UnADM
UNIVERSIDAD ABIERTA Y A DISTANCIA DE MÉXICO

Programa de la asignatura:

Simulación dinámica de bioprocesos

U2 | Simulación de procesos y sus variables

DCSBA

BIOTECNOLOGÍA

Índice

Presentación de la unidad......2
Propósito......2
Competencia específica......2
Temario......3
2.1. Sistemas dinámicos......3
2.1.1. Definición de sistema......9
2.1.2. Elementos, complejidad, variables y optimización de sistemas......11
2.2 Simulación de diferentes problemas
2.2.1. Desarrollo de ejercicios básicos......15
2.2.2. Desarrollo de ejercicios aplicables a la ingeniería......20
Actividades......65
Autorreflexiones......65
Cierre de la unidad......65
Fuentes de consulta......66

Presentación de la unidad

En esta unidad conocerás lo que es la simulación y sus tipos. Una vez entendidos estos conceptos, conoceremos algunas de las principales funciones que tiene Matlab para poder simular procesos y obtener gráficas de ellos.

Así, estarás listo para simular varios tipos de problemas de una forma sencilla y rápida, utilizando las formulas básicas y avanzadas de química y termodinámica.

Propósito

Aplicar el conocimiento del software Matlab para simular diversos problemas de bioprocesos.

Competencia específica

Determinar elementos básicos para diseñar sistemas simulados, con el fin de dar soluciones a problemas complejos en termodinámica mediante el uso de un simulador.

Temario

2. Simulación de procesos y sus variables.

2.1. Sistemas dinámicos
- 2.1.1. Definición de sistema
- 2.1.2. Elementos, complejidad, variables y optimización de sistemas

2.2. Simulación de diferentes problemas
- 2.2.1. Desarrollo de ejercicios básicos
- 2.2.2. Desarrollo de ejercicios aplicables a la ingeniería

2.1. Sistemas dinámicos

Introducción

La finalidad de utilizar modelos matemáticos, es poder representar fenómenos que ocurren en procesos biológicos, mediante plantear soluciones basados en ecuaciones matemáticas. Gracias al desarrollo de programas de computadora de diferente índole que son capaces de resolver ecuaciones matemáticas avanzadas, las ciencias biológicas han avanzado de una forma muy importante en los últimos años, ya que el perfeccionamiento de los modelos ha hecho posible el desarrollo de la industria biotecnológica. Todo esto sirve también como apoyo en la experimentación y reología de los medios donde reaccionan productos de origen biológico.

Los reactores agitados son ampliamente utilizados en industria bioquímica; las reacciones biológicas, la transferencia de oxígeno al medio de cultivo, la mezcla en el fermentador son elementos importantes a considerar para su eficiente y adecuada operación. La- representación- de los fenómenos anteriores a través de la utilización de modelos matemáticos, nos permite simular y diseñar de forma adecuada varios tipos de procesos como por ejemplo bioreactores, se pueden usar para simular el comportamiento de fermentadores a escala industrial y en diversas aplicaciones químicas industriales (Matlab nos permitirá resolver problemas desde balancear ecuaciones hasta resolver ecuaciones diferenciales en procesos bioquímicos).

La síntesis de diversos productos biológicos de interés comercial como los antibióticos, las enzimas, los ácidos orgánicos, las vitaminas, se realizan en fermentadores o biorreactores. Estos son recipientes donde se desarrollan los organismos, se provee un ambiente controlado para asegurar el crecimiento y la formación óptima de productos.
En la actualidad los fermentadores industriales presentan un alto grado de equipamiento, en ellos se incluyen sistemas de agitación y dispersión de aire, para garantizar una buena oxigenación del medio y un eficiente mezclado, así como sensores para conocer el cambio de las variables del cultivo tales como: la temperatura, el oxígeno disuelto, el pH, la concentración celular, entre otros.

La caracterización hidrodinámica de los fermentadores apoyada en modelos matemáticos, permite predecir mediante simulación numérica: el tiempo de mezclado, la transferencia de masa, los perfiles de temperatura, los patrones de flujo y propiedades como densidad y viscosidad, a través del fermentador, entre otros.
El contar con simulaciones numéricas de los aspectos antes mencionados, contribuye a la mejor comprensión de los fenómenos y procesos biológicos para su posterior implantación.

La clasificación de matemática de los modelos incluye determinísticos (entradas y salidas son valores fijos) o estocásticos (al menos un valor de salida o entrada es probabilístico); estático (variación del tiempo no es tomada en cuenta) o dinámico (la variación del tiempo es tomada en cuenta junto con las variables). Comúnmente los modelos de simulación son estocásticos y dinámicos.

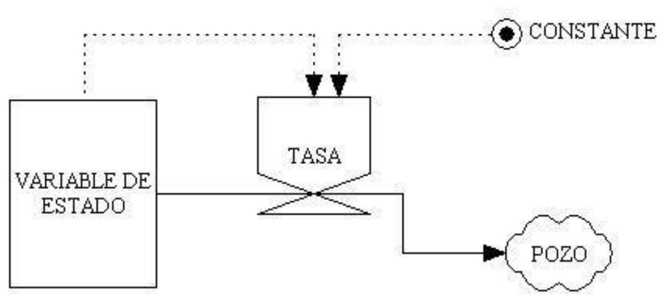
Sistemas dinámicos o dinámica de sistemas

Es una metodología que permite el análisis, en el tiempo, de un sistema, a través de un **modelo de simulación**; es decir, es el estudio de **CAMBIOS** en un sistema.
Sirve para representar un **problema** y proponer alternativas de manejo, mediante la evaluación y comparación de **escenarios**.

Modelo de simulación

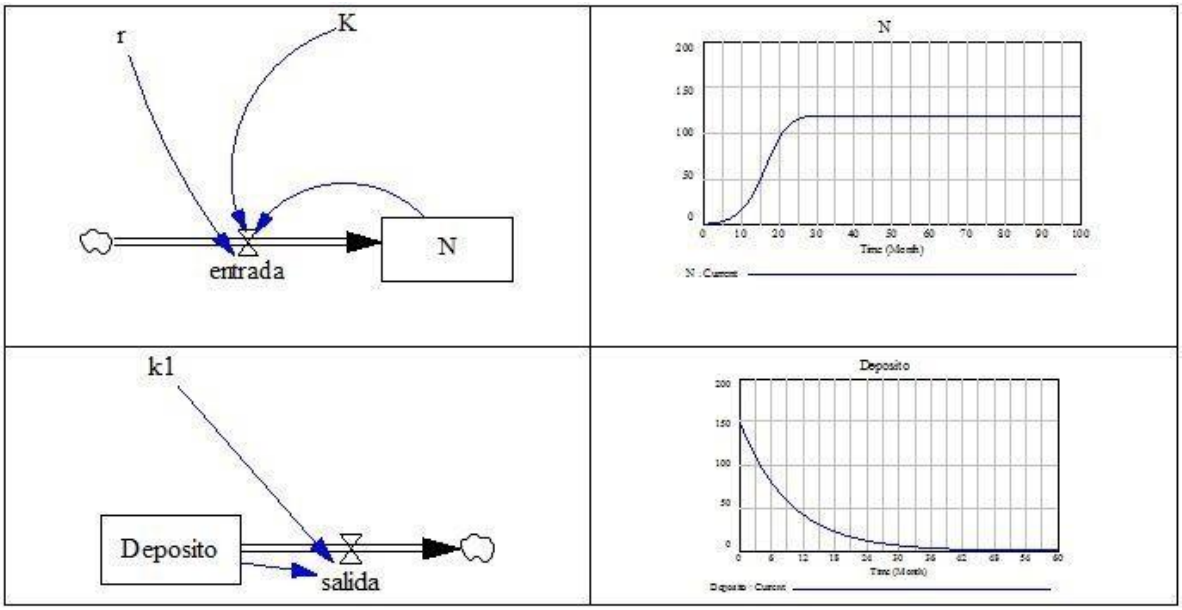
El modelo se construye con base en ciertos componentes, que incluyen:

- variables de estado,
- variables de tasa o flujos,
- constantes y
- variables auxiliares



Las variables de estado (ve) son variables de interés, tales como densidad poblacional, biomasa, peso, altura, plantas enfermas, ingreso económico, etc. Generalmente son características observables y medibles.
Las variables de tasa (vt) representan los flujos entre las variables de estado, son expresadas en forma de ecuaciones diferenciales de primer orden (ED1). Son estimadas a través de cambios en el estado de una variable.
Las constantes (c) son valores considerados fijos en el modelo. Limitan la complejidad del sistema.

Las variables auxiliares (va) se consideran como aquellas que no podemos controlar, pero que influyen en nuestro sistema.



Solución numérica contra solución analítica

La integración numérica nos ayuda a obtener los cambios en el sistema (Euler, Runge-Kutta) de las ecuaciones diferenciales de primer orden que representan las interacciones entre los componentes de nuestro modelo. Modelos analíticos son tomados como *ideales*. El modelo reúne "individuos" que son datos y los considera con valores típicos.

El trabajo más pesado consiste en construir el modelo y recolectar los datos.

Para el procesamiento de datos se requiere poco tiempo y se basa en la utilización de simuladores por ejemplo: Matlab, Simulink, SIMBIO, STELLA, etc.).

Aplicaciones

Las primeras aplicaciones se hicieron en el ámbito industrial, ámbito social, empresarial, ecosistemas, sí también agroecosistemas, problemas físicos.

Pasos

- Una vez identificado el problema, definirlo y establecer objetivos.
- Construir modelo conceptual que incluya la raíz del problema (diagrama).

- Transcribir modelo a notación forrester: ve, vt, va.
- Definir las ecuaciones que ayudan a predecir el comportamiento del sistema.
- Recabar datos.
- Diseñar y probar en el modelo, políticas alternativas que solucionen el problema.
- Depurar, verificar y validar. Proceso iterativo.
- Aplicar (evaluar escenarios).

¿Qué problemas de la vida cotidiana podemos representar mediante un simulador?

Un ejemplo es como afecta la pesca a la población de ballenas.

Figura 1. Modelo en notación forrester para población de ballenas. Más información diagrama de forrester: http://modelaciondinamicasi.blogspot.mx/

Simulación: Es un proceso que consiste en construir un modelo descriptivo de un sistema real, con el propósito de estudiar el comportamiento de dicho sistema a través del tiempo; con la ventaja de que no es necesario interrumpirlo (si es muy costoso), destruirlo (si se desea saber sus límites máximos de resistencia) o construirlo (si es sólo un propuesto). Además la simulación es una herramienta de la investigación de operaciones que nos permite conocer y analizar el comportamiento de un sistema real o propuesto para decidir cursos de acción: modificarlo, aceptarlo o rechazarlo. Utilizando el ejemplo de la población de ballenas que es afectada por la pesca, es fácil cambiar datos en nuestro modelo, pero es muy difícil por tamaño, costo y tiempo el reproducirlo en escala real; el estudio adecuado de las poblaciones de ballenas que se ven reducidas debido a la pesca nos ayuda a generar medidas de regulación internacional que protegerán a estas especies.

Figura 2. Foto tomada de: http://www.greenpeace.org/mexico/es/Campanas/Oceanos-y-costas/Las-ballenas/Santuario-mexicano-para-ballenas1/

Figura 3. Tomada de: http://www.freepik.es/foto-gratis/dos-ballenas_362515.htm

2.1.1. Definición de sistema.

Pueden darse varias definiciones de sistema:

“Conjunto de elementos que interactúan entre ellos” Pierre Delattre 1971."

- “La palabra sistema se refiere a una colección de procesos o eventos interrelacionados, abarcados por una frontera reconocible” (F. K. BERRIEN).

"Unidad consistente en partes mutuamente interactuantes” (ACKOFF)

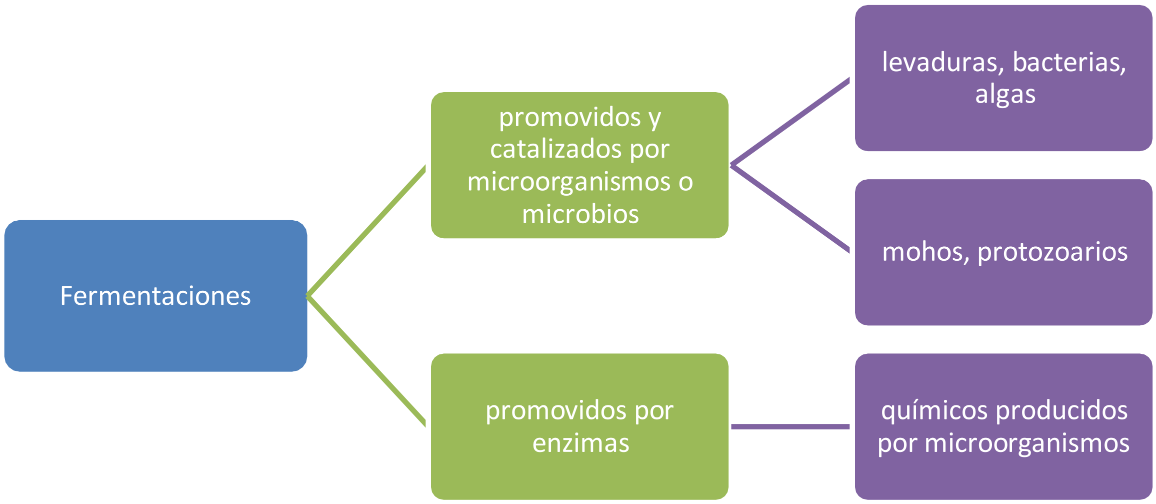
- "Totalidad arbitraria de variables que el investigador escoge de un gran número de variables que pertenecen al sistema real.” (ASHBY)

*Se mencionaron algunas de estas definiciones para darse una idea del mismo.

Un sistema biotecnológico es aquel donde el cambio estructural de la materia se realiza mediante la participación de microorganismos o enzimas en un dominio determinado. Por sí mismo, el sistema biotecnológico constituye un universo extremadamente grande de procesos que van desde las clásicas fermentaciones, hasta la producción de organismos transgénicos. Estos bioprocesos en general, pueden ser descritos y analizados en términos de la técnica y la información disponible utilizada para analizarla.

Generalizando podemos clasificar los sistemas biotecnológicos en procesos elementales fisiológicos, fermentaciones y la acción de entidades vivas.

Podemos clasificar las fermentaciones en:



Recordemos que las fermentaciones son reacciones en que una alimentación de materia prima orgánica se convierte en productos por la acción de enzimas o microbios.

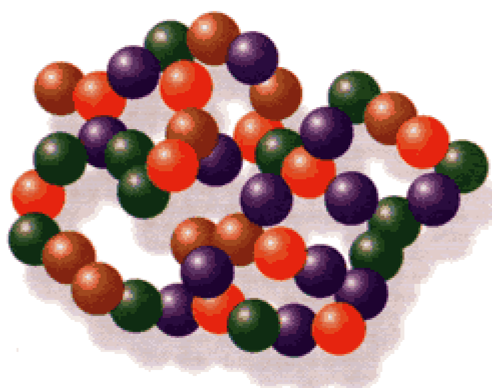
En este tema se utiliza mucho la optimización, explicada más adelante) donde se busca para unos casos minimizar la cantidad de bacterias, en el caso de la utilización de antibióticos y para otros casos maximizar su crecimiento como en el caso de las microalgas para producción de biodisel.

2.1.2. Elementos, complejidad, variables y optimización de sistemas

Elementos

Entidad, actividades, recursos y controles:

- **Entidad:** componente de interés en el sistema, responsable de que el sistema cambie, un ejemplo en biotecnología es un nuevo antibiótico para contrarrestar los efectos de bacterias.
- **Actividades:** Tareas que se llevan a cabo en el sistema; pueden estar relacionadas directa o indirectamente. Todo cambio que sea realizado por un proceso se le determina como actividad. Retomando el tema de poblaciones de ballenas la pesca es una actividad con la que su población se ve afectada.
- **Recursos:** medios indispensables para llevar a cabo las actividades.
- **Controles:** Administran como, donde y cuando se ejecutan las actividades, también determina acciones a tomar dada cierta circunstancia.

