Bisección

$\sqrt{x^2+1} = tg(x)$ criterio de parada: long $< 0{,}001$

$0 < x < \frac{\pi}{2}$

dibujar gráfica para ver intervalo

```
f[x_] = -√(x²+1) + Tan[x];
Plot[f[x], {x, 0, π/2}, PlotRange → {-1,1}]
```

definimos el intervalo (según lo anterior

```
a = 0,8;
b = 1;
c = (a+b)/2;
lon = Abs[b-a];
```

⚠ definimos antes del while

iteramos

```
While[
  test
  lon > 0.001 && Abs[f[c]] > 0,0001,
  If[f[c]*f[a] < 0, b = c, a = c];
  c = (a+b)/2;
  lon = Abs[b-a];
  Print[c, " ", lon, " ", f[c]]
  body
]
```

Newton-Raphson

$0.5 \cdot e^{x/3} - \sin(x) = 0$

$\varepsilon < 10^{-15}$ para raíces (+)

dibujar gráfica para aproximación inicial

```
f[x_] = 0.5*Exp[x/3] - Sin[x];
Plot[f[x], {x, 0, 4}]
```

coger 1 de las raíces (o las dos)

⚠ se ve corte en x≈0.6 ∧ x≈2.0

definimos intervalo, criterio de parada ∧ Nº de itera-

```
xv = 0.6;      → xvieja = xanterior
error = 1;
eps = 10^-5;   → epsilon
nmax = 50;
n = 1;
```

iteramos

```
While[
  test
  error > eps && n < nmax,
  xn = SetPrecision[xv - f[xv]/f'[xv], 15];
  error = Abs[xn - xv];
  Print["En la iter.", n, "la sol.aprox:", xn];
  xv = xn;
  n++
  body
]
```

SECANTE

$0.5 \cdot e^{\frac{x}{3}} - \sin(x) = 0$

— dibujamos como hemos hecho en N-R —

```
f[x_] = 0.5 * Exp[x/3] - Sin[x];

Plot[f[x], {x,0,4}]
```

— iteramos la definición —

```
                                                        start
For[ eps = 10^-15; nmax = 50; x0 = 0.6; x1 = 0.8; n=1; error=1,
                                 test
   error > eps && n < nmax,
   n++,   incr
   x2 = SetPrecision[x1 - f[x1] * (x1-x0)/f[x1]-f[x0], 15];
   error = Abs[x2-x1];
   Print["En la iter ", n, "la sol. aprox: ", x2];
   x0 = x1;   body
   x1 = x2
]
```

i) Valores de x que verifican la ecuación:

$\text{Solve}[\underbrace{x^2 + 2x - 7 == 0}_{\text{ecuación}}, \; x]$ → var. indep.

ii) Valor aprox. de x

$\text{NSolve}[\underbrace{x^2 + 2x - 7 == 0}_{\text{ecuación}}, \; x]$ → var. indep

iii) Extraer la solución:

```
Sol = NSolve[------];
a = x /. Sol[[1]]
```

iv) Encontrar raíz con ec. no lineales

1º dibujar: $\text{Plot}[\{\underbrace{\text{Cos}[x], x}_{\text{ecuación}}\}, \underset{\text{var. indep.}}{\{x}, \underbrace{0, \text{Pi}\}}_{\text{intervalo}}]$

2º raíz: FindRoot[Cos[x]==x, {x,1}]

02 - ecuacionesnolineales.nb

• Resolver sist. simbólicamente:

$$\begin{cases} ax+by=1 \\ x-y=2 \end{cases}$$

1º) $Sol = Solve[\{\underbrace{ax+by==1}_{ec.1}, \underbrace{x-y==2}_{ec.}\}, \{x,y\}]$

da como resultado = {{x → ec. , y → ec.}}

2º) Extraer soluciones

x1 = x /. Sol [[1]]

y1 = y /. Sol [[1]]

3) Obtener todas las soluciones

Reduce [a*x == 0 , x]

4) Soluciones en base a la var. indep.

