Puncte de echilibru. Stabilitate

In cadrul acestui laborator sunt prezentate instructiunile necesare studiului calitativ al solutiilor in jurul punctelor de echilibru in cazul ecuatiilor scalare autonome si a sistemelor planare de ecuatii autonome.

Ecuatii scalare autonome

Ecuatiile scalare autonome sunt de forma:

$$x' = f(x)$$

Solutiile constante $x(t) \equiv x^*$ ale ecuatiei diferentiale autonome se numesc **solutii de echilibru**, valoarea x^* se numeste **punct de echilibru**. Punctele de echilibru sunt solutiile reale ale ecuatiei:

$$f(x) = 0$$

Fie ecuatia diferentiala autonoma:

$$x' = x\left(1 - x^2\right)$$

In [1]:

```
reset()
t=var('t')
x=function('x')(t)
```

In [2]:

```
s=var('s')
f=function('f')(s)
f(s)=s*(1-s^2)
```

Punctele de echilibru se determina prin rezolvarea ecuatiei $f(s) = 0$

In [3]:

```
eqp=solve(f(s)==0,s)
eqp
```

Out[3]:

```
[s == -1, s == 1, s == 0]
```

Pentru studiul stabilitatii punctelor de echilibru pot fi aplicate doua metode, fie se aplica Teorema stabilitatii in prima aproximatie sau metoda grafica (analiza campului de directii).

Teorema stabilitatii in prima aproximatie

Fie x^* un punct de echilibru al ecuatiei diferentiale autonome:

$$x' = f(x)$$

unde f este de clasa C^1. Atunci:

1. Daca $f'(x^*) < 0$ atunci x^* este local asimptotic stabil;
2. Daca $f'(x^*) > 0$ atunci x^* este instabil.

Prin aplicarea Teoremei stabilitatii in prima aproximatie

Punctele de echilibru ale ecuatiei autonome date sunt stocate in variabila de tip lista *eqp*

In [4]:

```
eqp[0]
```

Out[4]:

```
s == -1
```

In [5]:

```
eqp[0].rhs()
```

Out[5]:

```
-1
```

In [6]:

```
x1=eqp[0].rhs()
x1
```

Out[6]:

```
-1
```

In [7]:

```
x2=eqp[1].rhs()
x2
```

Out[7]:

```
1
```

In [8]:

```
x3=eqp[2].rhs()
x3
```

Out[8]:

```
0
```

Pentru a aplica Teorema stabilitatii in prima aproximatie trebuie sa obtinem valoarea $f'(-1)$

In [9]:

```
diff(f,s)(x1)
```

Out[9]:

```
-2
```

Deoarece $f'(x_1) = f'(-1) = -2 < 0$ atunci punctul de echilibru $x_1 = -1$ este local asimptotic stabil.

Vom proceda similar pentru celelalte doua puncte de echilibru $x_2 = 1$ and $x_3 = 0$:

In [10]:

```
diff(f,s)(x2)
```

Out[10]:

```
-2
```

Deoarece $f'(x_2) = f'(1) = -2 < 0$ atunci punctul de echilibru $x_2 = 1$ este local asimptotic stabil.

In [11]:

```
diff(f,s)(x3)
```

Out[11]:

```
1
```

Deci, $f'(x_3) = f'(0) = 1 > 0$ ceea ce implica faptul ca punctul de echilibru $x_3 = 0$ este instabil.

Metoda grafica (analiza campului de directii)

In cazul ecuatiilor scalare autonome, Sagemath nu are o comandă pentru a genera portretul fazic, stabilitatea soluțiilor de echilibru se poate obtine prin analiza campului de directii si reprezentarea grafica a solutiilor reprezentative.

In general, pentru ecuatia diferentiala

$$x' = f(t,x)$$

campul de directii poate fi obtinut prin comanda

plot_slope_field(f(t,x),(t,a,b),(x,c,d),headaxislength=n, headlength=m,color='color_name')

In cazul ecuatiilor diferentiale autonome variabila independenta t nu apare in mod explicit in expresia ecuatiei

