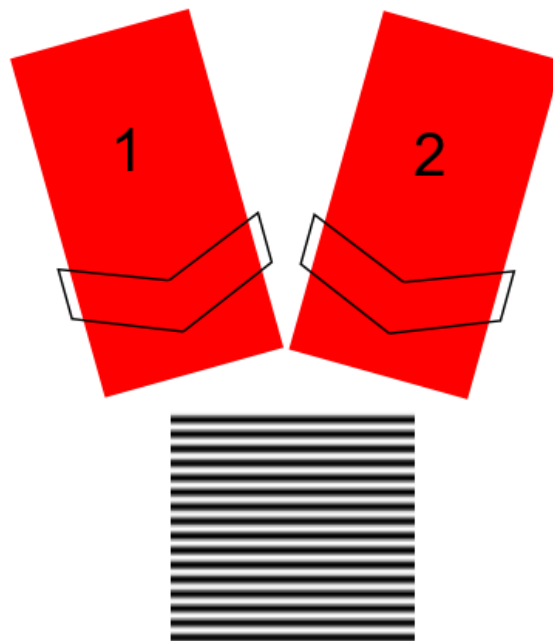


Fig. 15 Interferencia de dos ondas planas.

La interferencia de una onda plana con una fuente puntual se muestra en la Fig. 16. Las franjas de interferencia tienen la forma de círculos. Podemos decir que tal patrón de interferencia contiene información sobre un punto en el espacio. Si logramos registrar esta distribución, por ejemplo, en una película holográfica, entonces es posible reconstruir el punto decodificado iluminando el patrón con una onda plana.

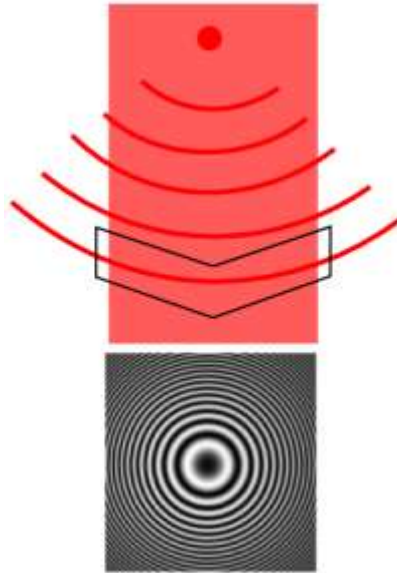


Fig. 16 La interferencia de una onda plana con una fuente puntual.

Cualquier objeto iluminado puede tratarse como un conjunto de muchas fuentes de puntos secundarias (principio de Huygens-Fresnel). El patrón de interferencia resultante de la interferencia de muchas fuentes puntuales con una onda plana es generalmente complicado (ver Fig. 17). Podemos decir que el objeto está codificado como un patrón de interferencia.

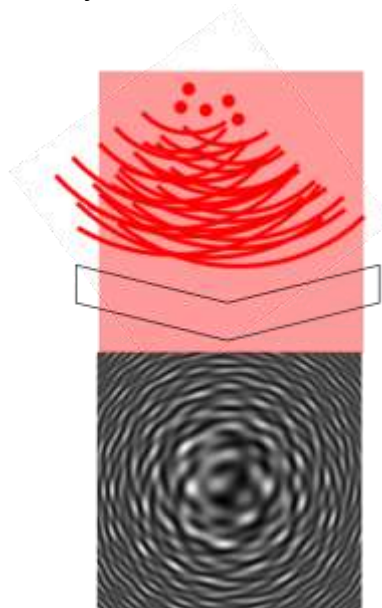


Fig. 17 La interferencia de una onda plana con muchas fuentes puntuales.

Tal patrón es un holograma. Este holograma se puede reconstruir iluminándolo con una onda plana. La figura 18 ilustra un ejemplo de la grabación y reconstrucción de un holograma. En este caso, durante la reconstrucción del holograma, obtenemos una imagen real (visible en la pantalla) y una imagen virtual (visible cuando miramos directamente al holograma iluminado).

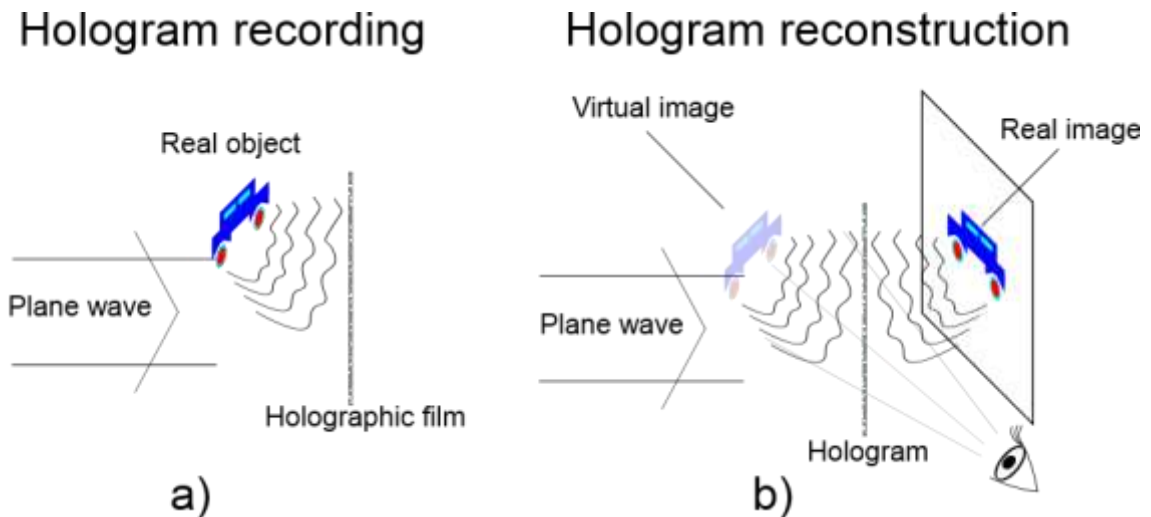


Fig. 18 Registro de un holograma (a) y su reconstrucción (b)

2.3. Tipos y propiedades de los hologramas

2.3.1. Tipos de hologramas

Los hologramas se pueden clasificar de diferentes maneras según sus propiedades. Una de las divisiones puede ser cómo la luz pasa a través del holograma. Decimos que el holograma puede ser modulado en amplitud o fase. La modulación de amplitud significa que el patrón de interferencia se registra en forma de áreas más o menos transparentes (Fig. 19a). El holograma de fase modulada es totalmente transparente y el patrón de interferencia se registra en forma de áreas con diferente índice de refracción (Fig. 19b).

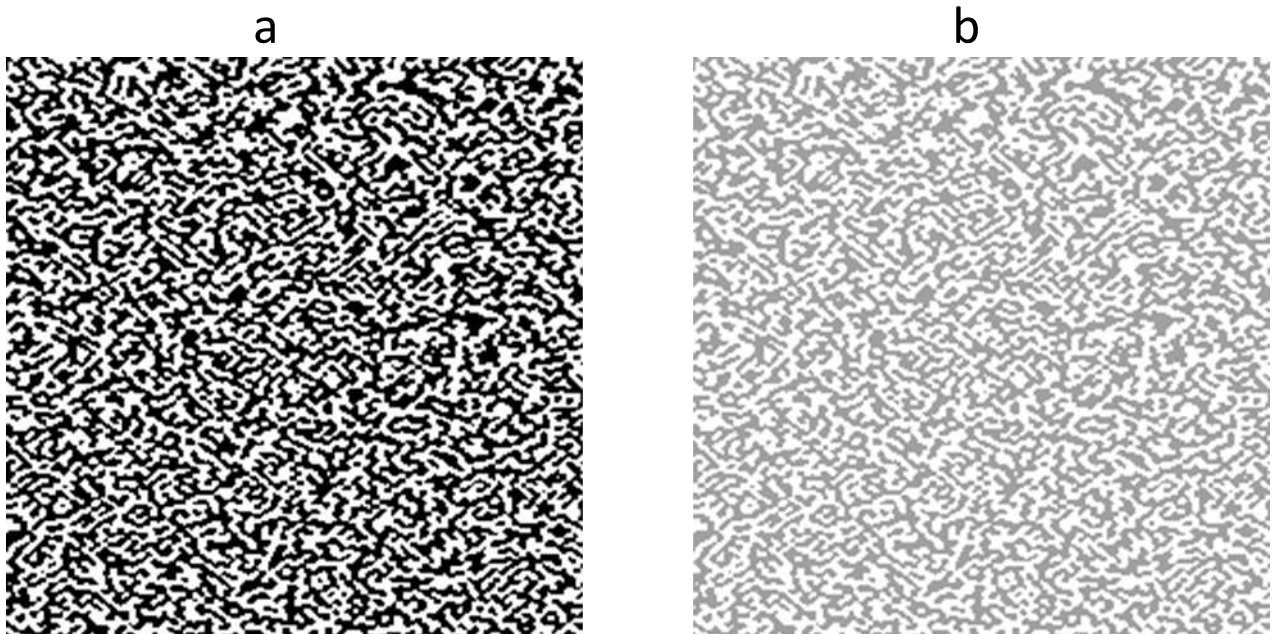


Fig. 19 Holograma modulado de amplitud (a) y fase (b).

Los hologramas también se pueden dividir en los llamados hologramas delgados y gruesos. Los hologramas delgados son aquellos que se comportan como si el patrón de interferencia se registrara solo en la superficie de la placa holográfica. Los hologramas gruesos son aquellos en los que el patrón de interferencia se registra en todo el volumen del holograma.



Fig. 20 Holograma delgado (a) y grueso (b)

La siguiente forma de describir hologramas es la transmisión y la reflexión de hologramas. El holograma de transmisión puede observarse transmitiendo luz a través de un holograma, mientras que el holograma de reflexión se observa reflejando la luz del holograma.



Fig. 21 Holograma de transmisión (a) y reflexión (b)

2.3.1. Propiedades de los hologramas

La holografía es un método fotográfico: la información sobre la imagen se registra en la placa holográfica. Sin embargo, las diferencias entre los hologramas y las fotos regulares son esenciales:

- En el caso de la holografía, la película debe tener una resolución mucho más alta (medida en líneas por milímetro).
- El negativo fotográfico contiene información solo sobre la intensidad de la luz, mientras que en el holograma (en forma de patrón de interferencia), la información sobre la intensidad y la fase de la luz (es decir, la distancia de los puntos del objeto desde el holograma) se codifica.
- Cada fragmento del negativo fotográfico contiene información sobre una parte diferente de la imagen, mientras que en el caso de un holograma, cada fragmento contiene información sobre la imagen completa. Esto significa que si cortamos el holograma en dos partes, cada una de ellas reconstruirá la imagen completa. Aunque habrá una diferencia en el brillo de la imagen y el rango de ángulo en el que podemos observar que el holograma disminuirá.

2.4. Configuraciones básicas para la grabación de hologramas

2.4.1. Holograma de Gabor

Históricamente, la primera configuración para grabar hologramas fue propuesta por Denis Gabor en 1948. Los láseres aún no se habían inventado en ese momento. Esta configuración utiliza una fuente de luz puntual, gracias a la cual es parcialmente coherente y es posible grabar un holograma. El objeto es una película translúcida (por ejemplo, un cartel o una imagen simple). Parte de la luz pasa directamente a través de la película y se dispersa. Ambos haces interfieren entre sí grabando en la película como patrón de interferencia (ver Fig. 22).

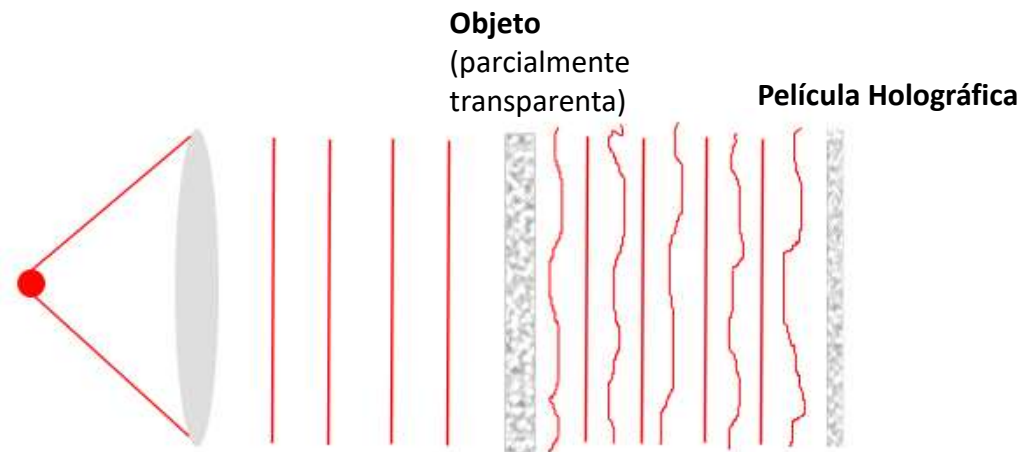


Fig. 22 1.1.1. Holograma de Gabor

2.4.1. Configuración de Leith-Upatnieks

En esta configuración, uno puede grabar el llamado holograma de Fresnel, que luego puede reconstruirse con la luz láser. El rayo láser se divide en dos (denominado haz de referencia y haz de objeto). La luz dispersada desde el objeto interfiere con el haz de referencia y se registra como un patrón de interferencia en la película holográfica (Fig. 23). En la Fig. 24 se muestra un ejemplo de un holograma de Fresnel.