Solve [a*x == 0, x]

• Resolver sistemas matriciales:

$$\begin{pmatrix} 1 & 5 \\ 2 & 1 \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a \\ b \end{pmatrix} \Rightarrow$$

mat = {{1,5},{2,1}};

c = {a,b};

sol = LinearSolve[m, c] * da 2 soluciones < x, y

sol[[1]] da la solución de x

sol[[2]] da la solución de y

para varios sistemas con la misma matriz:

1) lsf = LinearSolve [mat]

2) lsf [c] ← ponemos el vector c que queremos

Norma

b = {..., ..., ..., ...};

maxvect = Norm [b, ∞];

vgrad = Grad[x^4 + y^4 + z^4, {x,y,z}] ⇒ da {4x^3, 4y^3, 4z^3}

VectorPlot3D [vgrad, {x,-1,1}, {y,-1,1}, {z,-1,1}, VectorColorFunction → Hue]

Gradiente

f[r_, fi_, z_] = r^3 * Sin[fi]

Grad[f[r, fi, z], {r, fi, z}, "Cylindrical"]

Rotacional

f[R_, theta_, fi_] = {R^2 * Cos[theta], Cos[fi], 0}

Curl [f[R, theta, fi], {R, theta, fi}, "Spherical"]

Divergencia

Div[Curl[f[R, theta, fi], {R, theta, fi}, "Spherical"], {R, theta, fi}, "Spherical"]

JACOBI

A = {{--, --, --}, {--, --, --}}; (una matriz)

b = {--, --, --}; (un vector)

n = Length[b];

xant = {--, --, --}; (aprox. inicial)

xnuev = xant;

(A) si nos piden un número de iteraciones

i de 1→n

$$\text{Do}\Big[\ \text{Do}\Big[\text{xnuev}[[i]] = \frac{1}{A[[i,i]]} * \Big(b[[i]] - \sum_{j=1}^{i-1} A[[i,j]] * \text{xant}[[j]] - \sum_{j=i+1}^{n} A[[i,j]] * \text{xant}[[j]]\Big), \{i,n\}\Big];$$

Print[xnuev];
eabs = Norm[xnuev - xant, ∞];
erel = eabs/Norm[xnuev, ∞];
Print["El error absoluto es: ", eabs];
Print["El error relativo es: ", erel];
xant = xnuev,

{k, 1, 5}

Nº de iteraciones.

]

(B)

tolerancia

error = 1; tol = 10^(-4); k=0;

While[error > tol,
k = k+1;
Do[, {i,1,n}];
Print["En la iter. ", k];
Print["La nueva aprox es", xnuev];
error = Norm[xnuev - xant, ∞]/Norm[xnuev, ∞];
Print["El error relat. es: ", error];
xant = xnuev
]

Gauss-Seidel

```
A = { {--, --, --}, {--, --, --} };   (una matriz)
b = {--, --, --};   (un vector)
n = Length[b];
xant = {--, --, --};   aprox. inicial
xnuev = xant
```

```
Do[
  Do[xnuev[[i]] = 1/A[[i,i]] * (b[[i]] - Sum_{j=1}^{i-1} A[[i,j]] * xnuev[[j]] - Sum_{j=i+1}^{n} A[[i,j]] * xant[[j]]), {i, n}];

  Print[xnuev];
  eabs = Norm[xnuev - xant, ∞];
  erel = eabs / Norm[xnuev, ∞];
  Print["error absoluto: ", eabs];
  Print["error relativo: ", erel];
  xant = xnuev,
  {k, 1, 5}
]
```

la única diferencia con Jacobi

(B)

```
error = 1; tol = 10^(-4); k = 0;
While[error > tol,
  k = k+1;
  Do[ ];
  Print["En la iter ", k];
  Print["La nueva aprox es ", xnuev];
  error = Norm[xnuev - xant, ∞] / Norm[xnuev, ∞];
  Print["Error relativo = ", error];
  xant = xnuev
]
```

