

Introducción

14/enero/19

Modelo de sistemas dinámicos

① Fuerzas que actúan en caída libre

Tarea: investigar porque no se cae un satélite

$$\vec{F} = m\vec{a}$$

$$a = \frac{\vec{F}}{m}$$

$$a = \frac{dv}{dt}$$

$$\frac{dv}{dt} = g$$

$$v = \frac{dy}{dt}$$

$$v \rightarrow \text{tiende a velocidad terminal}$$

$$\frac{dv}{dt} = g - \left(\frac{cd}{m}\right) v^2$$

$$v \rightarrow \text{tiende a velocidad terminal}$$

$$\frac{dv}{dt} = g - \left(\frac{cd}{m}\right) v^2$$

$$\frac{dv}{dt} = \lim_{h \rightarrow 0} \frac{v(t+h) - v(t)}{h}$$

$$v(t+h) = v(t) + h \underbrace{\left(g - \frac{cd}{m} v^2(t)\right)}_{\text{pendiente}}$$

$$v(i+1) = v(i) + h \left(g - \frac{cd}{m} v^2(i)\right)$$

$$v(i+1) = v(i) + h \left(g - \frac{cd}{m} v^2(i)\right)$$

Taylor: approxima funciones con polinomios.

$$v(t+h) = v(t) + \frac{h * v'(t)}{1!} + \frac{h^2 * v''(t)}{2!} + \dots$$

Matlab

Script: van funciones y lo que quiero ejecutar

Necesitamos vector de tiempo y de velocidad, cuanto vale G y cuanto vale la masa.

Llamamos al script g si queremos correrlo ponemos RUN o escribimos el nombre del método

Si queremos que al correrlo no se impriman valores, les ponemos ;

Definir vectores: se pueden escribir con llaves o escribiendo valor inicial : incremento : valor final

Size: para saber el tamaño del vector usamos size

Los índices en Matlab empiezan en 1

FOR: declaras variable y el valor. El indice de un for es un vector

Ejemplo: for i = 1:1:10

$$v(i) = t(1)$$

end

se debe asignar el tamaño del vector al principio

Tamaño: Para asignar el tamaño del vector $v = \text{zeros}(1, 10)$

con un tamaño de vector fijo ya no hay problema que se este cambiando.

Ahora se puede poner la fórmula completa del problema dentro del for que va hasta tf.

Length(v): calcula el tamaño de v.

Gráfico: para graficar vectores usamos plot: $\text{plot}(x, y)$

Tarea sugerida: revisale bien el código

$$\vec{F} = m\vec{g} - Cd v^2$$

$$\frac{dv}{dt} = a = \frac{F}{m} = g - \frac{Cd}{m} v^2 \Rightarrow v(t)$$

$$v = \frac{dy}{dt}$$

$$y(t) = y(i+1) + v(i)^* h.$$

f
 $\frac{dy}{dt}$

Para crear funciones se puede crear una nueva función o dentro del mismo script:

```
function f = ...
end
```

Función interna para calcular la f .

Primer problema: no reconoce las variables afuera, se tienen que declarar adentro.

Declaradas adentro no es necesario que estén afuera. El parámetro que recibe se pone en lugar de $v(i)$ y el resultado es el nombre de la función = resultado.

Para poner **bbels** en matlab se escribe `xlabel('');`
`y label('');`

Para **titub** se escribe `title('');`

Para que aparezcan cuadriculas se escribe `grid on;`

Para calcular la velocidad terminal despejamos $v \Rightarrow \sqrt{m * g / Cd}$.

La función **plot** de matlab sob grafica vectores.

Para hacer vectores de 1, se escribe `ones(tamaño);`

`hold on` sirve para mantener gráficas a la vez, se termina con `hold off`.

`plot([0, tf], ...)`: 0 y tf marcan los extremos del vector y los une

con `-b` vuelve la gráfica azul (blue) y punteada `--`

Con **global** se cambian las variables declaradas varios veces

`= global g } global g después.`
`g = 9.81; }`

Para calcular de manera exacta alguna función escribimos:

`syms` parámetros;

`eqn` = operación a realizar $\Rightarrow \text{diff}(v, t)$ derivada de v en función de t

Al trabajar con `syms` nos habla de variables simbólicas.

`dsolve`: resuelve todo.

`subs`: objeto simbólico.

Método de Montecarlo.

$$x^2 + y^2 = 1$$

$$y = \sqrt{1-x^2}, \quad x \geq 0$$

Para poner "dados" en matlab generamos vectores alatorios

Tarea: calcular intervalo de confianza al 95% (proporción)

$$n = 1000$$

$$x = \left[\begin{smallmatrix} \text{rand} \\ \vdots \end{smallmatrix} \right] \quad \left\{ \begin{array}{l} \text{genera valores} \\ \text{aleatorios} \end{array} \right.$$

condición:
 $x(i)^2 + y(i)^2 \leq 1$

Al poner en punto antes de una operación a un vector, se realiza la operación a CADA ELEMENTO de la matriz.

