# Unidad 1

## Graficador

Considerar:

* Rango de valores de los ejes
* Escala de valores de los ejes
* Posibilidad de graficar más cerca o lejos (separación de los puntos)
* Controlar puntos para los cuales la función no esté definida
* Controlar puntos que deben graficarse fuera del área del gráfico.

## Raíces de funciones

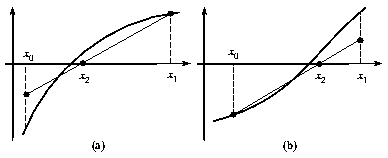
### Método cerrado: bisección

* Parte de un intervalo que debe contener la raíz
* Se toma el centro del intervalo y se ve de qué lado queda la raíz
* Se sigue iterando hasta condiciones de corte: cantidad máxima de iteraciones, error relativo, punto muy cercano a la raíz

**Datos a ingresar:**

* Intervalo (punto a y b)
* Cantidad máxima de iteraciones
* Error relativo

**Algoritmo**

Xri=0

Ci=1

Si f(a)\*f(b)<=0 entonces

Si f(a)=0 entonces xr=a

Si f(b)=0 entonces xr=b

Sino

Xr= (a+b)/2

Error=1

Xri=Xr

Mientras ci<=”cant \_maxima” y error>”e relativo” y abs(f(xr))>”e\_ relativo”

Ci=ci+1

Si f(a)\*f(xr)<0

b=xr

sino

a=xr

xr=(a+b)/2

Error= abs ((xr-xri)/xr)

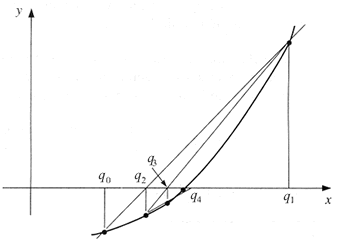
xri=xr

Fin mientras

Sino

Mensaje “Vuelva a ingresar un intervalo que contenga la raíz”

### Método cerrado: regla falsa

* Supone que la raíz no está cercana al medio del intervalo sino a uno de los extremos, donde abs(f(x)) es más cercano a 0
* Dado el intervalo se estima una recta que une los valores de la función en los puntos y se toma como xr el punto en donde la recta corta el eje x
* Se sigue iterando hasta las condiciones de paro: cantidad máxima de iteración, error relativo entre raíces o valor cercano a 0
* Los ángulos opuestos por el vértice tienen = tangente => f(a) / (xr-a) = f(b) /(b-xr)
* Formula: xr= (f(a)\*b – f(b)\*a) / (f(a)-f(b))

**Algoritmo**

Xri=0

Ci=1

Si f(a)\*f(b)<=0 entonces

Si f(a)=0 entonces xr=a

Si f(b)=0 entonces xr=b

Sino

Xr= (f(a)\*b – f(b)\*a) / (f(a)-f(b))

Xri=Xr

Error= 1

Mientras ci<=”cant \_maxima” y error>”e relativo” y abs(f(xr))>”e\_ relativo”

Ci=ci+1

Si f(a)\*f(xr)<0

b=xr

sino

a=xr

Xr= (f(a)\*b – f(b)\*a) / (f(a)-f(b))

Error= abs ((xr-xri)/xr)

xri=xr

Fin mientras

Sino

Mensaje “Vuelva a ingresar un intervalo que contenga la raíz”

## Método abierto: secante

* A partir de 2 puntos iniciales (xi y xi+1) que es conveniente que no contengan la raíz se traza la secante entre ellos (recta que une las función evaluada en los puntos) y donde corta el eje se toma Xi+2
* Xi+2= xi+1 – ( (f(xi+1) – (xi-xi+1)) / (f(xi) – f(xi+1))
* Si xi+2 no es raíz entonces xi= xi+1; xi+1=xi+2
* Si converge lo hace rápido (cuando la función es ln(x) LA SECANTE SE VA FUERA DEL DOMINIO

**Algoritmo**

Si f(x1)\*f(x2) =0

Si f(x1)=0 entonces xr=x1

Si f(x2)=0 entonces xr=x2

sino

Xr=x2 – (f(x2)\* (x1-x2)) / (f(x1) – f(x2))