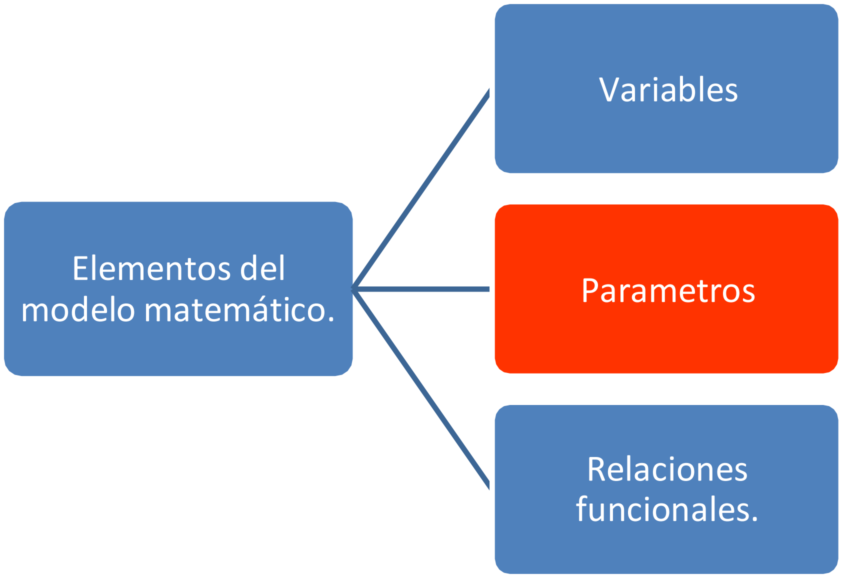


Complejidad

Un sistema se hace complejo por el número de actividades, recursos y controles que lo componen; de manera que se hace difícil de analizar el sistema, dicha complejidad depende de dos factores:

- Interdependencia. Los elementos están relacionados y se afectan entre sí.
- Variabilidad. La incertidumbre produce variación en el comportamiento de los elementos del sistema.

Los modelos matemáticos de sistemas constan de tres elementos:



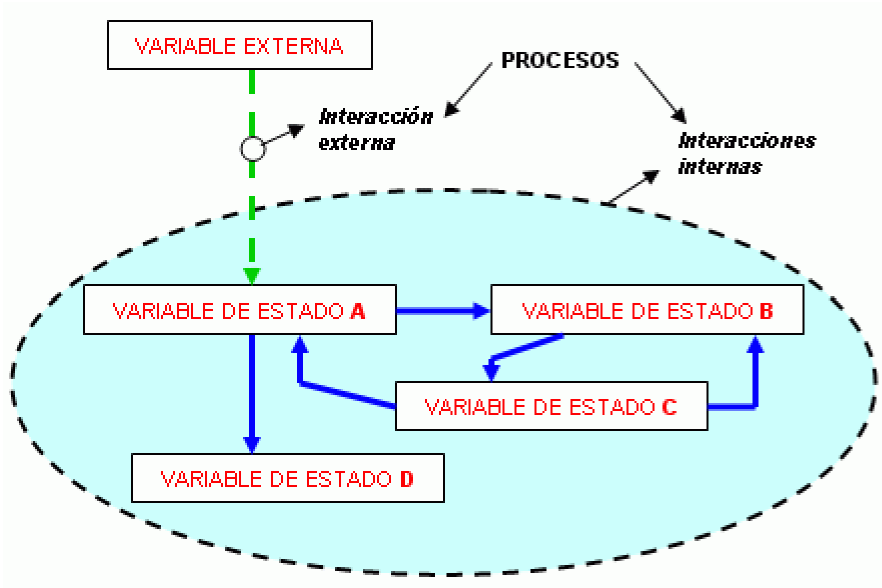
Las variables que aparecen en los modelos se emplean para relacionar un componente con otro y se clasifican como ***variables exógenas, variables de estado y variables endógenas***.

- Las ***variables exógenas*** (variables independientes) o de entrada al modelo, generalmente representada en el modelo matemático como **"x"**. Puede considerarse que estas variables actúan sobre el sistema, pero no reciben acción alguna de parte de él.

- Las ***variables endógenas*** (variables dependientes) o salida del sistema, comúnmente representada en el modelo matemático como **"y"**, son generadas por la interacción de las variables exógenas con las del estado, de acuerdo con las características de operación del último.

Las ***variables de estado*** permiten representar el comportamiento del sistema mediante ecuaciones, para buscar una relación matemática entre las salidas y las entradas del sistema, estas variables están en función del tiempo. En termodinámica, el estado de un sistema está caracterizado por un cierto número de parámetros llamados variables de estado tales como el volumen, la temperatura, la presión, la cantidad de materia, etc. Otro ejemplo es el de una población de náufragos en una isla desierta, la población y los recursos son variables de estado.

El hecho que una variable en particular esté, clasificada como exógena, de estado o endógena, depende del propósito de la investigación.



Parámetros

En el modelo son objetos o símbolos que representan a entidades o atribuciones del sistema que permanecen constantes durante el estudio. Las muy comúnmente utilizadas A, B, etc., en matemáticas, que para nuestro estudio en biotecnología pueden ser cuando un proceso es isocórico, isobárico o isotérmico.

Relaciones funcionales

Son los procesos físicos o las relaciones entre los símbolos de un modelo, que representan a las actividades y a las relaciones entre los elementos de un sistema. Describen la forma en que cambian las variables y como las afectan los parámetros.

Optimización

Concepto de optimización

La Investigación de Operaciones se encarga de encontrar la mejor solución, o la solución óptima, al problema bajo consideración. En lugar de contentarse con sólo mejorar el estado de las cosas, la meta es identificar el mejor curso de acción posible. Aun cuando debe interpretarse con todo cuidado, esta "búsqueda de la más óptima solución" es un aspecto muy importante dentro de la Investigación de Operaciones.
Un problema de optimización consiste en minimizar o maximizar el valor de una variable. En otras palabras se trata de calcular o determinar el valor mínimo o el valor máximo de

una función de una variable; un ejemplo es la optimización de medios de cultivo para microorganismos en la producción de biopreparados de interés agrícola, donde el diseño de un medio de cultivo responde a las exigencias del microorganismo en cuestión, aquí nuestro interés es maximizar su multiplicación, para ello la concentración y selección apropiada de los nutrientes son un factor elemental.

Para ejemplo de minimización utilizaremos uno que debemos tener en cuenta para la sobrevivencia humana y de otras especies, la huella de carbono, relacionado con la contribución del hombre al cambio climático, actualmente se da un seguimiento a varios productos o servicios para determinar la cantidad de rastros de carbono que tienen, la investigación se centra en minimizar las emisiones de CO_2.

Debemos de contestar adecuadamente las siguientes preguntas que nos servirán de guía para resolver el problema:

¿Qué se solicita en el problema?
¿Qué restricciones aparecen en el problema?
La respuesta correcta a la primera pregunta nos lleva a definir la función que deberá ser minimizada o maximizada.

Desde el punto de vista del diseño, para llevar a cabo la optimización del proceso, es importante seguir los siguientes pasos:

1. Definición del problema y selección de la función objetivo.
2. Descripción del modelo matemático del proceso o problema.
3. Solución del modelo matemático mediante una técnica adecuada.
4. Análisis de los resultados.

Métodos de optimización

Programación Lineal: Método Simplex

a) Programación No Lineal
    - Optimización sin restricciones
    - Optimización con Restricciones de Igualdad
    - Optimización con Restricciones de Desigualdad
b) La Programación Mixta-Entera en el Diseño de Procesos
c) Programación Mixta Entera Lineal: Método de “Branch and Bound”
d) Programación Mixta-Entera No Lineal: Método “Outer Approximation”

Para más información:
http://www.iqcelaya.itc.mx/~vicente/NotasSeminario1.pdf

2.2 Simulación de diferentes problemas

2.2.1. Desarrollo de ejercicios básicos

¿Qué es Matlab?

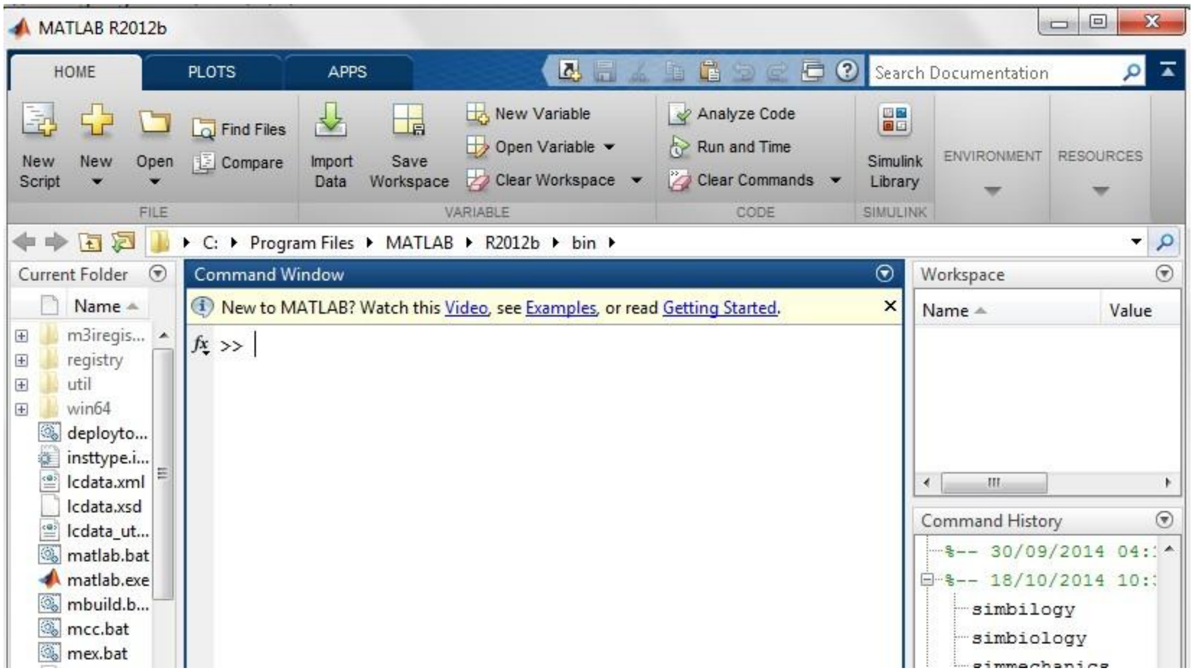
MATLAB es conocido como una herramienta de software matemático que nos permite resolver cualquier tipo de operaciones matemáticas desde sumas simples, ecuaciones de segundo grado, pasando por matrices y derivadas.

Otra definición es que MATLAB es un lenguaje de alto rendimiento para la informática técnica. Se integra computación, visualización y programación en un entorno fácil de usar donde los problemas y soluciones se expresan en notación matemática familiar. Los usos típicos incluyen:

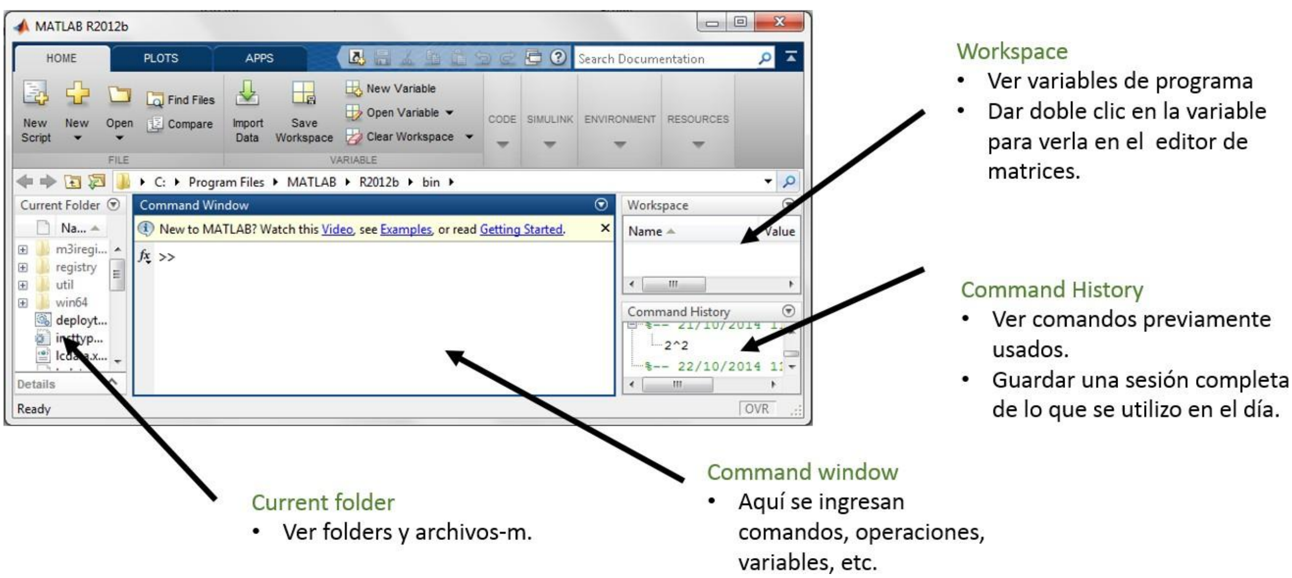
- Matemáticas y computación
- El desarrollo de algoritmos
- Modelado, simulación y creación de prototipos
- Análisis de datos, exploración y visualización
- Gráficos científicos y de ingeniería
- Desarrollo de aplicaciones, incluyendo la construcción de la interfaz gráfica de usuario

Entorno de Matlab

Su interfaz se compone de varias ventanas, que nos proporcionan información sobre la localización de los archivos, ventana de comandos para programación, espacio de trabajo, entre otras. A continuación se muestra la pantalla que se muestra al iniciar el programa, podemos ver GUI (Graphical User Interface) la Interface Gráfica del Usuario.



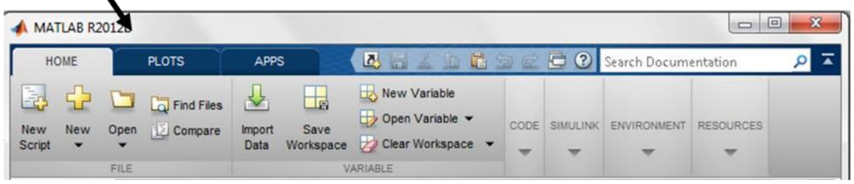
Ventanas principales:



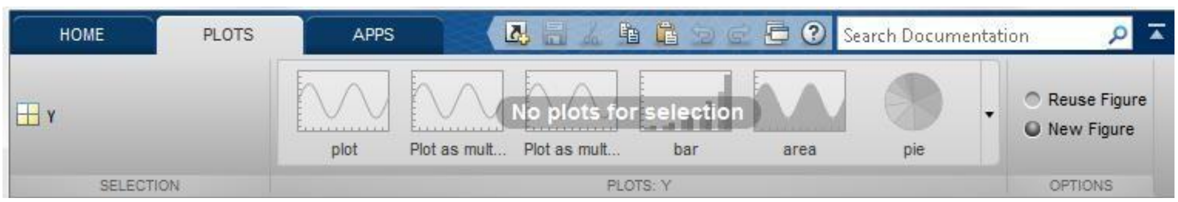
Otras ventanas incluyen editor window, figure window, variable editor, help, etc.

Barra de menús
A través de la barra de menús se accede a las operaciones que no están disponibles en la barra de herramientas.

Barra de Menú.