21/enero/19

Problema en Matlab: windows 10 necesitaba parche, ya funciona.

h va a ser más pequeña para disminuir errores.

Hay que convertir m/s a km/h.

Con paracaídas aumenta el cd, aumenta cd y por lo tanto disminuye la velocidad terminal.

Legenda: en orden en el que hicimos plot se escriben las legendas de la gráfica.

eje y o velocidad Terminal * 1.2 o tf.

Axis: define límites para los ejes x y y.

Es importante que no se deje al usuario sin información para analizar la gráfica.

X variable aleatoria. $f(x) = \frac{1}{b-a}$ si $a \leq x \leq b$

$g(x) = \sqrt{1-x^2}$ $E[g(x)]$ es el promedio $\Rightarrow \int_a^b g(x) f(x) dx$

rand: la función rand en Matlab devuelve números aleatorios normalmente distribuidos.

$$= \frac{1}{b-a} \int_a^b g(x) dx = \int_0^1 g(x) dx = \frac{\pi}{4}$$

mean: con esta función sacamos el promedio.

primero se declara la variable aleatoria con rand y después se saca la función con cada una de las x y al final se saca la esperanza de y y se multiplica por 4.

Tarea: buscar una serie para sacar pi. (π)

Ejercicio 3: satélite geoestacionario / Luna / + Sol.

*Ceros: $R = 6371 \text{ km} = 6.371 \times 10^6 \text{ m}$ r_{th}

$$\vec{a} = \frac{\vec{F}}{m} \quad \text{Atracción gravitacional.}$$

Proyecto: sistema de 3 cuerpos: tierra, sol y luna.

$$\gamma \begin{bmatrix} v_{t+1} \\ r_{t+1} \end{bmatrix} = \begin{bmatrix} v_t \\ r_t \end{bmatrix} + \begin{bmatrix} \frac{du}{dt} * h \\ \frac{dr}{dt} * h \end{bmatrix}$$

Ahora queremos saber la posición de n.

Redondeo y truncamiento.

La computadora usa standard para doble precisión y números reales.

Doble precisión: utilizamos más bits para representar los números deseados.

Redondeo: -173 en binario? Usaremos un bit para indicar si es negativo

$\overbrace{10000.10101101}^{16 \text{ bits}}$

esto sirve para representar enteros en cierto rango.

$$2^{15} - 1 = 2^{14} + 2^{13} + 2^{12} + \dots + 2^2 + 2^1 + 2^0$$

en matlab si tenemos un número más grande se escribe Inf

000031416
entero
 3.1416×10^0 e
fracción

- Punto flotante
- Normalización de números.

Ejemplo: Base-10 computer-5-digit word size
 $5.d_1d_2 \times 10^{d_0}$

s	d ₁	d ₂	d ₀	d _b
9	9	9		

11 bits	52 bits
signo	Exponente Montissa.

Como la f es finita, tenemos precisión finita

Como la e es finita tenemos un rango acotado

Los puntos flotantes tienen espacio discreto y un máximo y mínimo.

Errores: redondeo y truncamiento.

Serie de pi: $4 * \sum_{i=0}^{\infty} \frac{(-1)^i}{2i+1} = 4 * \left(\underbrace{1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \frac{1}{9} \dots}_{n=10} \right) \rightarrow \text{fors}$

Mientras el error sea más grande de lo que quiero, sigo dando vueltas. \Rightarrow 2 problemas: ¿Cómo calculo mi error? ¿Cuál quiero que sea mi error máximo?

while (error > errorMax)
 $\left| \frac{\pi_i - p}{\pi_i} \right| \rightarrow \text{error relativo}$ 0.00001
 1416

abs: función que permite sacar el valor absoluto de un valor.

El número de iteraciones nos dice que al hacer menos que las que hace el programa, tendríamos un error más grande. \rightarrow truncamiento.

Para no usar pi en Matlab, usamos $\left| \frac{\text{actual} - \text{previo}}{\text{actual}} \right|$

Tarea: leer todos los códigos que hemos hecho

Redondeo:

realMax: valor más grande que Matlab puede representar

realMin: valor más pequeño que se tiene en Matlab normalizado (standarizado).

$\frac{1}{\text{realMax}}$ da un número más pequeño que realmin pero no está normalizado

NaN: not a number = no es un número para matlab.

format long: te muestra más decimales en números

format hex: cambia números a formato hexadecimal

$e = \frac{000 \dots 05}{- \dots \mp \dots} \cdot 111 \dots 11s$

2^3 0 - $\frac{7}{3}$
 $0 - 2^3 - 1$

$2^{-1} = 2047$ Inf f=0
 NaN f>0

0 { subnormal f>0
 cero f=0

Tarea (viernes): 1) ¿Cuántos números normalizados hay?

