



## FACULTAD DE INGENIERÍA DE SISTEMAS INGENIERÍA EN COMPUTACIÓN

**Materia:** ICCR 334 Matemáticas Discretas

### Práctica 4: Relaciones

**Nombre:** Ing. Andrés Larco, MSc.

**Fecha:** 02/08/2021

### GUÍA PARA EL DESARROLLO DE LA PRÁCTICA

#### 1. OBJETIVOS

- Entender la utilidad del uso de herramientas en Matemáticas Discretas.
- Enseñar al estudiante los fundamentos de diseño de programas y desarrollo de algoritmos en MATLAB.
- Identificar funciones de MATLAB y utilidades de importación y exportación de datos.

#### 2. MARCO TEÓRICO

El Marco teórico tendrá los conceptos relacionados con MATLAB.

Deberá consultar:

- Diseño de programas y desarrollo de algoritmos.
- Funciones de MATLAB y utilidades de importación y exportación de datos.

#### 3. DESARROLLO

### Program Design and Algorithm Development

El objetivo de esta práctica es introducirlo en:

- El diseño de programas.
- Realizar funciones definidas por el usuario en Matlab.

### MATLAB Functions and Data Import-Export Utilities

- Permitirle familiarizarse con algunas funciones más comunes de MATLAB.
- Presentarle brevemente las formas de importar y exportar datos dentro y fuera del espacio de trabajo de MATLAB usando:
  - Los comandos de carga y guardado (`load` and `save`)
  - El asistente de importación.
  - Las funciones de entrada / salida (E / S) de archivos de bajo nivel

# Program Design and Algorithm Development

## Programming style

Algunos programadores se deleitan escribiendo un código conciso y oscuro.

Un gran número de programadores responsables, sin embargo, creen que es extremadamente importante desarrollar el arte de escribir programas bien diseñados, con toda la lógica y claramente descritos.

Por lo tanto, los programadores serios prestan bastante atención a lo que se llama estilo de programación, con el fin de hacer sus programas más claros y legibles tanto para ellos mismos como para otros usuarios potenciales.

Puedes encontrar esto irritante, si estás comenzando a programar por primera vez, porque naturalmente, estas impaciente por continuar con el trabajo. Pero un poco más de atención extra a tu diseño de programa pagará enormes dividendos a largo plazo.

A continuación, se ofrecen algunas sugerencias sobre cómo mejorar su estilo de programación.

- ▶ Debe hacer un uso generoso de los comentarios, tanto al comienzo de un guion como para describir brevemente lo que hace y cualquier método especial que pueda tener sea utilizado, y también a lo largo de la codificación para introducir diferentes secciones lógicas.
- ▶ El significado de cada variable debe describirse brevemente en un comentario cuando se inicializa. Debe describir las variables de forma sistemática, p. ej. en orden alfabético.
- ▶ Las líneas en blanco deben usarse libremente para separar secciones de codificación (por ejemplo, antes y después de las estructuras de bucle).
- ▶ La codificación (es decir, las declaraciones) dentro de estructuras (fors, ifs, whiles) se deben sangrar (tabular) en algunas columnas para que se destaquen.
- ▶ Deben usarse espacios en blanco en las expresiones para hacerlas más legibles, p.ej. a ambos lados de los operadores y signos iguales. Sin embargo, los espacios en blanco pueden ser omitidos en lugares donde las expresiones son complicadas, esto puede hacer que la lógica sea más clara.

Los programas son como cualquier otra forma de arte literario; deben estar diseñados y escritos con una cantidad razonable de previsión que conduzca a una declaración clara del problema que se va a resolver, para satisfacer una necesidad reconocida de una solución científica, técnica, de ingeniería o matemática.

La solución podría estar destinada a mejorar el conocimiento o ayudar al científico o ingeniero a hacer una decisión sobre un diseño propuesto de un equipo de laboratorio o un artefacto para mejorar nuestro nivel de vida.

El desarrollo del enunciado del problema es, probablemente, la parte más difícil de cualquier proceso de diseño; no es diferente para el diseño de un programa.

Por lo tanto, aprender a diseñar buenos programas (o códigos) proporciona una buena experiencia en la práctica del diseño creativo.

Se requiere un plan estratégico que lleve al desarrollo de un algoritmo (p. ej. la secuencia de operaciones necesarias para resolver un problema en un número finito de pasos) programado para que MATLAB se ejecute en orden, con el fin de proporcionar una respuesta al problema planteado.

El objetivo esencial es crear un plan de estructura de arriba hacia abajo, detallando todos los pasos del algoritmo que deben implementarse en MATLAB para obtener la solución deseada.

## The program design process

Los objetivos son introducir el concepto de programa, diseño y el concepto de estructura de plan (pseudocódigo) (también llamado esquema o script) como medio para diseñar la lógica de un programa.

La estructura de diseño de arriba hacia abajo está elaborada para ayudarlo a pensar en el desarrollo de buenas estrategias de resolución de problemas en lo que respecta al diseño de procedimientos al utilizar software como MATLAB.

Consideraremos que la versión de MATLAB tiene su propia caja de herramientas, p. ej. SIMULINK, la Symbolics toolbox y la Controls toolbox. MATLAB te permite personalizar tu entorno de trabajo para satisfacer tus propias necesidades.

Tu habilidad de utilizar todo el poder de MATLAB está limitada únicamente por tu experiencia y antecedentes educativos.

Cuanto más crezca en su conocimiento, podrá usar más productivamente tu imaginación y habilidades creativas para desarrollar métodos que aprovechen las profundidades de esta herramienta para ayudarte a resolver problemas técnicos. Los problemas pueden ser problemas asignados en el curso, en la investigación de pregrado, proyectos de ingeniería y diseño, así como (después de la graduación) en el trabajo.

Consideraremos programas relativamente simples; sin embargo, el proceso descrito tiene la intención de proporcionar una idea de lo que enfrentará cuando se tenga problemas más complejos.

Estos son, por supuesto, en la ingeniería, la ciencia, y los problemas matemáticos que aprenderá durante los últimos años de su educación formal, sus búsquedas de aprendizaje de por vida después de la graduación y su continuidad de responsabilidades educativas según lo requiera su profesión.

Sin duda, los ejemplos que hemos examinado hasta ahora han sido muy sencillos desde el punto de vista lógico. Esto porque nos hemos concentrado en los aspectos técnicos, en la escritura correcta de las declaraciones de MATLAB.

Es muy importante aprender cómo MATLAB realiza las operaciones aritméticas que forman la base de más programas informáticos complejos diseñados para calcular información numérica.