MATHEMATICA

valores exactos!

valores aproximados → poner un punto a cualquier operando para que sea decim

poner ; no muestra output "numerical value" N[, 5] ó //N operación Nº de cifras significativas | x=0 Clear[x] ...

operador reemplazamiento: /. eg: x^2-3x+1 /. x→1 resultado = −1 el valor es local, no se asigna

definir funciones: f[x_, y_] = ... ó f[x_, y_] := ... no la evalua, sólo define, usar en caso de errores

para que x e y puedan sustituirse por cualquier expresión.

funciones definidas a trozos: f[x_] = Piecewise[{{-1, x<-1}, {x, -1<=x<=1}, {1, x>1}}]

Expand[$(x-2)^2(x+1)^3(x-1)$] → $-4-4x+7x^2+6x^3-4x^4-2x^5+x^6$

Factor al revés, del polinomio a los factores.

Apart para separar una fracción
Together para unir (una suma de fracciones)
Cancel para simplificar

e.g. TrigExpand[Sin[x+y]] ⇒ Cos[y] Sin[x] + Cos[x] Sin[y]

Derivada D[Sin[x], x] | D[Sin[x], {x, 3}] derivada 3ª

Diferencial Dt[Sin[x]] → Cos[x] Dt[x]

Integrales $\int \ldots dx$ ó Integrate[, {x, a, b}]

Gradiente: da un vector que da idea de cómo varía el campo/magnitud (es un vector)
$f(x,y,z) = 2x+3y^2-\sin(z)$; $\nabla f = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}, \frac{\partial f}{\partial z}\right) = (2, 6y, -\cos(z))$

Divergencia: es el flujo por unidad de volumen, indica si en una superficie "..." hay puntos ó sumideros del campo (función vectorial) (es un escalar)
$\nabla \cdot \vec{F} = \left(\frac{\partial F_x}{\partial x} + \frac{\partial F_y}{\partial y} + \frac{\partial F_z}{\partial z}\right)$

Rotacional: aunque tengamos un campo siempre en la misma dirección, si en un punto hay ≠ magnitud de ese campo, se crea un rotacional (regla de la mano derecha)
$\nabla \times \vec{F} = \begin{vmatrix} \hat{x} & \hat{y} & \hat{z} \\ \frac{\partial}{\partial x} & \frac{\partial}{\partial y} & \frac{\partial}{\partial z} \\ F_x & F_y & F_z \end{vmatrix}$ $\vec{J} = \nabla \times \vec{F}$

(sigue en p. 6)

· Matrices de paso a radios espectrales: T_J

Jacobi: A = {{...}, {...}, {...}, {...}};
n=3 ; (Rango de A)

T_J = Table[If[i==j, 0, −A[[i,j]]/A[[i,i]]], {i,1,n}, {j,1,n}];

MatrixForm[T_J]

Max[Abs[Eigenvalues[T_J]]]

Gauss-Seidel $T_G = -(D+L)^{-1} \cdot U$

diag = DiagonalMatrix[Diagonal[A]]
L = LowerTriangularize[A] − diag;
U = UpperTriangularize[A] − diag;
T_G = −Inverse[diag+L].U ;

MatrixForm[ ]; L, U, T_G

Max[Abs[Eigenvalues[T_G]]].

Si hay que reordenar para tener estric. d. d.
p = {3, 1, 2, 4} nuevo orden de filas.
A = A[[p]]; MatrixForm[A]
b = b[[p]]; MatrixForm[b]
y ahora ya usar Jacobi o Gauss-Seidel

1º Comprobar convergencia con T_G, T_J a $\rho(T_J)$, $\rho(T_G)$ radios espectrales
2º Si convergen (<1), entonces aplicar el método que creamos o que nos digan.