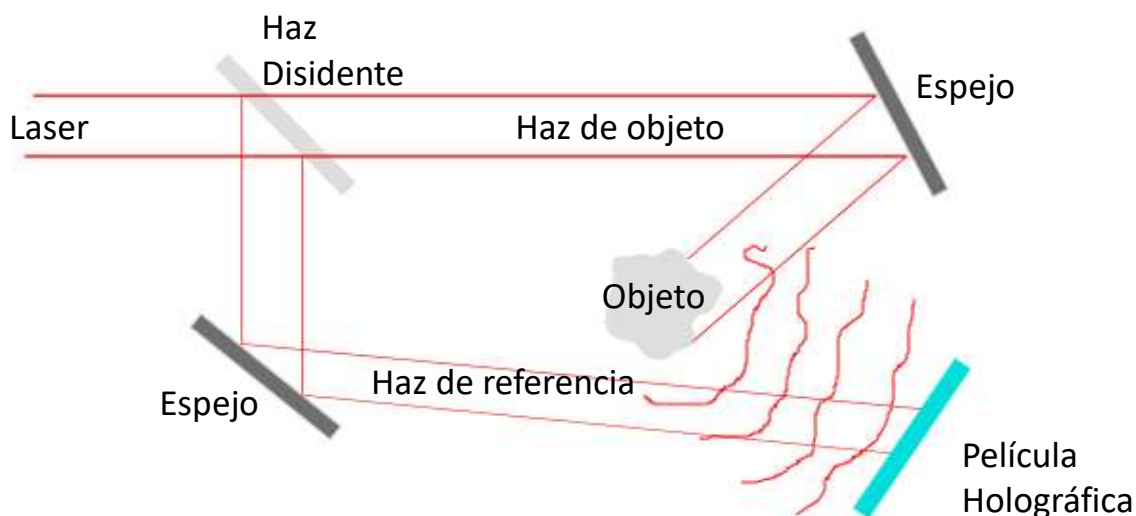


Fig. 23 Holograma de Fresnel grabado en la configuración de Leith-Upatnieks



Fig. 24 Un ejemplo del holograma de Fresnel.

2.4.1. Holograma del Arco Iris (Benton)

En esta configuración (Fig. 25), se utiliza un holograma de Fresnel (por ejemplo, previamente grabado en el sistema Leith-Upatnieks) como objeto. El holograma obtenido tiene la ventaja de que puede verse en luz blanca. Al girar el holograma ligeramente en una dirección, el efecto 3D es visible (es decir, llamado paralaje) mientras que en la otra dirección se puede observar el cambio de colores. Esto se ilustra en la Fig. 26.

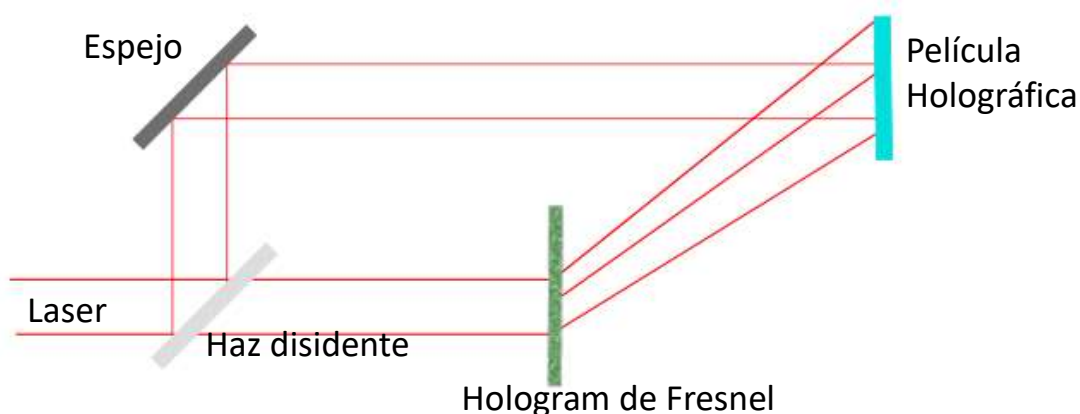


Fig. 25 Grabando un holograma de arcoiris

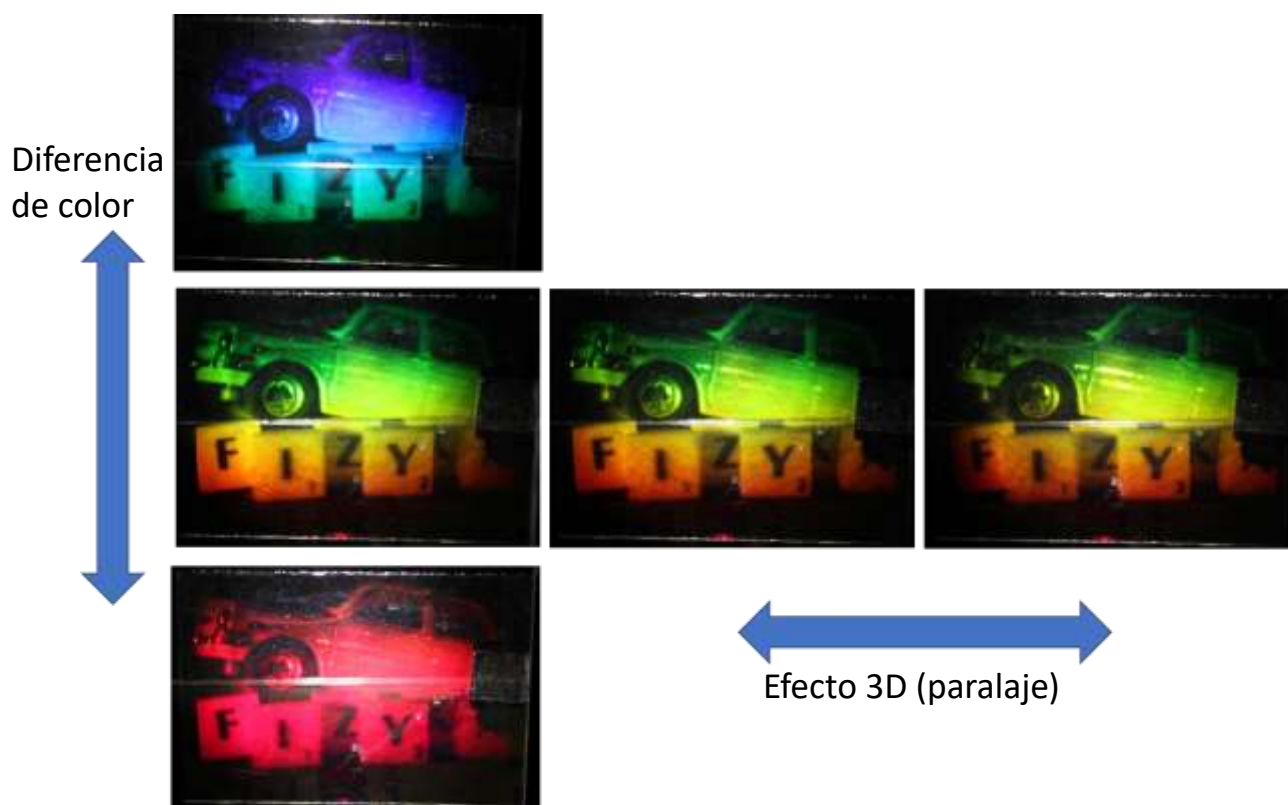


Fig. 26 Un ejemplo de holograma de arco iris.

2.4.1. Holograma de volumen

Este holograma se puede grabar en la configuración que se muestra en la Fig. 27. El holograma de volumen se puede ver en modo de transmisión en luz blanca. Un ejemplo de un holograma de volumen se muestra en la Fig. 28.

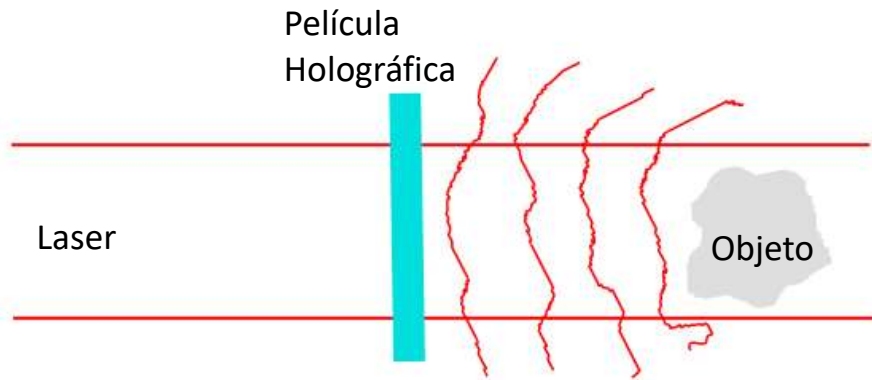


Fig. 27 Configuración para la grabación del holograma de volumen.



Fig. 28 Un ejemplo de holograma de volumen.

2.4.1. Hologramas generados por ordenador

No se necesita ningún objeto físico para hacer un holograma generado por computadora (CGH). En el caso más simple, la función del objeto reproduce un archivo gráfico con un signo o una forma bidimensional. Sobre la base de dicho archivo, un patrón de interferencia (también un archivo gráfico) se calcula utilizando la computadora. Se usa un algoritmo iterativo para los cálculos (el algoritmo Saxton de Gerchberg es el más popular). El patrón de interferencia calculado se puede realizar en forma de un elemento óptico (diapositiva). Si iluminamos las diapositivas con un rayo láser (por ejemplo, con un puntero láser), la luz que pasa formará una forma previamente diseñada. Un ejemplo de un holograma de este tipo puede ser el popular puntero láser, que generan una forma de diseño. La fabricación de un holograma de alta calidad requiere un equipo sofisticado. Sin embargo, en las condiciones del hogar, se puede hacer un CGH adecuado para fines educativos. Para este propósito, el patrón de interferencia calculado debe imprimirse, por ejemplo, en papel A4. Luego, una foto del patrón impreso debe tomarse con una cámara analógica. Después de eso, todo lo que tienes que hacer es desarrollar la película y recibir un

holograma generado por computadora. El proceso de obtención de dichos hologramas se presenta en la Fig. 29.

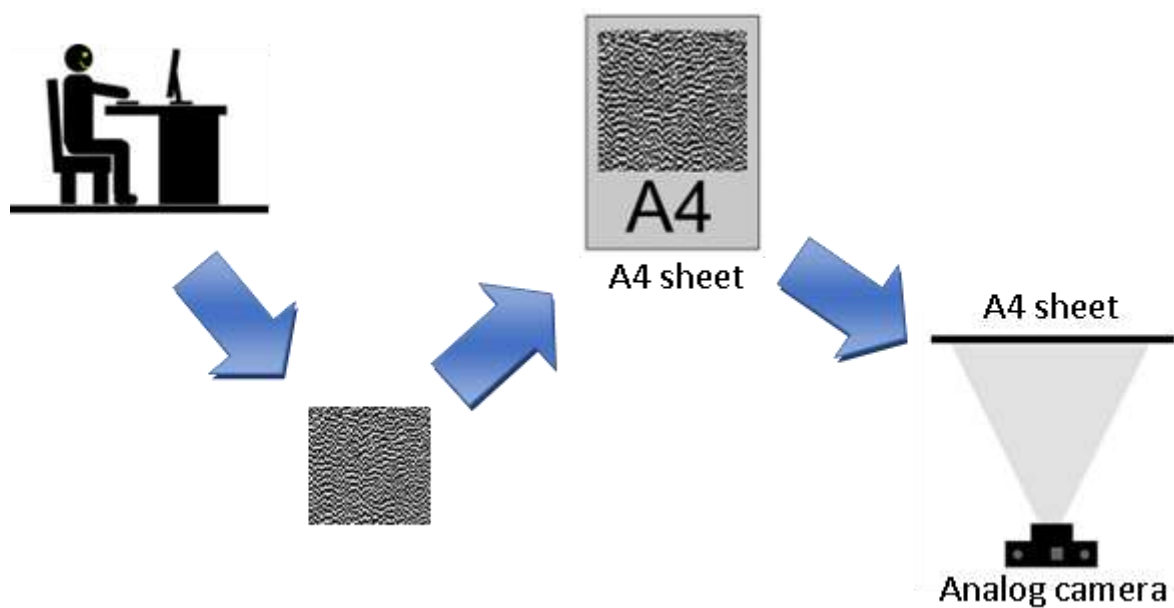


Fig. 29 El proceso de obtención de CGH en condiciones de hogar.