Para diseñar un programa exitoso se necesita comprender un problema a fondo, y descomponerlo en sus etapas lógicas más fundamentales. En otras palabras, tienes que desarrollar un procedimiento sistemático o algoritmo para resolver el problema.

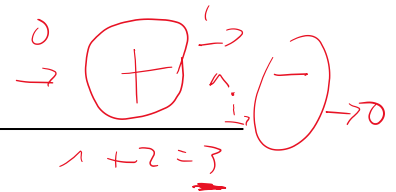
Hay varios métodos que pueden ayudar en este proceso de desarrollo del algoritmo. Examinamos uno de estos enfoques, la estructura del plan, en el contexto de la resolución de problemas técnicos con la ayuda de una computadora. Tú ya conoces brevemente el concepto de plan de estructura. El desarrollo de este plan es la parte principal del proceso de diseño del software (o código) porque son los pasos de este plan los que se traducen a un idioma que la computadora puede entender, p. ej. en comandos de MATLAB, algunos de los cuales ya han sido introducidos en las prácticas anteriores.

## Computer program design process

En el lenguaje MATLAB, se utilizan secuencias de líneas de comando o funciones (*functions*), y se guardan como M-files en su directorio de trabajo `work`, estos son los principales objetos y más útiles para los usuarios de herramientas como MATLAB.

En el directorio de trabajo predeterminado, `\ work`, o en su propio directorio de trabajo, p. ej. `\ mytools`, seguramente comenzará a acumular un conjunto de M-files que tú has creado a medida que continúas usando MATLAB. Una forma de crear y llegar a tu propio directorio de trabajo es ejecutar los siguientes comandos:

```
mkdir mytools  
cd mytools
```



¿Qué significa crear programas bien escritos?

El objetivo del diseño de herramientas de software es que funcione, que se puedan leer fácilmente y entenderse, y, por lo tanto, se pueda modificar sistemáticamente cuando sea necesario.

Para que los programas funcionen bien, deben satisfacer los requisitos asociados con el problema o la clase de problemas que se pretende resolver.

Las especificaciones, es decir, la descripción detallada del propósito (o función del programa), sus entradas, el método de procesamiento, salidas y cualquier otro requisito especial, deben ser conocidos para diseñar un algoritmo y un programa eficaz.

Debería funcionar completa y correctamente, es decir, todas las opciones deben poder utilizarse sin errores dentro de las limitaciones de las especificaciones.

El programa debe ser legible y, por lo tanto, claramente comprensible.

Por lo tanto, es útil descomponer las tareas principales (o el programa principal) en sub tareas (o subprogramas) que realicen partes específicas de la tarea principal.

Es mucho más fácil leer subprogramas que tienen menos líneas que un gran programa principal que no segrega las sub tareas de manera eficaz, en particular si el problema a resolver es relativamente complicado.

Cada sub tarea debe ser diseñada para que pueda ser evaluada de forma independiente antes de su implementación en el esquema más amplio de las cosas (es decir, en el plan del programa principal).

Un código bien escrito, cuando funciona, se evalúa mucho más fácilmente en la fase de prueba del proceso de diseño. Si es necesario realizar cambios para corregir errores de signos se pueden implementar fácilmente.

Una cosa para tener en cuenta al agregar comentarios es describir el proceso programado: agregue suficientes comentarios y referencias para que un año después de escribir el programa sepa exactamente lo que se hizo y con qué propósito. Tenga en cuenta nombrar a sus archivos también es un arte.

El proceso de diseño se describe a continuación. Los pasos se pueden enumerar de la siguiente manera:

→ **Paso 1: Análisis del problema.** El contexto de la investigación propuesta debe ser establecido y proporcionar la motivación adecuada para el diseño de un programa. El diseñador debe reconocer plenamente la necesidad y debe desarrollar una comprensión de la naturaleza del problema a resolver.

→ **Paso 2: Enunciado del problema.** Desarrolle un enunciado detallado de los problemas a resolver con un programa.

→ **Paso 3: Esquema de procesamiento.** Defina las entradas y las salidas requeridas que serán producidas por el programa.

→ **Paso 4: Algoritmo.** Diseñe el procedimiento paso a paso utilizando el diseño de arriba hacia abajo proceso que descompone el problema general en problemas subordinados. Las subtarefas para resolver luego se refinan diseñando una lista detallada de pasos a programar.

(Esta lista de tareas es el plan de estructura; está escrito en pseudocódigo, es decir, una combinación de inglés, matemáticas y comandos de MATLAB. El objetivo es diseñar un plan comprensible y esto se traduce fácilmente a un lenguaje informático. **MATLAB**)

→ **Paso 5: Algoritmo del programa.** Traduzca y convierta el algoritmo en un lenguaje para MATLAB, depure los errores de sintaxis hasta que la herramienta se ejecute exitosamente.

→ **Paso 6: Evaluación.** Pruebe todas las opciones y realice un estudio de validación del programa, p. ej. compare los resultados con otros programas y tareas similares, compara con datos experimentales si es apropiado, y compara con predicciones teóricas basadas en metodología teórica relacionada con los problemas que debe resolver el programa.

El objetivo es determinar que las subtarefas y el programa general son correctos y precisos. La depuración adicional en este paso es encontrar y corregir errores lógicos (p. ej. escritura incorrecta de expresiones colocando un signo + donde se suponía debe ser colocado un signo -), y errores de tiempo de ejecución que pueden ocurrir después de que el programa se ejecute con éxito (por ejemplo, casos en los que ocurre una división por 0 involuntariamente).

→ **Paso 7: Aplicación.** Resuelva los problemas para los que fue diseñado el programa. Si el programa está bien diseñado y es útil, podría guardarse en su directorio de trabajo. (es decir, en su caja de herramientas (toolbox) desarrollado por el usuario) para uso futuro.

SWP U (Código)  
100% reutilizar

100% C  
Nuevo / Re  
70% / 30%

## Projectile problem example

**Paso 1:** Consideremos el problema de los proyectiles, examinado en el primer semestre de física. Se asume que los estudiantes de ingeniería y ciencias comprenden este problema.

En este ejemplo queremos calcular el vuelo de un proyectil (por ejemplo, una pelota de fútbol) lanzado a una velocidad y un ángulo preestablecido.

Queremos determinar la trayectoria de vuelo y la distancia horizontal que recorre el proyectil (u objeto) antes de tocar el suelo. ??

Asumamos que la resistencia del aire es cero y una fuerza gravitacional constante que actúa sobre el objeto en dirección opuesta a la distancia vertical desde el suelo.