2) ¿Qué número es: 3f 847ae 147ae 147b?

$$3) f(x) = -0.1x^4 - 0.15x^5 - 0.5x^2 - 0.28x + 1.2$$

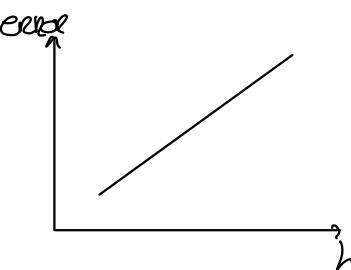
$$x = 0.5$$

$$f'(x) \approx \frac{f(x+h) - f(x-h)}{2h}$$

$$h=1 \quad h=\frac{h}{2}$$

n veces

$$\begin{cases} n=6 \\ 7 \\ 8 \\ 11 \end{cases}$$



código en Matlab

Número normalizado: los que tienen un 1.

real max: 1.7977e308

real min: 2.2251e-308

$$0 \leq e \leq 2^{32}-1 \leq 2047$$

$$er = e - (2^{32}-1) = e - 1023$$

$$1 + \left(\frac{1}{2} + \frac{1}{4} + \frac{1}{8} \right) \Rightarrow \left(\frac{7}{8} \right) = \left(\frac{2^3 - 1}{2^3} \right) = 1 - \frac{1}{2^3}$$

$$+ \left[\boxed{1 + (1 - 2^{-52})} \times 2^{\boxed{1023}} \right] \text{ número real max.}$$

$$+ \left[\boxed{1 + (2^{-52})} \times 2^{\boxed{-1022}} \right] \text{ número real min.}$$

$1 - 2^{-52} \neq 1 \quad g \neq 0$

Estas cifras dependen de el número de bits.

signo exponent	matissa
11 bits	52 bits

$$\text{hexadecimal: } \frac{1}{\text{realmax}} = 0004000000 \times 10^{-12}$$

En formato long = $\frac{1}{\text{realmax}}$ da un número más pequeño que el realmin.

$$\left[\boxed{(0 + 2^{-52})} \times 2^{-1022} \right] \quad \left[\boxed{(0 + 2^{-52})} \times 2^{-1022} \right]$$

* Precisión de la máquina: existen dígitos que no puedo representar.

Eps: epsilon de la máquina \rightarrow máximo error que podemos tener

Programa que $|t-x|$ no sea mayor a ϵ .

$x = 1$
while $(1 + \frac{x}{2} > 1)$



se pueden representar números más pequeños después de 1.

$$x = \frac{x}{2}$$

El epsilon es el máximo error relativo.

¿Cuántos números represento de manera exacta? = 1^{era} pregunta de la tarea.

$$1.000\ 000\ 000\ 000 \neq 1.$$

isequal: dice si es igual una cosa a otra: isequal(C suma, 1);

$\therefore 0.1$ no se puede representar de manera exacta.

vector 0:0.1:1: vector que va de 0 a 1 aumentando en 0.1

$$1 - 0.9 - 0.1 \neq 0 \text{ porque } 0.1 \text{ no se representa exactamente.}$$

\therefore error casi nunca será 0.

$1e^{-16}$ = número tan chiquito que Matlab lo descarta.

$$d = (2^{53} + 1) - 2^{53} = 0 \text{ porque el } 1 \text{ es un número muy pequeño en relación con } 2^{53} \text{ entonces no lo toma en cuenta.}$$

El orden de como van las operaciones en matlab si importa.

Ejempb: $1e^{-16} + 1 - 1e^{-16} = 1.0000$
 $1e^{-16} - 1e^{-16} + 1 = 1.$

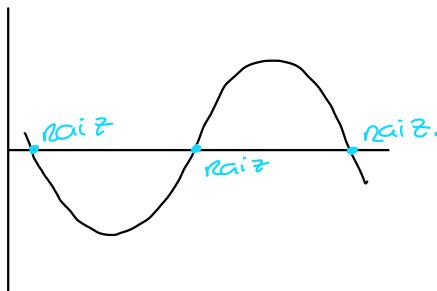
$$\sin(\pi) = 1.224646 \neq 0$$

Ejempb: $405.900000\dots 0 = \underline{\hspace{1cm}}0\underline{\hspace{1cm}}1000000\underline{\hspace{1cm}}0101\underline{\hspace{1cm}}100100000$

$$\begin{array}{r} 100\ 0000\ 0101 \\ \hline 1029 \\ - 1023 \\ \hline 6 \end{array} \quad \frac{1001}{2^{\frac{1}{2^1}} 2^{\frac{1}{2^2}}} \left(1 + \frac{1}{2^1} + \frac{1}{2^2} \right) \times 2^6 = 100$$

Raíces y optimización

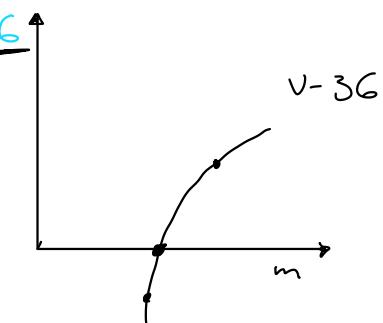
Raíces: Métodos cerrados: bisección, interpolación lineal
 Métodos abiertos: Newton-Raphson, secante.