$$x' = f(x)$$

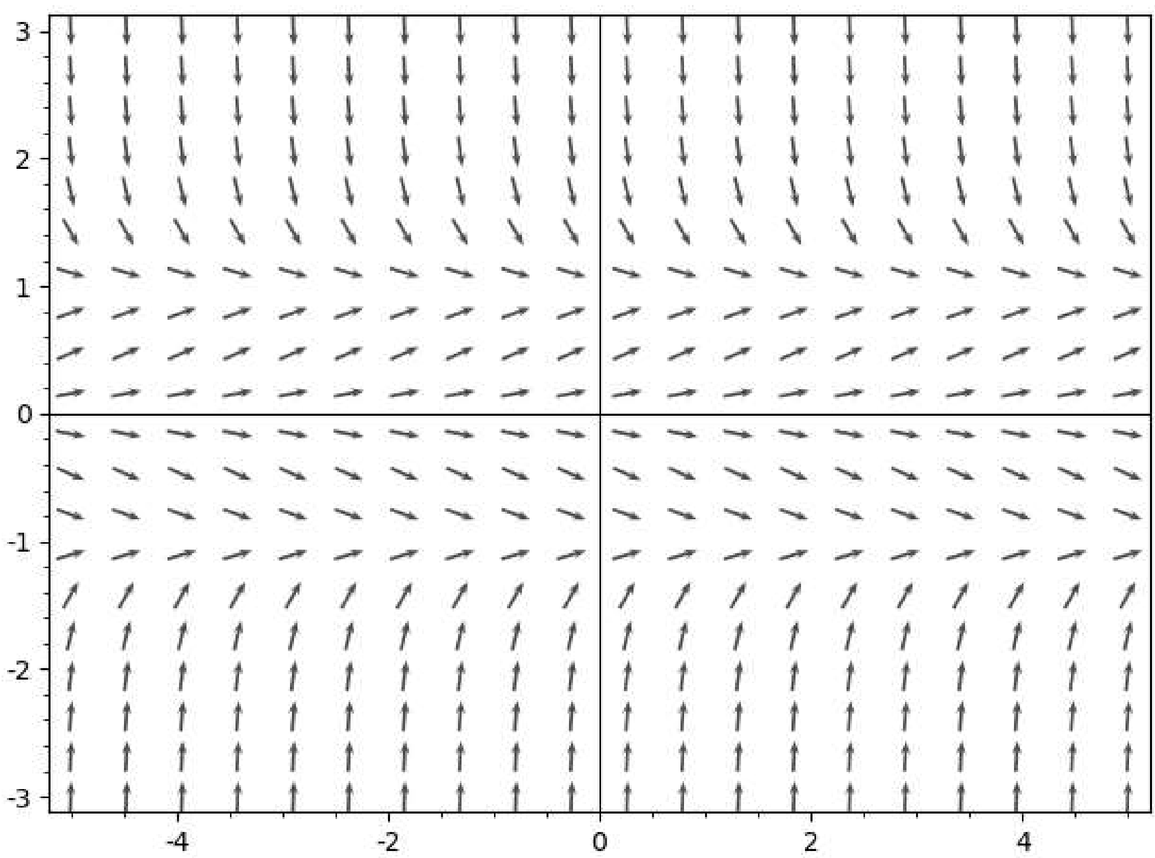
astfel, campul de directii se obtine utilizand comanda in forma:

plot_slope_field(f(x),(t,a,b),(x,c,d),headaxislength=n, headlength=m,color='color_name')

In [12]:

```
sf=plot_slope_field(f(s),(t,-5,5),(s,-3,3),headaxislength=3, headlength=4,color='red')
sf
```

Out[12]:



Pentru a vizualiza comportamentul pe termen lung al solutiilor este necesara reprezentarea grafica a unor solutii reprezentative. Deoarece in general solutiile ecuatiilor autonome nu pot fi obtinute in mod explicit este necesara utilizarea unei metode numerice pentru reprezentarea grafica a solutiilor.

Comanda *desolve_rk4()* rezolva numeric problema Cauchy atasata ecuatiei prin utilizarea metodei numerice Runge-Kutta si returneaza fie o lista a valorilor solutiei pe o anumita diviziune a intervalului considerat sau graficul solutiei pe acest interval.

Structura comenzii este:

desolve_rk4(de, dvar, ics=None, ivar=None, end_points=None, step=0.1, output='list')

- Varianta 1 (functie de doua variabile)
  - *de* - membrul drept al ecuatiei, adica functia $f(x, y)$ din ecuatia $y' = f(x, y)$
  - *dvar* - variabila dependenta (functia necunoscuta)

- Varianta 2 (ecuatia diferentiala simbolica)
  - *de* - ecuatia diferentiala incluzand termenul diff(y,x)
  - *dvar* - variabila dependenta (functia necunoscuta)

- Alti parametrii:
  - *ivar* - trebuie specificat in cazul in care ecuatia este autonoma sau apar mai multi parametrii in ecuatia diferentiala
  - *ics* - conditia initiala de forma [x0,y0]
  - *end_points* - capetele intervalului
    - daca *end_points* este a sau [a], integrarea se face pe intervalul min(ics[0],a) si max(ics[0],a)
    - daca *end_points* este None, este utilizat implicit end_points=ics[0]+10
    - daca *end_points* este [a,b] integrarea se face pe intervalul min(ics[0], a) and max(ics[0], b)
  - *step* - (optional, default:0.1) marimea pasului metodei (numar pozitiv)

- output - (optional, default: 'list') este una din 'list', 'plot', 'slope_field' (graficul contine solutia si campul de directii)

In [13]:

```
deq=diff(x,t)==f(x)
deq
```

Out[13]:

```
diff(x(t), t) == -(x(t)^2 - 1)*x(t)
```

In [14]:

```
desolve_rk4(deq, x, [0,0.5], step=0.5, end_points= [-5,5], output='list')
```