El ángulo de lanzamiento  $\theta_0$ , se define como el ángulo medido desde la horizontal (plano del suelo) hacia arriba, hacia la dirección vertical, es decir,  $0 < \theta \leq \pi/2$ , donde  $\theta_0 = 0$  implica un lanzamiento en la dirección horizontal y  $\theta_0 = \pi/2$  implica un lanzamiento en la dirección vertical (es decir, en la dirección opuesta a la gravedad).

Si  $g = 9,81 \text{ m/s}^2$  se utiliza como aceleración de la gravedad, entonces la velocidad de lanzamiento,  $V_0$ , debe ingresarse en unidades de m/s.

Por lo tanto, si el tiempo,  $t > 0$ , es el tiempo en segundos desde el momento del lanzamiento de  $t = 0$ , la distancia recorrida en  $x$  (la dirección horizontal) y  $y$  (la dirección vertical) están en metros (m).

Queremos determinar el tiempo que tarda el comienzo del movimiento para golpear el suelo, la distancia horizontal recorrida y la forma de la trayectoria. Además, grafique la velocidad del proyectil frente a la dirección de este vector. Necesitamos, por supuesto, la teoría (o expresiones matemáticas) que describan la solución a este problema con el fin de desarrollar un algoritmo para obtener soluciones al problema del proyectil de resistencia cero.

**Paso 2:** Las fórmulas matemáticas que describen la solución al problema descrito del proyectil en el Paso 1 se proporciona en este paso. Dado el ángulo de lanzamiento y velocidad de lanzamiento, la distancia horizontal recorrida desde  $x = y = 0$ , la cual describe la coordenada de ubicación del lanzador, es

$$x_{max} = 2 \frac{V_0^2}{g} \sin \theta_0 \cos \theta_0$$

El tiempo desde  $t = 0$  en el lanzamiento para que el proyectil alcance  $x_{max}$  (es decir, su rango) es

$$t_{x_{max}} = 2 \frac{V_0}{g} \sin \theta_0$$

El objeto alcanza su altitud máxima,

$$y_{max} = \frac{V_0^2}{2g} \sin^2 \theta_0$$

el tiempo

$$t_{y_{max}} = \frac{V_0}{g} \sin \theta_0$$

Para este problema, la distancia horizontal recorrida cuando el objeto alcanza la altitud máxima es  $x_{y_{max}} = x_{max}/2.0$

La trayectoria (o trayectoria de vuelo) se describe mediante el siguiente par de coordenadas en un instante dado de tiempo entre  $t = 0$  y  $t_{x_{max}}$

$$x = V_0 t \cos \theta_0$$

$$y = V_0 t \sin \theta_0 - \frac{g}{2} t^2$$

Necesitamos resolver estas ecuaciones en el intervalo de tiempo  $0 < t \leq t_{x_{max}}$  para condiciones de lanzamiento prescritas  $V_0 > 0$  y  $0 < \theta \leq \pi/2$ . Entonces el valor máximo de la altitud y el rango se calculan junto con su respectivo tiempo de llegada. Finalmente, queremos graficar  $V$  versus  $\theta$ , donde

$$V = \sqrt{(V_0 \cos \theta_0)^2 + (V_0 \sin \theta_0 - gt)^2}$$

y

$$\theta = \tan^{-1} \left( \frac{V_y}{V_x} \right)$$

Debemos tener en cuenta que cuando estudiamos las soluciones basadas en estas fórmulas que la resistencia del aire se asumió como **cero** y la aceleración gravitacional se asumió era constante.



**Paso 3:** Las entradas requeridas son  $g$ ,  $V_0$ ,  $\theta$  y un número finito de tiempo entre  $t = 0$  y el momento en que el objeto vuelve al suelo.

Las salidas son el alcance y el tiempo de vuelo, la altura máxima y el tiempo en que se alcanza, y la forma de la trayectoria en forma gráfica.

**Pasos 4 y 5:** El plan de estructura desarrollado para resolver este problema se presenta a continuación como programa MATLAB.

Por tanto, se debe comenzar por estudiar el diseño de programas relativamente simples como el que se describe en esta sección. La evaluación y comprobación del código es la siguiente:

```
%
% The projectile problem with zero air resistance
% in a gravitational field with constant g.
%
% Written by D. T. Valentine ..... september 2006
% An eight-step structure plan applied in MATLAB:
%
% 1. Definition of the input variables.
%
g = 9.81; % Gravity in m/s/s.
vo = input('What is the launch speed in m/s?');
tho = input('What is the launch angle in degrees?');
tho = pi*tho/180; % Conversion of degrees to radians.
% 2. Calculate the range and duration of the flight.
%
txmax = (2*vo/g) * sin(tho);
xmax = txmax * vo * cos(tho);
%
% 3. Calculate the sequence of time steps to compute trajectory.
%
dt = txmax/100;
t = 0:dt:txmax;
%
% 4. Compute the trajectory.
%
x = (vo * cos(tho)) .* t;
y = (vo * sin(tho)) .* t - (g/2) .* t.^2;
%
% 5. Compute the speed and angular direction of the projectile.
% Note that vx = dx/dt, vy = dy/dt.
%
vx = vo * cos(tho);
vy = vo * sin(tho) - g .* t;
v = sqrt(vx.*vx + vy.*vy);
th = (180/pi) .* atan2(vy,vx);
%
% 6. Compute the time, horizontal distance at maximum altitude
```

```

tymax = (vo/g) * sin(tho);
xymax = xmax/2;
ymax = (vo/2) * tymax * sin(tho);
%
% 7. Display output.
%
disp(['Range in m = ', num2str(xmax), 'Duration in s = ', num2str(txmax)])
disp(' ')
disp(['Maximum altitude in m = ', num2str(ymax), 'Arrival in s = ', num2str(tymax)])
plot(x,y,'k',xmax,y(size(t)), 'o',xmax/2,ymax, 'o')
title(['Projectile flight path, vo = ', num2str(vo), ' th = ', num2str(180*th/pi)])
xlabel('x'), ylabel('y') % Plot of Figure 1.
figure % Creates a new figure.
plot(v,th,'r')
title('Projectile speed vs. angle')
xlabel('V'), ylabel('\theta') % Plot of Figure 2.
%
% 8. Stop.
%

```

**Pasos 6 y 7:** El programa se evaluó ejecutando una serie de valores del ángulo de lanzamiento y velocidad de lanzamiento dentro de las especificaciones requeridas.

El ángulo de 45 grados se verificó para determinar que se produjo el rango máximo en este ángulo para todas las velocidades de lanzamiento especificadas.