Ci=1

Error=1

Mientras ci<=”cant \_maxima” y error>”e relativo” y abs(f(xr))>”e\_ relativo”

Ci=ci+1

Error= abs((xr-x2) / xr)

X1=x2

X2=xr

Xr=x2 – (f(x2)\* (x1-x2)) / (f(x1) – f(x2))

## http://3.bp.blogspot.com/_TNRUVu5MLqE/S7rvBTJJgfI/AAAAAAAAABA/YjFK8qCVMlg/s320/imagen1.JPG

## Método abierto: Newton Rawson (tangente)

* Parte de un punto inicial,
* Calcula la recta tangente al punto y ve donde corta el eje (la derivada)
* Si converge lo hace rápido.
* Si la raíz coincide con un punto de inflexión puede producir un ciclo

Ci=0

Error=1

Xr=x1

Mientras ci<=”cant \_maxima” y error>”e relativo” y abs(f(xr))>”e\_ relativo”

Ci=ci+1

Derivada=(f(x1+0.00001) – f(x1))/0.00001

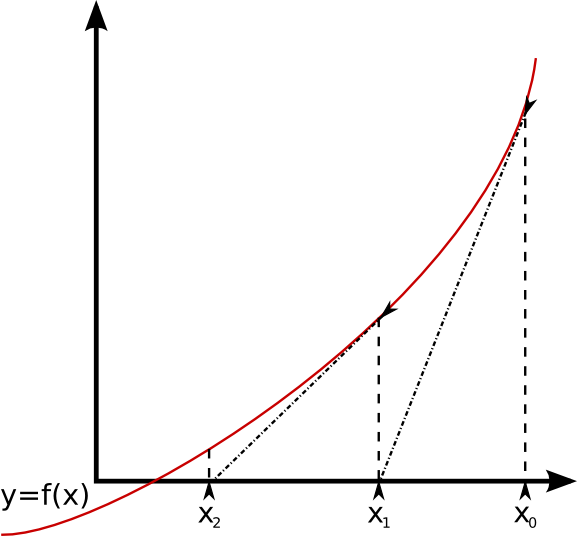
Si derivada<>0

Xr=x1- (f(x1)/derivada)

Error= abs ((xr – x1) /xr)

X1=xr

Fin mientras



# Unidad 2

## Sistema de ecuaciones

La solución de los sistemas de ecuaciones lineales encuentra una amplia aplicación en la ciencia y la tecnología. En particular, se puede afirmar, que en cualquier rama de la Ingeniería existe al menos una aplicación que requiera del planteamiento y solución de tales sistemas.