Problema 1: caída libre: $v < 36 \text{ m/s}$ $t = 45$ ¿masa máxima?

Problema 2: ¿sueldo bruto? sueldo neto: \$4,500.

Problema 1: $v = \sqrt{\frac{g \cdot m}{C_d}} \tanh\left(\sqrt{\frac{g \cdot C_d}{m}} \cdot t\right)$



Bisección: en Matlab escribir una función con una a, una b.

Crear una nueva función en matlab: `function [x,i] = bisección(f,a,b)`
 error relativo entre valbr actual y anterior.

El punto después de operación significa que va a operar en cada elemento de la matriz.

Tarea: Calcular el sueldo bruto para tener saldo neto de \$45,000
 Necesitamos una instrucción en Matlab para leer el archivo
 - Buscar operaciones lógicas para determinar cosas. ($=1$, $=0$).
 La función x le pasamos la tabla, la x y nos da número positivo o negativo. Bisección nos va a dar el sueldo bruto.

06 / febrero / 19

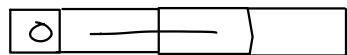
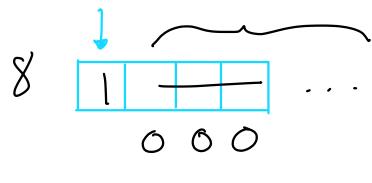
		$n=2$
		2^n
0	0	
0	1	
1	0	
1	1	
		2^{n-1}

$$n=64$$

$$\text{posibles} = 2^64 = 2^{2^4} = 2^{16} = 2^{52}$$

$$2(2^{16}-2)2^{52} = 2^{64} - 2^{54} = \text{números normalizados.}$$

hex 2 dec (sc1): convierte a decimal un numero hexadecimal en matlab.



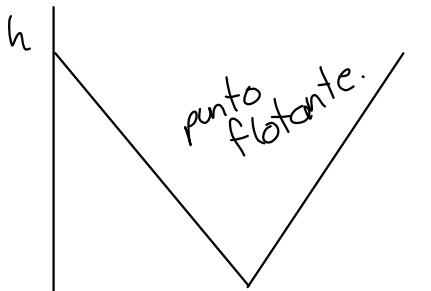
$$\left[\frac{1}{16} \quad \frac{1}{16^2} \quad \frac{1}{16^3} \right]$$

$$c = 1016 \begin{cases} 00 \rightsquigarrow \\ 11 \rightsquigarrow \end{cases}$$

$$re = e - (2^{10} - 1)^{1023}$$

no normalizado: signo * 2^{e-1023} * (1+f)

Ejercicio: $i = 0 \quad 1 \quad 2$
 $n = 1 \quad \frac{1}{10} \quad \frac{1}{10^2} \quad \frac{1}{10^3}$



Tarea de sueldos: sueldo Neto = f(sueldo bruto, tabla)

3-5 líneas.
probarlo con bisección y reescribirlo con 3-5 líneas.

11 / febrero / 19

Algoritmo de bisección.

$$\begin{matrix} a & b \\ xl & xm \end{matrix}$$

$$\begin{aligned} L_0 &= b-a \\ L_1 &= \frac{L_0}{2} \end{aligned}$$

$$x = \frac{b+a}{2}$$

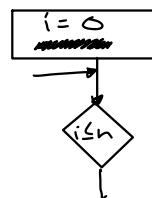
$$L_2 = \frac{L_1}{2} = \frac{L_0}{2^2} = \frac{L_0}{4}$$

$$L_n = \frac{L_0}{2^n}$$

Matlab: cambiamos $f(a)$ por una variable fa que tambien incó adop-
tando los nuevos valores de $f(a)$.

Problema: x es el penúltimo, tenemos que darle el valor nuevo al final.

while en Java:

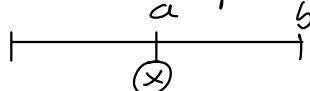


Se cambia iniciando el valor de x antes del while y el valor de $f(x)$. En la condición del while también se verifica que $f(x) \neq 0$.

Se crea una nueva variable que se llame intervalo = num2hex (abs(b-a)); $fprintf(\times .2i \times .20.16f \times .20.16f \times .5 \times , i, x, fx, intervalo); \rightarrow print con formato.$ Le entran 4 parámetros, el morado indica el formato. En cada iteración va cambiando el intervalo a la mitad. Va cambiando un bit.

Se sale del ciclo porque llegó a 0.

Se fija un max como $\log_2((b-a)/\text{eps}(\text{númMin})) \rightarrow$ para asegurarnos que va a dar las vueltas necesarias.