Capítulo 3. Procesamiento de imágenes con el uso del software Octave.

3.1. Instalación de Octave

El programa Octave debe descargarse del sitio web:

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

Para hacer esto, haga clic en el botón de descarga de acuerdo con la Fig. 30:

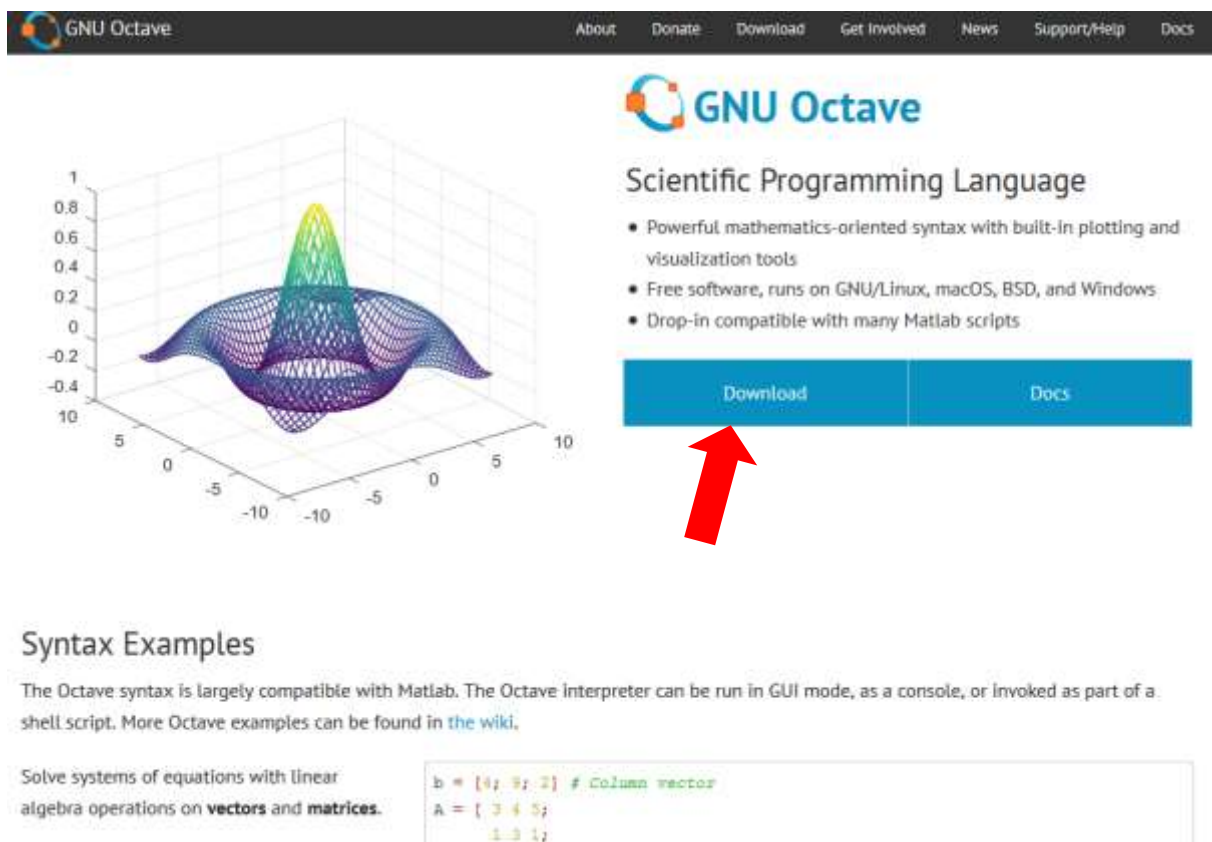


Fig. 30 Descargando Octave

Luego ve a la pestaña de Windows y descarga el archivo [octave-4.2.1-w32-installer.exe](#) o [octave-4.2.1-w64-installer.exe](#) dependiendo de si debe ser una versión de 32 bits o de 64 bits (Fig. 31).

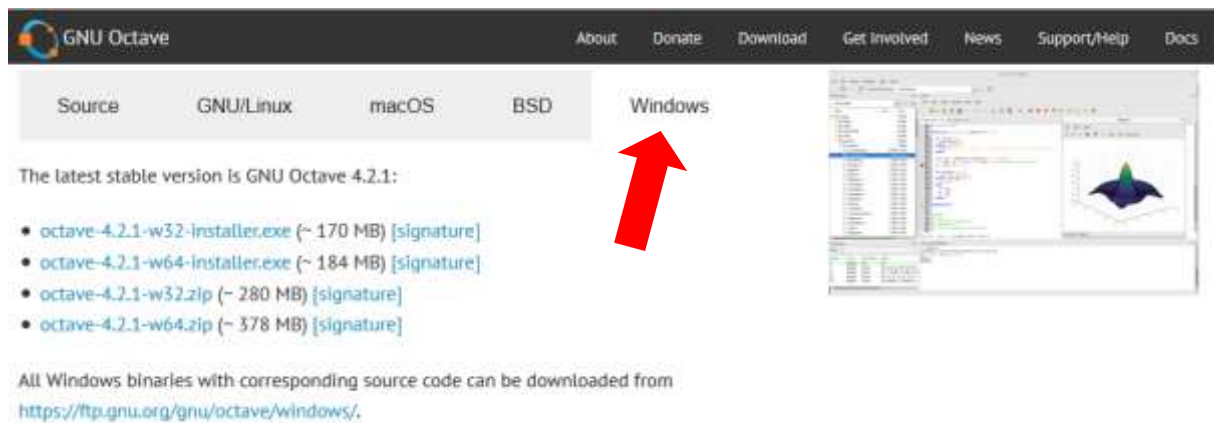


Fig. 31 Descargando Octave

3.2. Iniciando el programa

Si trabaja con Windows, haga clic en el menú Inicio y comience a escribir "Octave" en el campo de búsqueda. Luego abra el archivo "Octave (GUI)". Aparecerá la vista principal, en la que hay varias ventanas (Fig. 32). El más importante de ellos es el área para ingresar comandos (marcada con el número 1), luego una ventana que permite elegir una carpeta en la que trabajaremos (marcada con el número 2) y, por último, un explorador de archivos en la carpeta actual (número 3).

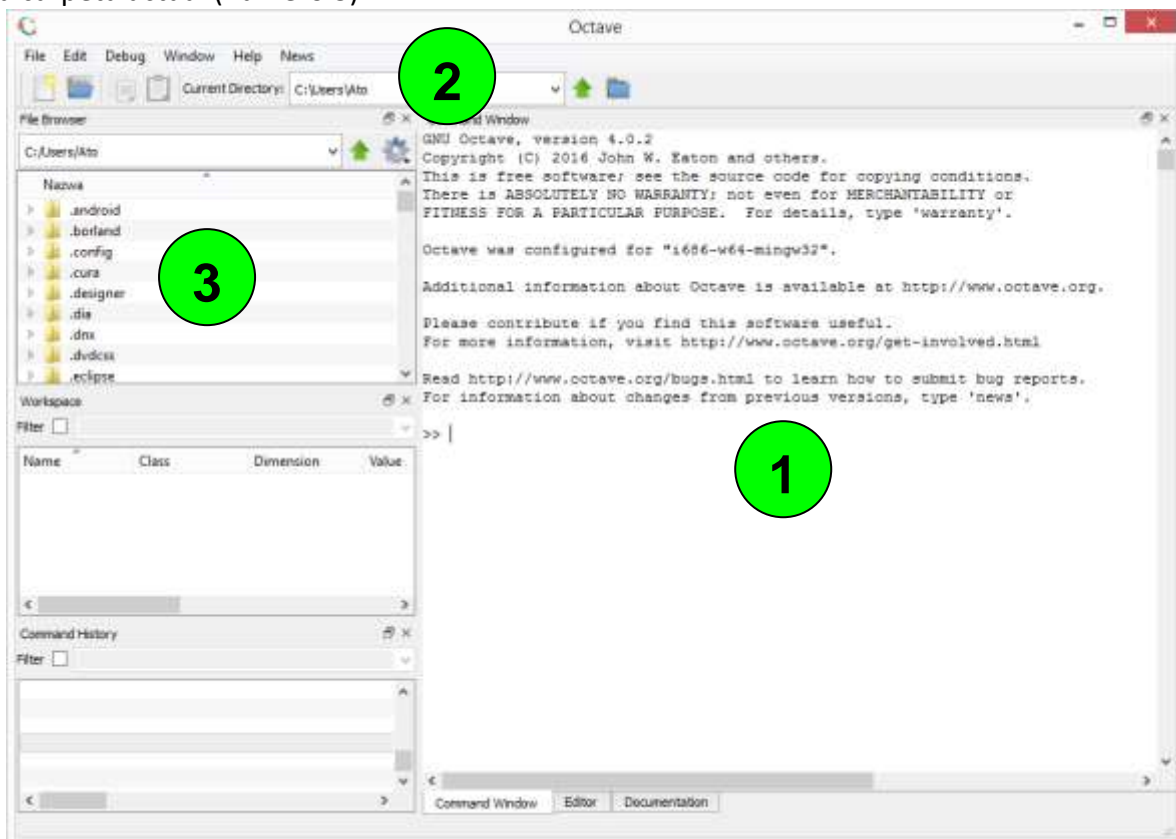


Fig. 32 Iniciando el programa Octave

3.3. Ventana de Comandos

El ingreso de comandos se puede hacer de dos maneras: usando la "Ventana de comandos" y usando el "Editor". En el primer caso, cada comando se ejecuta inmediatamente después de presionar la tecla Intro. En el segundo caso, el llamado script debe escribirse (una serie de comandos), que después de ejecutarse el script se ejecutará línea por línea. En este momento, puede ser un poco incomprensible, por lo que a continuación se presentan algunos ejemplos.

El cambio entre la ventana de comandos y el editor se muestra en la Fig. 33.

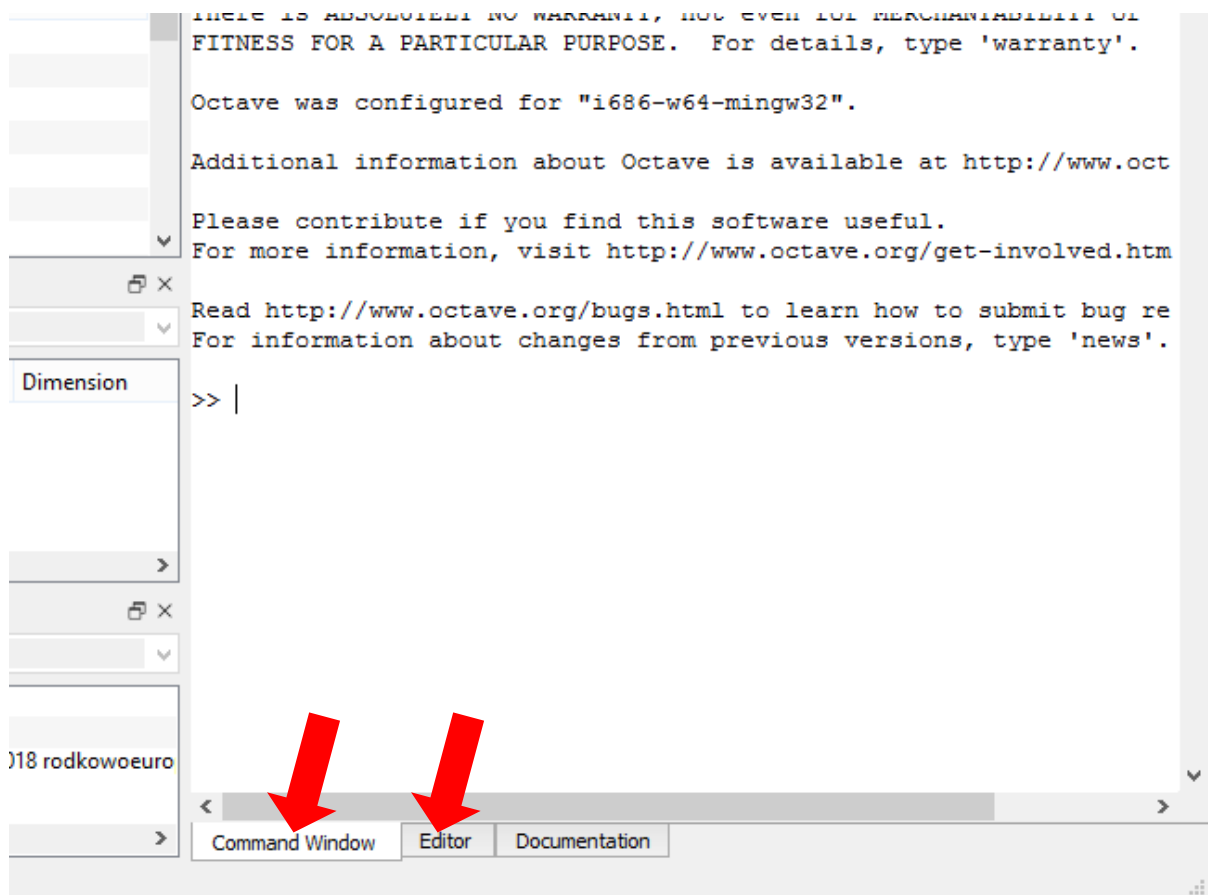


Fig. 33 Ventana de comandos y editor

La ventana de comandos se utiliza para realizar cálculos cortos y rápidos. Por ejemplo, si queremos calcular el valor de la expresión $\cos^2\left(\frac{\pi}{4}\right)$ luego en la ventana solo ingrese la expresión `(cos(pi/4))^2`. El operador `^` significa elevar a la potencia (en nuestro caso a la potencia de 2). Después de ingresar la expresión anterior y confirmarla con la tecla Intro, recibiremos inmediatamente el resultado como se muestra en la Fig. 34.