Esto es bien conocido por el caso de resistencia del aire cuyo valor es cero en un campo de fuerza  $g$  constante.

→ Ejecutando este código de computadora para  $V_0 = 10 \text{ m/s}$  y  $\theta_0 = 45^\circ$ , se crea el gráfico de la trayectoria de vuelo y orientación versus velocidad, están en las Figuras 1 y 2, respectivamente.

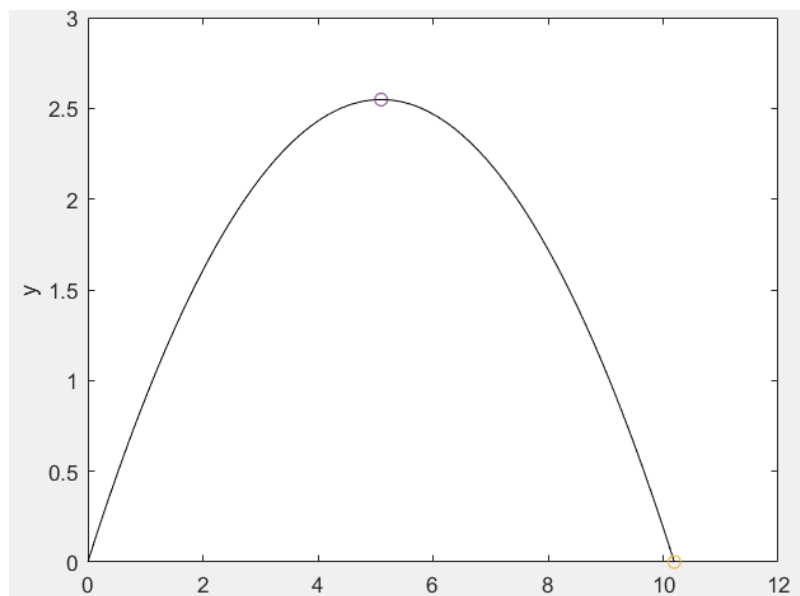


Figura 1 Trayectoria de vuelo del proyectil,  $V_0 = 10$ ,  $\theta_0 = 45^\circ$

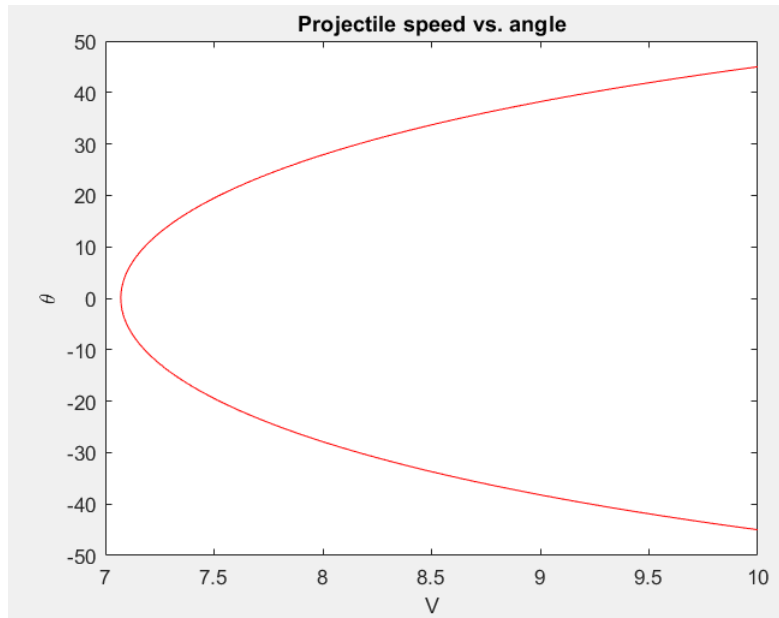


Figura 2 Ángulo del proyectil versus la velocidad

Es muy importante que el ingeniero o el científico prueben a sí mismos que la herramienta que están usando es válida para el problema que están tratando de resolver.

Para ilustrar lo fácil que es encontrar ejemplos escriba `MATLAB examples` en uno de los motores de búsqueda disponibles y encuentre el siguiente script (entre muchos otros):

```
t = (0:.1:2*pi)';  
subplot(2,2,1)  
plot(t,sin(t))  
subplot(2,2,2)  
plot(t,cos(t))  
subplot(2,2,3)  
plot(t,exp(t))  
subplot(2,2,4)  
plot(t,1./(1+t.^2))
```

Este script ilustra cómo poner cuatro gráficos en una ventana de una sola figura.

Verifique que funciona, escriba cada línea en la ventana de comandos seguida de Enter. Note que las posiciones de cada gráfico; su ubicación está determinada por los tres números enteros en la lista de argumentos de la función de `subplot`. Busque en la Ayuda a través `Help` marque (?) para obtener más información sobre la `subplot`.

## Programming MATLAB functions

function  
M Files

MATLAB le permite crear sus propios archivos M-files. Un archivo de función Mfiles es similar a un archivo de secuencia de comandos en que también tiene una extensión .m. Sin embargo, ~~una función de archivo M-files difiere de un archivo de script en que un archivo M-files se comunica con el espacio de trabajo de MATLAB solo a través de argumentos de entrada y salida especialmente designados.~~

Las funciones son herramientas indispensables a la hora de solucionar un problema en piezas lógicas manejables.

Las funciones matemáticas breves se pueden escribir en una línea como objetos en línea. La capacidad de esto se ilustra con un ejemplo en la siguiente subsección. En la siguiente subsección se introducen ideas esenciales incorporadas en la creación de MATLAB function como archivos M-files .

### Inline objects: Harmonic oscillators

Si dos osciladores armónicos acoplados, por ejemplo, dos masas conectadas con un resorte en una mesa muy suave, se consideran como un solo sistema, la salida del sistema en función del tiempo  $t$  podría estar dado por algo como

$$h(t) = \cos(8t) + \cos(9t).$$

Puede representar  $h(t)$  en la línea de comando creando un objeto en línea de la siguiente manera:

```
h = inline( 'cos(8*t) + cos(9*t)' );
```

Ahora escriba algunas declaraciones de MATLAB en la ventana de comandos que usen su función  $h$  para dibujar el gráfico en la figura,