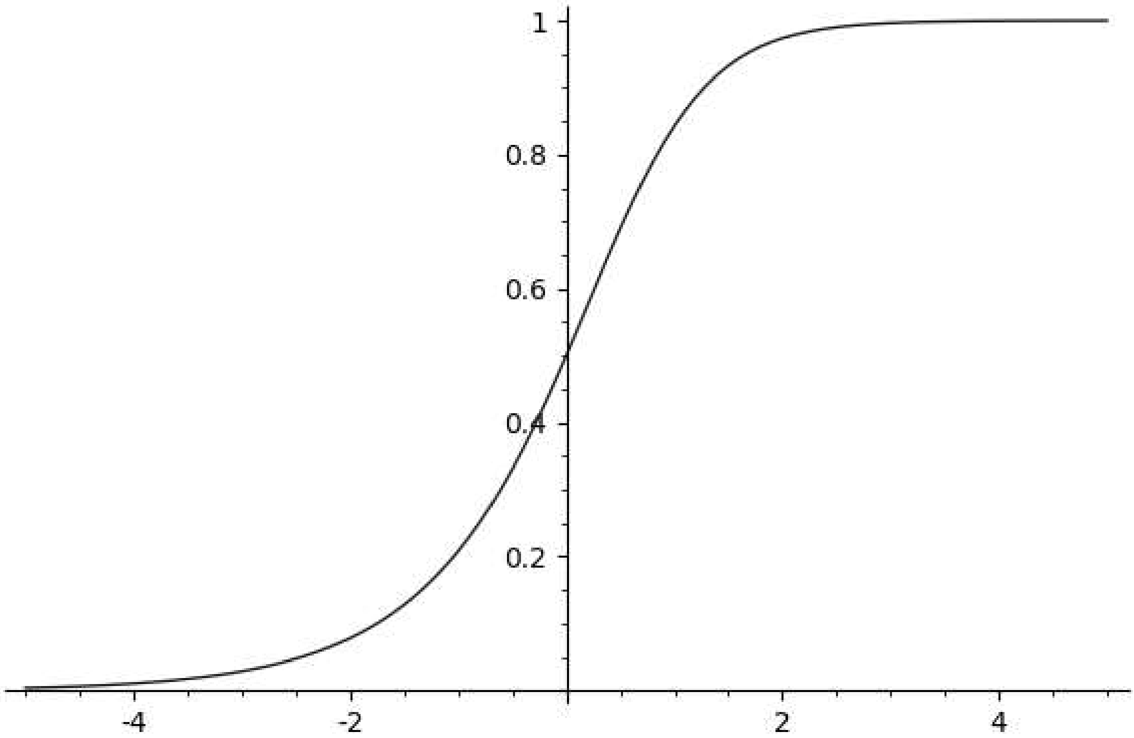
Out[14]:

```
[[-5.0, 0.003903791065104986],
 [-4.5, 0.006433631876193118],
 [-4.0, 0.01060269171808395],
 [-3.5, 0.01747228389973102],
 [-3.0, 0.02878800682685936],
 [-2.5, 0.04741101298323355],
 [-2.0, 0.07798679679354956],
 [-1.5, 0.1278641214708751],
 [-1.0, 0.2078424109397569],
 [-0.5, 0.3305423618917383],
 [0, 0.500000000000000],
 [0.5, 0.689416022585013],
 [1.0, 0.8431442383331245],
 [1.5, 0.9322733589084158],
 [2.0, 0.9730785242120609],
 [2.5, 0.9896725418081366],
 [3.0, 0.9960935743967014],
 [3.5, 0.998530305589941],
 [4.0, 0.9994481887463996],
 [4.5, 0.9997929755801175],
 [5.0, 0.9999223524467028]]
```

In [15]:

```
desolve_rk4(deq, x, [0,0.5], step=0.1, end_points= [-5,5], output='plot')
```

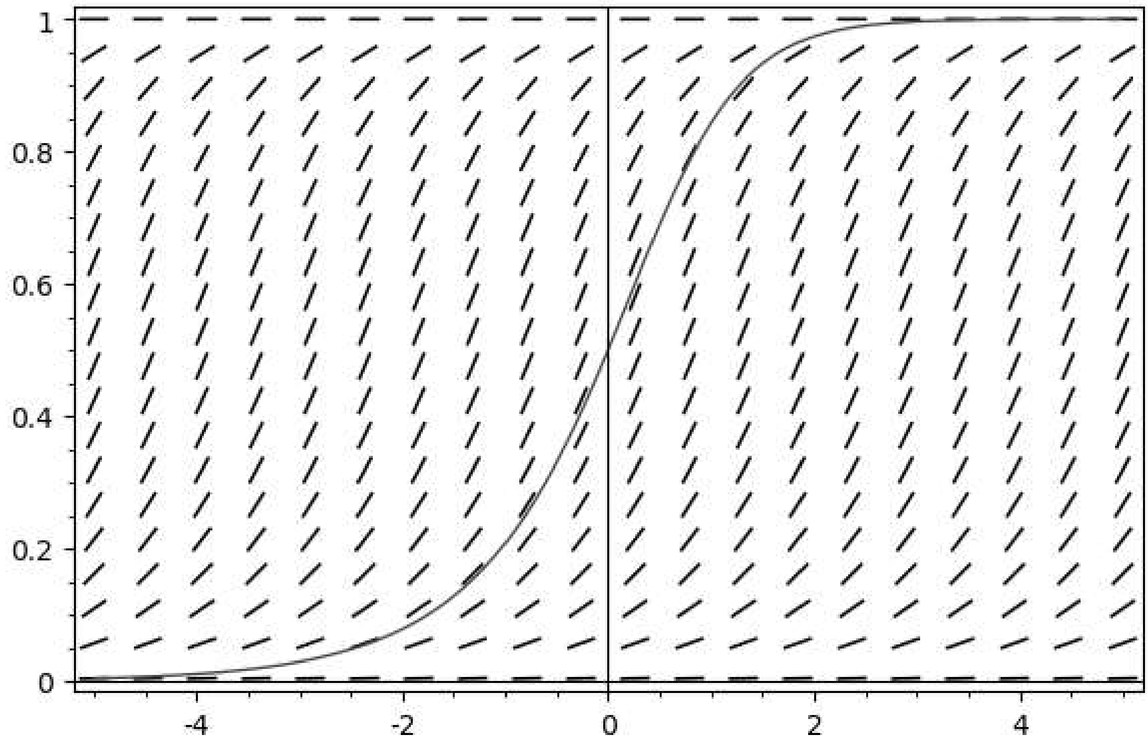
Out[15]:



In [16]:

```
desolve_rk4(deq, x, [0,0.5], step=0.1, end_points= [-5,5], output='slope_field',color='red'
```