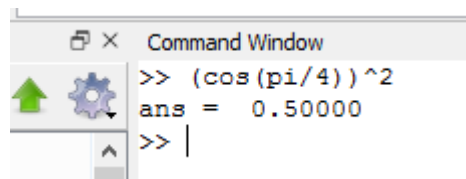


Fig. 34 Resultado del cálculo

El resultado de nuestra expresión se asigna automáticamente con la variable llamada "ans". Podemos usar esta variable de una manera muy simple para cálculos posteriores. Por ejemplo, si queremos que el resultado vuelva a elevarse a la potencia de 2, simplemente escriba la expresión de teclado `ans ^ 2` como se muestra en la Fig. 35. Luego, la variable "ans" tomará un nuevo valor igual en este caso a 0,25.

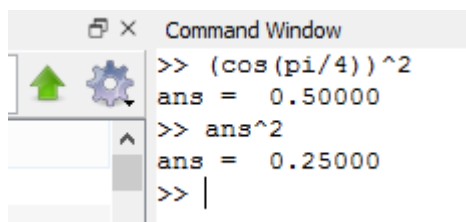


Fig. 35 Cálculos adicionales

Por supuesto, podemos almacenar diferentes valores en variables creadas independientemente. Si, por ejemplo, nos gustaría recordar un valor de 5 en la variable que llamaremos, por ejemplo, "a", entonces podemos escribir `a = 5` como se muestra en la Fig. 36.

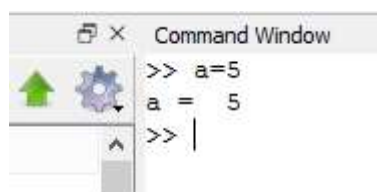


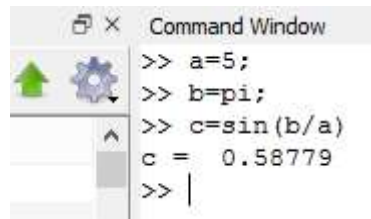
Fig. 36 Declaración de variable

De esta manera, declaramos una nueva variable llamada "a" y al mismo tiempo le asignamos un valor de 5. Después de presionar la tecla Intro, el resultado de nuestro comando se imprimió en la pantalla (`a = 5`). Si queremos que los resultados no se muestren en la pantalla, el comando debe finalizar con un punto y coma como se muestra:



Fig. 37 El resultado del comando no se muestra explícitamente.

Puede crear varias variables de esta manera y realizar varios cálculos (Fig. 38).



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

Fig. 38 Los cálculos

Todas las variables declaradas hasta ahora se pueden ver en la ventana "Área de trabajo" (Fig. 39).

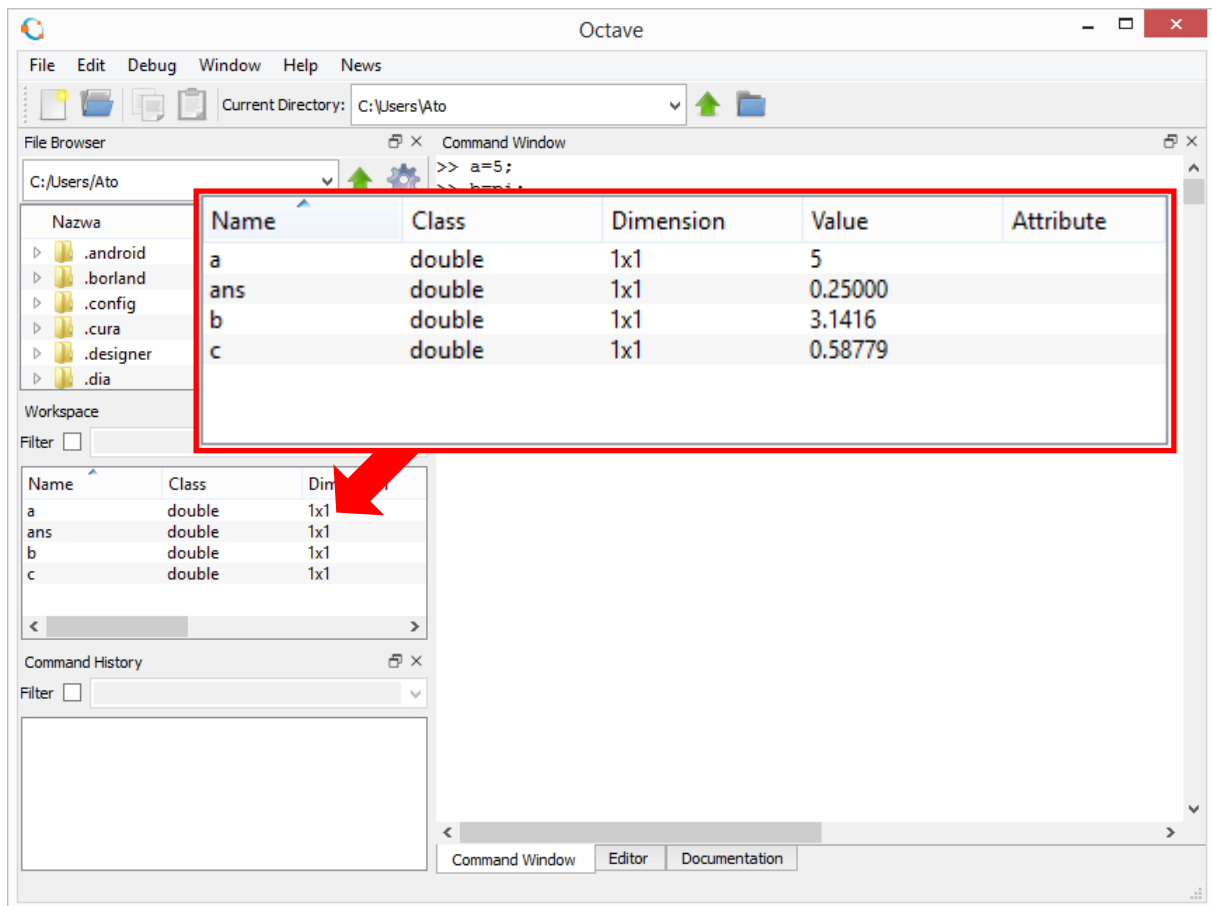


Fig. 39 Ventana "espacio de trabajo"

3.4. Editor

Como se dijo anteriormente, la segunda opción para ingresar comandos es usar el editor. En este caso, primero se debe escribir todos los comandos y luego "ejecutar" el script escrito. Vaya a la pestaña "Editor" (Fig. 33) e ingrese los comandos (denominado código

fuelle). Primero, sin embargo, vale la pena configurar la carpeta en la que desea trabajar. Para hacer esto, haga clic en el icono azul de acuerdo con la Fig. 40 y luego seleccione la carpeta adecuada (puede, por ejemplo, crear y elegir la carpeta C:\HOLOMAKERS\Octave).

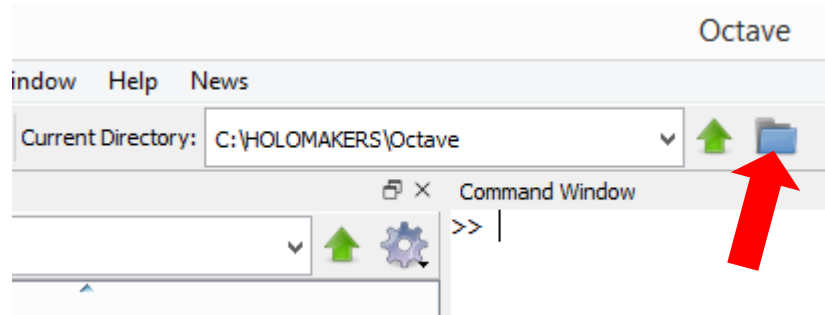


Fig. 40 Cambiando la carpeta de trabajo

Ahora podemos escribir nuestro propio guión. La figura 41 presenta algunos comandos simples. Cada comando está en una línea separada (esto le permite mantener el código legible y evitar posibles errores). Para que el cálculo tenga lugar, debe ejecutar el script con el botón "Guardar archivo y ejecutar" resaltado en la Fig. 41 con una flecha roja. Durante la primera ejecución, el programa le pedirá que guarde el archivo. El archivo se guardará en el directorio actual establecido previamente. Asigne al archivo el nombre "test01". El archivo se guardará como "test01.m", y luego se ejecutará el script (todos los comandos se ejecutarán en el orden desde la primera hasta la última línea).