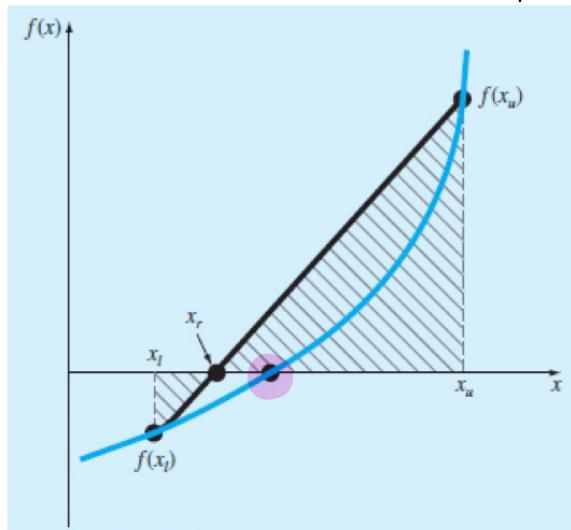
Rapidez de Convergencia

$$\lim_{n \rightarrow \infty} \frac{|p_{n+1} - p|}{|p_n - p|^\alpha} = \lambda$$

Orden de convergencia de biseción = 1, por lo tanto el algoritmo tiene convergencia lineal.

Interpolación lineal: (falsa posición)

Encontrar el punto suponiendo que sustituyo mi función por una recta. Lo mismo que biseción pero con otra fórmula en lugar de los extremos entre 2. Una vez encontrado el punto se hace lo mismo que en biseción: decidir con qué punto me quedo.



$$x_{n+1} = f(b, a).$$

Nos interesa ver donde cae x en el cje.

$$x_r = f(b, a).$$

Se calcula la función lineal con la función de la recta con 2 puntos.

$$y = y_1 + \frac{y_2 - y_1}{x_2 - x_1} (x - x_1)$$

En el dibujo: $f(x_u) + \frac{f(x_l) - f(x_u)}{x_1 - x_u} (x - x_u)$

$$\therefore x_r = f(b, a) = b - \frac{f(b)(b-a)}{f(b) - f(a)}$$

Método mejor porque encuentra la solución en menos iteraciones.

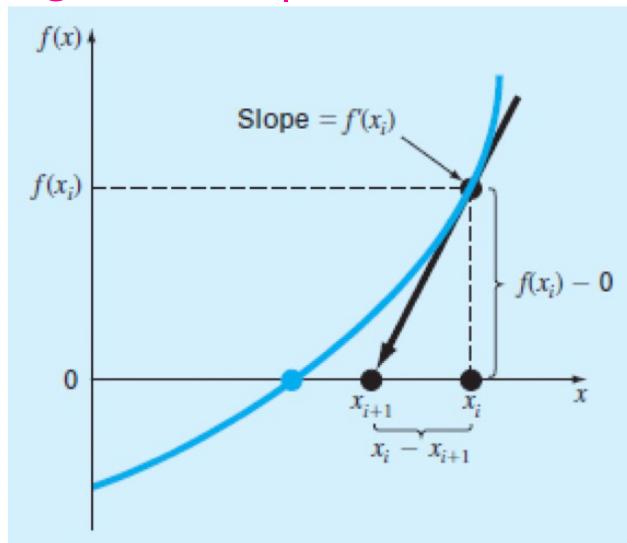
Si hacemos más grande el intervalo $(10, 200)$ tarda más interpolación lineal que bisección. \therefore Hay casos que interpolación tarda más.

Ejercicio Salarios:

$$a = C(1 + \text{datos}(1, \text{end}) / 100) * \text{neto}$$
$$b = (\text{end} - 1, \text{end}) / 100$$

En este ejercicio bisección hace 49 iteraciones, mientras que interpolación lineal solo hace 7, lo calcula a la primera.

Newton - Raphson



Es cuadrática, es el método más utilizado.

La función a calcular ahora es la derivada:

$$y - y_0 = m(x - x_0)$$
$$y - f(x_i) = f'(x_i)(x - x_i)$$

Si se sustituye con 0:

$$0 - f(x_i) = f'(x_i)(x_{i+1} - x_i)$$

$$\frac{-f(x_i)}{f'(x_i)} + x_i = x_{i+1}$$

Otra manera de calcularlo es:

$$f'(x_i) = \frac{f(x_i)}{x_i - x_{i+1}}$$

Tarea: programar Newton Raphson de la manera más bonita posible.

13 / febrero / 19

Condición en while de $f(x) \approx 0$ no es necesario porque hace una iteración de más.

Bisección: algoritmo lento.

66 es el máximo de iteraciones.

Hacer simbólica la función y luego derivarla es la mejor manera de hacerlo.

Actual = 7 y previo = 0 es para que entre al while pero se ve feo.
↳ forzar para que entre una primera vez.

do

while condición } Con el do while siempre entra y hasta el final checa la condición.

Do while en Matlab: cond = true;
 while cond
 {
 cond = error > cota && i < Max && f(x) ~= 0;

$$f(x) = x^2 - M$$

$$x_{n+1} = x_n - \frac{x_n^2 - M}{2x_n}$$

$$= \frac{1}{2} \left(x_n + \frac{M}{x_n} \right)$$

→ Raíz cuadrada de M.