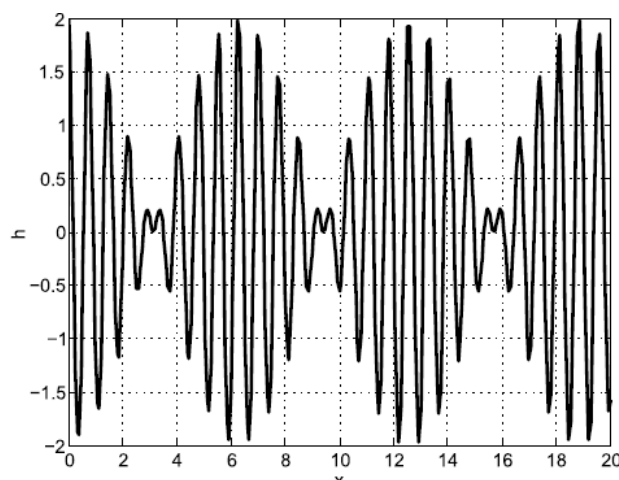


Figura  $\cos(8t) + \cos(9t)$ .

```
x = 0 : 20/300 : 20;  
plot(x, h(x)), grid
```



## FACULTAD DE INGENIERÍA DE SISTEMAS INGENIERÍA EN COMPUTACIÓN

---

Nota:

- ▶ La variable  $t$  en la definición en línea `inline` de `h` es el argumento de entrada. Es esencialmente una variable "ficticia", y sólo sirve para proporcionar información a la función del mundo exterior. Puede utilizar cualquier nombre de variable aquí; eso no tiene que ser el mismo que se usó cuando invoca la función.
- ▶ Puede crear funciones de más de un argumento con `inline`. Por ejemplo,

```
f1 = inline( 'x.^2 + y.^2' , 'x' , 'y' );  
f1(1, 2)  
ans =  
  
5
```

Tenga en cuenta que los valores de entrada de  $x$  e  $y$  pueden ser matrices y, por lo tanto, la salida `f1` será una matriz del mismo tamaño que  $x$  e  $y$ . Por ejemplo, después de ejecutar las dos líneas de comando anteriores para obtener `ans = 5`, intente lo siguiente

```
x = [1 2 3; 1 2 3]; y = [4 5 6; 4 5 6];  
f1(x,y)  
ans =  
  
17    29    45  
17    29    45
```

La respuesta es una operación elemento por elemento que conduce al valor de la función para cada combinación de elementos de matriz en ubicaciones similares.

## MATLAB function: $y = f(x)$

Las características esenciales de una función de MATLAB están incorporadas en el siguiente ejemplo. Consideremos la función  $y$  definida por la relación  $y = f(x)$ , es decir,  $y$  es una función de  $x$ . La construcción de un archivo de función comienza con la declaración de un comando de función. A esto le sigue la fórmula que es la función de interés que desea sustituir por un valor particular de  $x$  para obtener el valor correspondiente de  $y$ . El plan de estructura para la evaluación de un particular la función algebraica es la siguiente

- PL.
1. `function y = f(x) % where $x$ is the input and $y$ is the output.`
  2. `y = x.^3 - 0.95*x;`
  3. `end % This is not necessary to include; however, it plays the role of STOP`

El archivo de función M-file creado en base a este plan es el siguiente;

```
function y = f(x) % where $x$ is the input and $y$ is the output.  
y = x.^3 - 0.95*x;  
end % This is not necessary to include; however, it plays the role of  
STOP.
```

Se guardó como Ejem5funtion.m. Después de guardarlo, se usó de la siguiente manera en el Comannd Window. Prueba el siguiente ejemplo:

```
f(2)  
ans =  
6.1000
```

A continuación, vamos a crear una función que tome como entrada los tres coeficientes de la ecuación cuadrática, a saber,

$$ax^2 + bx + c = 0.$$

Queremos determinar las dos raíces de esta ecuación, es decir, queremos determinar la solución a esta ecuación. Una forma de abordar este problema es aplicar las soluciones conocidas de la ecuación cuadrática. Hagamos esto creando un archivo de función siguiendo el plan de estructura para el algoritmo completo para encontrar la (s) solución (es) de  $x$ , dados los coeficientes  $a$ ,  $b$  y  $c$  tomados de la siguiente manera:

QUADRATIC EQUATION STRUCTURE PLAN

```
1. Start
2. Input data (a, b, c)
3. If a = 0 then
    If b = 0 then
        If c = 0 then
            Display 'Solution indeterminate'
        else
            Display 'There is no solution'
    else
        x = -c/b
        Display x (only one root: equation is linear)
    else if b² < 4ac then
        Display 'Complex roots'
    else if b² = 4ac then
        x = -b/(2a)
        Display x (equal roots)
    Else
        x₁ = (-b + √(b² - 4ac))/(2a)
        x₂ = (-b - √(b² - 4ac))/(2a)
        Display x₁, x₂
4. Stop.
```

Un archivo de función basado en este plan es el siguiente:

```
a = 4; b = 2; c = -2;
function x = quadratic(a,b,c)
% Equation:
% a*x^2 + b*x + c = 0
% Input: a,b,c
% Output: x = [x1 x2], the two solutions of this equation.
if a==0 && b==0 && c==0
disp(' ')
disp('Solution indeterminate')
elseif a==0 && b==0
disp(' ')
disp('There is no solution')
elseif a==0
disp(' ')
disp('Only one root: equation is linear')
disp(' x ')
x1 = -c/b;
x2 = NaN;
elseif b^2 < 4*a*c
disp(' ')
disp(' x1, x2 are complex roots ')
disp(' x1 x2')
x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
elseif b^2 == 4*a*c
x1 = -b/(2*a);
x2 = x1;
disp('equal roots')
disp(' x1 x2')
else
x1 = (-b + sqrt(b^2 - 4*a*c))/(2*a);
x2 = (-b - sqrt(b^2 - 4*a*c))/(2*a);
disp(' x1 x2')
end
if a==0 && b==0 && c==0
elseif a==0 && b==0
else
disp([x1 x2]);
end
end
```

Esta función se guarda con el nombre de archivo quadratic.m.

```
a = 4; b = 2; c = -2;
quadratic(a,b,c)
x1 x2
    0.5000 -1.0000
who
Your variables are:
a b c
```

Las dos raíces en este caso son reales. Es un ejercicio útil para probar todas las posibilidades para evaluar si esta función se ocupa con éxito de todas las funciones cuadráticas y ecuaciones con coeficientes constantes. El propósito de este ejemplo fue ilustrar cómo construir un archivo de función. Tenga en cuenta que las únicas variables en el espacio de trabajo son los coeficientes  $a$   $b$   $c$ . Las variables definidas y necesarias





## **FACULTAD DE INGENIERÍA DE SISTEMAS INGENIERÍA EN COMPUTACIÓN**

---

en la función no están incluidas en el espacio de trabajo. Por lo tanto, las funciones no se desordenan el espacio de trabajo con las variables que solo se necesitan en la propia función para ejecutar los comandos que contiene.