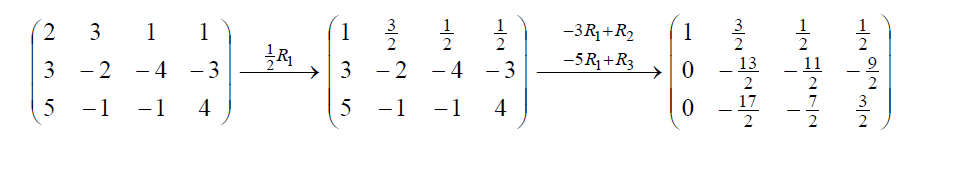
Métodos

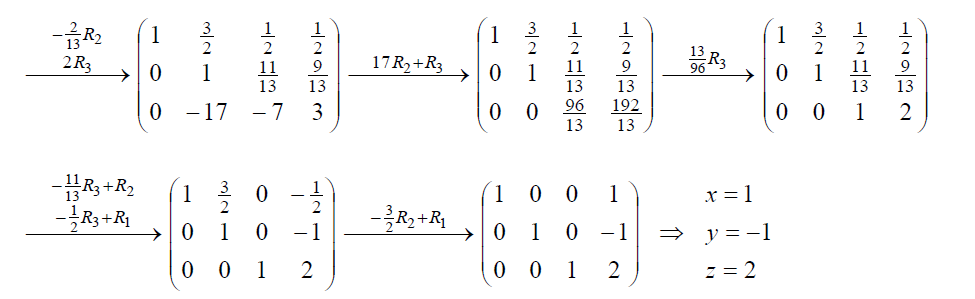
* Gauss Jordan
* Gauss Seidel

### Gauss Jordan

Ejemplo con 3 filas

1. Escribir la matriz aumentada
2. Scaling (dividir cada renglón por el mayor de la fila) => para poder calcular el determinante y por si está mal condicionada
3. Normalizar
   1. dividir 1° fila por la diagonal (hace 1 la diagonal)
   2. Restar fila 1 y 2 y restar fila 1 y 3 para hacer 0 los números debajo de la diagonal (a21\* R1 + R2) y –a31 \* R1 + R2)
   3. Divido fila 2 por el valor de la diagonal (hago 1 la diagonal)
   4. Resto fila 3 a32 \* R2 + R3 (hace 0 debajo de la diagonal). Luego a12 \* R2 + R1
   5. Divido fila 3 por la diagonal
   6. Hago 0 los de arriba – a23 \* R3 + R2 y a13 \* R3 + R1.
4. Los valores que quedan en el termino independiente son los valores de las incógnitas





**Algoritmo**

Aceptar n

Aceptar A(n,n+1)

Escaling (A,n)

Normalizar(A,n)

‘*mostrar resultados*

For i=1 to i

Mostrar A(i,n+1)

Next i

*Escaling (A matriz, n entero)*

For i=1 to n

Max=|A(i,1)|

For j=2 to n+1

Si |A(i,j)|>max

Max=|A(i,j)|

Fin si

Next j

For j=1 to n+1

A(i,j)=A(i,j) /max

Next j

Next i

*Normalizar (A matriz,n entero)*

For i=1 to n

Si A(i,i)<>0 Coe=A(I,i)

For j=i to n+1

Si A(i,j) <>0

A(i,j)=A(i,j) /coe

Sino

‘El método no puede continuar’

Fin si

Next j

‘triangulación

For j=1 to n

Si i<>j

Coe=A(j,i)

For k=i to n+1

A(j,k)=A(j,k) – coe \* A(i,k)

Next k

Fin si

Next j

Next i

### Gauss Seidel

La secuencia de pasos que constituyen el método de *Gauss-Seidel* es la siguiente:

1. Asignar un valor inicial a cada incógnita que aparezca en el conjunto.
2. Partiendo de la primera ecuación, determinar un nuevo valor para la incógnita, utilizando para las otras incógnitas los valores supuestos.
3. Pasar a la segunda ecuación y determinar en ella el valor de la incógnita, utilizando el valor calculado para la incógnita del paso 2 y los valores supuestos para las incógnitas restantes.
4. Continuar con las ecuaciones restantes, determinando siempre el valor calculado de la incógnita particular, y utilizando siempre los últimos valores calculados para las otras incógnitas de la ecuación.
5. Continuar iterando hasta que el valor de cada incógnita, determinado en una iteración particular, difiera del valor obtenido en la iteración previa, en una cantidad menor que cierto EPSILON(error relativo) seleccionado arbitrariamente o se supere una cantidad máxima de iteraciones previstas.

**Algoritmo**

Aceptar n

Aceptar A (n, n+1)

Aceptar maxIter, Aceptar errorRela

Escaling(A,n)

V(i)= Seidel (A, n, maxIter, errorRela)

‘*mostrar resultados*

For i=1 to i

Mostrar V(i)

Next i

*Seidel( A matriz, n entero, maxIter entero, ErrorRela doble) vector*

E(i)=0

Iter=0

ErrorR=1

For i=1 to n

Si A(i,i) =0

“El método no puede aplicarse”

Exit function

Fin si

Mientras iter<maxIter and ErrorR>ErrorRela

ErrorR=0

For i=1 to n

V(i)=A(i,n+1)

For j=i to n+1

Si i<>j

V(i)=V(i)- A(i,j) \* V(j)

Next j

V(i)= V(i) /A(i,i)

Si |(V(i)-E(i))/V(i)|>=ErrorRela

ErrorR=1

Fin si

Next i

Iter=iter+1

For i=1 to n

E(i)=V(i)

Next i

fin mientras

Retornar V(i)

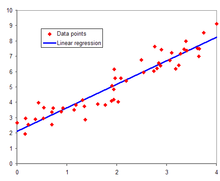
# Unidad 3

## Regresión

Permite realizar un ajuste de datos experimentales a una función

### Regresión lineal

Dados los puntos de las observaciones realizadas se trata de buscar la mejor recta que los ajuste.