Convergencia de Bisectión es lineal

Convergencia de Newton Raphson es cuadrática.
 Es la serie de Taylor.

$$f(x_r) = f(x_n) + f'(x_n)(x_r - x_n) + \frac{f''(\xi)}{2} (x_r - x_n)^2 = 0$$

$$f(x_n) = f'(x)(x_n - x_{n+1})$$

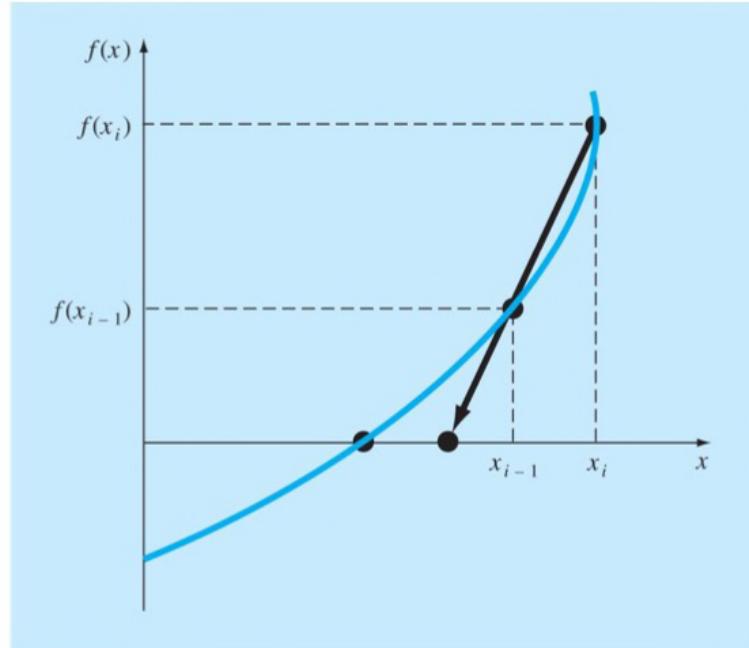
$$x_{n+1} = -\frac{1}{2} \frac{f''(\xi)}{f'(x_n)} x_n^2$$

$$x_{n+1} = O(x_n^2)$$

Complicaciones: Tiene que ser derivable. El valor inicial de la x tiene que estar cerca de la raíz. La derivada lo que nos da es la pendiente de la recta.

Método de la Secante.

La derivada se approxima con una diferencia distinta finita hacia atrás.



$$s_n = \frac{f(x_n) - f(x_{n-1})}{x_n - x_{n-1}}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{s_n}$$

Tarea: Problema de comisión utilizando los métodos vistos en clase.
 Se tiene que despejar h.

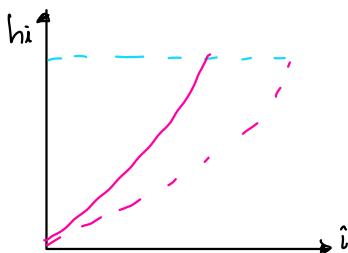
$$V(h) = 30.$$

$V(h) - 30 = 0 \rightarrow$ queremos encontrar la raíz de esto.

Calcular y graficar el resultado

se deben de ocupar los métodos que vimos.

Sabemos que Bisectión es el más lento y sabemos cuánto se tarda en resolverlo.



Se tienen que hacer el número de iteraciones posibles, no solo 3.

Raíces y optimización.

Supongamos la función unimodal en el rango que nosotros le damos.

Óptimo: el máximo o mínimo de acuerdo a un criterio.

Existe también el método de interpolación parabólica. Se parece a interpolación (lineal).

Fibonacci en Matlab: con un for. Sacarlos consecutivos. Fibonacci razón dorada.

Proporción áurea (golden ratio) razón dorada. (Se parece a biseción)

Proporción Áurea:

$$\frac{l_1 + l_2}{l_1} = \frac{l_1}{l_2} = \varphi$$
$$1 + \frac{l_2}{l_1} = \frac{l_1}{l_2} = 1 + \frac{1}{\varphi} = \varphi = \varphi + 1 = \varphi^2$$

$$\frac{1+\sqrt{5}}{2}$$

$$\varphi^2 - \varphi - 1 = \varphi$$



Se tienen 2 raíces pero se toma la positiva porque son distanciadas.

$$\therefore \text{en programa dc Matlab: } \varphi = \frac{1+\sqrt{5}}{2}$$

Algoritmo va a calcular 2 puntos internos para volver a usar el algoritmo. Dentro de esos 2 puntos tiene que estar la raíz. Intervalos se irán reduciendo siempre en la misma proporción. Se escoge entre 2 lados dependiendo en dónde esté la raíz.

$$x_1 = x_1 + d$$

$$x_2 = x_2 - d$$

$$d = (\varphi - 1)(x_u - x_l)$$

g intervalo actual.

$$\varphi = \varphi - 1.$$

$$= 1.618 - 1 = 0.618.$$

Para decidir que intervalo tomar se evalúa la función.

$$a < x_2 < x_1 < b$$

$f(a) f(x_2) f(x_1) f(b)$ Evaluas los extremos y los x .

$$f(x)$$

Eliminate

Minimum



$$a = x_2$$

$$x_1 = x_2$$

porque tiene la misma proporción.