## MATLAB functions & \*data import-export utilities

En este punto, debería saber cómo escribir un programa MATLAB cuyas entradas datos, realizan operaciones aritméticas simples en los datos, tal vez involucrando bucles y decisiones, y mostrar los resultados del cálculo en un comprensible formulario. Sin embargo, es probable que tenga problemas más interesantes en ciencia e ingeniería impliquen funciones matemáticas especiales como senos, cosenos, logaritmos, etc. MATLAB viene con una gran colección de tales funciones; tenemos que ya hemos visto algunos de ellos. Este capítulo presenta las funciones más comunes disponibles en MATLAB. Además, es posible que desee importar datos trazados u operados matemáticamente, y exportar datos para uso futuro. Por eso, Este capítulo también le presenta la importación de datos en MATLAB espacio de trabajo de varias fuentes. También analiza la exportación de datos a archivos en su directorio de trabajo para su uso posterior con MATLAB o con otras herramientas de software.

### COMMON FUNCTIONS

Una breve lista de funciones y comandos de MATLAB se incluye a continuación con algunas más comunes. Utilice `helpwin` al mando línea para ver una lista de categorías de funciones, con enlaces a descripciones de funciones. Alternativamente, vaya a la lista de Contenidos en el Navegador de ayuda (el panel izquierdo en el navegador de ayuda), y expanda sucesivamente MATLAB, **Function Reference**, Referencia de funciones de MATLAB donde puede elegir **Functions by Category** (Funciones por Categoría), o **Alphabetical List of Functions** (Lista alfabética de funciones).

Tenga en cuenta que, si el argumento de una función es una matriz, la función se aplica, elemento por elemento a todos los valores de la matriz, por ejemplo,

```
sqrt([1 2 3 4])
```

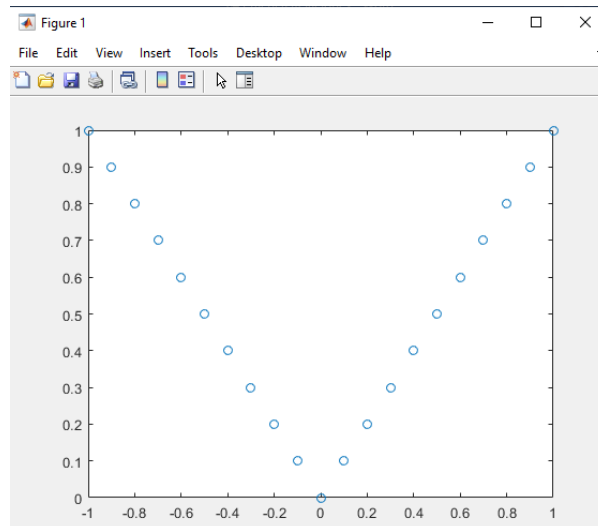
```
ans =
```

```
1.0000    1.4142    1.7321    2.0000
```

Dado que este texto está escrito en un estilo tutorial, se espera que examine la siguiente lista de algunas de las funciones más comunes en MATLAB. Es También, en cierta medida, esperaba que ya lo supiera, desde sus primeros cursos en matemáticas y ciencia, algo sobre estas funciones. Una forma de examinar estas funciones es trazarlas. Diviértete experimentando con ¡MATLAB en su investigación de las siguientes funciones! Por ejemplo, haz el siguiente ejercicio para todas las funciones de la variable  $x$  asignada, como se ilustra

```
x = -1:.1:1;  
plot(x,abs(x),'o')
```

Debería obtener una ilustración que se parezca a una V.



`abs (x)`  
valor absoluto de x.

`acos (x)`  
arco coseno (inverso de coseno) de x entre 0 y  $\pi$ .

`acosh (x)`  
inverso de coseno hiperbólico de x, por ejemplo  $\ln(x + \sqrt{x^2 - 1})$

`asin (x)`  
arco seno (inverso del seno) de x entre  $-\pi/2$  y  $\pi/2$ .

`asinh (x)`  
inverso del seno hiperbólico de x, por ejemplo  $\ln(x + \sqrt{x^2 + 1})$


`atan (x)`  
arco tangente de x entre  $-\pi/2$  y  $\pi/2$ .

`atanh (x)`  
inverso de tangente hiperbólica de x, por ejemplo,  $\frac{1}{2} \ln \left( \frac{1+x}{1-x} \right)$

`ceil (x)`  
entero más pequeño que excede x

clock

hora y fecha en un vector de seis elementos, por ejemplo, las declaraciones

 `t = clock;`

`fprintf( ' %02.0f:%02.0f:%02.0f\n', t(4), t(5), t(6) );`

resultado en 14:09:03. Observe cómo las horas, los minutos y los segundos se dejan rellenos con ceros si es necesario.

`cos(x)`

coseno de  $x$ .

`cosh(x)`

coseno hiperbolico de  $x$ , por ejemplo  $\frac{e^x + e^{-x}}{2}$  (ver figura)

`cot(x)`

cotangente de  $x$ .

`csc(x)`

cosecante de  $x$ .

`cumsum(x)`

suma acumulativa de los elementos de  $x$ , por ejemplo,

`cumsum(1:4)`

devuelve

`ans =`

1      3      6      10

`date`

tiempo en formato de caracteres de cadena `dd-mmm-yyyy`, por ejemplo, `02-Feb-2001`.

`exp(x)`

valor de la función exponencial

`fix(x)`

redondea al entero más cercano hacia cero, por ejemplo, `fix(-3.9)` devuelve `-3`

`fix(3.9)` devuelve `3`.

`floor(x)`

entero más grande que no exceda  $x$ , es decir, se redondea al número entero más cercano, por ejemplo

`floor(-3.9)` devuelve `-4`, `floor(3.9)` devuelve `3`.

`length(x)`

número de elementos del vector  $x$ .

`log(x)`

logaritmo natural de  $x$ .



## FACULTAD DE INGENIERÍA DE SISTEMAS INGENIERÍA EN COMPUTACIÓN

---

`log10(x)`

logaritmo base 10 de  $x$ .

`max(x)`

elemento máximo del vector  $x$ .

`mean(x)`

valor medio de elementos del vector  $x$ .

`min(x)`

elemento mínimo del vector  $x$ .

`pow2(x)`

$2^x$ .