##### Algoritmo

Aceptar n (cantidad de puntos)

For i=1 to n

Aceptar X(i), Y(i)

Sx=sx + X(i)

Sy= Sy+y(i)

Sx2=Sx2+X(i)2

Sxy=Sxy + X(i)\*Y(i)

Si ((n\*Sx2) – Sx22)<>0

a 1= ((n\* Sxy)-(Sx\*Sy))/((n\*Sx2)-Sx2)

a0= (Sy/n) – (a1\*(sx/n))

st=0, sr=0

for i=1 to n

st=st+((Sy/n)-y(i))2

sr= sr+ ((a1\*X(i)) + a0 – y(i))2

si st<>0

r=sqrt(|(st-sr)/st|)\*100  
 mostrar a1,a0,r

### Regresión polinomial

* Solicitar grado del polinomio
* Controlar que el grado no sea <1
* N indica la cantidad de puntos, grado el grado del polinomio

**Algoritmo**

For i=1 to grado

For j=1 to grado+1

M(I,j)=0

For i=1 to n

Aceptar x(i), y(i)

Sx=sx + x(i)

Sy=Sy+ y(i)

For j= 1 to grado+1

For k= 1 to grado+1

M(j,k) = M(j,k) + X(i)(k-1)+(j-1)

M(j,Grado+2)=M(j,grado+2)+(Y(i)\*x(i)(j-1))

R=gauss(M)

St=0, sr=0

For i=1 to n

St=st+((sy/n)-Y(i))2

S=0

For j=1 to grado+1

S=s+(r(j) \* (X(i)(j-1)))

Sr=sr+(s-y(i))2

si st<>0

corr=sqrt((st-sr)/st) \* 100

for i=1 to grado+1

mostrar r(i)

sino

division por 0

## Interpolación – Lagrange

* Considerar que se puedan ingresar “n” puntos a interpolar
* Permitir el ingreso de todos los puntos y luego mostrar el valor hallado de cada uno
* Hacer repetidas llamadas al método para obtener cada valor

**Algoritmo**

Aceptar n (cantidad de puntos) , x (valor a interpolar)

For i=1 to n

Aceptar X(i), Y(i)

S=0, i=0

Mientras i<=n

Sa=1, sb=1

For j=1 to n

Si i<>j

Sa=sa\* (x-x(j))

Sb= sb\* (X(i)-X(j))

S=s + y(i) \* sa/sb

i=i+1

retornar S

# Unidad 4: Integración

* El valor de la integral siempre debe ser positivo

## Trapecios

Aceptar a,b,n

S=0; h=(b-a)/n

For i=1 to (n-1)

S=s+f(a+(i\*h))

Int=h/2\* f(a) + 2\*s + f(b))

Mostrar int

## Simpson 1/3 intervalos múltiples

Aceptar a,b

Aceptar n hasta que (-1)n =1

H=(b-a)/n

Sp=0; Si=0

For i=1 to (n-1)

Si (-1)i =1

Sp=sp + f(a+(h\*i))

Sino

Si=si+f(a+(h\*i))

Int=h/3 \* f(a) + 4\*si + 2\*sp + f(b)

Retornar int

## Simpson 3/8

Aceptar a,b

Aceptar n hasta n>1

Si (-1)n =(-1)

h=(b-a)/n

Nb= b- h\*3

Nn=n-3

Int1= Simpson1/3 (a,nb,nn)

na=nb

nb=b

nc=b- h\*2

nd=b- h

Int2= 3/8\*h \* f(na) + 3\* f(nc) + 3\* f(nd) + f(nb)

Int= int1 + int2

Mostrar int