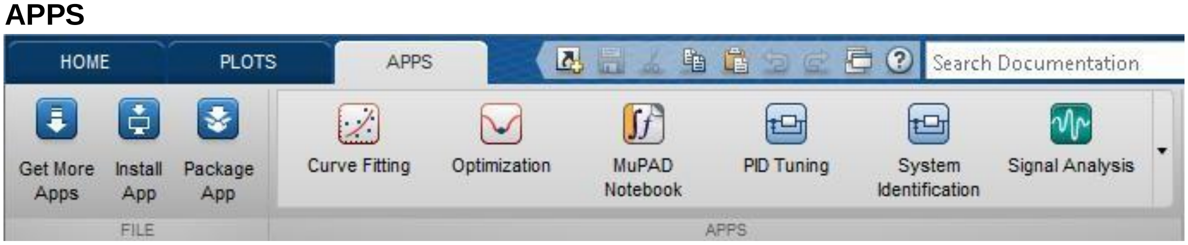


Home
En esta pestaña encontramos las opciones de archivo: Nuevo, Abrir, importar, etc.
Plots



En esta pestaña se encuentran las opciones con las que podrás graficar funciones, matrices, etc.

APPS



Esta pestaña contiene las aplicaciones dirigidas a tematicas concretas como Analisis de señales, optimización, ajuste de curvas, visor de sequencias ,etc.

¿Cómo reconoce los números?

- Enteros (sin punto decimal):
  38 501 -12
- Reales (con punto decimal):
  5. -8.4 31.521
- Reales (notación científica o exponencial):
  4.e-3 = 4 x 10-3 = 0.004

5.2e+4 = 5.2 x 104 = 52000
Nota: Se utilizará PUNTO DECIMAL para separar la parte entera de la decimal y en caso de
Operaciones Aritméticas básicas

OPERACIÓN	SIMBOLO
Suma a+b	+
Resta a-b	-
Multiplicación a*b	*
División a/b	/
Potencia a^b	^

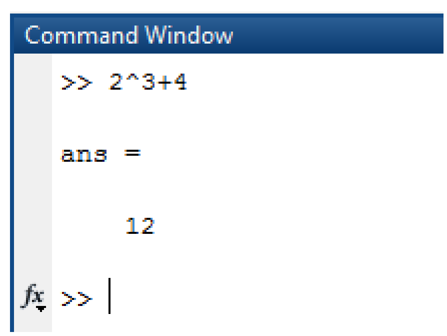
Puede reducirse la utilización de MATLAB al de una calculadora, escribiendo expresiones aritméticas y terminar presionando INTRO. Nos proporciona el resultado a través de la variable del sistema ans (answer).Para no crear eco (la respuesta inmediata de cada orden) en la terminal, se debe agregar a las órdenes "punto y coma" al final de la operación.

Al iniciar nuestras operaciones en command window aparecerán de inicio los símbolos >> después de ellos se iniciará la escritura de las funciones.

En caso de tener varios operadores aritméticos, el orden con que se realizan las operaciones es determinante: EL ORDEN EN QUE LAS OPERACIONES ESTÁN ESCRITAS NO SE EFECTÚAN DE LA FORMA EN QUE SE PRESENTAN.

El orden estará determinado por estas reglas:

1. Exponenciación.
2. Multiplicación y división
3. Suma y resta
4. Para cada grupo, de izquierda a derecha



Se usa paréntesis para modificar el orden:
5. Si hay paréntesis, su contenido se calcula antes que el resto
6. Para paréntesis anidados, se efectúan primero los más internos.

Probar las siguientes operaciones en matlab:

Operación 1.

```
>> 5+3*5
```

Se escribe la operación en command window, se presiona enter y se obtendrá el resultado

```
Ans =
      20
```

Operación 2.

```
>>5+(3*5)
Ans =
      20
```

Operación 3.

```
>> 2+3^4/2

ans =

   42.5000
```

Operación 3.

```
>> (2+3^4)/2

ans =

   41.5000
```

Parte 1. Ejercicios Aritmética básica
Actividad 1

Resuelva en Matlab las siguientes operaciones:

a) 1^3-5-6/3*2
b) (2+4)^3-27
c) (7+4)/(2+3)^2
d) (7-3)^2/(2+1)^3
e) 2*(5+4)^(1/2)/(1+3)^2
f) 5^3/2^2
g) 3*(5-3)^2-3*(2-1)^3
h) 5+3*(2+1)^3+25
i) 3*(3-1)^(1/3)/3*(1+2)^3
j) 7*(4-3)^(1/2)/(1+2)^2+4

2.2.2. Desarrollo de ejercicios aplicables a la ingeniería

Variables en Matlab

Variable representa a aquello que varía o que está sujeto a algún tipo de cambio. Nos referimos a algo que se caracteriza por ser inestable o inconstante, en matemáticas generalmente hablamos de variable con respecto al tiempo, tiende a cambiar su valor conforme transcurre el tiempo.

Matlab almacena resultados en la variable **ans** por defecto, las variables pueden contener hasta 19 caracteres y son sensibles a las máyusculas.
Deben comenzar con una letra.

Algunas variables especiales de Matlab:

Variable	Valor
ans	Nombre por defecto de la variable usada para los resultados
Pi	Razón de una circunferencia a su diámetro
Eps	Número más pequeño tal que, cuando se le suma 1, crea un número en coma flotante en el computador mayor que 1
Inf	Infinito
i y j	i = j = v-1
NaN	Magnitud no numérica
Realmin	El número real positivo más pequeño que es utilizable
Realmax	El número real positivo más grande que es utilizable

Al realizar un calculo Matlab utiliza los valores conocidos en el momento en que se evalua la orden asignada.Utilizando clear podemos borrar todas las variables en el espacio de trabajo.

Para crear una variable en Matlab, simplemente se ha de introducir, en la ventana command window, el nombre de la variable y su valor. Recordar que primero asignaremos las variables. Por ejemplo:

Operación 1.

```
>> x=4;
>> y= x^2
y =

    16
```

Operación 2
>> x=3;
>> y=(2+x)^2

y =   25

Operación 3.
>> a=2;b=1;
>> y=2*a^2+b

y =   9

Operación 4.
>> a=2;b=1;c=3;
>> y=2*a+3*b+4*c

y =   19

Operación 5.

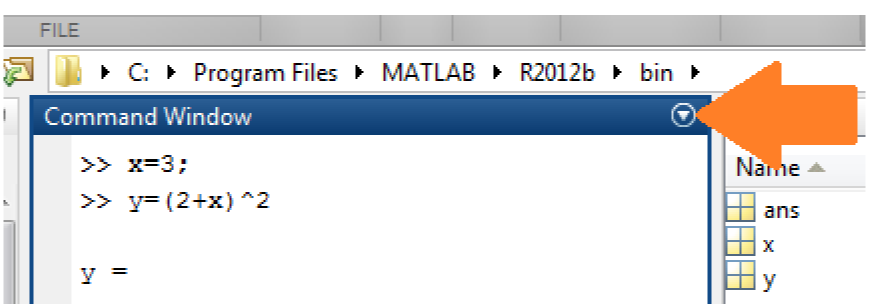
>> a=1; b=3; c=5;
>> y=(2*a+3*b-2*c)^2

y =   1

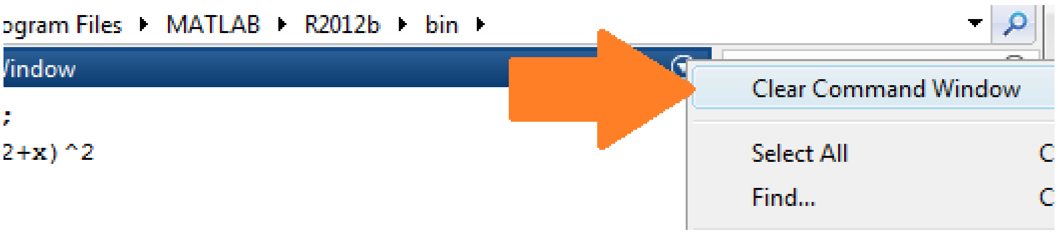
Para limpiar la pantalla de command window

Despues de haber hecho varias operaciones en Matlab es común querer limpiar la pantalla, para ello:

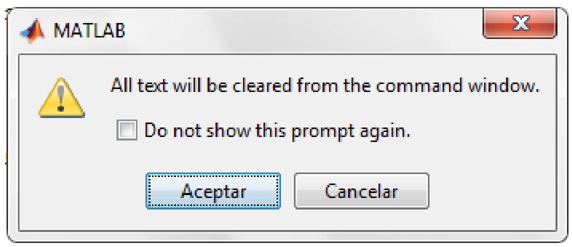
1. Nos dirigimos a la parte superior derecha de command window



2. Seleccionar clear command window.



3. Nos mostrará la siguiente advertencia, a la cual pondremos aceptar.



4. Ya esta libre la pantalla.

Ejercicio

Aplicación a la termodinamica

1. La calibración de un termopar se da manteniendo una de sus dos soldaduras en el punto de hielo t_h= 0.00 °C, y se obtiene:

$$\mathcal{E}(t) = \alpha t + \beta t^2$$

Donde α=0.004 mV/°C y β=4x10-6 mV/°C^2. Determinar la fuerza electromotriz que produce dicho termopar cuando una soldadura se encuentra en el punto de vapor, t_v=100 °C y la otra en el punto de fusión del zinc, t_{Zn}=419.53 °C.

Haremos algunos cambios en cuanto a literales, cambiaremos a α por a, $\mathcal{E}$ por E, β por B, t_v por t_1 y t_{Zn} por t_2 para que Matlab pueda trabajar con ellas. Quedan así nuestros datos: a=0.004, B=0.000004, t_1=100 y t_2=419.53.

$$\mathcal{E}_T = \mathcal{E}(t_2) - \mathcal{E}(t_1)$$

En Matlab registramos lo siguiente:

```
>> a=0.004; B=0.000004; t1=100; t2=419.53;
>>Et= (a*t2+B*(t2)^2)-( a*t1+B* (t1)^2)

Et =

   1.9421
```

Funciones en Matlab

Scripts vs. Funciones

El formato de archivo propio de MATLAB se denomina M-file (tiene extensión .m) y puede ser creado desde cualquier editor de texto externo o desde el editor integrado en MATLAB. Este fichero contiene el código que después será ejecutado desde la línea de comandos.

El M-file puede ser de dos tipos:

Script: se limita a ejecutar una serie de órdenes, trabajando sobre las variables del espacio de trabajo (workspace) de MATLAB.

Función: admite parámetros o argumentos de entrada y devuelve resultados. Opera con variables locales. Es importante tener en cuenta que las funciones sólo se comunican con el workspace a través de las variables de entrada y salida.

El uso de los scripts está recomendado en los casos en los que tengas que efectuar repetidamente una serie de pasos. En cambio, las funciones permiten la creación de nuestras propias aplicaciones para resolver problemas determinados.

Se debe de tener en cuenta lo siguiente:

La función debe llamarse igual que el fichero .m que la alberga.
El nombre de la función debe comenzar por una letra.

Scripts	Funciones
• No aceptan argumentos de entrada ni producen resultados de salida	• Aceptan argumentos de entrada y producen resultados
• Trabajan sobre las variables en el *workspace*	• Por defecto, las variables internas son locales a la función
• Útiles para automatizar una serie de pasos que se repiten muchas veces	• Útiles para extender el lenguaje MATLAB para tus aplicaciones

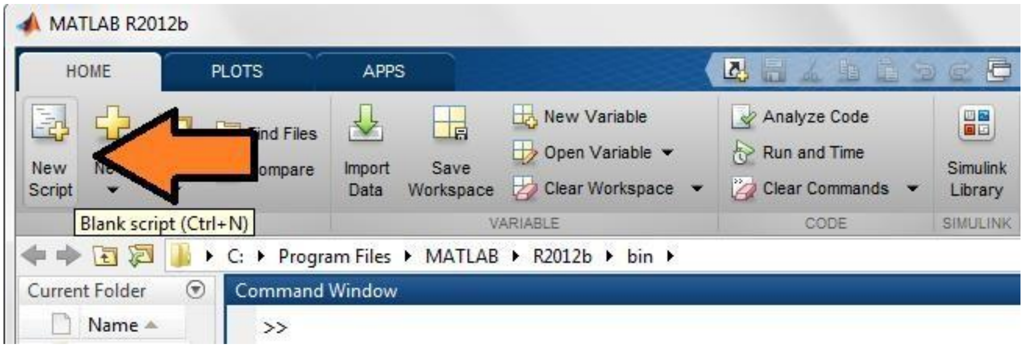
Los *scripts* trabajan sobre variables en el *Workspace* de la línea de comandos o crean nuevas variables que son añadidas a dicho *Workspace*, de modo que todas esas variables pueden ser luego manipuladas desde la línea de comandos.

Funciones en Matlab:
https://www.youtube.com/watch?v=aQInsnJU-qA
Operaciones y funciones en Matlab:
https://www.youtube.com/watch?v=ZUPlcfrmMuQ

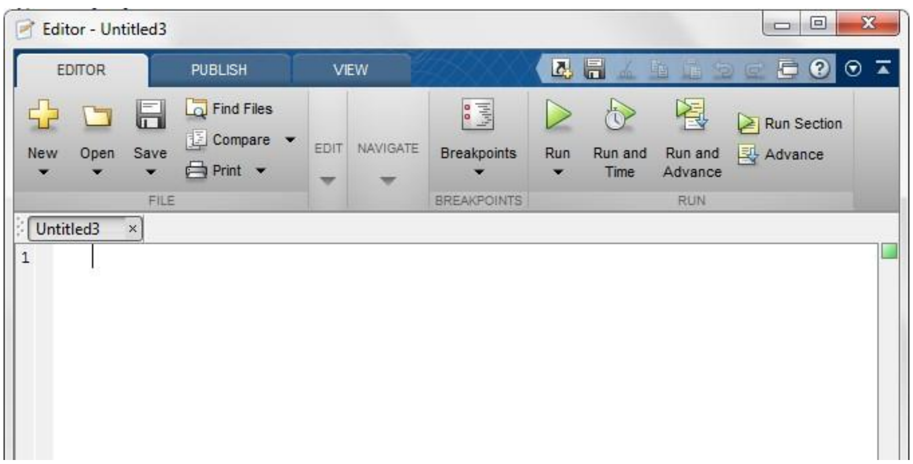
Generando función básica

Suma de dos números

Para generar la función damos clic en **New Script**.



Abre el editor que es la ventana donde trabajaremos y guardaremos nuestra función para después llamarla.

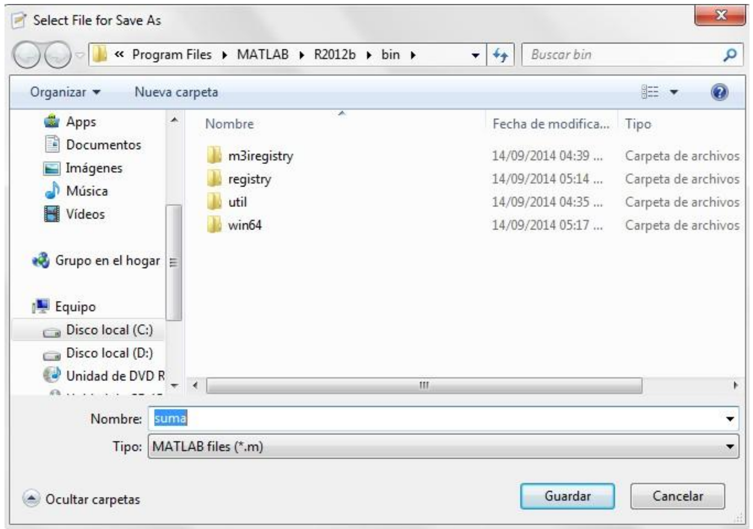


Empezaremos por una función simple de suma que suma dos números “x” e “y”, devuelve la suma z=x+y

Escribiremos lo siguiente en el editor:

function [z] = suma (x,y) %Esta función suma dos números x e y %y devuelve el resultado de la suma en z z=x+y; %efectúa la suma end	1. Definimos la función 2. Para agregar descripción usamos el símbolo de %. 3. Se asigna la formula. 4. Finalizamos la función.