`prod(x)`

producto de elementos de  $x$

→ Examinemos a continuación un ejemplo que, con suerte, lo inspirará a examinar todos los de las funciones enumeradas, así como cualquier otra función que pueda descubrir que admite MATLAB. Consideremos el arco-coseno, el arco-seno y las funciones arco-tangente, es decir, `acos(x)`, `asin(x)` y `atan(x)`, respectivamente. Si usted especifica a  $x$ , es decir, el coseno, el seno y la tangente, respectivamente, entre -1 y 1, ¿en qué cuadrante del círculo se seleccionan los ángulos de salida? Para proveer en respuesta, se creó y ejecutó el siguiente script de archivo M-files. El grafico la comparativo de los resultados calculados se ilustra en la Figura. Las REMARKS en el final del guión proporciona una interpretación de los resultados gráficos y, por lo tanto, una respuesta a la pregunta planteada.

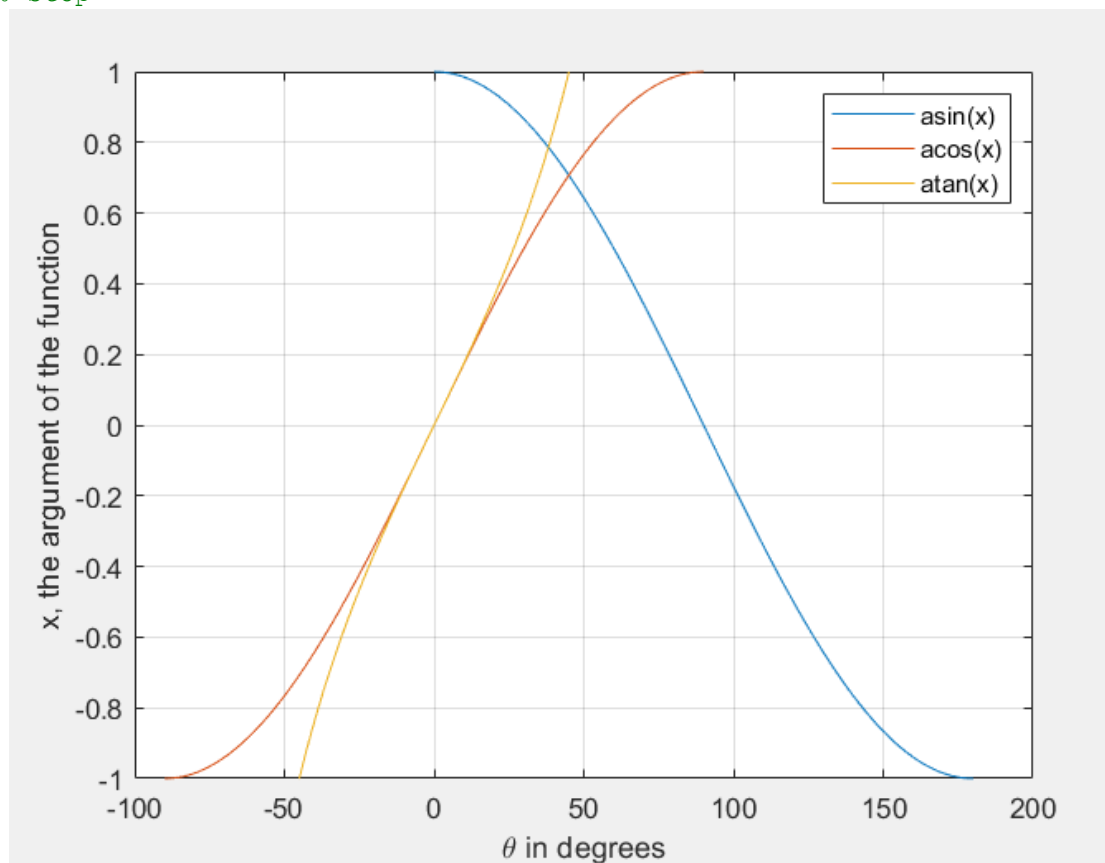
f

```
%  
% Script to compare the acos(x), asin(x), and atan(x)  
% functions over the range -1 < x < 1. The values are  
% converted to angles in degrees. The results are  
% compared graphically.  
%  
% Script prepared by D. T. Valentine - September 2006.  
% Comments modified by D.T.V. ... 2008/2012/2016/2018.  
%  
% The question raised is: What range of angles, i.e.,  
% which of the four quadrants of the circle from 0 to  
% 2*pi are the angular outputs of each of the functions?  
%  
% Assign the values of x to be examined:  
%  
x = -1:0.001:1;
```

```

%
% Compute the arc-functions:
%
y1 = acos(x);
y2 = asin(x);
y3 = atan(x);
%
% Convert the angles from radians to degrees:
%
y1 = 180*y1/pi;
y2 = 180*y2/pi;
y3 = 180*y3/pi;
%
% Plot the results:
%
plot(y1,x,y2,x,y3,x),grid
legend('asin(x)', 'acos(x)', 'atan(x)')
xlabel('\theta in degrees')
ylabel('x, the argument of the function')
%
% REMARKS: Note the following:
% (1) The acos(x) varies from 0 to 90 to 180 degrees.
% (2) The asin(x) varies from -90 to 0 to 90 degrees.
% (3) The atan(x) varies from -90 to 0 to 90 degrees.
% To check remark (3) try atan(100000000) *180/pi.
%
% Stop

```



Comparación de los resultados de las funciones  $\text{acos}$ ,  $\text{asin}$  y  $\text{atan}$ .

## IMPORTING AND EXPORTING DATA

Cuando empiece a programar en serio, a menudo necesitaré almacenar datos en un disco. El proceso de mover datos entre MATLAB y archivos de disco se llama importar (desde archivos de disco) y exportar (a archivos de disco).

Los datos se guardan en el disco *files* en uno de dos formatos: texto o binario. En formato de texto, los valores de los datos son códigos ASCII, y se puede ver en cualquier editor de texto. En formato binario, los valores de los datos son no son códigos ASCII y no se pueden ver en un editor de texto. El formato binario es más eficiente en términos de espacio de almacenamiento requerido.

R  
P  
SAS  
Wea

Esta sección proporciona un breve resumen de las principales formas en que MATLAB importa y exporta datos.

### The load and save commands

Si desea guardar datos entre sesiones de MATLAB, los comandos `save` y `load` son probablemente los mejores para usar.