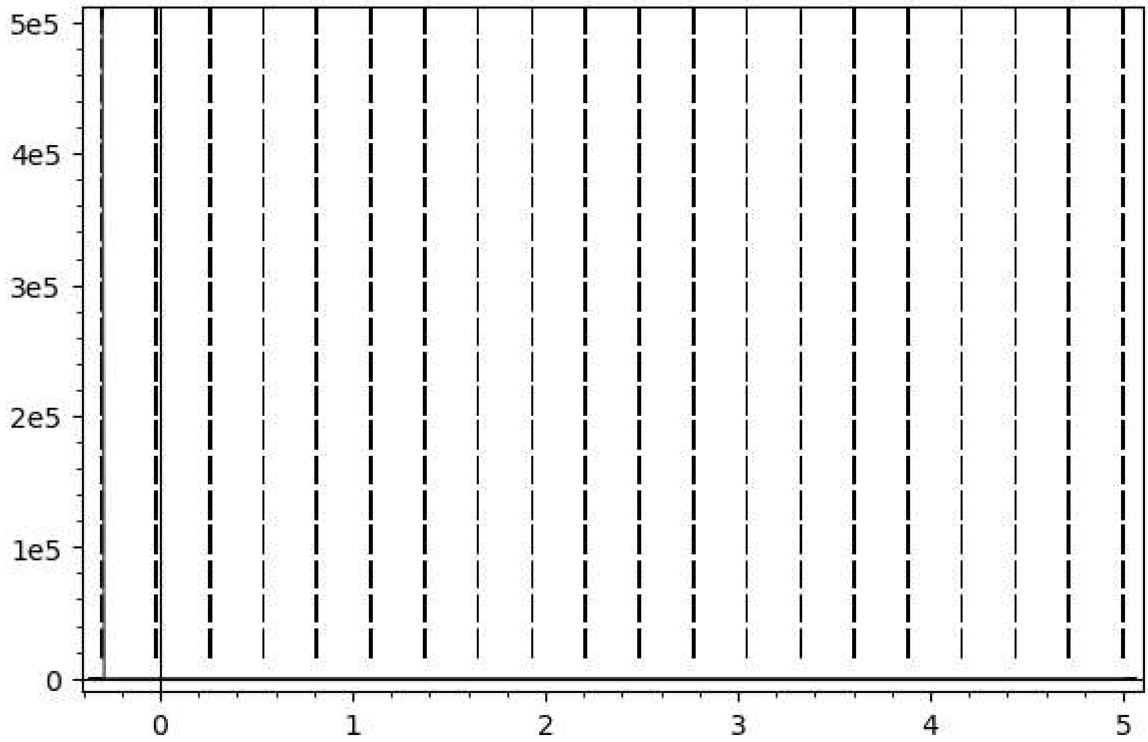
Out[16]:



In [17]:

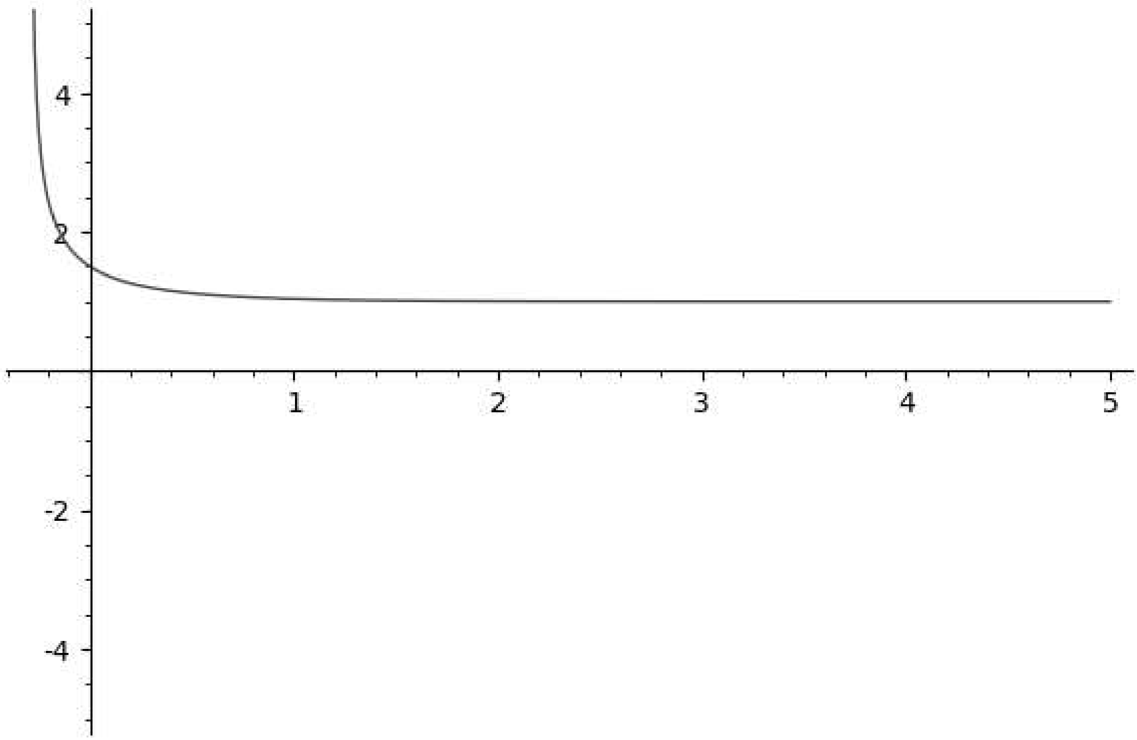
```
desolve_rk4(deq, x, [0,1.5], step=0.01, end_points= [-5,5], output='slope_field',color='rec
```

