P1. EKUAZIO DIFERENTZIAL ARRUNTAK

SOLUZIO OROKORRAREN KALKULUA

-**DSolve[**"ekuazioa", y[x],x**]** : soluzio orokorrak eman
-**DSolveValue[**"ekuazioa", y[x],x**]** : soluzio orokor bakarra eman
-**Simplify[**%**]** : soluzioa sinplifikatu
-**C[**n**]** : konstanteak
-**Plot[** {ekuazioak}, {x,xmin,xmax}**]** : Irudikatu

ADB

- *SO[x_] = DSolveValue[y'[x] - y[x] == Sin[x], y[x], x];*

- *SolOro = DSolve[y'[x] - y[x] == Sin[x], y[x], x]*
  yp[x_] = y[x] /. SolOro[[1]] *// ERAUZI BEHAR DA //*

- *Plot[{ yp[x] /. C[1] -> 0, yp[x] /. C[1] -> 1, yp[x] /. C[1] -> -2}, {x, -2, 2}];*

HASTAPEN ETA MUGALDE BALDINTZEN PROBLEMEN SOLUZIOEN KALKULUA

-**DSolve[**{ekuazioa,baldintzak}, y[x],x**]** : soluzio partikularrak eman
-**DSolveValue[**{ekuazioa,baldintzak}, y[x],x**]** : soluzio partikular bakarra eman

EKUAZIO DIFERENTZIALEN NORABIDE EREMUAK

-**VectorPlot[**{1,f},{x,xmin,xmax},{y,ymin,ymax},VectorColorFunction->Hue**]** : y'=f(x,y) -ren norabide-eremua irudikatu
-**StreamPoints** -> Coarse : marrak irudikatu

ADB

1.- *F=Solve[y' - y == Sin[x] ,y']*
2.- *F1[x_,y_]= y' /. F[[1]]*
3.- *NorEre = VectorPlot[{1, F1[x,y]}, {x, -1, 1}, {y, -1, 1}, Axes -> True, VectorColorFunction -> Hue]*

HASTAPEN ETA MUGALDE BALDINTZEN PROBLEMEN SOLUZIOEN HURBILDUAK

-**NDSolve[**{ekuazioa,baldintzak},y[x],{x,xmin,xmax}**]**: interpolatutako soluzioak adierazitako eremuan

ADB

1.- *SN = NDSolve[{y'[x] - y[x] == Sin[x], y[0] == 0}, y[x], {x, -1, 1}]*
2.- *SE[x_] = y[x] /. SN[[1]]* *// ERAUZI BEHAR DA //*
3.- *Table[SE[x], {x, -1, 1, 0.2}] // Chop // TableForm*
4.- *Plot[SE[x], {x, -1, 1}]*

EXISTENTZIA ETA BAKARTASUNAREN TEOREMA

ADB

1.- *F = Solve[y' - y == Sin[x] ,y']*
2.- *F1[x_,y_]= y' /. F[[1]]*
2.- *F1d = D[F1[x,y], y]*

P2. EKUAZIO DIFERENTZIALEN SISTEMAK

P2.1. LEHEN ORDENAKO SISTEMAK

SOLUZIO OROKORRAREN KALKULUA

-**DSolve[**{eku1,eku2, … ,ekuN}, { x1[t], …, xN[t] }, t **]** : soluzio orokorrak eman

HASTAPEN BALDINTZEN PROBLEMEN SOLUZIOEN KALKULUA

-DSolve[{ ekuazioak,baldintzak }, { x1[t], x2[t], … ,xn[t] } , t **]** : soluzio partikularrak eman
-ParametricPlot[{ sol1[t], sol2[t] }, {t, tmin, tmax} **]** : sistemaren soluzio funtzioa irudikatu

HASTAPEN BALDINTZEN PROBLEMEN SOLUZIOEN HURBILDUAK

-NDSolve[{ ekuazioa,baldintzak }, { x1[t],x2[t], ... ,xn[t] },{ t,tmin,tmax } **]** : soluzio hurbildua

P2.2. SISTEMA AUTONOMOAK

FASE-ESPAZIOAREN AZTERKETA

-StreamPlot[{eku1,eku2}, {X, Xmin, Xmin}, {Y, Ymin, Ymax}, StreamColorFunction -> "Rainbow" **]**

ADB

```
 sol = Solve[{2 x - y == 0, -x + 2 y == 0}, {x, y}]
Out[25]= {{x -> 0, y -> 0}}
```

1) (0,0) puntu kritikoa
2) Jakobiarra definitu

```
jacob[x_, y_] := {{D[2 x - y, x], D[2 x - y, y]}, {D[ -x + 2 y , x],
   D[ -x + 2 y , y]}}
A = jacob[x, y] /. sol[[1]]
Out[27]= {{2, -1}, {-1, 2}}

Det[A]
Out[28]= 3
```

3) A matrizearen autobalio eta autobektoreak

```
Eigenvalues[A]
Eigenvectors[A]
Out[31]= {3, 1}
Out[32]= {{-1, 1}, {1, 1}}
```

Autobalioak 1 eta 3 positiboak eta errealak direnez, NODO EZ EGONKOR bat edukiko dugu

4) Ardatzen eta funtzioen norabideen analisi sakona

```
SO = DSolve[{x'[t] == 2 * x[t] - y[t], y'[t] == -x[t] + 2 * y[t]}, {x[t], y[t]}, t];
xe[t_] = x[t] /. SO[[1,1,2]];
ye[t_] = y[t] /. SO[[1,2,2]];
Limit[xe[t] / ye[t], t → ∞]
Limit[xe[t] / ye[t], t → -∞]
Out[33]= -1
Out[34]= 1
```

5) Irudikatu

```
PF1 = StreamPlot[{2 x - y, -x + 2 y}, {x, -1, 1}, {y, -1, 1},
  StreamColorFunction -> "Rainbow"]

Ibilbide = Plot[{x, -x}, {x, -1, 1}]
Show[PF1, Ibilbide]
```