La función se guarda en un fichero. “Notar que queda con el nombre de suma”.



El fichero que guarda la función tiene el mismo nombre que la función, tal como vemos al seleccionar en el Editor File/Save as...

Llamada a la función

La llamada a la función se puede hacer desde la ventana de comandos

>> suma(3,5) ans = 8	1. Al escribir suma estamos llamando la función que guardamos, asignándole valores a sumar. 2. Al presionar enter genera el resultado.

Las variables x, y y z son locales a la función y por tanto, no aparecen en la ventana Workspace, no se puede acceder a ellas desde la ventana de comandos.

Ahora resolveremos una ecuación de segundo grado $ax^2+bx+c=0$.

Nuevamente abrimos el editor y escribimos:

function [x1,x2] = raices_2(p)
    dis= sqrt(p(2)*p(2)-4*p(1)*p(3));
    x1=(-p(2)+dis)/(2*p(1));
    x2=(-p(2)-dis)/(2*p(1));
end	

Guardamos el archivo. Quedará guardado con el nombre raices_2.

Ahora en command window llamamos la función, resolveremos la siguiente ecuación:
$2x^2+3x-6$.

>> [r1,r2]=raices_2([2 3 -6]) r1 = 1.1375 r2 = -2.6375	1. Se asigna a, b y c. 2. Presionamos enter. Se obtiene el valor de las raices.

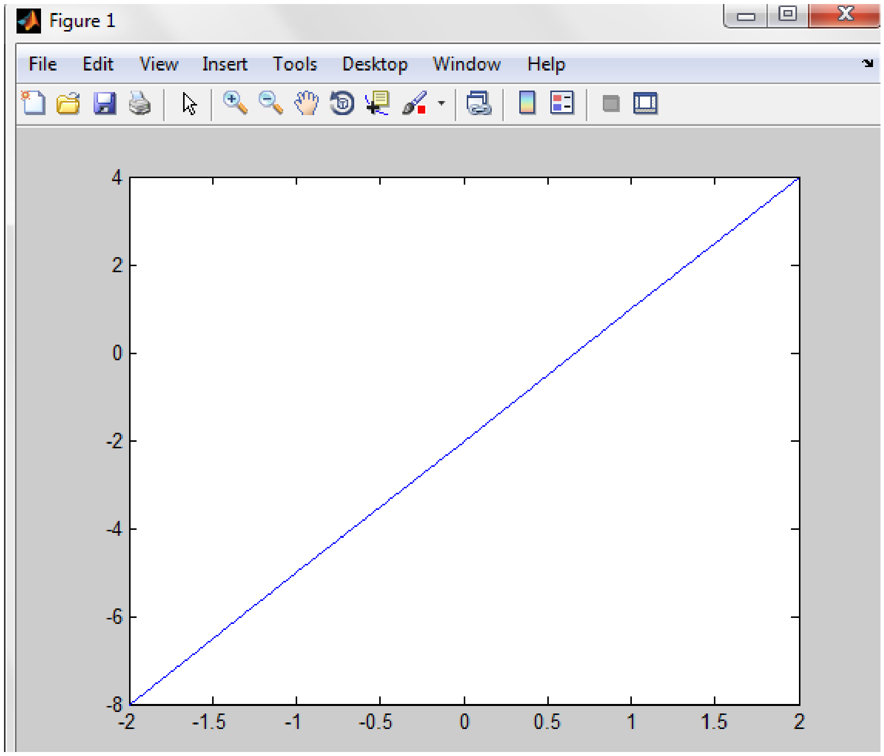
Graficación de Funciones

Matlab genera información a partir de datos, no a partir de funciones, debido a ello es que siempre debemos designar cual es el intervalo de datos en los que se evalurá la función. Para generar la grafica de funciones utilizaremos el comando **plot**.

Ejemplo 1. Para este ejemplo el intervalo será a partir de puntos.

>> x=[-2 -1 0 1 2]; >> y=3*x-2 y = -8 -5 -2 1 4 >> plot (x,y)	1. Asignamos intervalo. 2. Agregamos la función. 3. Presionamos enter y obtenemos el valor del intervalo en y. Ya que contamos con los intervalos podemos graficar. 4. Con el comando plot pedimos graficar los datos en “x” y “y”. Presionamos Enter.

Esperamos un poco a que se abra la siguiente ventana:

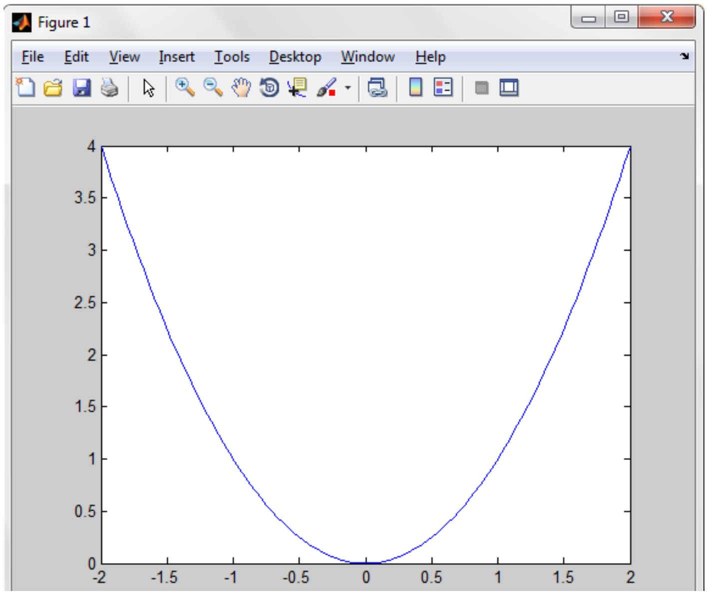


Ejemplo 2

En este ejemplo graficaremos una función cuadratica utilizando un itervalo completo utilizando el comendo linspace.

>>x=linspace(-2,2); >>y = x.^2; >>plot (x,y)	1. Asignamos intervalo.Enter. 2. Agregamos la función. Enter. (importante el punto despues de la x). 3. Graficar. Enter.

Esperamos un poco a que se abra la siguiente ventana:



Funciones elementales

Los nombres de las funciones elementales son las "generales matemáticas".

Los argumentos pueden ser, siempre que tenga sentido, reales o complejos y el resultado se devuelve en el mismo tipo del argumento.

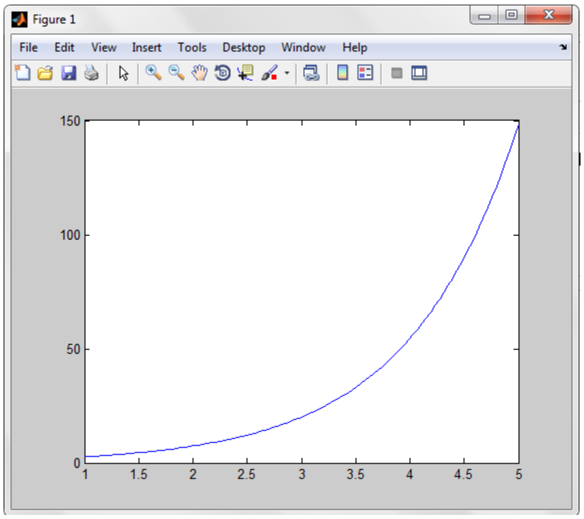
Algunas de ellas son:

sqrt(x)	raiz cuadrada	sin(x)	seno (radianes)
abs(x)	módulo	cos(x)	coseno (radianes)
conj(z)	complejo conjugado	tan(z)	tangente
real(z)	parte real	cotg(x)	cotangente
imag(z)	parte imaginaria	asin(x)	Arcoseno
exp(x)	exponencial	acos(x)	Arcocoseno
log(x)	logaritmo natural	atan(x)	arcotangente
log10(x)	logaritmo decimal	cosh(x)	cos. hiperbólico
rat(x)	aprox. racional	sinh(x)	seno hiperbólico
mod(x,y) rem(x,y)	resultado de dividir x por y. Iguales si x,y>0 .	tanh(x)	tangente hiperbólica
fix(x)	Redondeo hacia 0	acosh(x)	arcocoseno hiperb.
ceil(x)	Redondeo hacia + infinito	asinh(x)	arcoseno hiperb.
floor(x)	Redondeo hacia - infinito	atanh(x)	arcotangente hiperb.

round(x)	**Redondeo al entero más próximo**		

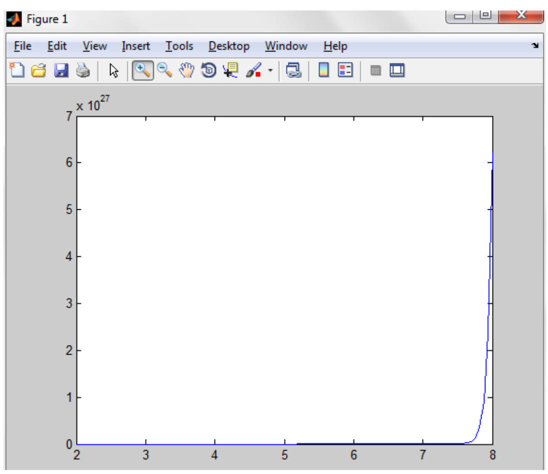
Ejemplo 1

```
>>  x=linspace(1,5);
>> y=exp(x);
>> plot (x,y)
```



Ejemplo 2.

```
>> x=linspace(2,8);
>> y=exp(x.^2);
>> plot(x,y)
```



Más ejemplos:

Conducción de calor:
http://www.sc.ehu.es/sbweb/energias-renovables/MATLAB/calor/conduccion/conduccion.html

Ejercicio

1. La presión atmosférica (p) varía en función de la altura (h) según la siguiente expresión: $p=1035*e^{-012h}$, donde la altura se mide en kilómetros y la presión en milibares.

a. Escribir una función presión que calcule la presión para una altura dada (utilizar la función de MATLAB exp). Determine el intervalo, explique resultados y gráfica.

2. Utiliza los cálculos y los datos del cuadro que sigue para calcular las tasas de natalidad, las tasas de mortalidad y las tasas de crecimiento de la población de tres países utilizando Matlab.

Número de nacimientos (%)	÷	población	X	100	=	tasa de natalidad
Número de muertes (%)	÷	población	X	100	=	tasa de mortalidad
	Tasa de natalidad (%)	–	tasa de mortalidad (%)	=	tasa de crecimiento de la población (%)	

	Nacimientos	Muertes	Población	Tasa de natalidad	Tasa de mortalidad	Tasa de crecimiento de la población
País A	662.000	297.000	33.100.000	[**2%**]	[**0,9%**]	[**1,1%**]
País B	411.000	191.800	27.400.000	[**1,5%**]	[**0,7%**]	[**0,8%**]
País C	211.200	96.800	4.400.000	[**4,8%**]	[**2,2%**]	[**2,6%**]

3. Una población animal está formada por 35.228 individuos. De éstos en un año mueren 25.128 y nacen 26.737. Calcula las tasas de mortalidad y natalidad anuales cada 100 individuos (en %).

4. *Escherichia coli* (E. Coli) es una bacteria que se multiplica por bipartición cada 20 minutos si las condiciones ambientales son favorables. Si partimos de un cultivo bacteriano formado por 1.000 individuos:

a. ¿Cuál será la población teórica si no hay mortalidad y se mantienen las mismas condiciones, al cabo de 6 horas?

b. Representa la curva de crecimiento.

5. En una población aislada en la que no hay migraciones, formada por 5.000 individuos, anualmente nacen 2.000 ejemplares y mueren 1.800. Calcular estos parámetros:

a. La tasa de aumento de la población o *potencial biótico* por individuo y por año. Indícalo también en %.

b. *El tiempo de duplicación.* (tiempo de duplicación **=** *70/tasa de crecimiento en %* o bien tiempo de duplicación **=** *0,7/r*)

6. Cierta población aislada, sin limitación de espacio y alimento, está formada por 1.000 ejemplares.

c. Calcula el n° de individuos que habrá en esa población al cabo de 8 años, sabiendo que la tasa de crecimiento de la misma es del 2% anual (potencial biótico en %). ($N_t = N_o \cdot e^{rt}$ se utiliza para calcular los individuos de cada generación, sabiendo que e = 2,72).

d. El tiempo de duplicación.

Matrices

Para Matlab matrices y vectores son la misma cosa, podemos introducir matrices de varias maneras:

- Por medio de una lista de elementos explicitos.
- Agregando datos desde un fichero externo.
- Utilizando funciones predefinidas de Matlab para generar matrices.
- Crear la matriz directamente utilizando el fichero M.

Crear una matriz de forma explicita.

Es la forma mas sencilla de generar una matriz, solo debemos de escribir sus elementos, pero debemos de considerar las siguientes reglas:

a) Los elementos de la matriz se deben introducir fila por fila.
b) Los elementos de cada fila deben estar separarse por comas o dejar espacios en blanco.
c) El listado de todos los elementos deben estar encerrado entre corchetes [ ].
d) Al final de una fila se debe escribir punto y coma “**;**”.

Nota: El número de elementos de cada fila debe ser el mismo; en caso de no ser así, MATLAB enviará un mensaje de error.

Ejercicio 1. Agreguemos la siguiente matriz a Matlab.

$$A = \begin{matrix} 16 & 3 & 2 & 13 \\ 5 & 10 & 11 & 8 \\ 9 & 6 & 7 & 12 \\ 4 & 15 & 14 & 1 \end{matrix}$$

Agregaremos a Matlab:

>> A=[16 3 2 13; 5 10 11 8; 9 6 7 12; 4 15 14 1] A= 16 3 2 13 5 10 11 8 6 7 12 15 14 1	Asignando filas. Presionamos enter. Nos muestra la Matriz

Función suma, transpuesta y diagonal

Podemos realizar sumas con Matlab con la función sum. Con esta función se suman los valores contenidos en las columnas de las matrices.

Apliquemos esta función a la matriz A:

>> sum (A) **ans =** **34 34 34 34**	**Aplicamos la función suma.** **Presionamos enter.** **Nos arroja el valor de la suma de las columnas.**

En caso de no asignar una variable a la matriz, Matlab usará la variable por defecto ans para poder almacenar el resultado del cálculo. En el ejercicio anterior se obtuvo un vector fila que contiene la suma de las columnas en A.

¿Y la suma de las filas?

Matlab está programado para trabajar en primera intención con columnas en una matriz, debido a ello para realizar la suma por filas será más fácil calcular la **transpuesta de la**

matriz, después hacer el cálculo de la suma de las columnas de la transpuesta y finalmente transponer el resultado.

Para calcular la transpuesta se utilizará una apostrofe ('). Para esta operación se reposicionará la matriz alrededor de la diagonal principal y se transformará de un vector fila a un vector columna.

>> A' ans = 16 5 9 4 3 10 6 15 2 11 7 14 13 8 12 1	Se asigna A, agregando la apostrofe. Presionar enter. Podemos ver el resultado.

Ahora utilizaremos la transpuesta de la transpuesta.

>> sum (A')' ans = 34 34 34 34	Se asigna A, agregando paréntesis y apostrofes. Presionar enter. Podemos ver el resultado, la suma en columna.

La función diag genera un vector con los elementos de la diagonal principal.

>> diag (A) ans = 16 10 7 1	Se asigna la diagonal de A. Presionar enter. Podemos ver el resultado.

Ahora sumaremos las cantidades de esta columna.

>> sum(diag(A))	Se asigna la suma de la diagonal de A. Presionar enter.

ans = 34	Podemos ver la suma.

Operaciones con matrices.
Los operadores básicos en MATLAB para escalares y matrices son los mismos.

OPERACIÓN	SIMBOLO
Suma a+b	+
Resta a-b	-
Multiplicación a*b	*
División a/b	/
Potencia a^b	^
Conjugada	'

Operaciones matriciales puntuales.