Out[17]:



In [18]:

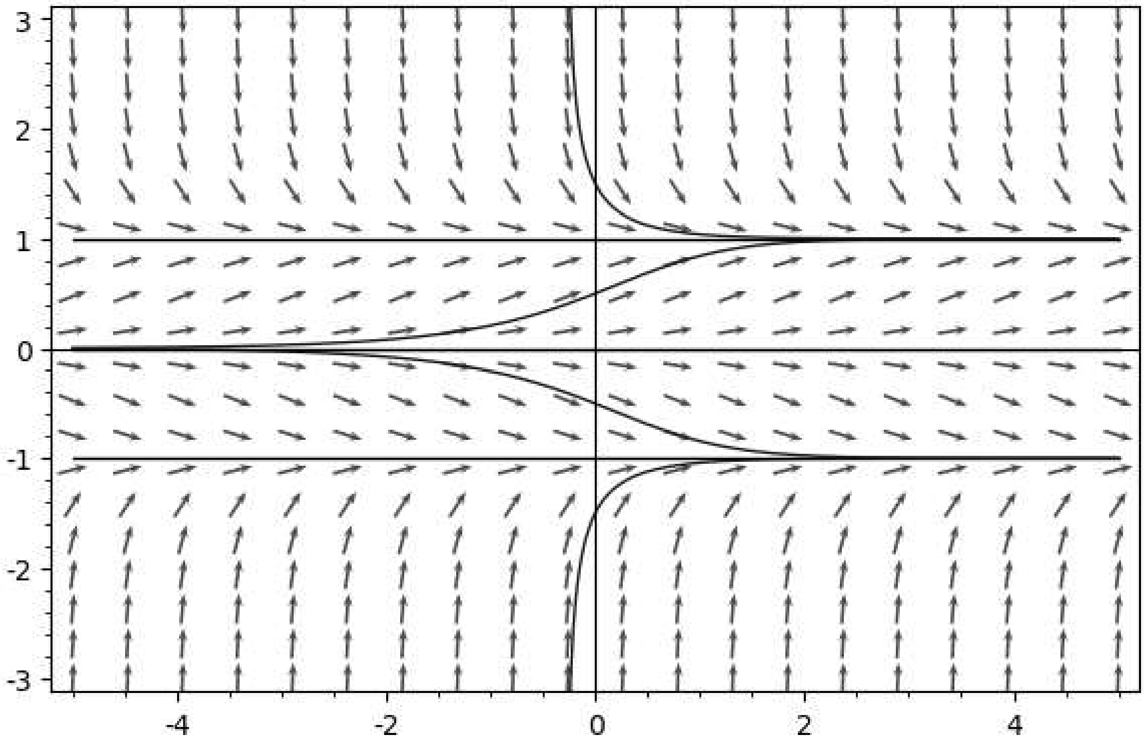
```
g=desolve_rk4(deq, x, [0,1.5], step=0.01, end_points= [-5,5], output='plot',color='red')
g.show(ymin=-5,ymax=5)
```



In cazul ecuatiei considerate avem punctele de echilibru $x_1 = -1$, $x_2 = 1$ and $x_3 = 0$, solutiile reprezentative sunt solutiile ale caror conditii initiale $x(0) < -1$, cele cu conditii initiale $-1 < x(0) < 0$, cele cu $0 < x(0) < 1$ si cele cu $x(0) > 1$. Pentru a vizualiza si solutiile de echilibru trebuiesc utilizate si conditiile initiale $x(0) = -1$, $x(0) = 0$ si $x(0) = 1$ corespunzatoare valorilor punctelor de echilibru:

In [19]:

```
sf=plot_slope_field(f(s),(t,-5,5),(s,-3,3),headaxislength=3, headlength=4,color='red')
g1=desolve_rk4(deq, x, [0,-1.5], step=0.01, end_points= [-5,5], output='plot',color='blue')
g2=desolve_rk4(deq, x, [0,-1], step=0.01, end_points= [-5,5], output='plot',color='black')
g3=desolve_rk4(deq, x, [0,-0.5], step=0.01, end_points= [-5,5], output='plot',color='blue')
g4=desolve_rk4(deq, x, [0,0], step=0.01, end_points= [-5,5], output='plot',color='black')
g5=desolve_rk4(deq, x, [0,0.5], step=0.01, end_points= [-5,5], output='plot',color='blue')
g6=desolve_rk4(deq, x, [0,1], step=0.01, end_points= [-5,5], output='plot',color='black')
g7=desolve_rk4(deq, x, [0,1.5], step=0.01, end_points= [-5,5], output='plot',color='blue')
g=sf+g1+g2+g3+g4+g5+g6+g7
g.show(ymin=-3,ymax=3)
```



Sisteme planare. Puncte de echilibru. Stabilitate

Printr-un sistem planar de ecuatii diferentiale autonome intelegem un sistem de forma:

$$\begin{cases} x' = & f_1(x, y) \\ y' = & f_2(x, y) \end{cases}$$

Solutiile constante ale sistemului planar

$$\begin{cases} x(t) \equiv & x^* \\ y(t) \equiv & y^* \end{cases}$$

se numesc **solutii de echilibru**, punctul $X^*(x^*, y^*)$ se numeste **punct de echilibru**.