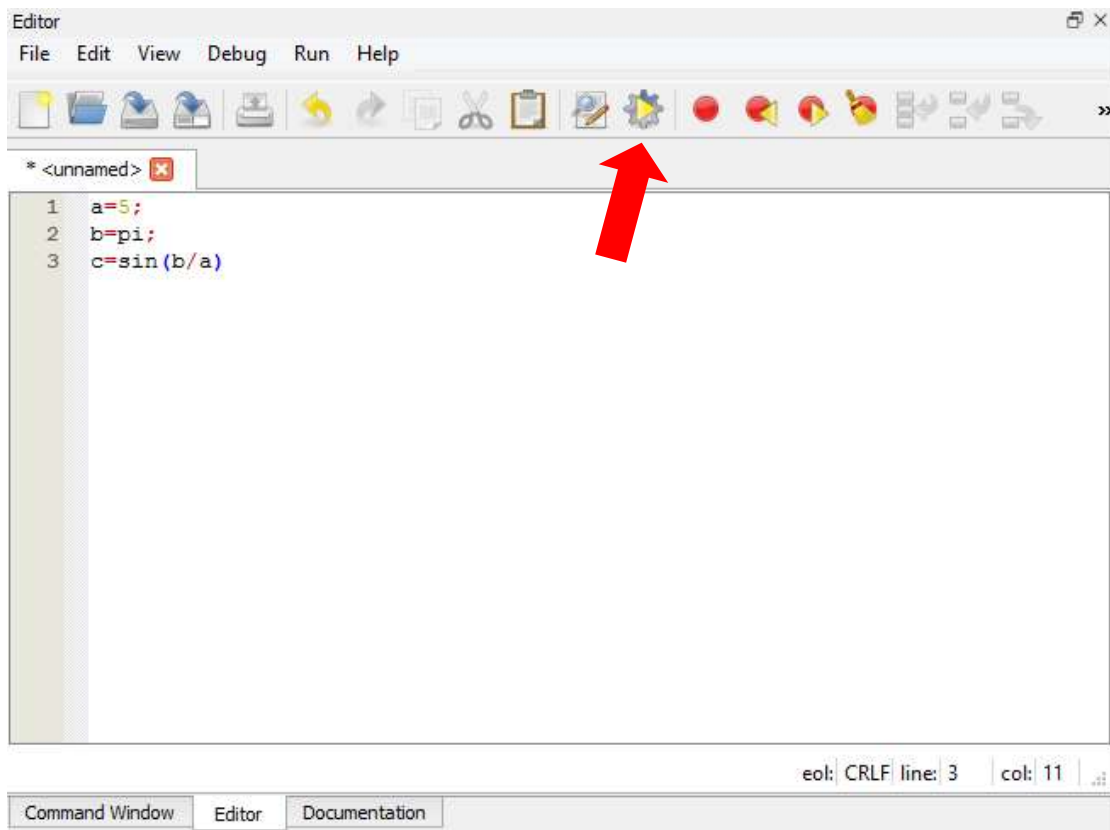
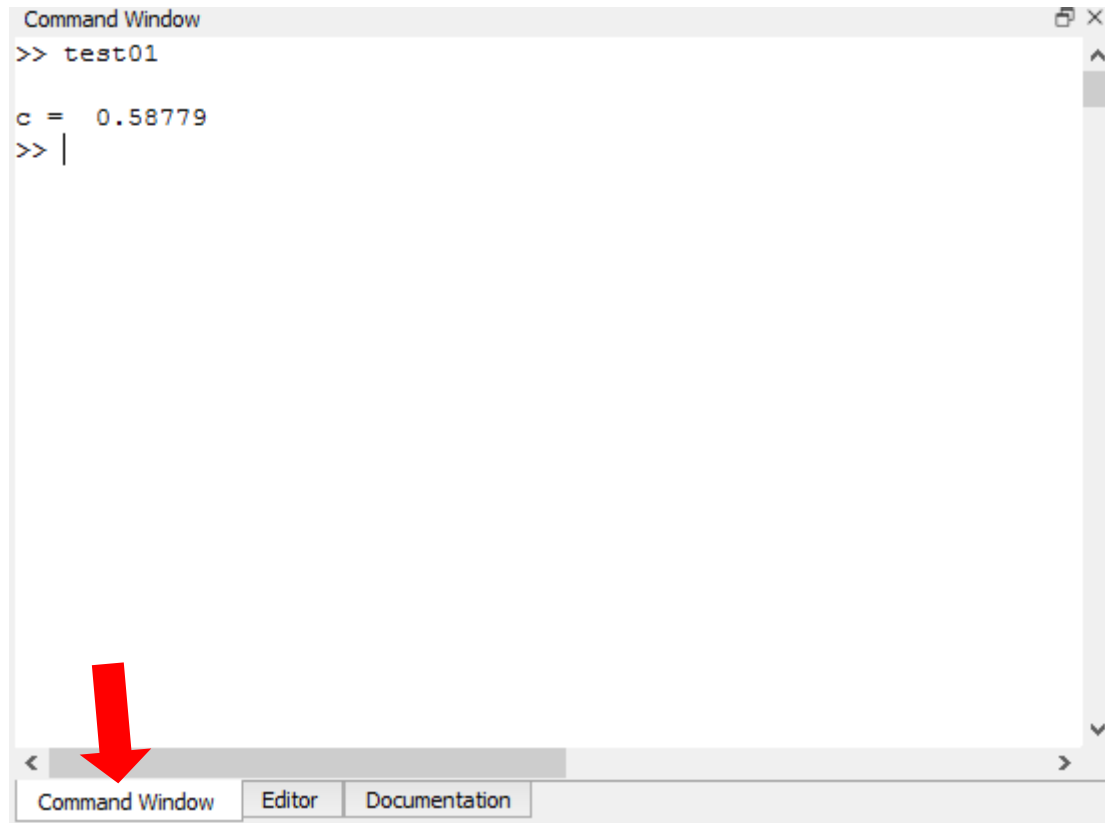


Fig. 41 Editor

Para ver el resultado de nuestras acciones, vuelva a la "Ventana de comandos" (Fig. 42). Solo se mostró la variable "c" porque el comando $c = \sin(b/a)$ no se terminó con un punto y coma. Los primeros dos comandos se terminan con un punto y coma por lo que no se muestran.



```
Command Window
>> test01

c = 0.58779
>> |
```

Fig. 42 Resultado del cálculo

3.5. Problemas básicos de programación en el software de Octave

3.5.1. Variables

Las variables se utilizan para almacenar los datos que necesitamos (números, caracteres, texto) en la memoria de la computadora. Gracias a esto, podemos leer el valor de la variable en cualquier momento. En Octave, es muy fácil declarar una variable. Simplemente escriba su nombre y luego asígnele un valor como se mencionó en el último capítulo. En Octave, una variable puede almacenar muchos números. Entonces tal variable se llama una matriz. Las matrices pueden ser unidimensionales o bidimensionales, como se ilustra en la Fig. 43.

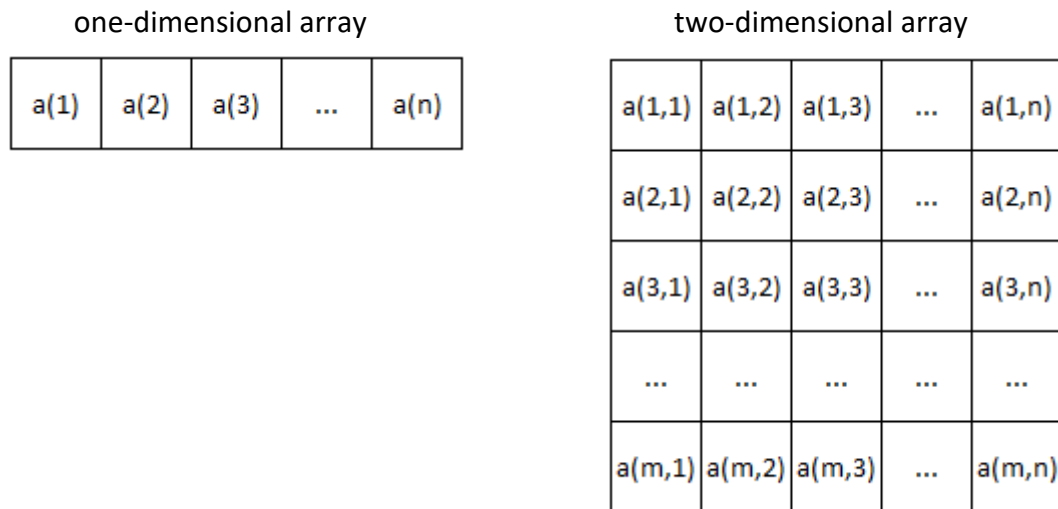


Fig. 43 Estructura de datos de matriz

Como puede ver, cada elemento de una matriz unidimensional se define por un índice y, en el caso de una matriz bidimensional, dos índices. Imagina que queremos declarar una variable llamada `tab1`, en la que almacenaremos 5 números. En Octave, simplemente escribe:

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

Es una matriz que tiene 5 filas y una columna. Cada uno de los cinco elementos de la matriz tiene un valor de 0. Ahora declaremos una matriz bidimensional de 3x3 también llena de ceros:

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

Ingrese las instrucciones anteriores en Octave en la "Ventana de comandos" para ver las matrices que creó.

El siguiente es un ejemplo interesante de la creación de una matriz unidimensional que consiste en números equidistantes de un rango determinado. Vamos a crear una matriz y llamémosla `X`, que contiene 5 elementos en un rango de 0 a π . Usaremos la función `linspace` (comienzo de rango, fin de rango, número de elementos): `X=linspace(0,pi,5)`

Ingrese la instrucción anterior en Octave en la "Ventana de comandos" para ver el resultado.

3.5.2. Operaciones básicas de entrada/salida

La operación de entrada que discutiremos aquí es ingresar los datos desde el teclado, donde el usuario le da un valor por teclado, que luego será recordado por el programa en la variable apropiada. La operación de salida es, por ejemplo, mostrar el resultado del cálculo en la pantalla.

Para leer un número desde el teclado y guardarlo en una variable llamada "a", use el siguiente comando: `a = input('Texto del usuario')`. Escriba el siguiente comando en la ventana de comandos y luego confirme con Enter.

```
a = input('Inserte el valor: ')
```

La siguiente línea mostrará el texto: Ingrese un valor, y el programa esperará hasta que ingrese un valor desde el teclado. Escriba el número 5 y confirme con Enter. A partir de ahora, la variable "a" contiene el valor 5. Debería obtener un resultado como en la Fig. 44.



Fig. 44 Cargando datos desde el teclado

La operación de salida más simple es mostrar el texto en la pantalla con el comando `disp`. En la "Ventana de comandos" ingrese el siguiente comando y confírmelo con la tecla Enter:

```
disp('La variable "a" contiene el valor: '), disp(a)
```

Se mostrará el siguiente texto: La variable "a" contiene el valor: y en la siguiente línea se mostrará el valor de la variable "a". Por lo tanto, al utilizar la función `disp`, puede mostrar tanto el texto como los valores variables en la pantalla. Si queremos mostrar solo el texto, también podemos usar la función de poner.

3.5.3. Operadores

Discutiremos tres clases de operadores: aritméticos, relacionales y lógicos.

1. Los operadores aritméticos no son más que el operador de sumar (+), restar (-), multiplicar (*), dividir (/) y asignar valores de operador (=). Así, por ejemplo, la expresión $a = c + 2 * b$ contiene tres operadores (=, +, *).
2. Los operadores relacionales se utilizan para comparar dos valores (por ejemplo, $a > b$). Estos operadores son: operador de igualdad (==), menor que (<), mayor que (>), menor o igual que (<=), mayor o igual que (>=), diferente de (!=). El resultado de la comparación de dos valores es el llamado valor lógico, que es verdadero o falso. Por ejemplo, si escribimos la expresión $2 == 5$, será falsa, mientras que la expresión $2 < 5$ será verdadera.
3. Los operadores lógicos como su nombre sugiere se utilizan para realizar operaciones lógicas básicas como la suma lógica (| |), la conjunción lógica (&&) y el operador de negación (!). El resultado de una expresión lógica es verdadero o falso. Por ejemplo, la expresión $(5 > 2) \&\& (2 < 3)$ es verdadera porque los términos $5 > 2$ y $2 < 3$ son verdaderos. Una expresión utilizada aquí contiene operadores relacionales (<, >) y el operador lógico (&&).

Un concepto digno de mención es la llamada prioridad de los operadores. Aunque suena bastante misterioso, todos los estudiantes se han encontrado con esto desde los primeros años de aprendizaje de las matemáticas. Piense en el resultado de la siguiente expresión: $5 + 2 * 3 = ?$. Así es, el resultado es 11. Tenga en cuenta que intuitivamente primero se multiplicó y luego se agregó. ¿Por qué no agregó los números 5 y 2 primero y luego no multiplicó el resultado por 3? Esto se debe a que es consciente de que la operación de multiplicación se realiza primero y luego la suma. Esta es la prioridad de los operadores. Nos dice qué operaciones en la expresión se llevan a cabo primero y cuáles siguen. La tabla 1 muestra los operadores ordenados según su prioridad. El operador de negación lógica tiene la más alta prioridad. Esto significa que si tal operador está en algún lugar de la expresión, se

ejecutará primero. Los operadores de relaciones tienen la prioridad más baja (se realizan al final).

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

Table 1 Prioridad de operadores

Algo obvio también conocido de las clases de matemáticas es el hecho de que si queremos cambiar el orden en que se ejecutan los operadores, tenemos que colocar los corchetes en el lugar correcto. Por lo tanto, la expresión $5 + 2 * 3$ da un valor de 11, mientras que el resultado de la expresión $(5 + 2) * 3$ será el número 21. A veces es necesario insertar corchetes para que la expresión tenga sentido. Este ejemplo ya se dio anteriormente: $(5 > 2) \&\& (2 < 3)$.

3.5.4. Sentencia condicional

Cada uno de nosotros toma decisiones diferentes cada día. Por ejemplo, si el sol brilla, podemos decidir que vamos a ir en bicicleta y nos quedaremos en casa de lo contrario. Este es un ejemplo de una declaración condicional simple. Primero, verificamos alguna condición (por ejemplo, si el sol está brillando) y luego, dependiendo del resultado de la condición, tomamos las medidas apropiadas (por ejemplo, vamos en bicicleta o nos quedamos en casa). La condición está representada por una expresión lógica que es verdadera o falsa. Expresión lógica puede ser por ejemplo, la frase "Ahora el clima es soleado". Esta oración (expresión lógica) es verdadera o falsa según las condiciones climáticas.

En la programación, las declaraciones condicionales son tan necesarias como en la vida cotidiana. La figura 45 presenta la estructura de una instrucción condicional simple. Dependiendo de si la condición es verdadera o no, se ejecutan otras instrucciones (declaraciones).