Estas operaciones se ejecutan elemento por elemento, las más comunes:

OPERACIÓN	SIMBOLO
Producto	.*
Potencia	.^
División derecha	./

Funciones Matriciales

SIMBOLO	FUNCIÓN
Det	Determinante
Inv	Inversa
Rref	Matriz escalonada
rank	Rango
size	Dimensión de una matriz.
trace	Traza
sqrt	Raíz cuadrada de c/elemento de la matriz.
sqrtm	Raíz cuadrada de una matriz.

Matrices especiales.

Matlab tiene funciones para crear matrices de una forma sencilla, estas son:

SIMBOLO	FUNCIÓN
Eye	Matriz unidad
Zeros	Matriz de ceros
Ones	Matriz de unos
Rand	Matriz de números aleatorios entre 0 y 1.
Magic	Matriz mágica

Agregaremos las siguientes matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 5 \\ 2 & 0 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 1 & 0.2 \end{bmatrix} \qquad C = \begin{bmatrix} -1 & 2 \\ 1 & 3 \\ 1/2 & 0 \end{bmatrix}$$

D= eye (3)   E=ones (2)

Realizaremos algunas operaciones con estas matrices.

Multiplicación

Trabajaremos con la matriz A y B.

<table>
<tr>
<td>

```
>> A=[1 2 3; -1 0 5; 2 0 -1]; B=[1 2; -1 0; 1 0.2];


>> A*B

ans =

    2.0000    2.6000
    4.0000   -1.0000
    1.0000    3.8000
```

</td>
<td>

1. Ingresamos A y B.
2. Presionamos enter.
3. Ingresamos A*B
4. Presionamos enter.

</td>
</tr>
</table>

Potencia

>> A=[1 2 3; -1 0 5; 2 0 -1]; >> A^2 ans = 5 2 10 9 -2 -8 0 4 7	1. Ingresamos A. 2. Presionamos enter. 3. Ingresamos A al cuadrado. 4. Presionamos enter.

Matrices especiales

Eye

>> D=eye (3) D = 1 0 0 0 1 0 0 0 1	1. Ingresamos funcion eye para 3x3. 2. Presionamos enter. 3. Tenemos ya la matriz.

Ones

>> E=ones(2) E = 1 1 1 1	1. Ingresamos funcion ones (unos) para 2x2. 2. Presionamos enter. 3. Tenemos ya la matriz.

Ejercicios:

Con las siguientes matrices:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ -1 & 0 & 5 \\ 2 & 0 & -1 \end{bmatrix} \qquad B = \begin{bmatrix} 1 & 2 \\ -1 & 0 \\ 1 & 0.2 \end{bmatrix} \qquad C = \begin{bmatrix} -1 & 2 \\ 1 & 3 \\ 1/2 & 0 \end{bmatrix}$$

D= eye (3)   E=ones (2)

Encuentre:

a) A*B+A*C
b) B^2
c) B/C
d) C/B
e) C + ones(3,2)
f) A*B Y B*A
g) A*(B+C)
h) C*B'+A^2

Resolviendo un sistema de ecuaciones

Dado el siguiente sistema de ecuaciones encontrar x, y y z.

$$\begin{aligned} 3x+2y-z&=10 \\ 8x-3y+4z&=21 \\ 5x+2y+z&=13 \end{aligned} \qquad \begin{bmatrix} 3 & 2 & -1 \\ 8 & -3 & 4 \\ 5 & 2 & 1 \end{bmatrix} * \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} 10 \\ 21 \\ 13 \end{bmatrix}$$

La forma más sencilla para obtener los valores es asignar a la primera matriz como A y a la ultima como B, y despues hacemos A/B.

Notar que para esta operación la barra es invertida \.

>> A=[3 2 -1;8 -3 4;5 2 1]; B=[10;21;13]; >> A\B ans = 3.2250 -0.7000 -1.7250	1. Ingresamos el valor de A y B. 2. Presionamos enter. 3. Marcamos A\B 4. Presionamos enter. 5. Tenemos el resultado.

Aplicación: balanceo de ecuaciones por el método algebraico

Balanceo de ecuaciones por el método algebraico.

Para balancear ecuaciones se deben considerar los siguientes puntos

1) A cada formula de la ecuación se le asigna una literal y a la flecha de reacción el signo de igual. Ejemplo:

$Fe + O_2$ __________> Fe_2O_3

A B C

2) Para cada elemento químico de la ecuación, se plantea una ecuación algebraica

Para el Fierro $A = 2C$

Para el Oxigeno $2B = 3C$

3) Armaremos nuestro sistema de ecuaciones:

$$1A + 0B = 2C$$
$$0A + 2B = 3C$$

>> A=[1 0; 0 2]; C=[2;3]; >> A\C ans = 2.0000 1.5000	1. Ingresamos el valor de A y C. 2. Presionamos enter. 3. Marcamos A\C 4. Presionamos enter. 5. Tenemos el resultado.

$2Fe + 1.5O_2$ __________> Fe_2O_3

Para convertir a enteros multiplicaremos todo por dos por lo que queda:

$4Fe + 3O_2$ __________> $2Fe_2O_3$

De esta manera podemos balancear ecuaciones por el método algebraico utilizando Matlab.

Ecuaciones diferenciales ordinarias. Aplicación a dinámica poblacional

Las ecuaciones diferenciales permiten representar matemáticamente muchos fenómenos de la naturaleza (la física está llena de ecuaciones diferenciales) y del aspecto social (como la evolución de poblaciones). En biología el estudio de dinámica de poblaciones es importante para conocer la variación de individuos a través del tiempo, se sustenta en ecuaciones diferenciales.

Para conocer más sobre ello:

http://escalera.bio.ucm.es/recursos/claroline/claroline/backends/download.php?url=L1RlbWE0L3Q0Mi5wZGY%3D&cidReset=true&cidReq=MBB2010

Ejercicio: Ecuación diferencial ordinaria (EDO) valor inicial.

En los ejercicios donde se trabaja con ecuaciones diferenciales para resolver problemas de valor inicial, el tipo general es:

$$(d)\ \begin{cases} y' = f(t,y) \\ y(to) = yo \end{cases}$$

En estos problemas se evalua, de entre todas las soluciones de la EDO, aquella en la que **to** vale **yo**.

Por ejemplo:

La ED y' = y -1 tiene innumerables soluciones:

$$y= Ce^{t} + 1$$

En cada valor que evaluemos C se obtendrá una solución diferente:

$y = -e^{t}+ 1$ $y = 2e^{t} + 1$ etc.

Ahora, al buscar entre todas las funciones halladas una que tome el valor de y=2 en t = 0, se encuentra que la solo la siguiente cumple esa condición:

$$y= e^{t}+1$$

Para lo cual tenemos entonces:

$$(d)\ \begin{cases} y' = y - 1 \\ y(0) = 2 \end{cases}$$

Pero como te habrás dado cuenta hacerlo de forma manual consume mucho tiempo, por lo cual utilizaremos matlab, debemos indicar que Matlab no calcula la expresión exacta pero aproxima los valores de la función solución en algunos puntos.

Para calcular en Matlab la aproximación de la solución de:

$$(d)\ \begin{cases} y' = f(t,y) \\ y(to) = yo \end{cases}$$

Se utilizará:
>>f= inline('expresion de f(t,y)', 't','y')
>>ode 23 (f,[t0,tf],y0)

Estos comandos calculan y grafican la aproximación a la solución del problema (d).

>> [t,y]=ode23(f,[t0,tf],y0)

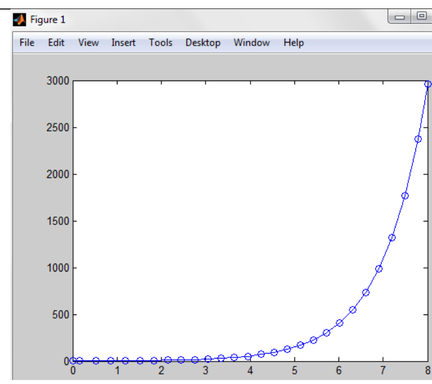
Este último comando genera la aproximación numérica de la solución sin dibujar gráfica. La aproximación de la solución queda guardada en vectores, con la dimensión de t e y, que significa: el valor solución de (d) en t(k) es la aproximación y(k).

Para obtener la longitud de los vectores debemos utilizar el comando:

>> lenght(t)	Nos muestra el número de componentes del vector t.

Probar lo siguiente para probar la Solución de (d).

$$(d)\ \begin{cases} y' = y - 1 \\ y(0) = 2 \end{cases}$$

Usando el intervalo [0,8] Ingresar el Matlab >> f=inline('y-1','t','y') >> ode23(f,[0,8],2)	 Grafica.

Ejemplo dinámica de poblaciones

El número de individuos en poblaciones de bacterias, en ciertas circunstancias, están regidos por la ley:

$$y'(t)=0.3y(t) \quad o \quad y'= 0.3y$$

Donde t es el tiempo en horas, y(t) el número en miles de bacterias que existen en el momento **t**.

Se sabe que al iniciar nuestro experimento se tienen 40 mil bacterias. Queremos saber lo siguiente:

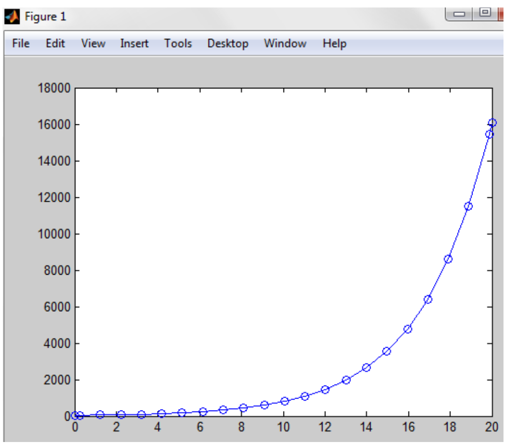
- ¿Cuántas habrá 20 horas más tarde?
- ¿En qué instante habrá 2 000 000 de bacterias?

El modelo matemático es este:

$$\begin{cases} y' = 0.3y \\ y(0) = 20 \end{cases} en\ [0,20]$$

f=inline('0.3*y','t','y') f = Inline function: f(t,y) = 0.3*y >> ode23(f,[0,20],40)	1. Ingresamos los paremetros con el comando inline. 2. Presionamos enter 3. Nos muestra el valor de f. 4. Ingresamos los parámetros para el intervalo en el que se graficará nuestra gráfica.

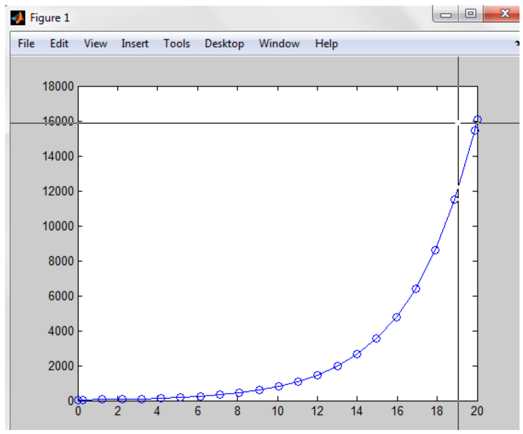
De la gráfica podemos obtener que a las 20 horas se tendrán aproximadamente $16x10^6$ bacterias.



Si queremos un resultado mucho más exacto ingresamos en matlab:

>> ginput(1)

Nos mostrará un cursor que utilizaremos dentro de la gráfica para indicar el punto dentro de la curva que satisface al dato buscado, damos clic sobre el dato y obtenemos su dato numérico.



Aparecerá en la ventana de comandos el punto donde se dio clic lo siguiente:
ans =

1.0e+04 *

0.0020   1.6079

¿En qué instante habrán 2 000 000 bacterias?

Nuevamente utilizaremos el mismo comando ginput(1).

>>ginput (1) ans = 1.0e+03 * 0.0129 2.0263	1. Se ingresa el comando. 2. Se presiona enter. 3. Aparece la gráfica. 4. Dar clic en el punto que corresponde a 2000 (recordar que el valor está en miles). 5. Nos arroja el resultado.

Nota: el resultado obtenido es aproximado

Ley del enfriamiento de Newton

En química o biología contamos con esta ley que se basa en ecuaciones lineales, que nos permite conocer la variación de temperaturas con respecto al tiempo.

Ejercicio: Un horno que se utiliza para fabricar pasteles está a temperatura constante de 45°. Se coloca un pastel con una constante de enfriamiento de 0.03. Si al comienzo la temperatura del pastel es de 18°.

- ¿Qué temperatura tendrá después de 40 min?
- ¿Cuánto tiempo tardará en ponerse a temperatura de 46°?

Apliquemos la ley del enfriamiento

$$\text{(t)} \left\{ \begin{array}{l} y' = 0.03(45-y) \\ y(0) = 18 \end{array} \right. \quad en\ [0{,}40]$$

Ingresemos estos valores a Matlab.

>> f=inline('0.03*(45-y)','t','y') >> ode23(f,[0,40],18)	

Ejercicio:

Para algunos modelos de pesca utilizamos la ecuación:

$$y' = \alpha \ln\left(\frac{k}{y}\right) y - qy$$

Donde el término -qy mide el efecto negativo que ejerce la pesca sobre el crecimiento de alguna población en específico. Estos modelos ayudan a analizar la sostenibilidad de los bancos de pesca.

Calcula y grafica (en el cuadro [0,100] x [0,50]) las soluciones del problema.

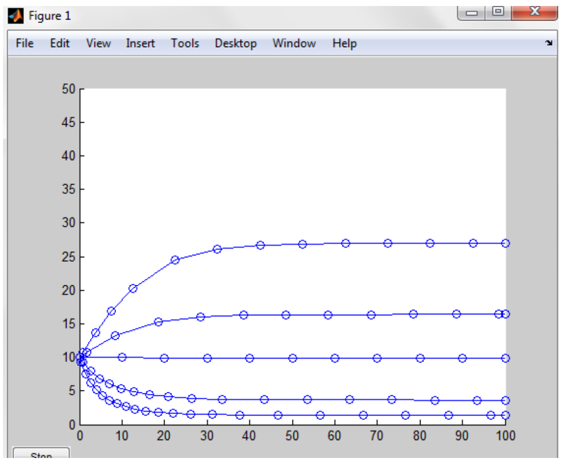
$$\left\{ \begin{array}{l} y' = 0.1 \ln\left(\frac{27}{y}\right) y - qy \\ y(0) = 10 \end{array} \right.$$

Para los valores del parámetro q : 0 , 0.05 , 0.1 , 0.2 , 0.3

Nos interesa tener en una sola ventana las diferentes gráficas para ello utilizaremos el comando hold de matlab.

>> axis([0,100,0,50]); hold on >> f=inline('0.1*log(27/y)*y-0*y','t','y') >> ode23(f,[0, 100], 10) >> f=inline('0.1*log(27/y)*y-0.05*y','t','y') >> ode23(f,[0, 100], 10) >> f=inline('0.1*log(27/y)*y-0.1*y','t','y') >> ode23(f,[0, 100], 10) >> f=inline('0.1*log(27/y)*y-0.2*y','t','y') >> ode23(f,[0, 100], 10) >> f=inline('0.1*log(27/y)*y-0.3*y','t','y') >> ode23(f,[0, 100], 10)	1. Ingresamos las dimensiones requeridas y hold on para la gráfica. 2. También ingresamos todas las ecuaciones. 3. Después de haber ingresado todo presionamos enter.

Nuestra gráfica queda así:



Sistema diferencial ordinario para el modelo Lotka-Volterra para poblaciones que interactúan presa-depredador.

Para un sistema diferencial ordinario se cuenta con varias ecuaciones diferenciales e incógnitas. Este sistema de ecuaciones permite modelar situaciones en las que varias poblaciones conviven en el mismo hábitat.
El modelo de Lotka-Volterra, conocido como modelo de presa-depredador, simula matemáticamente la situación en la que hay dos especies que conviven y una de ellas es depredada por la otra.