Proporción Áurea.

$$\frac{l_1}{l_1 + l_2} = \frac{l_2}{l_1}$$

$$\frac{l_1 + l_2}{l_1} = \frac{l_1}{l_2} = \phi$$

$$\phi = \frac{1+\sqrt{5}}{2}$$

$$g = \phi - 1$$

minimizar. $\frac{a+b}{2}$ a b $a+b$

$$[x_1, i] = \text{optAurea}(f, a, b)$$

$$x = \frac{a+b}{2}$$

$$f(x_1) < f(x_2)$$

Queremos dos puntos mínimos

x. Se convierte en x_2 y x_2 en x_u . La idea es no repetir los cálculos.

$$x_1 = x_1 + d$$

$$x_2 = x_u - d$$

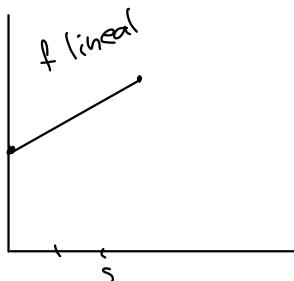
$$d = (\phi - 1)(x_u - x_1)$$

$$d = g(x_u - x_1)$$

Se redifinen f_1 y f_2 en cada caso así como x_1 y x_2 para no repetir los cálculos dentro del while en Matlab.

$d = g * d;$ va partiendo el intervalo en la misma proporción. Siempre se va acercando.

Problema: Parabólico. $g = 9.81; v\phi = 55; m = 80; c = 15; z\phi = 100$
 $z = @C(t) = -Gz\phi + m/c * (v\phi + m*g/c) * C1 - \exp(-c/m*t) - m*g/c * t;$



x	f(x)
0	-
5	-
10	-
15	-

$$f(3) = ?$$

Interpolación con polinomios

Parabólica: grado 2.

Lineal: grado 1.

Polinomios: Son continuos y derivables. Son amigables

Encontrar parábola que pase por 3 puntos. Polinomio es único. Parábola tiene que valer lo mismo en los 3 puntos.

Problema se reduce a tener 3 puntos, encontrar la parábola que pasa por ahí. Hay 2 métodos muy conocidos para interpolar.

Lo que queremos es:

$$x_1 \quad f(x_1) = p(x_1)$$

$$x_2 \quad f(x_2) = p(x_2)$$

$$x_3 \quad f(x_3) = p(x_3)$$

$$p(x) = f(x_1) + \dots + f(x_3)$$

$$= 0 + f(x_2) + 0$$

$$= 0 + 0 + f(x_3)$$

$$p(x_1) = \underbrace{\frac{(x-x_2)(x-x_3)}{(x_1-x_2)(x_1-x_3)} f(x_1) +}_{\text{grado } 2} f(x_2) + f(x_3)$$

Al agregarle otro $f(x)$ se convierte en grado 2.

$$p'(x) = \underline{\quad} = 0 \text{ y despejo } x = \underline{\quad}$$

Fórmula desarrollada. Intentarlo a mano o buscáelo en internet.

Lagrange: $p(x_k) = y_k, k=1, \dots, n$

$$p(x) = \sum_k \left(\prod_{j \neq k} \frac{x - x_j}{x_k - x_j} \right) y_k$$

$$\begin{array}{ll} x_1 & f(x_1) \\ x_2 & f(x_2) \\ x_3 & f(x_3) \end{array} \quad \left. \begin{array}{l} \boxed{} \\ \boxed{} \end{array} \right\} \text{diferencias divididas.}$$

$$x = f(r, s, t) \Rightarrow p(x) = \boxed{}, \quad p'(x) = \underline{\quad} = 0$$

$$\underline{\quad} = \text{opt} | PSC(f, r, s, t)$$

$$f_r = f(r)$$

$$f_s = f(s)$$

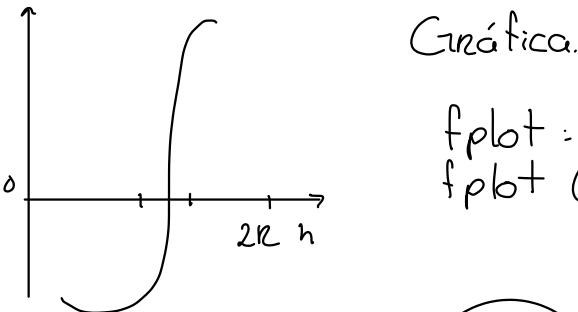
$$f_t = f(t)$$

while = (as recorre y sustituye
 $s=r$, $f_s = f_r$; calcula $f(r)$)

Newton: buscáelo de tarea. Buscar polinomio Newton, derivarlo, igualar a 0 y sacar x .