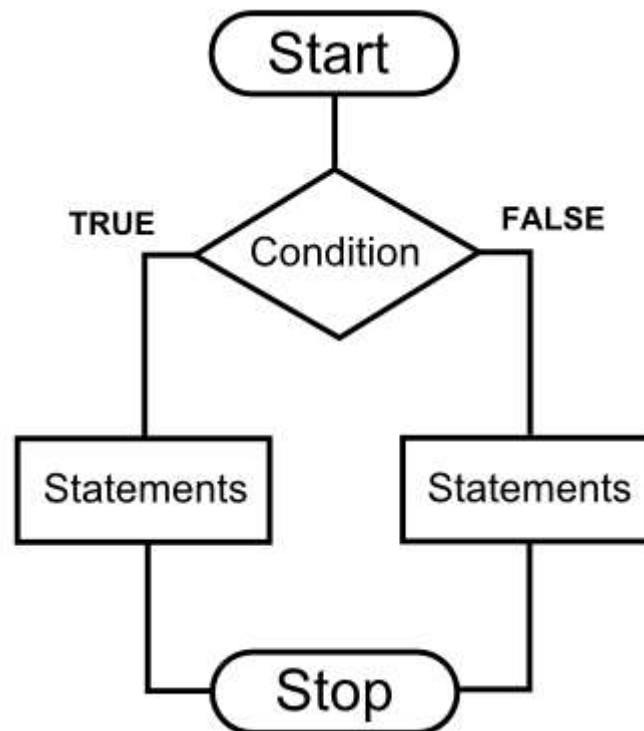


Fig. 45 Algoritmo de una declaración condicional simple

En Octave, la declaración condicional simple se ve como sigue:


```
if (condición)
```

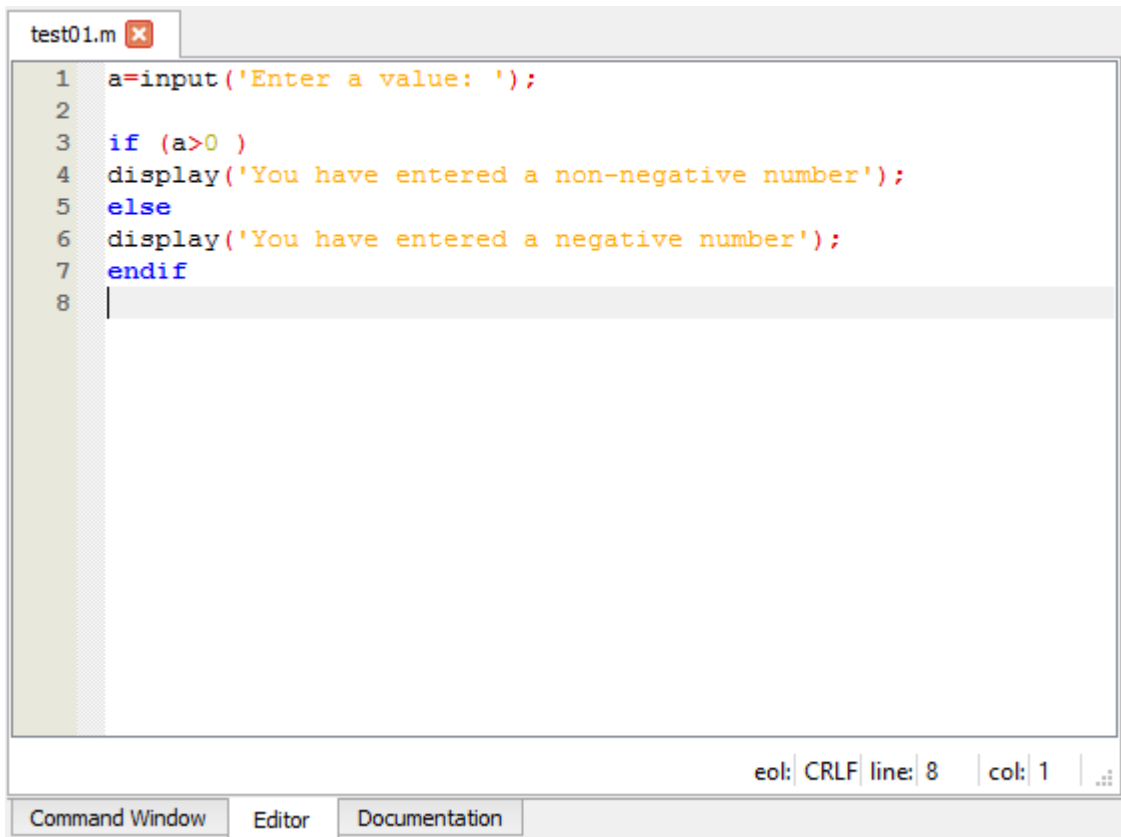
Otras declaraciones que se ejecutarán si la condición es verdadera.

```
else
```

Otras declaraciones que se ejecutarán si la condición es falsa.

```
endif
```

Como ejemplo de uso de la declaración condicional, escribimos un breve guión que verificará si el número ingresado por el usuario es negativo o no negativo. Escriba en la ventana del editor la secuencia de comandos como en la Fig. 46. Luego presione el botón "guardar archivo y ejecutar"  y vaya a la ventana de comandos. Después de ingresar el número y presionar la tecla Enter, aparecerá la oración correspondiente.



```
test01.m
1 a=input('Enter a value: ');
2
3 if (a>0 )
4 display('You have entered a non-negative number');
5 else
6 display('You have entered a negative number');
7 endif
8
eol: CRLF line: 8 col: 1
Command Window Editor Documentation
```

Fig. 46 Ejemplo de uso de la declaración condicional

A menudo, sin embargo, tenemos una situación en la que, después de verificar una condición, queremos verificar otra. En Octave, por lo tanto, debemos escribir:

```
if (condición1)
```

Otras declaraciones que se ejecutarán si la condición1 es verdadera

```
elseif (condición2)
```

Otras declaraciones que se ejecutarán si la condición1 es falsa y la condición2 es verdadera

```
else
```

Otras declaraciones que se ejecutarán si tanto condición1 como la condición2 son falsas

```
endif
```

Para ilustrarlo mejor, modifiquemos un poco el último guión para que el programa también indique si el usuario ha proporcionado el número cero (Fig. 47).


```

1 a=input('Enter a value: ');
2
3 if (a>0 )
4 display('You have entered a positive number');
5 elseif (a<0)
6 display('You have entered a negative number');
7 else
8 display('You have entered 0');
9 endif

```

Fig. 47 Ejemplo de uso de la declaración condicional

3.5.5. Bucle "For"

A menudo es necesario ejecutar algunos comandos muchas veces. Para evitar ingresar los mismos comandos muchas veces, el bucle se usa en la programación. Discutiremos brevemente solo un tipo de bucle: el bucle "for". Te permite ejecutar un conjunto de comandos un cierto número de veces. La estructura del bucle for se muestra en la Fig. 48. Es muy simple y nos dice que las declaraciones se llevarán a cabo 10 veces en este caso. ¿Cómo sabes que solo 10 veces? Esto se define al principio del bucle. Una variable, que llamamos "i" en cada ejecución de bucle posterior, aumenta su valor en 1, de 1 a 10. Cuando la variable "i" alcanza el valor final (en este caso 10), el bucle finaliza su operación.


```

1 for i=1:10
2
3 statements...
4
5 end

```

Fig. 48 Bucle "For"

Un ejemplo muy simple de usar el bucle "for" puede ser escribir varios caracteres en la pantalla. Imagina que queremos poder mostrar una línea que consta de los caracteres "-". Podemos escribir un script muy simple para este propósito, que se muestra en la Fig. 49.

Crea un nuevo script usando el icono . Durante la primera ejecución, guárdala con el nombre test02. Puede ver que nuestra línea tendrá 15 caracteres de longitud (el bucle "for" se ejecutará 15 veces).

```

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

```

Fig. 49 Mostrando una línea de 15 caracteres

El resultado de la operación de script se puede ver en la ventana de comandos (Fig. 50).

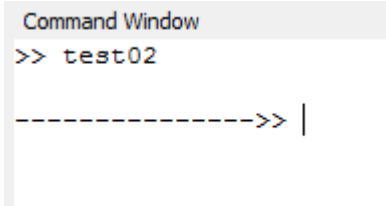


Fig. 50 Una línea de 15 caracteres.

Podemos modificar el script un poco para que el usuario pueda decidir qué longitud de la línea mostrar (Fig. 51). Tenga en cuenta que el bucle se ejecutará n veces, donde n es el número proporcionado por el usuario.

```
1 n=input ('enter the length of the line: ');
2
3 for i=1:n
4
5 puts ('-');
6
7 end
```

Fig. 51 Mostrando una línea con cualquier número de caracteres

El bucle for se usa muy a menudo para varios tipos de cálculos. ¿Puede escribir un guión en el que ingrese 5 números y el programa calculará su promedio? Si no es así, mira la Fig. 52.

```
1 for i=1:5
2
3 a(i) = input('Enter a value: ');
4
5 end
6
7 sum = 0;
8
9 for i = 1:5
10
11 sum = sum + a(i);
12
13 end
14
15 average = sum/5
```

Fig. 52 Cálculo de la media aritmética

Analicemos este script ahora. Primero, el usuario ingresa 5 números desde el teclado. Estos números se escriben en una matriz llamada "a". A continuación, se declara la variable denominada suma y su valor se establece en 0. El siguiente bucle ejecuta la suma de 5 números ingresados por el usuario. La última línea es el cálculo de la media aritmética. Como no se termina con un punto y coma, el resultado se mostrará en la pantalla.

3.5.6. Dibujando cuadros

Octave te permite dibujar una gráfica de una función de una manera muy simple. Dibujar una función $y = f(x)$ primero debemos crear una matriz de argumentos de la función x y la matriz correspondiente de valores de función $f(x)$. Supongamos que quieres dibujar una función $y = \sin(x)$ en un rango de $[0, 2\pi]$. Octave permite declarar una matriz con n elementos equidistantes del rango $[a, b]$. Esto se hace usando la función `linspace(a, b, n)`. Supongamos que queremos dividir nuestra $[0, 4\pi]$ rango en 100 valores equidistantes. En este caso, hay que escribir:

```
x = linspace(0, 4*pi, 100);
```

De esta manera, declaramos la matriz de 100 elementos. La obtención de una matriz de valores de funciones es incluso más simple, porque es suficiente para escribir:

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

Como x es una matriz de 100 elementos, entonces y se convertirá automáticamente en una matriz con el mismo número de elementos. Ahora solo muestra el gráfico. Para hacer esto, escribe:

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

El guión completo se presenta en la figura 53.

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

Fig. 53 Preparación de datos y visualización de gráficos.

Después de iniciar el guión, aparecerá una nueva ventana con un gráfico (ver Fig. 54).

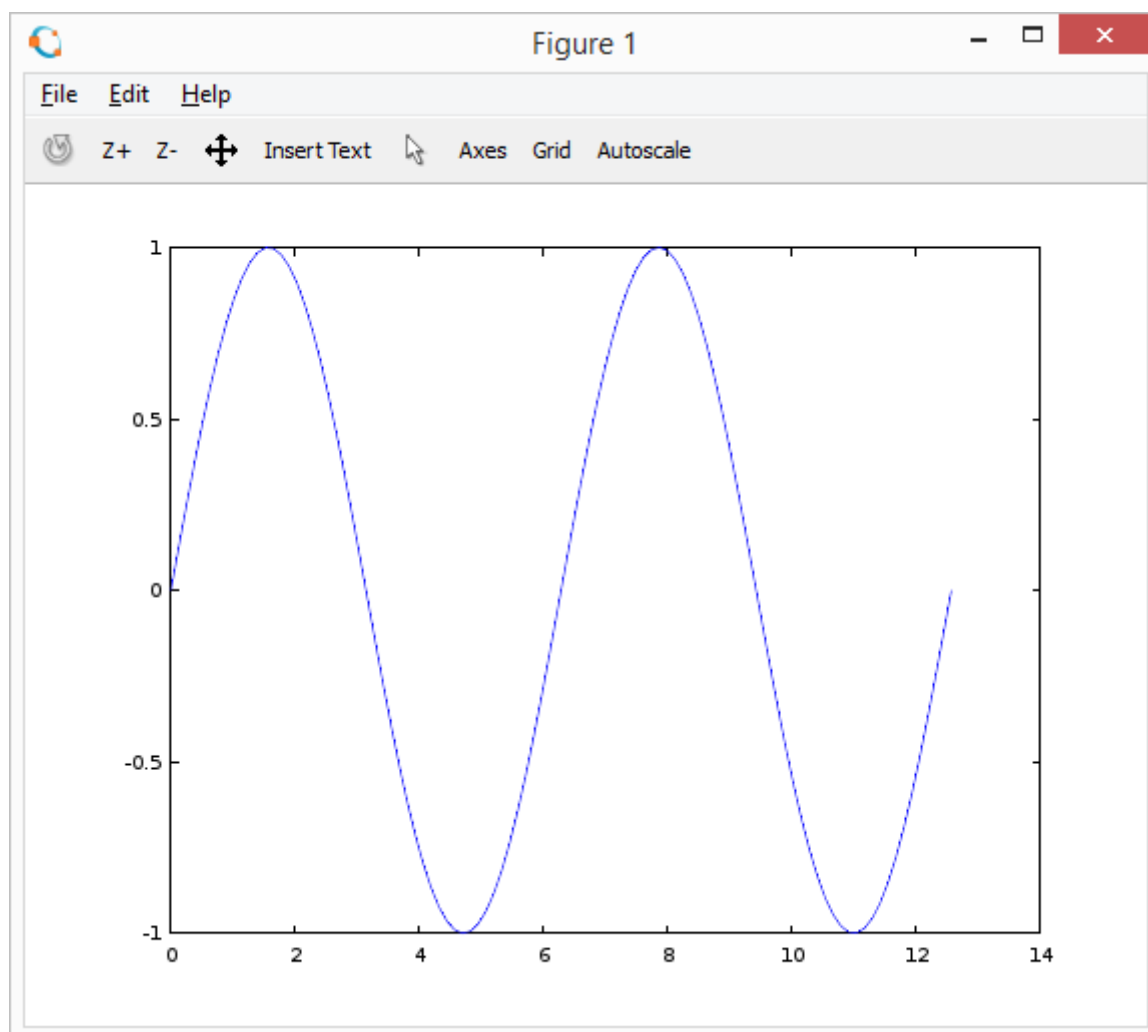


Fig. 54 Gráfica

3.6. Procesamiento de imágenes

3.6.1. Creando una imagen

Las imágenes grises se representan en Octave como una matriz bidimensional, donde cada elemento corresponde a un píxel (Fig. 55).