Si asignamos a $\mathbf{y_1(t)}$ como el número de presas en el instante **t** y por $\mathbf{y_2(t)}$ el número de depredadores en el instante **t**, el modelo de Lotka-Volterra establece que el número de individuos de cada especie evoluciona en el tiempo de acuerdo con el sistema diferencial:

$$\begin{cases} y1' = ay1 - by1y2 \\ y2' = -cy2 + dy1y2 \end{cases}$$

en el que las constantes **a, b, c** y **d** varían en cada caso debido a que dependen de la natalidad y agresividad de cada especie. Obsérvese que ahora se tienen dos incógnitas y dos ecuaciones.

A este sistema habrá que añadir, como en el caso de una sola ecuación, unas condiciones iniciales que indiquen cuál es la situación de partida, es decir, cuántos individuos de cada especie hay en el instante inicial:

Utilizando matlab debemos escribir el sistema en notación vectorial.	$\begin{cases} \begin{pmatrix} y_1' \\ y_2' \end{pmatrix} = \begin{pmatrix} a\ y_1 - b\ y_1\ y_2 \\ -c\ y_2 + d\ y_1\ y_2 \end{pmatrix} \\ \begin{pmatrix} y_1(t_0) \\ y_2(t_0) \end{pmatrix} = \begin{pmatrix} A \\ B \end{pmatrix} \end{cases}$

Para definir f que toma adquiere valores vectoriales, la función **f** depende de **t** y del vector **y**.	$f(t,y) = f\left(t, \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}\right) = \begin{pmatrix} a\ y_1 - b\ y_1\ y_2 \\ -c\ y_2 + d\ y_1\ y_2 \end{pmatrix}$

La forma general que ingresaremos a Matlab es la siguiente:

```
>> f=inline('[a*y(1)-b*y(1)*y(2);-c*y(2)+d*y(1)*y(2)]','t','y')
```

Contamos entonces con una solución análoga, donde notamos que la condición inicial es también un vector:

```
>> ode23(f,[t0,tf],[A;B])
```

Ejercicio:

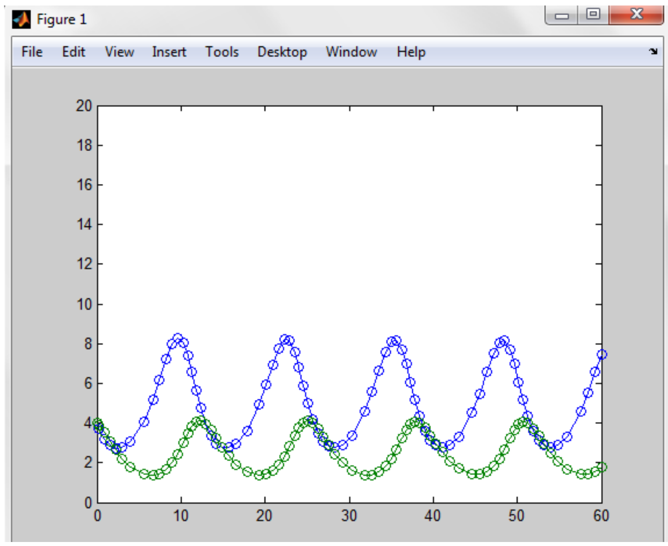
Encontrar la solución del siguiente sistema:

$$\begin{cases} y_1' = \ \ 0.5\ y_1 - 0.2\ y_1\ y_2 \\ y_2' = -0.5\ y_2 + 0.1\ y_1\ y_2 \\ y_1(0) = 4 \\ y_2(0) = 4 \end{cases}$$

Ingresamos a matlab lo siguiente:

```
>> f=inline('[0.5*y(1)-0.2*y(1)*y(2); -0.5*y(2)+0.1*y(1)*y(2)]','t','y')
>> ode23(f,[0,60],[4;4])
>> axis([0,60,0,20])
```

Presionamos enter.



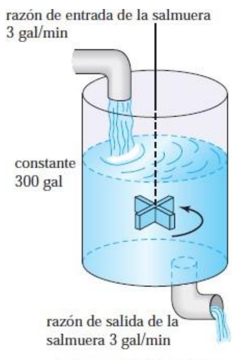
Podemos observar un comportamiento periódico singular de este tipo de modelo.

Modelo lotka-volterra:
http://matema.ujaen.es/jnavas/web_modelos/labiologia/practica5.pdf

Desarrollo y simulación en mezclas

Un tanque contiene inicialmente 300 galones de una solución de salmuera. Al tanque entra y sale sal porque se bombea una solución a un flujo de 3 gal/min, se mezcla con la solución original y sale del tanque con un flujo de 3 gal/min. La concentración de la solución entrante era 2 lb/gal, por consiguiente, la entrada de sal era Rentra = (2 lb/gal) • (3 gal/min) = 6 lb/min y salía del tanque con una razón R_{sale} = (A/300 lb/gal) • (3 gal/min) = A/100 lb/min. Si había 50 lb de sal disueltas en los 300 galones iniciales, ¿cuánta sal habrá en el tanque pasado un gran tiempo?

SOLUCIÓN. Para encontrar la cantidad de sal A(t) en el tanque al tiempo t, resolvemos el problema con valores iniciales.



Tanque de llenado.

$$\frac{dA}{dt} = \begin{pmatrix} \text{razón de} \\ \text{entrada} \\ \text{de la sal} \end{pmatrix} - \begin{pmatrix} \text{razón de} \\ \text{salida} \\ \text{de la sal} \end{pmatrix} = Rentra - Rsale \qquad (7)$$

Ahora, puesto que la solución sale del tanque con la misma razón con la que entra, el número de galones de la salmuera en el tanque al tiempo *t* es una constante de 300 galones. Por lo que la concentración de la sal en el tanque así como en el flujo de salida es *c(t)* = A(t)/300 lb/gal, por lo que la razón de salida R_{sale}, de sal es

R_{sale} = Concentración de sal en el flujo de salida*razón de salida de salmuera = razón de salida de sal.

$$Rsale = \left(\frac{A(t)}{300} lb/gal\right) * (3\ gal/min) = \frac{A(t)}{300}\ lb/mi$$

La razón neta, ecuación (7) entonces será:

$$\frac{dA}{dt} = 6 - \frac{A}{100}$$

Ecuación (8)

$$\frac{dA}{dt} + \frac{1}{100}A = 6, \quad A(0) = 50.$$

Aquí observamos que la condición adjunta es la cantidad inicial de sal A(0) = 50 en el tanque y no la cantidad inicial de líquido. Ahora como el factor integrante de esta ecuación diferencial lineal es $e^{t/100}$, podemos escribir la ecuación como:

$$\frac{d}{dt}[e^{t/100}A] = 6e^{t/100}.$$

Integrando la última ecuación y despejando A se obtiene:

Integrando la última ecuación y despejando A se obtiene la solución general $A(t) = 600 + ce^{-t/100}$. Conforme t = 0, A = 50, de modo que c = —550. Entonces, la cantidad de sal en el tanque al tiempo t está dada por:

$$\mathbf{A(t) = 600 - 55Oe^{-t/100}.} \qquad \mathbf{(6)}$$

La solución (6) se usó para construir la gráfica de la ***Ilustración 1***. En la ecuación (6) y en la figura 3.1.4a también se puede ver, que A(t) —> 600 conforme t—>∞. Por supuesto que esto es lo que se esperaría intuitivamente en este caso; cuando ha pasado un gran tiempo la cantidad de libras de sal en la solución debe ser (300 gal)(2 lb/gal) = 600 Ib.

Podemos suponer que la razón con que entra la solución al tanque es la misma que la razón con que sale. Sin embargo, el caso no necesita ser siempre el mismo; la salmuera mezclada se puede sacar con una razón r_{sale}, que es mayor o menor que la razón r_{entra}, con la que entra la otra salmuera. Por ejemplo, si la solución bien mezclada del ejemplo 5 sale con una razón menor, digamos de $r_{sale} = 2$ gal/min, entonces se acumulará líquido en el tanque con una razón de $r_{entra}-r_{sale}$=(3—2)gal/min=1 gal/min. Después de t minutos (1 gal/min) • (t min) = t gal se acumularán, por lo que en el tanque habrá 300 + t galones de salmuera. La concentración del flujo de salida es entonces c(t) = A/(300 + t) y la razón con que sale la sal es $R_{sale} = c(t) \cdot r_{sale}$, o

$$R_{sale} = \left(\frac{A}{300 + t}\,\text{lb/gal}\right) \cdot (2\ \text{gal/min}) = \frac{2A}{300 + t}\,\text{lb/min.}$$

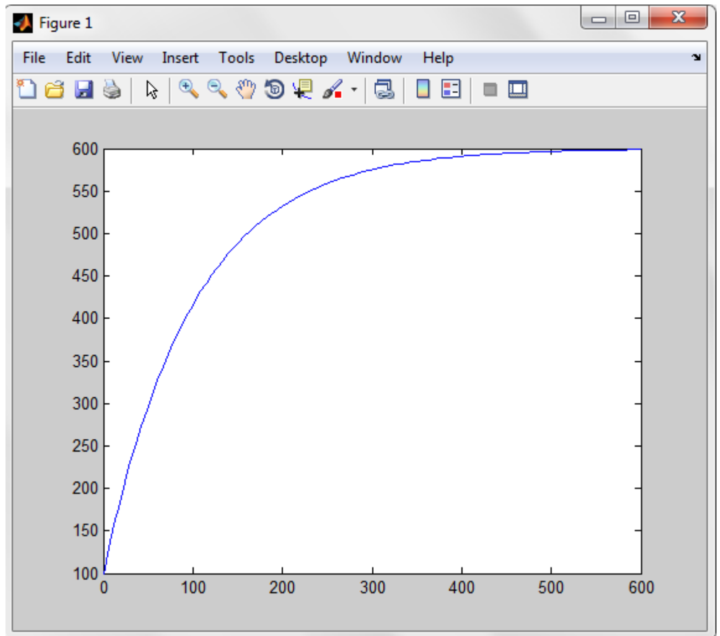
Por tanto, la ecuación (5) se convierte en:

$$\frac{dA}{dt} = 6 - \frac{2A}{300 + t} \qquad \text{o} \qquad \frac{dA}{dt} + \frac{2}{300 + t}A = 6.$$

Debe comprobar que la solución de la última ecuación, sujeta a A(0) = 50, es $A(t) = 600 + 2t — (4.95 \times 10^7)(300 + t)^{-2}$.

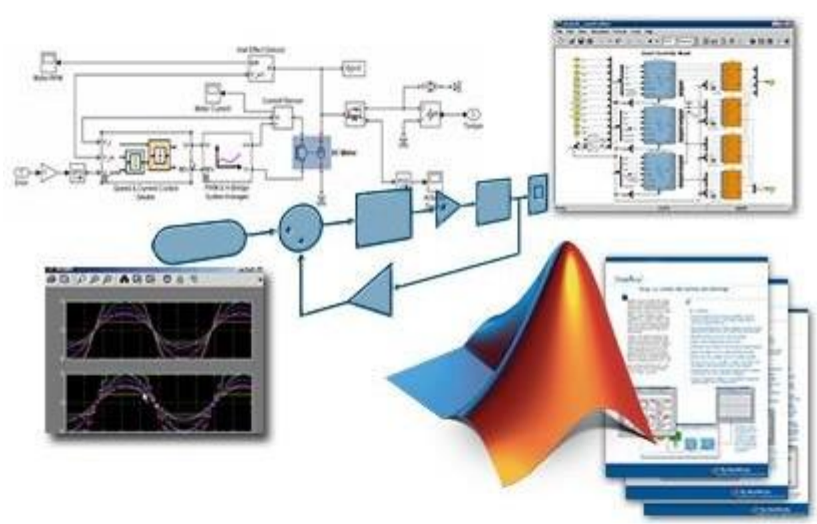
Graficar en Matlab.

>> x=linspace (0,600); >> y= 600-500*exp(-x/100); >> plot (x,y)	1. Ingresamos el intervalo. 2. Ingresamos la ecuación. 3. Impresión de la gráfica.



Aplicación con Simulink

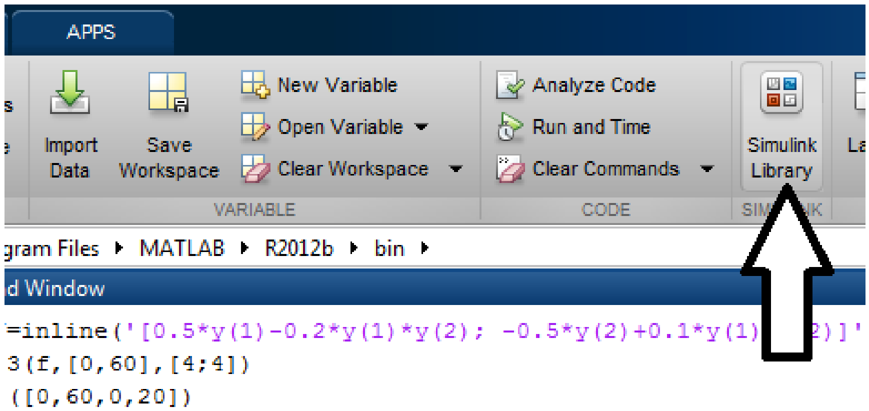
Simulink es una parte de Matlab, nos permite también simular modelos matemáticos, la diferencia es que está organizado por bloques que ya tienen predefinidas funciones que solamente tenemos que posicionar en nuestra hoja y determinarles sus parámetros.



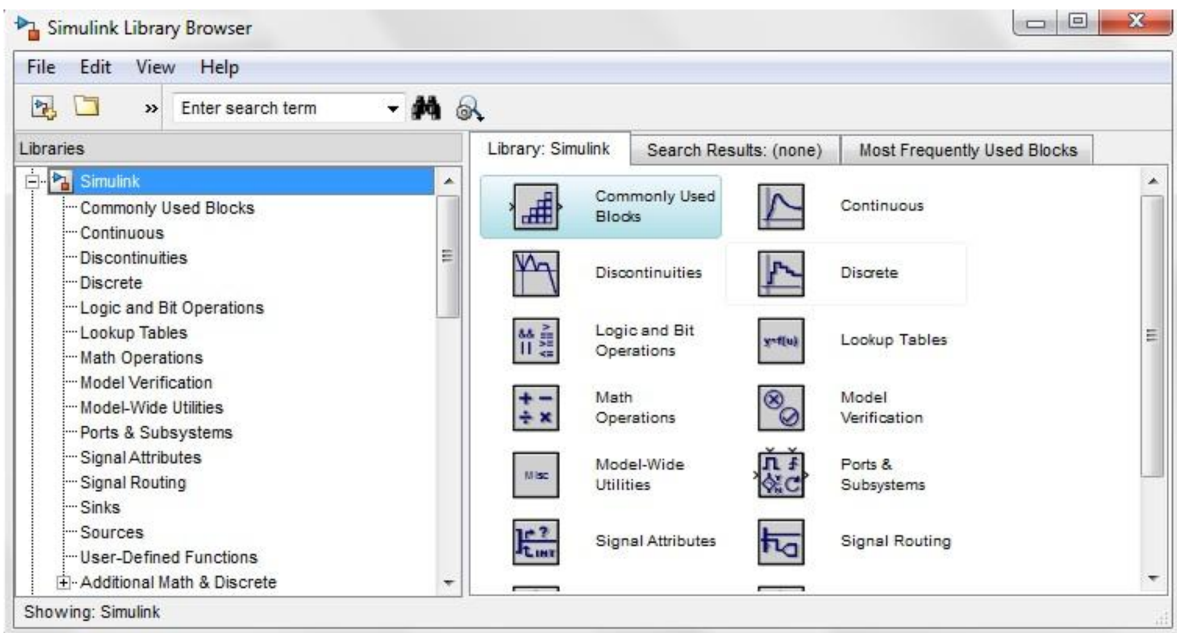
Su principal tarea es resolver Ecuaciones Diferenciales de Primer Orden de manera numérica.