Punctele de echilibru $X^*(x^*, y^*)$ sunt solutiile reale ale sistemului algebric:

$$\begin{cases} f_1(x,y) = & 0 \\ f_2(x,y) = & 0 \end{cases}$$

Cazul sistemelor liniare

In cazul sistemelor liniare:

$$\begin{cases} x' = & a_{11}x + a_{12}y \\ y' = & a_{21}x + a_{22}y \end{cases}$$

sau in forma vectoriala

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = A \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

unde

$$A = \begin{pmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{pmatrix}$$

originea $(0,0)$ este un punct de echilibru.

Criteriul de stabilitate in cazul sistemelor liniare

Punctul de echilibru $(0,0)$ este:

1. local stabil $\Longleftrightarrow \mathrm{Re}\,\lambda \leq 0$, $\forall\lambda$ valoare proprie a matricii A, egalitatea cu 0 avand loc pentru valori proprii simple;
2. asimptotic stabil $\Longleftrightarrow \mathrm{Re}\,\lambda < 0$, $\forall\lambda$ valoare proprie a matricii A;
3. instabil $\Longleftrightarrow$ nu are loc 1.

Clasificarea punctului de echilibru (0,0):

Spunem ca punctul de echilibru (0,0) este:

- de tip **nod** daca $\lambda_1, \lambda_2 \in \mathbb{R}$ si $\lambda_1 \cdot \lambda_2 > 0$;
- de tip **sa** daca $\lambda_1, \lambda_2 \in \mathbb{R}$ si $\lambda_1 \cdot \lambda_2 < 0$;
- de tip **focus** daca $\lambda_{1,2} = \alpha \pm i\beta \in \mathbb{C}$, $\alpha \neq 0$;
- de tip **centru** daca $\lambda_{1,2} = \pm i\beta \in \mathbb{C}$.

Fie sistemul liniar:

$$\begin{cases} x' = & x + y \\ y' = & x - y \end{cases}$$

In acest caz avem:

$$A = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}$$

Valorile proprii ale matricii A pot fi determinate utilizand comanda *eigenvalues*

In [20]:

```
reset()
A=matrix([[1,1],[1,-1]])
A
```

Out[20]:

```
[ 1  1]
[ 1 -1]
```

In [21]:

```
A.eigenvalues()
```

Out[21]:

```
[-1.414213562373095?, 1.414213562373095?]
```

Sagemath avertizeaza faptul ca valorile proprii nu au fost determinate exact prin semnul ?, comanda returneaza o valoare aproximativa a acestora, dar pentru analiza stabilitatii acest lucru este suficient.

Astfel, pentru acest exemplu, punctul de echilibru $(0, 0)$ este instabil deoarece una dintre valorile proprii este pozitiva, $1.414... > 0$, fiind de tip sa.

Daca dorim detereminarea exacta a valorilor proprii acest lucru se poate face prin determinarea radacinilor polinomului caracteristic al matricii A:

In [22]:

```
cp(x)=A.charpoly()
cp
```

Out[22]:

```
x |--> x^2 - 2
```

In [23]:

```
solve(cp==0,x)
```

Out[23]:

```
[x == -sqrt(2), x == sqrt(2)]
```

Pentru reprezentarea portretului fazic trebuie sa folosim comanda de rezolvare numerica *desolve_system_rk4* deoarece in general sistemele planare nu sunt rezolvabile.

Structura comenzii *desolve_system_rk4* este:

desolve_system_rk4(des, vars, ics=None, ivar=None, end_points=None, step=0.1)

INPUT:

- *des* - lista mebrilor drepti al ecuatiilor sistemului
- *vars* - lista variabilelor dependende (functiile necunoscute)
- *ivar* - (optional) trebuie specificat daca sistemul este autonom sau contine mai multi parametrii

- *ics* - lista conditiilor initiale de forma [x0,y01,y02,y03,....]
- *end_points* - capetele intervalului
  - daca *end_points* este a sau [a], integrarea se face pe intervalul min(ics[0],a) si max(ics[0],a)
  - daca *end_points* este None, este utilizat implicit end_points=ics[0]+10
  - daca *end_points* este [a,b] integrarea se face pe intervalul min(ics[0], a) and max(ics[0], b)
- *step* – (optional, default: 0.1) marimea pasului metodei (numar pozitiv)

OUTPUT:

Se returneaza o lista de forma $[t_i, x(t_i), y(t_i)]$.

Observatie. Comanda *desolve_system_rk4* nu are ca si output reprezentarea grafica. Pentru reprezentare grafica trebuie utilizata in plus comanda *list_plot*.

In [24]:

```
x,y,t=var('x,y,t')
f1(x,y)=x+y
f2(x,y)=x-y
```

In [25]:

```
desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,1,-1], ivar = t, end_points = [-1,1],
```

Out[25]:

```
[[-1.0, 2.178175582521938, -4.914760225614616],
 [-0.9, 1.925422767612532, -4.252331374129021],
 [-0.8, 1.711242398200907, -3.675090485140899],
 [-0.7000000000000001, 1.531343747591895, -3.17147355560485],
 [-0.6000000000000001, 1.382122862773177, -2.73139150871942],
 [-0.5, 1.260590365556255, -2.346028077070541],
 [-0.4, 1.164311565637939, -2.007663184267604],
 [-0.3, 1.091357685852782, -1.709518286834819],
 [-0.2, 1.0402672225, -1.445620578055556],
 [-0.1, 1.010016666666667, -1.210683333333333],
 [0, 1, -1],
 [0.1, 1.010016666666667, -0.80935],
 [0.2, 1.0402672225, -0.6349138669444444],
 [0.3, 1.091357685852782, -0.4731970848707454],
 [0.4, 1.164311565637939, -0.3209599470082735],
 [0.5, 1.260590365556255, -0.1751526540419697],
 [0.6000000000000001, 1.382122862773177, -0.0328542168269349],
 [0.7000000000000001, 1.531343747591894, 0.1087860604210598],
 [0.8, 1.711242398200906, 0.2526056887390845],
 [0.9, 1.925422767612531, 0.4014858389039571],
 [1.0, 2.178175582521938, 0.5584090605707386]]
```

Se observa ca *desolve_system_rk4* returneaza o lista de forma $[t, x(t), y(t)]$ pentru $t \in [a, b]$, unde $[a, b]$ este intervalul pe care se face integrarea, specificat prin optiunea *end_points*.