20 / febrero / 19

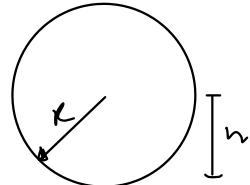
Tarea Resuelta: Sistema físico. $\rightarrow v = f(h)$
 $\hookrightarrow v - 30 = 0$.



Gráfica.

f plot : supongo que es conocido.

f plot ($f, []$) \rightarrow debe aceptar vectores de entrada.
 $h \circ 2$ Se ponen puntos para esto.



Matlab examen. Usan OCTAVE lo único que no funciona en OCTAVE son las variables simbólicas. Estudiarán los problemas y métodos utilizados

Números.

$$\mathbb{N} \quad \frac{6}{2} = 3 \text{ naturales.}$$

$$\mathbb{R} \quad \frac{5}{2} =$$

`fplot (y, linspace);
grid on;
axis [-4,`

Teorema: polinomio de coeficientes reales tiene n raíces complejas.

SIEMPRE se tienen que poner puntos para vectorizar la función.

$$\sqrt{-1} \text{ en matlab} = 0.0000 + 1.0000i$$
$$a = 2 + 3i$$

Los números complejos (*i*) se pueden cambiar por cualquier letra

Se puede usar en el examen para verificar cosas. Una manera de definir un polinomio.

$$\begin{bmatrix} 1 & 0 & 1 \end{bmatrix} = \text{vector horizontal}$$
$$\begin{bmatrix} 1, & 0, & 1 \end{bmatrix} = \text{vector horizontal.}$$
$$\begin{bmatrix} 1; & 0; & 1 \end{bmatrix} = \text{vector vertical.}$$

Con apóstrofe calcula la transpuesta.

$p = \begin{bmatrix} 1 & 0 & 1 \end{bmatrix}$ es un polinomio, con roots (*p*) calcula las raíces de ese polinomio. Las raíces las saca en números complejos.
 $x^2 + 1 = 0 \Rightarrow 0 + 1i$
 $0 - 1i$

Funcióen en matlab: `fzero` encuentra el cero de la función que le pones. Funciona para cualquier función

$a = 2 + 3i$, $a' = a^*$ → para números complejos, el apóstrofe saca el conjugado de la función.

Funcióen `angle`: nos da la norma del vector. o el ángulo que forma

`real`: da la parte real de un número complejo.

`imag`: da la parte imaginaria de un número complejo

$$(2+3i)(2-3i) = 2^2 - 3^2 i^2$$

$$r(\cos \theta + i \sin \theta)$$



$z = r^* \exp(i\theta)$ nos da el número entero complejo.

$z^n = (re^{i\theta})^n = r^n (e^{i\theta})^n = \underbrace{r^n e^{in\theta}}_{\cos n\theta + i \sin n\theta} \rightarrow$ se eleva un número complejo.

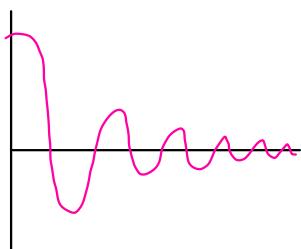
$$e^{i\pi} + 1 = 0$$

$$\underbrace{e^{i\pi} = -1}_{\cos \pi + i \sin \pi}$$

Números complejos nos interesan para lo que sigue del curso.

Ecuaciones dinámicas modelan cambios en el tiempo.

Al resolver ecuaciones diferenciales nos va a dar resultados del tipo $e^{at+bi} = e^a e^{bi} \rightarrow$ se des compone en seno y coseno con la frecuencia que depende de b .



$$z = r e^{i\theta} \quad z^n = r^n e^{in\theta}$$

$$z^n = r^n [e^{i\theta_n}]$$

Números complejos se pueden expresar con senos y cosenos y con exponentes.

Sistemas dinámicos son continuos y pueden ir decayendo.

27/febrero/19

Corrección Examen

$$F, \text{ peso} = m \cdot g$$

$$= \rho e V e g$$

$$= \rho e \frac{4}{3} \pi r^3 g.$$



flotación

09/marzo/19

Matrices

$$\text{Matlab: } A = \begin{bmatrix} 2, 1; 1, 2 \end{bmatrix} \quad \left. \begin{array}{l} \\ x = [1; 2] \end{array} \right\} A = \begin{bmatrix} 2 & 1 \\ 1 & 2 \end{bmatrix}, \quad x = \begin{bmatrix} 1 \\ 2 \end{bmatrix} \Rightarrow A^* x = \begin{bmatrix} 4 \\ 5 \end{bmatrix}$$

$$RC = \begin{bmatrix} 0 & 1 \\ -1 & 1 \end{bmatrix} \Rightarrow RC^* x = \begin{bmatrix} 2 \\ -1 \end{bmatrix} \rightarrow \text{matriz de rotación.}$$

atan2d = función en Matlab que saca el ángulo de una matriz z . La d al final significa decimales.

Matriz: transformación lineal que convierte un vector x en un vector y .