Resolviendo EDO con simulink

Para iniciar simulink daremos un clic en:



Que es el icono correspondiente a la librería de simulink. Abrirá la siguiente ventana.



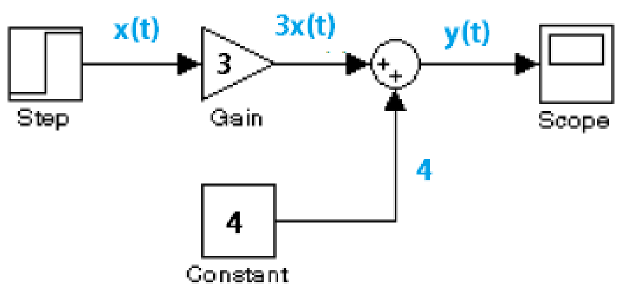
Podemos notar que contiene varias sublibrerias dentro de esta opción.

Modelando una expresión algebraica.

Iniciaremos con el ejemplo de una población en donde su crecimiento sigue la forma y=mx+b, donde m y b son constantes. Suponga m=3 y B=4, entonces x es la variable independiente relacionada al tiempo. Una vez dado x en la ecuación es fácil obtener la variable dependiente y, hagámoslo ayudados por simulink.

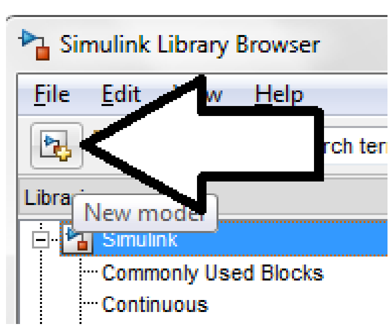
La siguiente imagen muestra una forma de configurar el modelado matemático en Simulink (se agregaron indicaciones en azul para que se entienda mejor). En este caso, x está siendo modelado (arbitrariamente) como una función de paso (bloque de Paso). La señal x entra en un bloque de ganancia, lo que multiplica su entrada por una constante especificada (m = 3 en este caso). Por lo tanto, la salida del bloque de ganancia es 3x.

A esto le sumamos la salida de un bloque constante, que se envía un valor de b = 4 aquí. De este modo, la señal de entrada en el **Scope** es y = 3x + 4.

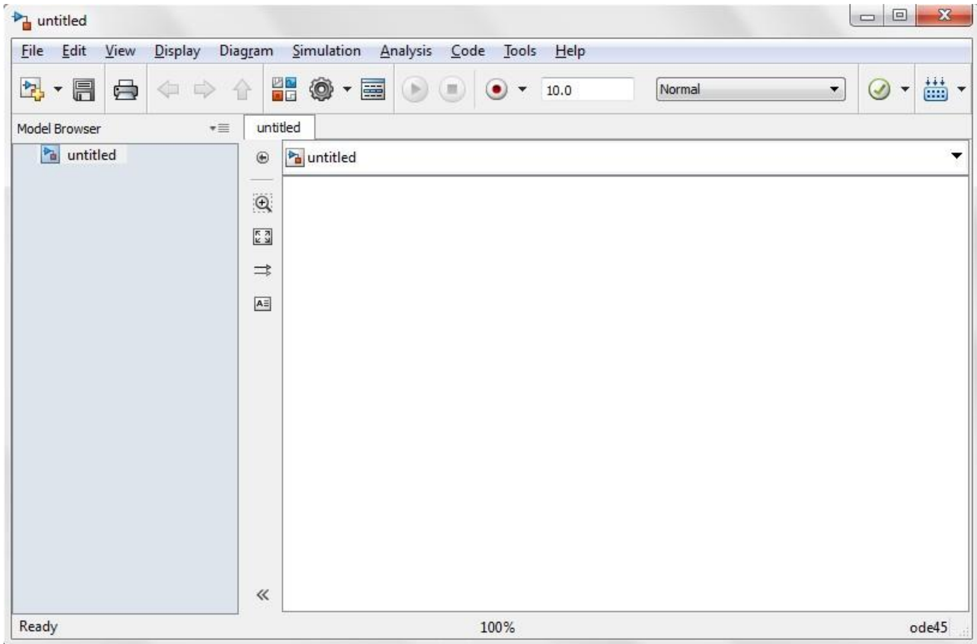


El bloque **Scope** proporciona una representación gráfica de su entrada como una función del tiempo (encargado de graficar inmediatamente), y es sólo una de muchas cosas que podríamos hacer con nuestra señal de salida **"y"**. Otras aplicaciones incluyen el almacenamiento en el disco o en el espacio de trabajo de MATLAB.

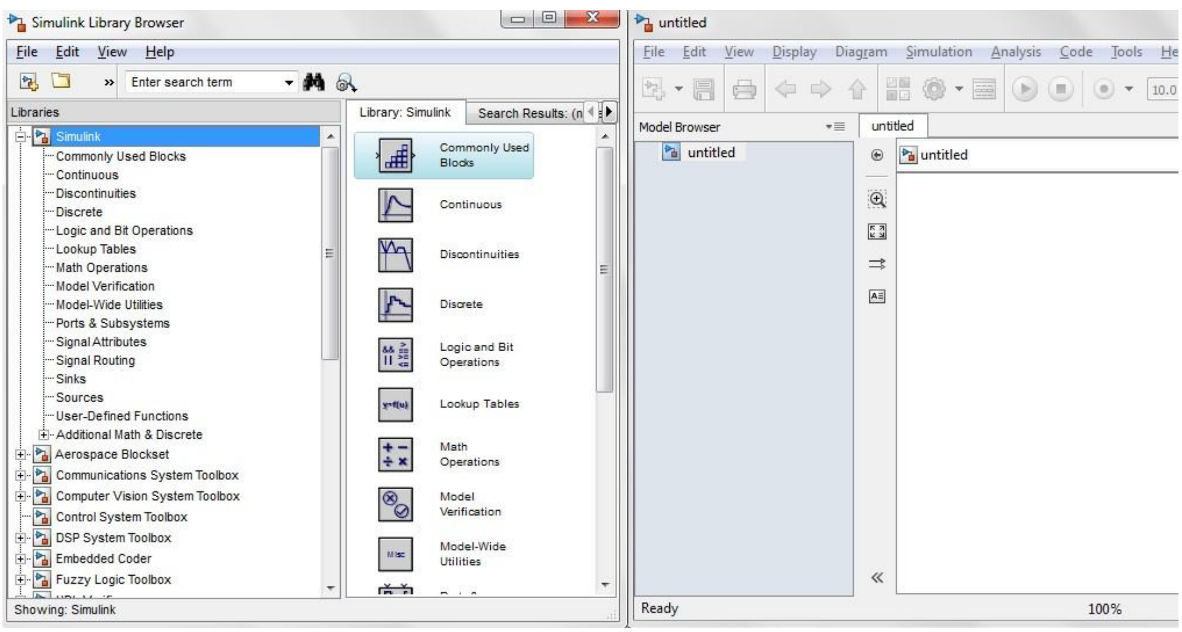
Para iniciar daremos clic en El icono mostrado que es new model.



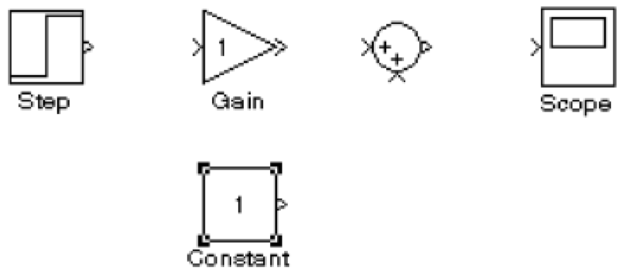
Se abrirá la siguiente ventana que es donde posicionaremos los bloques:



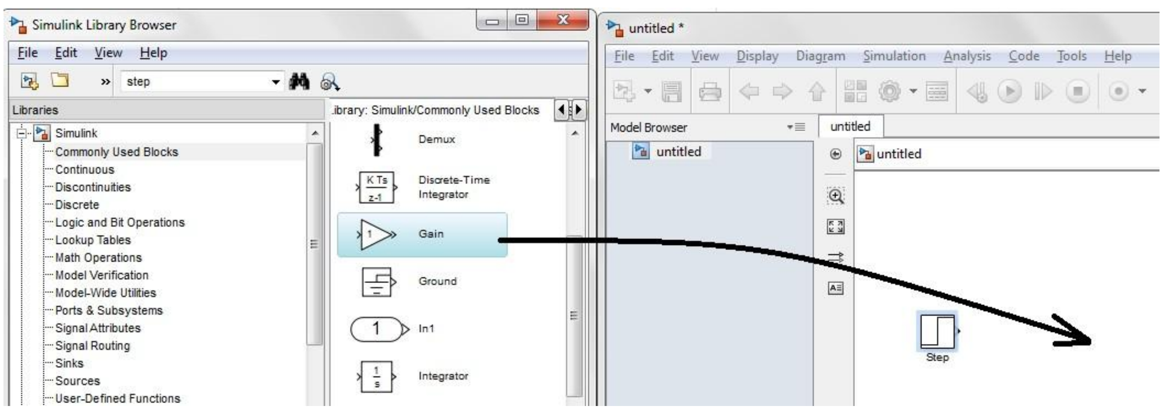
Para comenzar a trabajar con simulink te recomiendo tener las ventanas minimizadas dispuestas de esta manera para poder jalar los bloques desde la ventana de librería.



Dentro de las librerías debemos identificar los siguientes bloques:

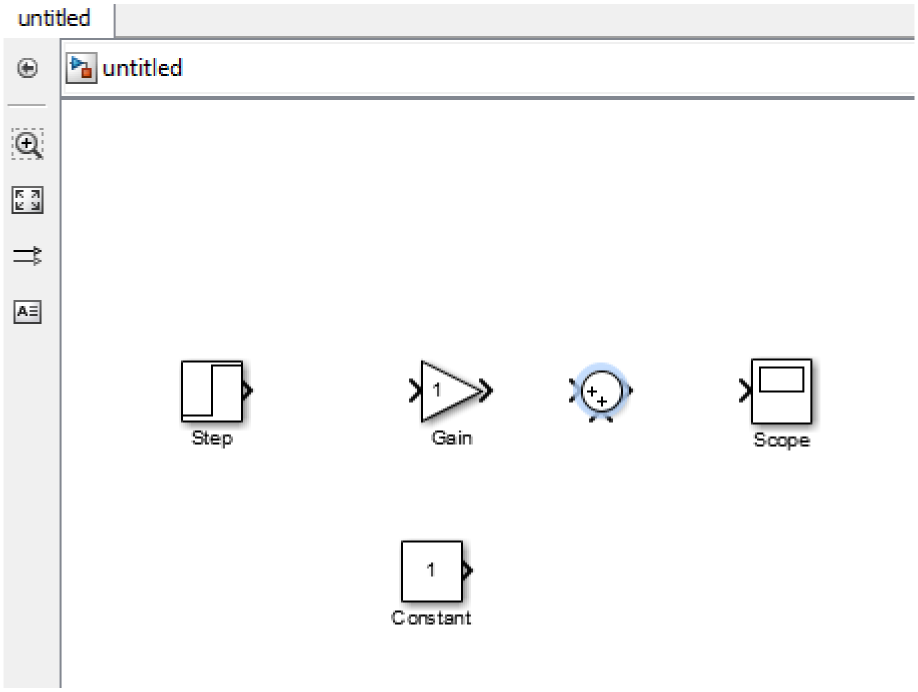


Para ello daremos doble clic sobre commonly used blocks, ahí encontraremos gain, sum, constant y scope. Simplemente tenemos que seleccionar el bloque y arrastrarlo a la ventana de simulink.



Después doble clic sobre **sources** y encontramos el bloque step.

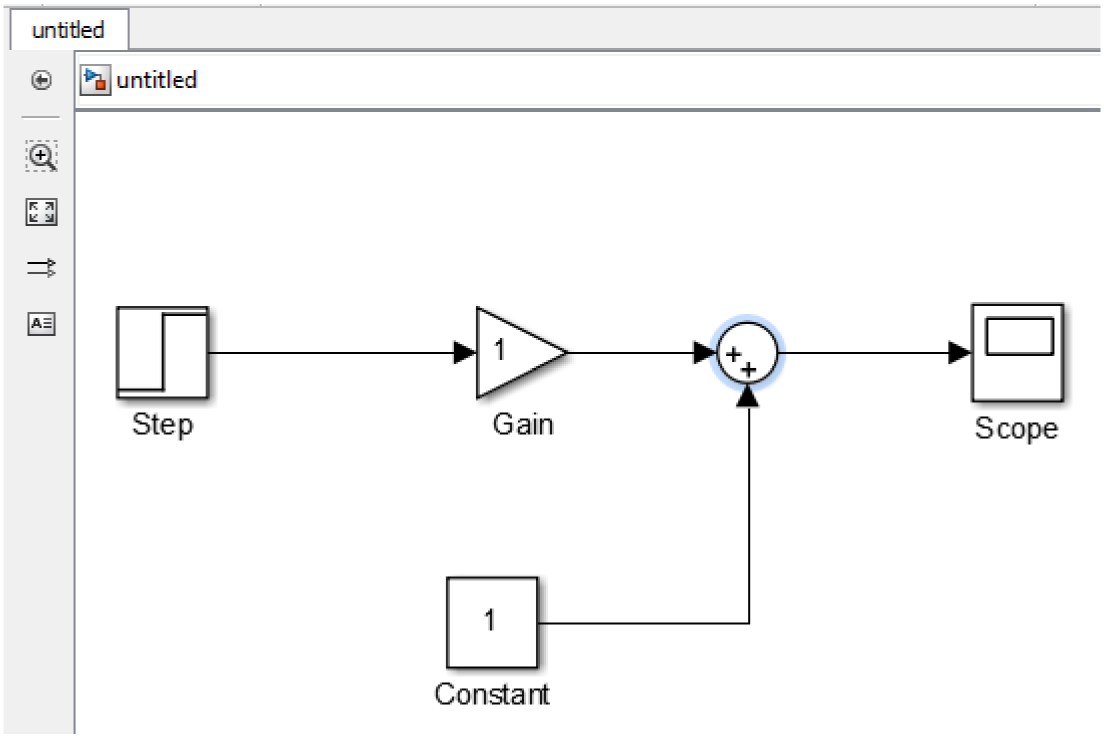
Así se verán los componentes dentro de la ventana.



Lo que necesitamos ahora es unirlos, lo que tenemos que hacer para ello es dar clic donde muestra la flecha de color azul.

Mantener sostenido el clic hasta llegar al punto que muestra la siguiente figura del bloque gain.

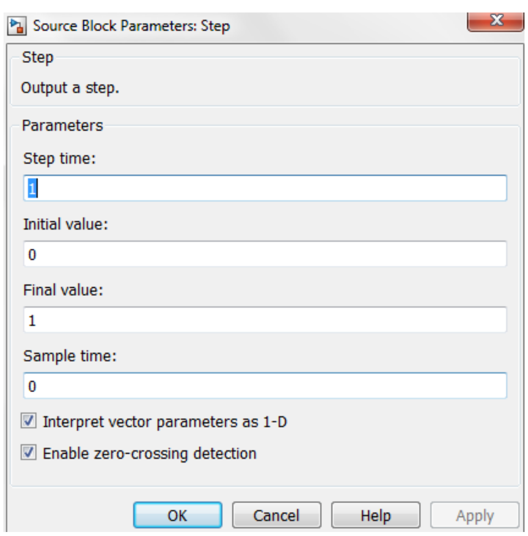
Esto lo haremos con todos los bloques, hasta que queden como en la siguiente figura.