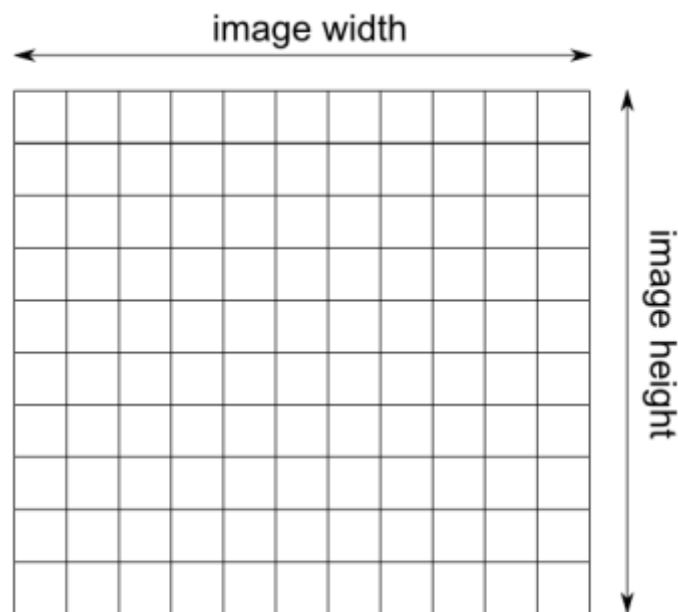


Fig. 55

Crear una nueva imagen es simplemente declarar una matriz bidimensional. Si desea crear una imagen con un tamaño de 200 x 200 píxeles, simplemente escriba:

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

De esta manera, a la variable image1 se le asignó una matriz de 200 x 200 rellena con ceros (cada elemento de la matriz es cero). Utilizamos la función de ceros aquí.

3.6.2. Lectura/escritura de imagen y visualización en pantalla

A menudo es necesario cargar un archivo de imagen que ya existe en el disco. Para hacer esto, primero copie la imagen al "directorio actual" previamente configurado. Entonces solo escribe el comando:

```
image1 = imread('filename');
```

El archivo de imagen con el nombre especificado ahora se almacena en la variable `image1`.

Para mostrar la imagen de la variable `image1`, use el comando:

```
imshow(image1);
```

Para mostrar varias imágenes, a cada imagen se le debe asignar un número diferente usando la figura de comando (número). También vale la pena hacer la ventana titulada usando el título del comando ('Título de la imagen'). A continuación se muestra un ejemplo de visualización de dos imágenes diferentes en dos ventanas diferentes:

```
figure(1);imshow(image1);title('Image no 1');  
figure(2);imshow(image2);title('Image no 2');
```

La imagen se guarda en un archivo usando el comando:

```
imwrite(image1, 'name.extension');
```

La imagen se puede guardar en varios formatos, por ejemplo, bmp, png, jpg, tiff, gif.

Como ejemplo, cargue un archivo de imagen con la extensión bmp al directorio de trabajo actual y luego escriba un script que muestre esta imagen en la pantalla y guárdelo con varias extensiones.

```
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');
```

Fig. 56 Convertir la imagen en diferentes formatos.

3.6.3. Conversion de color

Octave tiene muchas funciones diferentes para la conversión de color. Entre ellas hay una función que convierte una imagen en color a gris y una función que convierte una imagen (gris o coloreada) en una imagen binaria que se compone solo de dos colores: blanco y negro. Estas funciones, sin embargo, no están disponibles de forma predeterminada cuando se inicia el entorno Octave. Para poder usar estas funciones, necesita cargar el llamado paquete llamado imagen. Así que, al comienzo del guión, escribe:

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

A partir de ahora, podemos utilizar libremente las muchas funciones incluidas en este paquete. Para convertir una imagen en color a gris, use la función `rgb2gray`. Por ejemplo, si la variable `colorImage` almacena una imagen en color, podemos reemplazarla con gris y guardarla en la variable `grayImage` de la siguiente manera:

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

Del mismo modo, la imagen se puede convertir a binario.:

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

3.6.4. Rotación

Puede rotar la imagen de Octave con respecto al centro con cualquier ángulo utilizando la función de `imrotate`. Para usar esta función, necesitamos indicar la imagen girada y el ángulo de rotación. Por ejemplo, si desea colocar `Image1` girada 45 ° en la variable `Image2`, escriba:

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

3.6.5. Suma, resta, multiplicación y división

Como recordamos, la imagen en Octave se representa como una matriz bidimensional con dimensiones correspondientes al ancho y alto de la imagen en píxeles. Al tener dos imágenes con las mismas dimensiones, podemos realizar fácilmente operaciones como agregar o multiplicar imágenes. Por ejemplo, agregar dos imágenes entre sí simplemente agregará los valores de píxeles correspondientes entre sí (Fig. 57).

Image A		Image B		Image 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

Fig. 57 Añadiendo dos imágenes

De manera similar, la resta se realiza así como la multiplicación y la división. La Tabla 2 presenta los nombres de funciones particulares en el entorno de Octave.

Operación	Octave
Suma	imadd
Resta	imsubtract
Multiplicar	immultiply
División	imdivide

Table 2 Operaciones en imágenes

Deje que las variables `imgA` e `imgB` representen dos imágenes con las mismas dimensiones. Luego podemos crear una imagen `imgC` que es la suma o multiplicación de estas dos imágenes:

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

3.6.6. Transformada de Fourier

La transformación de Fourier es un tipo de transformación ampliamente utilizada en muchos campos, como la electrónica, la óptica o la música. En el caso del procesamiento de imágenes, también se puede realizar una transformada de Fourier. La transformada de Fourier de una imagen a menudo se parece a una colección de píxeles aleatorios, pero contiene la misma información que la imagen original (ver Fig. 58).

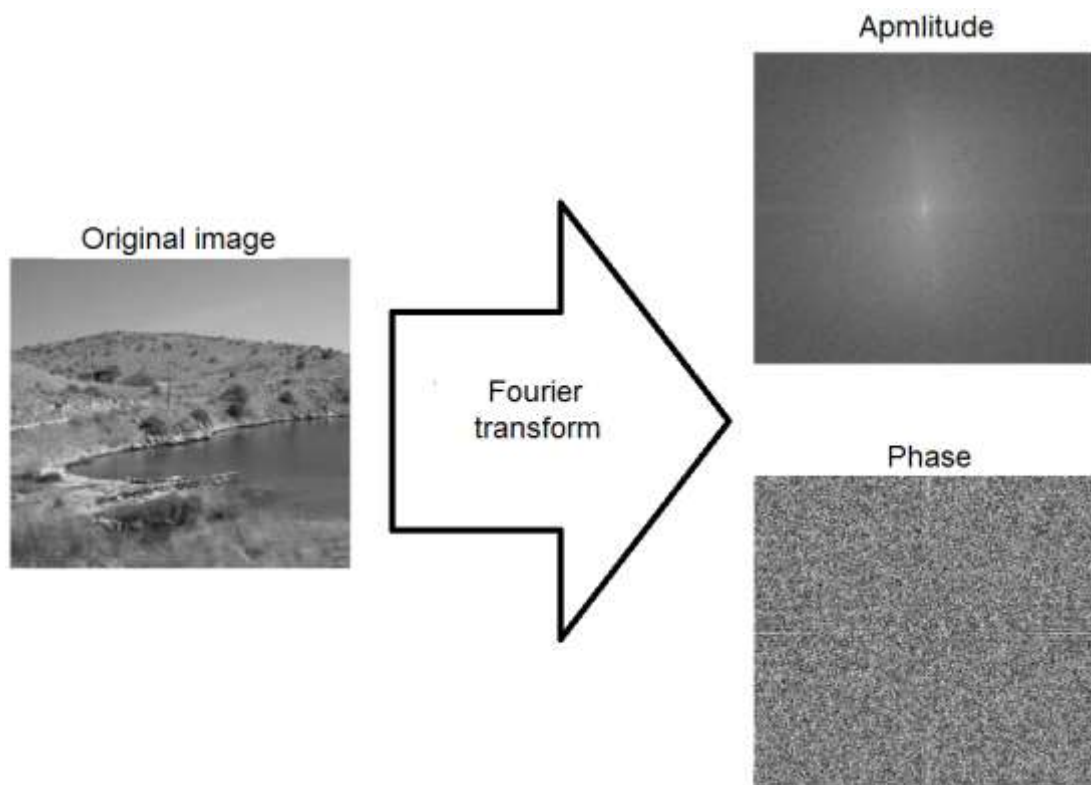


Fig. 58 La imagen y su transformada de Fourier

Cada punto de transformación de Fourier no solo tiene una amplitud sino también una fase. Una discusión detallada de este tema va más allá del alcance de esta guía. Solo recordemos que la fase es necesaria para reproducir la imagen original (a través de la transformada de Fourier inversa), mientras que la amplitud se distribuye de la siguiente manera: los puntos en el centro de la transformación corresponden a las áreas suaves de la imagen, mientras que los puntos de transformación distantes del centro corresponden a los bordes de la imagen (el borde es, por ejemplo, el borde entre el cielo liso y las montañas, entre el agua y la tierra, etc.). Esto se puede ilustrar en un ejemplo simple. La figura 59a presenta el cuadrado blanco y su transformada de Fourier. El punto medio de la transformada de Fourier corresponde a la parte central del cuadrado (una imagen suave), mientras que cada uno de los cuatro "brazos" corresponde al borde correspondiente del cuadrado (Fig. 59b).