### Exporting text (ASCII) data

Para exportar (guardar) la matriz

```
A = [1 2 3; 4 5 6]
```

A =

1	2	3
4	5	6

en formato ASCII "delimitado" en el archivo `myData.txt` utilice el comando

```
save myData.txt A -ascii
```

Si ve `myData.txt` en un editor de texto (o lo escribe en la ventana de comandos) le parece a esto:

```
1.0000000e+000 2.0000000e+000 3.0000000e+000  
4.0000000e+000 5.0000000e+000 6.0000000e+000
```

Los delimitadores son los caracteres que se utilizan para separar los valores de los datos en el archivo: espacios por defecto. Puede utilizar tabulaciones en lugar de espacios especificando el calificador `-tabs` en lugar de `-ascii`. Si guarda matrices (cadenas) de esta manera, los códigos ASCII de los caracteres se escriben en el archivo.

## Importing text (ASCII) data

El comando `load` es el reverso de `save`, pero tiene un giro curioso. Si la matriz `A` se ha guardado en `myData.txt` como arriba, el comando

```
load myData.txt
```

se crea una variable en el espacio de trabajo con el mismo nombre que el archivo, menos la extensión, es decir, `myData`. Si no desea que el nombre de archivo sea el nombre de la variable, utilice la forma funcional del comando, por ejemplo,

```
A = load('myData.txt')
```

Los datos importados de esta manera no tienen que ser creados por MATLAB. Usted puede crearlo en un editor de texto, o podría ser creado por cualquier otro programa que exporte datos en formato ASCII.

## Exporting binary data

El comando

```
save filename x y z
```

guarda las variables `x`, `y` e `z` en el archivo `filename.mat` en propiedad de MATLAB formato binario, es decir, un archivo M-files de este tipo que solo puede ser utilizado por MATLAB.

Nota:

- ▶ Si no se enumeran variables, se guarda todo el espacio de trabajo.
- ▶ La extensión `.mat` es la predeterminada; puede especificar una extensión diferente.
- ▶ Busque ayuda para todas las opciones de guardado.





## FACULTAD DE INGENIERÍA DE SISTEMAS INGENIERÍA EN COMPUTACIÓN

---

### Importing binary data

El comando

`load filename`

carga todas las variables de `filename.mat` en el espacio de trabajo; ver `help` para todas las opciones de carga.

### SUMMARY

- ▶ Las funciones de MATLAB se pueden utilizar para realizar una variedad de tareas matemáticas, trigonométricas y otras operaciones.
- ▶ Los datos se pueden guardar en archivos de disco en formato de texto (ASCII) o en formato binario.
- ▶ `load` y `save` se puede utilizar para importar / exportar tanto texto como datos binarios (este último en forma de archivos MAT).

---

#### 4. INFORME DE LA PRÁCTICA

El informe de la practica deberá seguir los siguientes lineamientos:

- Utilizar el formato para la presentación del laboratorio.
- Revisar la **presente** guía de Práctica 4 Relaciones.
- El archivo entregar con el Nombre: Apellido.Nombre.Practica.04
- En la sección de resultados de la práctica deberá presentar los resultados de los siguientes ejercicios:

The problems in these exercises should all be structure-planned before being written up as MATLAB programs (where appropriate).

3.1 The structure plan in this example defines a geometric construction. Carry out the plan by sketching the construction:

1. Draw two perpendicular  $x$ - and  $y$ -axes
2. Draw the points  $A (10, 0)$  and  $B (0, 1)$
3. While  $A$  does not coincide with the origin repeat: Draw a straight line joining  $A$  and  $B$  Move  $A$  one unit to the left along the  $x$ -axis Move  $B$  one unit up on the  $y$ -axis
4. Stop

3.2 Consider the following structure plan, where  $M$  and  $N$  represent MATLAB variables:

1. Set  $M = 44$  and  $N = 28$
2. While  $M$  not equal to  $N$  repeat: While  $M > N$  repeat: Replace value of  $M$  by  $M - N$  While  $N > M$  repeat: Replace value of  $N$  by  $N - M$
3. Display  $M$
4. Stop
  - (a) Work through the structure plan, sketching the contents of  $M$  and  $N$  during execution. Give the output.
  - (b) Repeat (a) for  $M = 14$  and  $N = 24$ .
  - (c) What general arithmetic procedure does the algorithm carry out (try more values of  $M$  and  $N$  if necessary)?

3.3 Write a program to convert a Fahrenheit temperature to Celsius. Test it on the data in Exercise 2.11 (where the reverse conversion is done).

3.4 Write a script that inputs any two numbers (which may be equal) and displays the larger one with a suitable message or, if they are equal, displays a message to that effect.

3.5 Write a script for the general solution to the quadratic equation  $ax^2 + bx + c = 0$ . Use the structure plan in Section 3.2.2. Your script should be able to handle all possible values of the data  $a$ ,  $b$ , and  $c$ . Try it out on the following values:

- (a) 1, 1, 1 (complex roots)
- (b) 2, 4, 2 (equal roots of  $-1.0$ )
- (c) 2, 2,  $-12$  (roots of 2.0 and  $-3.0$ )

The structure plan in Section 3.2.2 is for programming languages that cannot handle complex numbers. MATLAB can. Adjust your script so that it can also find complex roots. Test it on case (a); the roots are  $-0.5 \pm 0.866i$ .

4.1 Write some MATLAB statements which will:

- (a) find the length  $C$  of the hypotenuse of a right-angle triangle in terms of the lengths  $A$  and  $B$  of the other two sides;
- (b) find the length  $C$  of a side of a triangle given the lengths  $A$  and  $B$  of the other two sides and the size in degrees of the included angle  $\theta$ , using the cosine rule:

$$C^2 = A^2 + B^2 - 2AB \cos(\theta).$$

4.2 Translate the following formulae into MATLAB expressions:

- (a)  $\ln(x + x^2 + a^2)$
- (b)  $[e^{3t} + t^2 \sin(4t)] \cos^2(3t)$
- (c)  $4 \tan^{-1}(1)$  (inverse tangent)
- (d)  $\sec^2(x) + \cot(y)$
- (e)  $\cot^{-1}(|x/a|)$  (use MATLAB's inverse cotangent)

4.3 There are 39.37 inches in a meter, 12 inches in a foot, and three feet in a yard. Write a script to input a length in meters (which may have a decimal part) and convert it to yards, feet and inches. (Check: 3.51 meters converts to 3 yds 2 ft 6.19 in.)

---

## 5. CONCLUSIONES Y RECOMENDACIONES

- Realizar las conclusiones y recomendaciones de la práctica.

---

## 6. BIBLIOGRAFÍA

- Para la bibliografía utilizar un estilo estándar por ejemplo IEEE.
- La bibliografía deberá ser de libros, artículos científicos, material académico de instituciones de educación superior, páginas oficiales de organismos de estandarización y propietarios de tecnologías, otros sitios similares.