Definir señales de entrada

Vamos a configurar la simulación. Primero definimos los parámetros en la señal de entrada, damos doble clic en el bloque Step.

Al dar doble clic sobre el bloque Step nos aparecerá la ventana de parámetros del bloque.



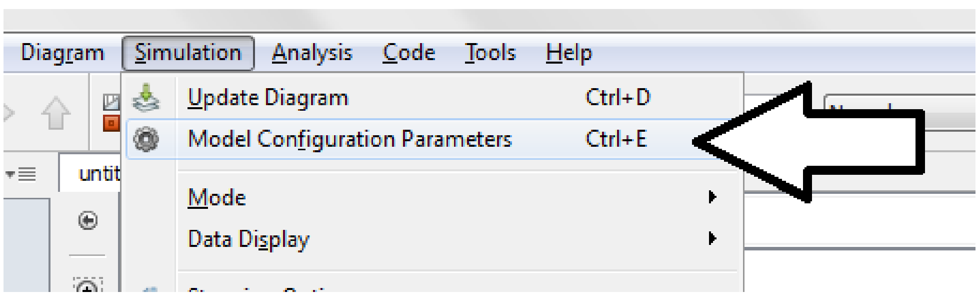
Los valores por defecto iniciaran a ***x*** en valor cero hasta que se produzca un incremento (en t =1).

El tiempo de muestra cero significa que la salida de los bloques será una función continua de tiempo. Si en lugar de eso tuviéramos que utilizar un valor positivo, la salida del bloque se definiría en integrales múltiples de la muestra de tiempo, pero sin definir el resto del tiempo.

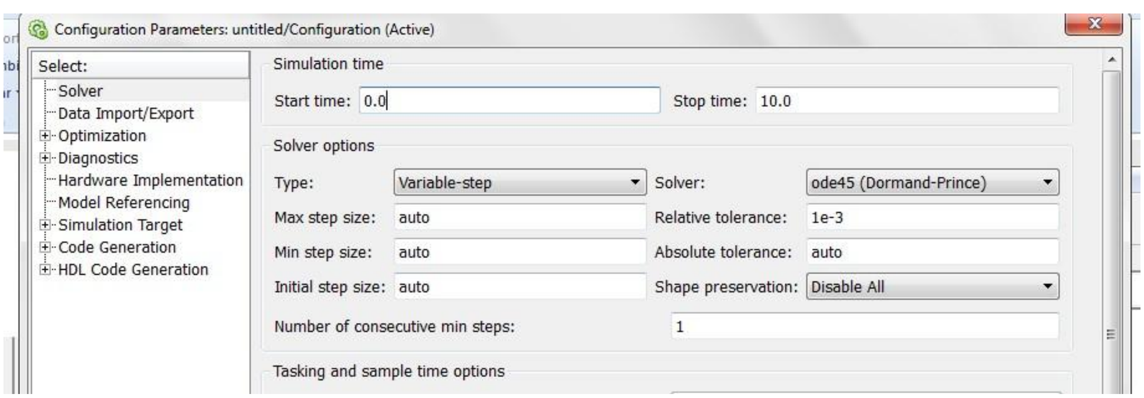
Deja todos los parámetros a sus valores por defecto para este ejemplo.

Parámetros de la simulación

Ahora definiremos los parámetros de la simulación, para esto nos dirigimos a la pestaña simulation y del menú colgante seleccionamos Model configuration parameters.



Se abre la siguiente ventana.



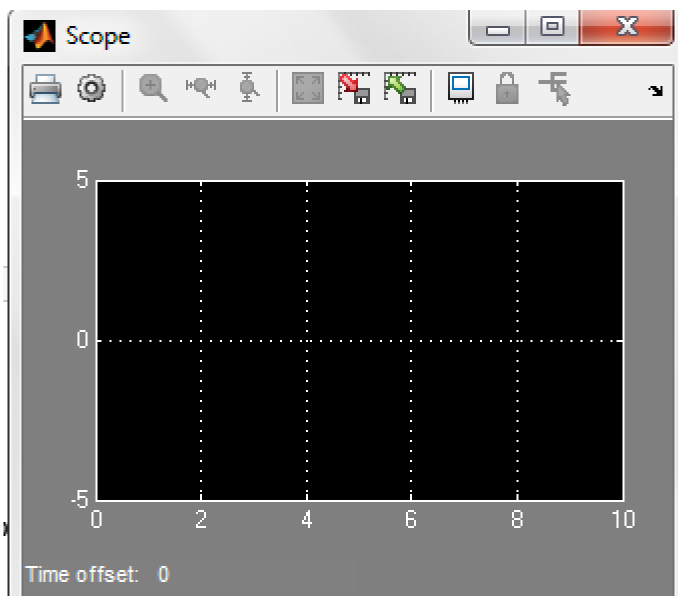
La ficha Solver le permite especificar la hora a la que comienza la simulación y donde se detiene (que depende de la escala de tiempo de su problema).

La otra área clave en la opción Solver es la sección solver options. El valor predeterminado, es un tamaño de **paso variable**, **ode45,** utilizando el método de Runge Kutta, que es el adecuado para esta simulación, y hay situaciones en las que un algoritmo de paso de tamaño fijo sería mejor, pero para casos generales aparece por defecto. Consulte la Ayuda en Simulink para más detalles.

Por último, las tolerancias para la solución numérica pueden tener un impacto. Si alguna vez sospecha que sus resultados son inexactos, pruebe a reducir la tolerancia en un orden de magnitud o dos. Si los resultados cambian significativamente, disminuir aún más. También asegúrese de que su definición del problema es razonable, y de que está utilizando la opción del solucionador apropiado. Usted debe verificar las entradas de la ficha de área de trabajo de E / S; **utilizaremos los valores predeterminados aquí.**

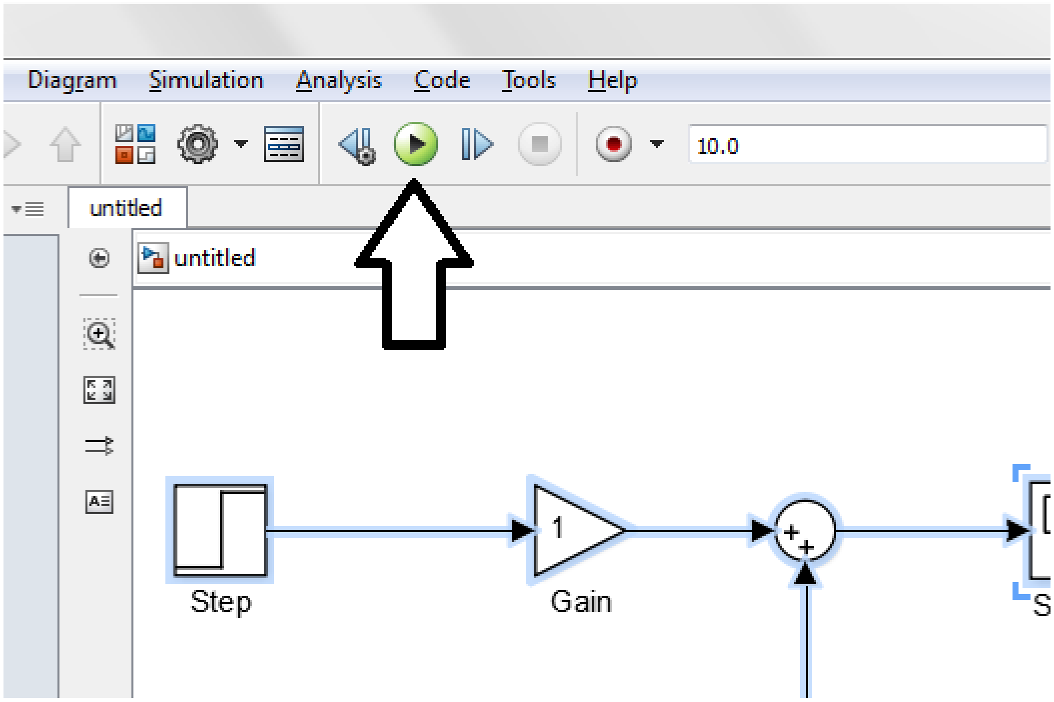
Cierre la ventana de parámetros de simulación.

Ahora daremos doble clic sobre scope para abrir la pantalla gráfica.

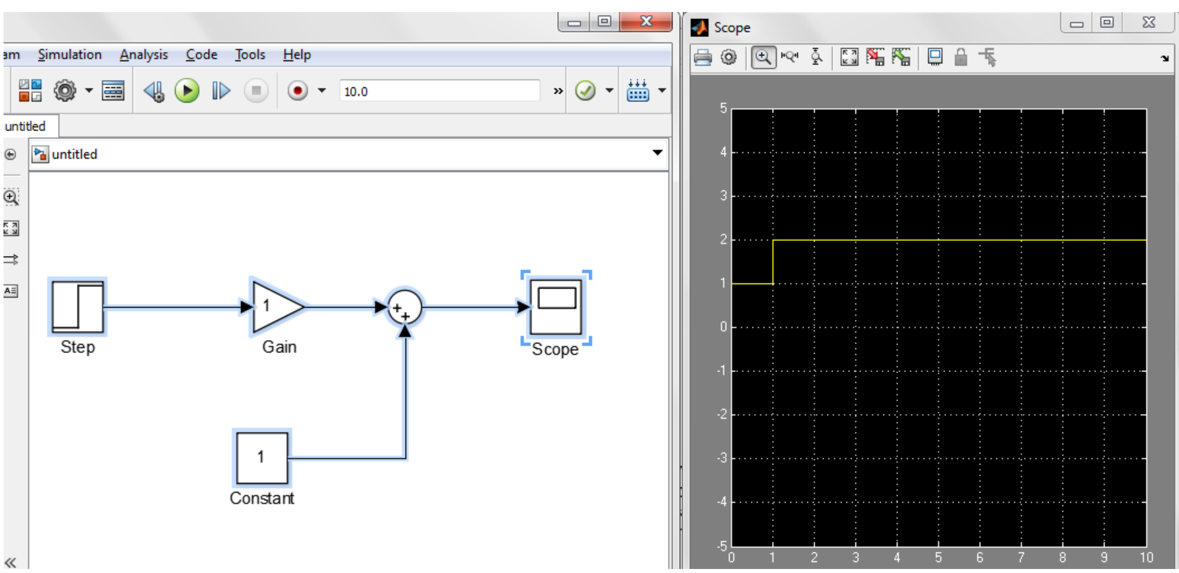


Corriendo la simulación

Ahora haga clic en el botón de inicio.



La simulación comienza (el botón de inicio cambia a un icono de pausa, y el botón de parada a su derecha se activa). El tiempo transcurrido aparece en el cuadro a la izquierda de la opción de simulación (parte inferior de la ventana del modelo) y la traza de la señal y aparece en el alcance. Debe mostrarse de la siguiente forma.



La grafica muestra el resultado final que es la línea amarilla.

De esta forma se demuestra la forma sencilla en que trabaja **simulink**.

Recursos para tu ruta de aprendizaje

Himmelblau, E. J. (1976). *Analisis y simulacion de procesos.* Reverte.
Videgaray, M. G. (1998(1a. reimpresión)). *Modelos y Simulación.* Ed. UNAM-FES Acatlán.
Herniter, M. E. (2001). *Programing in Matlab.* Thomson learning.
Rojas-Espinoza. Modelado y análisis de sistemas lineales. Costa Rica. Recuperado de: http://cursos.eie.ucr.ac.cr/claroline/backends/download.php?url=L0ZvbGxldG9BbmFsaXNpczIwMTQucGRm&cidReset=true&cidReq=IE409_001
Moore, H. (2007). Matlab para ingenieros. Utah, USA. Recuperado de: http://cursos.itcg.edu.mx/libros/matlab%20para%20ingenieros.pdf

Gonzalez, C. (2000). Simulador de una columna de destilacion binaria. México, D.F. Recuperado de: http://www.cesargm.com/Projects/Simulator%20of%20a%20binary%20distillation%20column.pdf

Sandoval, C. (2010). Laboratorio Virtual de Procesos. Cuernavaca, Morelos, México. Recuperado de: http://www.cenidet.edu.mx/subaca/web-mktro/submenus/investigacion/tesis/54%20Cinda%20Luz%20Sandoval%20Torres.pdf

En estos documentos encontraras información importante sobre simulación que te ayudaran a comprender mejor los conceptos sobre simulaciones de procesos biotecnológicos.

Las ecuaciones diferenciales en biología:
http://escalera.bio.ucm.es/recursos/claroline/claroline/backends/download.php?url=L1RlbWE0L3Q0Mi5wZGY%3D&cidReset=true&cidReq=MBB2010

Modelo lotka-volterra:

http://matema.ujaen.es/jnavas/web_modelos/labiologia/practica5.pdf

Videos:
Funciones en Matlab:
https://www.youtube.com/watch?v=aQInsnJU-qA

Operaciones y funciones en Matlab:
https://www.youtube.com/watch?v=ZUPlcfrmMuQ

En estas páginas encontraras información importante sobre los temas que debes de repasar y que pueden ser simulados con matlab o simulink.

Actividades

La elaboración de las actividades estará guiada por tu figura académica, mismoque te indicará, a través de la Planificación de actividades de la figura académica,la dinámica que tú y tus compañeros (as) llevarán a cabo, así como los envíos que tendrán que realizar.

Para el envío de tus trabajos usarás la siguiente nomenclatura: BSDP_U2_A1_XXYZ, donde BSDP corresponde a las siglas de la asignatura, U2 es la unidad de conocimiento, A1 es el número de actividad, el cual debes sustituir considerando la actividad que se realices, XX son las primeras letras de tu nombre, Y la primera letra de tu apellido paterno y Z la primera letra de tu apellido materno.

Autorreflexiones

Para la parte de **autorreflexiones** debes responder las *Preguntas de Autorreflexión* indicadas por tu figura académica y enviar tu archivo. Cabe recordar que esta actividad tiene una ponderación del 10% de tu evaluación.

Para el envío de tu autorreflexión utiliza la siguiente nomenclatura: BSDP_E2_ATR _XXYZ, donde BSDP corresponde a las siglas de la asignatura, E2 es la unidad de conocimiento, XX son las primeras letras de tu nombre, y la primera letra de tu apellido paterno y Z la primera letra de tu apellido materno

Cierre de la unidad

En esta unidad se aplicaron herramientas informáticas que nos ayudan a comprender mejor varios procesos; el utilizar la simulación nos ayuda a reducir el tiempo de análisis de datos y a obtener resultados de forma numérica y gráfica que son indispensables en cualquier experimento o resolución de problemas.

Para poder utilizar esas herramientas de manera eficaz debemos de tener bien fundamentadas las bases matemáticas e identificar qué tipo de modelo matemático simulará el experimento que estamos llevando a cabo.

Es muy recomendable dar un repaso a temas relacionados de biología, fisicoquímica, termodinámica, etc.

Fuentes de consulta

Himmelblau, E. J. (1976). *Analisis y simulacion de procesos.* Reverte.
Videgaray, M. G. (1998(1a. reimpresión)). *Modelos y Simulación.* Ed. UNAM-FES Acatlán.
Herniter, M. E. (2001). *Programing in Matlab.* Thomson learning.
Rojas-Espinoza. Modelado y análisis de sistemas lineales. Costa Rica.
Moore, H. (2007). Matlab para ingenieros. Utah, USA. Recuperado de: https://web.archive.org/web/20150509105704/http:/cursos.itcg.edu.mx/libros/matlab%20para%20ingenieros.pdf

González, C. (2000). Simulador de una columna de destilación binaria. México, D.F.
Sandoval, C. (2010). Laboratorio Virtual de Procesos. Cuernavaca, Morelos, México. Recuperado de:

https://web.archive.org/web/20131031174144/http:/www.cenidet.edu.mx/subaca/web-mktro/submenus/investigacion/tesis/54%20Cinda%20Luz%20Sandoval%20Torres.pdf