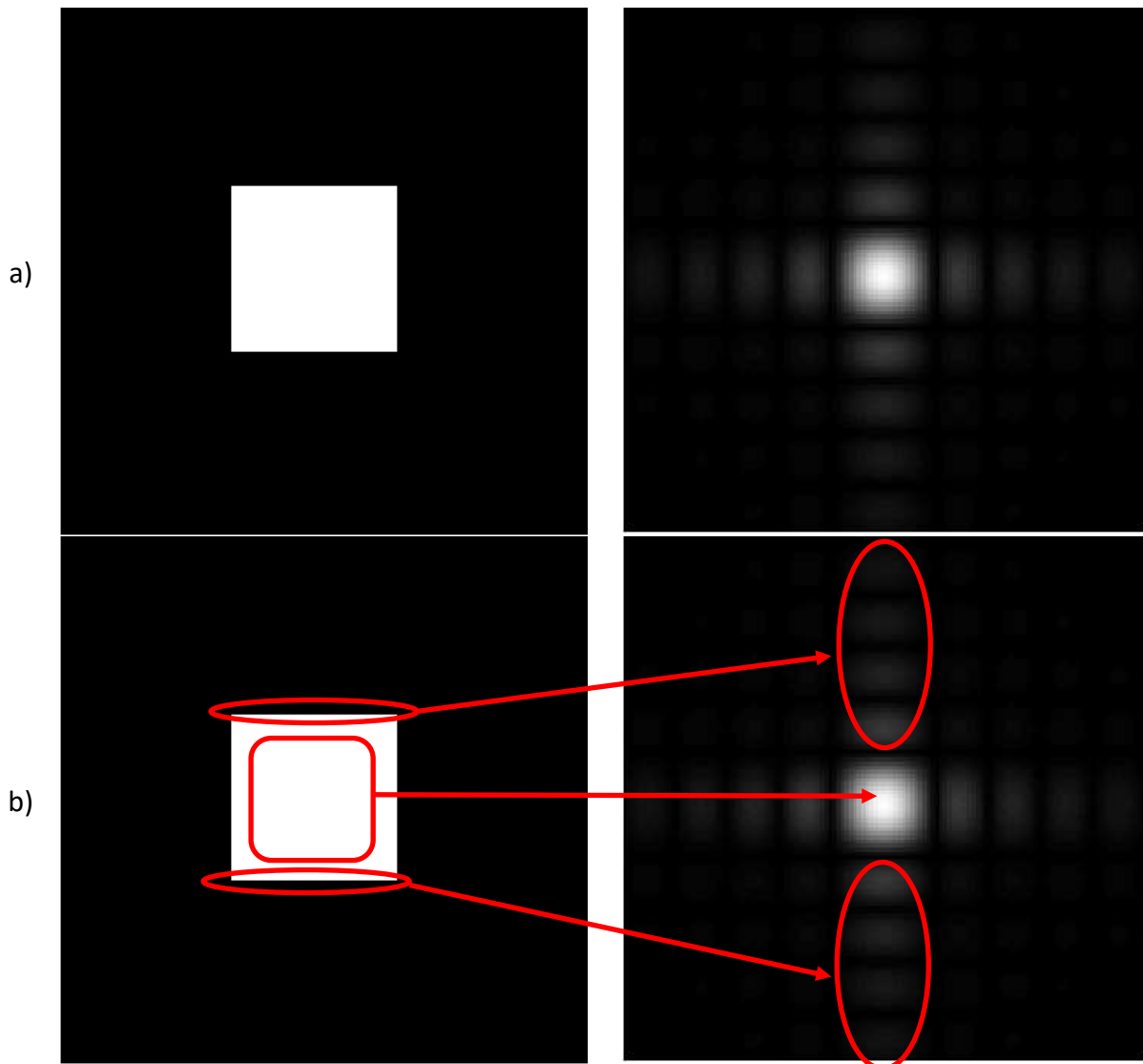


Fig. 59 Transformada de Fourier de la plaza

En el caso de figuras geométricas simples, es claramente visible. La figura 60 presenta la imagen del triángulo y su transformación. El punto medio de la transformación corresponde a la parte central del triángulo. Los tres lados del triángulo se pueden reconocer en la transformación como tres "brazos". También son visibles tres brazos adicionales. Estos brazos representan los vértices del triángulo.

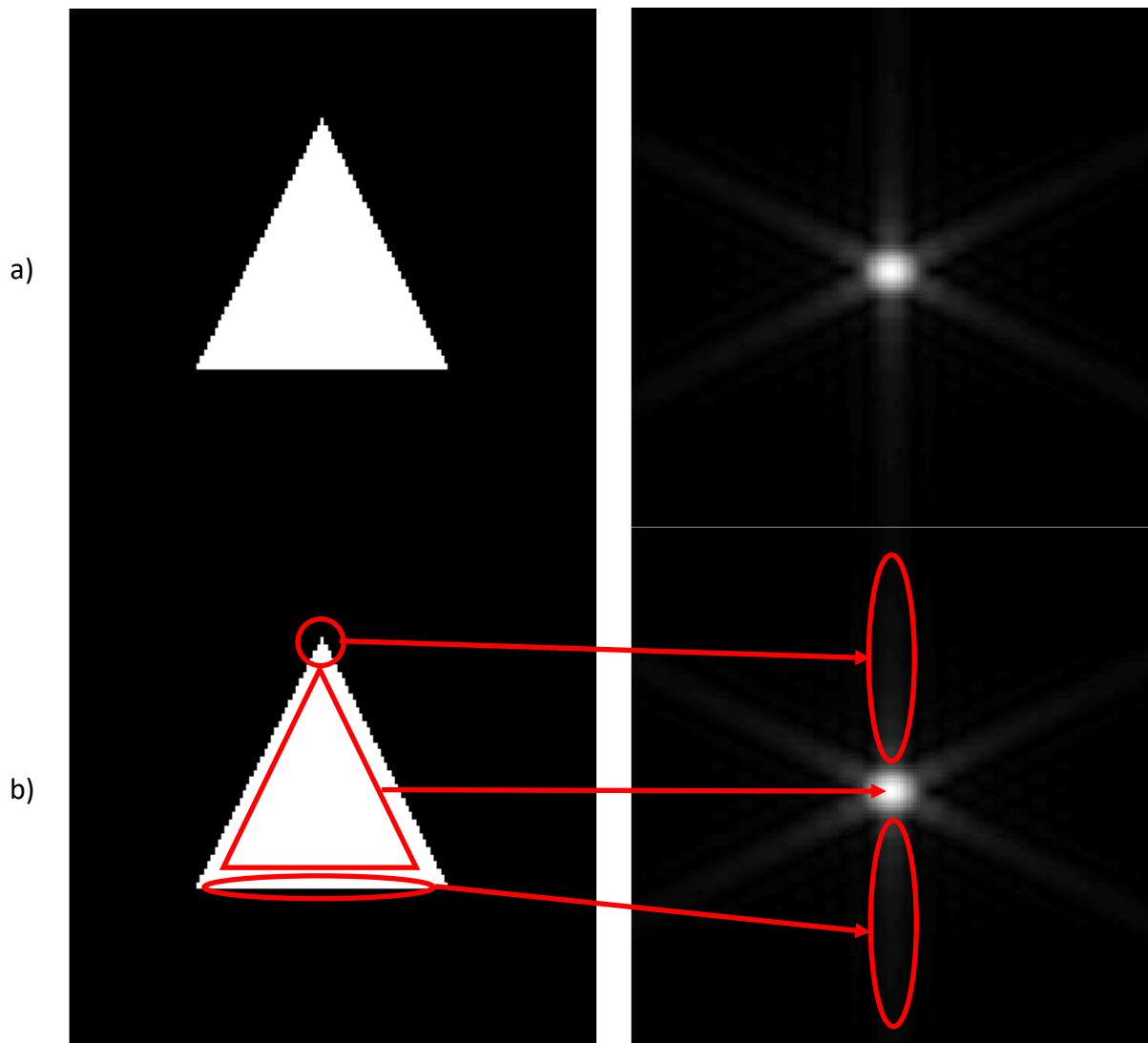


Fig. 60 Transformada de Fourier del triángulo

En Octave, la función de transformación de Fourier se llama `fft2`. Esta función funciona de tal manera que la transformada de Fourier se desplaza desde el centro. Por lo tanto, es necesario utilizar adicionalmente la función `fftshift`. Por ejemplo, si desea hacer una transformada de Fourier de imagen `img1`, escriba:

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

Para visualizar la amplitud de la transformada se debe escribir:

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

Por otro lado, para escribir la fase de la transformación, uno debe escribir:

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

3.7. Algoritmo para el cálculo de hologramas generados por computadora (CGH)

Para crear un holograma analógico, necesitamos tener un objeto real, que se registrará como un holograma. En el caso de hologramas generados por computadora, el objeto real se reemplaza con un archivo gráfico que contiene cualquier forma (por ejemplo, un archivo gráfico que contiene un texto blanco sobre un fondo negro). La generación de un holograma implica el cálculo por computadora del patrón de interferencia. El patrón se puede grabar en una placa o película holográfica. Después de pasar por tal holograma, la luz láser se doblará de tal manera que creará un patrón diseñado previamente (por ejemplo, texto). La generación de computadora del holograma se lleva a cabo utilizando un algoritmo determinado llamado el algoritmo de Gerchberg-Saxton. Este algoritmo requiere proporcionar la distribución de entrada de la intensidad diseñada (es decir, el archivo gráfico mencionado anteriormente con el texto) y la distribución de intensidad del haz de luz en el que se reproducirá el holograma (también un archivo gráfico). Como resultado de la operación del algoritmo, se puede obtener un archivo gráfico con una distribución de fase que es un holograma diseñado. La idea general se presenta en la figura 61.

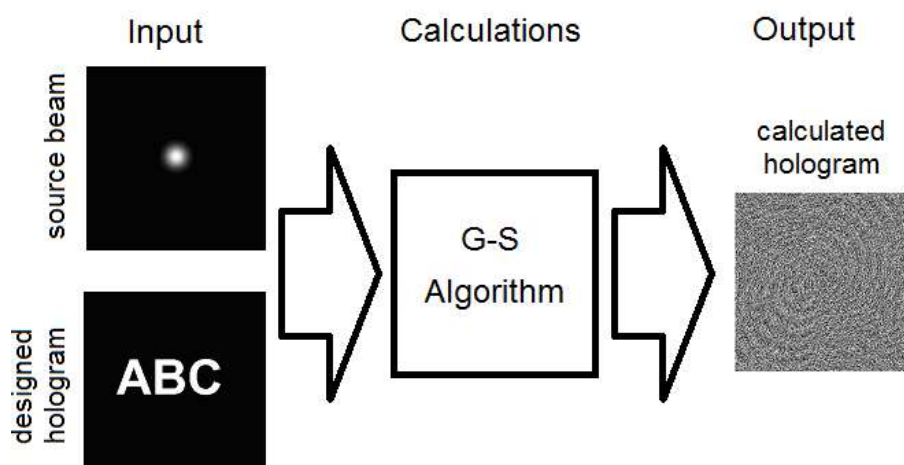


Fig. 61 Los datos de entrada en base a los cuales se calcula el holograma.

El algoritmo de Gerchberg-Saxton es un algoritmo iterativo, lo que significa que algunas operaciones (cálculos) se realizan muchas veces. Cada ejecución posterior del cálculo hace que obtengamos un holograma de mejor calidad. El esquema del algoritmo G-S se presenta en la Fig. 62.

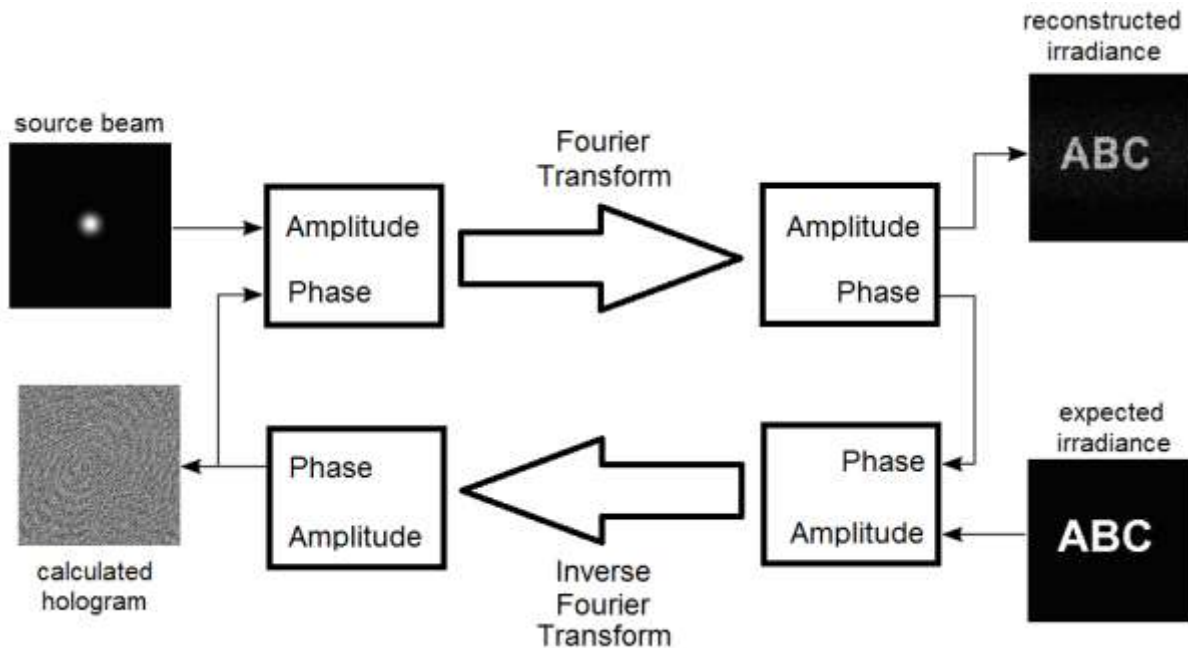


Fig. 62 El algoritmo de Gerchberg-Saxton

A continuación se muestra un script que utilizaremos para generar hologramas. El holograma calculado se guardará en el archivo CGH.bmp. Los archivos de entrada de muestra se pueden descargar desde el sitio web 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(ifft2(fftshift(D)));  
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');
```