Pentru a reprezenta orbita corespunzatoare solutiei ce satisface conditia initiala $x(0) = x0, y(0) = y0$ data prin optiunea $[0, x0, y0]$ vom reprezenta punctele $[x(t), y(t)]$ utilizand *list_plot* cu optiunea *plotjoined=True*.

Mai intai generam aceasta lista de valori extragand-o din lista returnata de *desolve_system_rk4*, apoi aplicam *list_plot*.

In [26]:

```
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,1,-1], ivar = t, end_points = [-1,1
XY=[ [j, k] for i,j,k in P]
XY
```

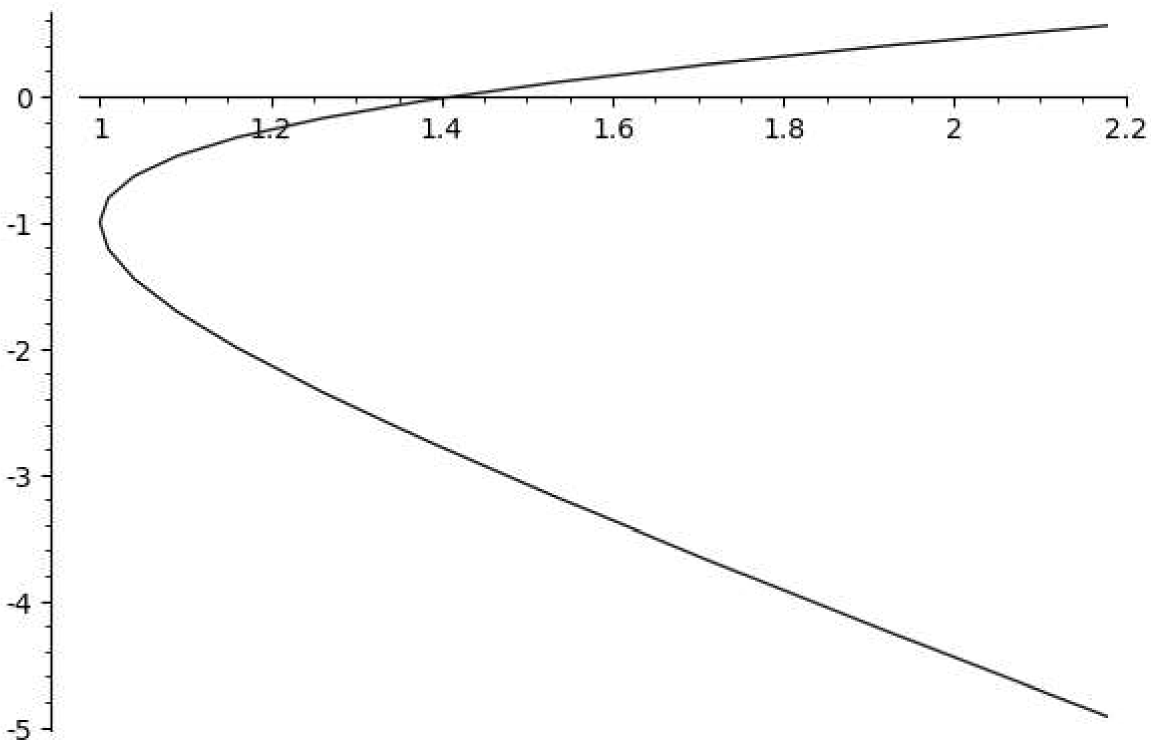
Out[26]:

```
[[2.178175582521938, -4.914760225614616],
 [1.925422767612532, -4.252331374129021],
 [1.711242398200907, -3.675090485140899],
 [1.531343747591895, -3.17147355560485],
 [1.382122862773177, -2.73139150871942],
 [1.260590365556255, -2.346028077070541],
 [1.164311565637939, -2.007663184267604],
 [1.091357685852782, -1.709518286834819],
 [1.0402672225, -1.445620578055556],
 [1.010016666666667, -1.210683333333333],
 [1, -1],
 [1.010016666666667, -0.80935],
 [1.0402672225, -0.6349138669444444],
 [1.091357685852782, -0.4731970848707454],
 [1.164311565637939, -0.3209599470082735],
 [1.260590365556255, -0.1751526540419697],
 [1.382122862773177, -0.0328542168269349],
 [1.531343747591894, 0.1087860604210598],
 [1.711242398200906, 0.2526056887390845],
 [1.925422767612531, 0.4014858389039571],
 [2.178175582521938, 0.5584090605707386]]
```

In [27]:

```
list_plot(XY,plotjoined=True, color='blue')
```

Out[27]:



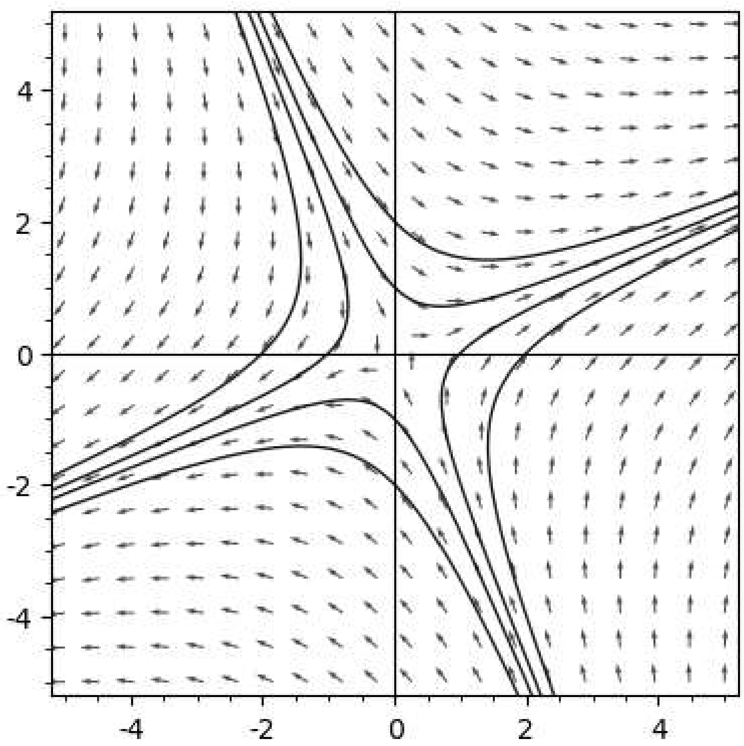
Daca dorim reprezentarea mai multor orbite trebuie generat graficul pentru fiecare orbita corespunzatoare

Daca dorim reprezentarea mai multor orbite trebuie generat graficul pentru fiecare orbita corespunzatoare setului de conditii initiale ales, adica trebuiesc executati pasii anteriori pentru fiecare orbita. In plus, pentru a vizualiza sensul de parcurgere al acestora trebuie generat si campul de directii prin utilizarea comenzii *plot_vector_field*.

Pentru sistemul considerat sa reprezentam campul de directii si orbitele corespunzatoare punctelor $(1, 0)$, $(2, 0)$, $(-1, 0)$, $(-2, 0)$, $(0, 1)$, $(0, 2)$, $(0, -1)$, $(0, -2)$:

In [28]:

```
n=sqrt(f1(x,y)^2+f2(x,y)^2)
g=plot_vector_field( [f1(x,y)/n,f2(x,y)/n], [x,-5,5], [y,-5,5],color='red' ,aspect_ratio =
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,1,0], ivar = t, end_points = [-5,5]
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,2,0], ivar = t, end_points = [-5,5]
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,-1,0], ivar = t, end_points = [-5,5
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,-2,0], ivar = t, end_points = [-5,5
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,0,1], ivar = t, end_points = [-5,5]
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,0,2], ivar = t, end_points = [-5,5]
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,0,-1], ivar = t, end_points = [-5,5
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,0,-2], ivar = t, end_points = [-5,5
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
g.show(xmin=-5,xmax=5,ymin=-5,ymax=5)
```



Cazul sistemelor neliniare

In cazul sistemelor neliniare de forma:

$$\begin{cases} x' = & f_1(x,y) \\ y' = & f_2(x,y) \end{cases}$$

mai intai determinam punctele de echilibru $X^*(x^*, y^*)$ prin determinarea solutiilor reale ale sistemul algebric:

$$\begin{cases} f_1(x,y) = & 0 \\ f_2(x,y) = & 0 \end{cases}$$

Stabilitatea punctului de echilibru se obtine prin aplicarea *Teoremei stabilitatii in prima aproximatie*.

Teorema stabilitatii in prima aproximatie

Fie $X^*(x^*, y^*)$ un punct de echilibru al sistemului neliniar.

- Daca $\mathrm{Re}(\lambda) < 0$ pentru orice valoare proprie λ a matricii $J_f(X^*) = J_f(x^*, y^*)$ atunci X^* este local asimptotic stabil.
- Daca exista o valoare proprie λ a matricii $J_f(X^*) = J_f(x^*, y^*)$ cu $\mathrm{Re}(\lambda) > 0$ atunci X^* este instabil, unde $J_f(X)$ este matricea jacobiana a functiei vectoriale $f = (f_1, f_2)$

Fie sistemul neliniar:

$$\begin{cases} x' = & x \cdot (1 - \frac{1}{2}x - y) \\ y' = & y \cdot (x - 1 - \frac{1}{2}y) \end{cases}$$

In [29]:

```
reset()
x,y,t=var('x,y,t')
f1(x,y)=x*(1-1/2*x-y)
f2(x,y)=y*(x-1-1/2*y)
```

In [30]:

```
EquilP=solve([f1(x,y)==0,f2(x,y)==0],x,y)
EquilP
```

Out[30]:

```
[[x == 0, y == 0], [x == 2, y == 0], [x == 0, y == -2], [x == (6/5), y ==
(2/5)]]
```

Generam matricea jacobiana a functiei vectoriale $f = (f_1, f_2)$ utilizand comanda *jacobian*:

In [31]:

```
jacobian((f1(x,y),f2(x,y)), (x,y))
```

Out[31]:

```
[-x - y + 1          -x]
[         y  x - y - 1]
```

In [32]:

```
J=jacobian((f1(x,y),f2(x,y)), (x,y))
J
```

Out[32]:

```
[-x - y + 1          -x]
[         y  x - y - 1]
```

Pentru fiecare punct de echilibru $X^*(x^*, y^*)$ evaluam $J_f(x^*, y^*)$, calculam valorile proprii corespunzatoare si aplicam Teorema stabilitatii in prima aproximatie.

Pentru punctul de echilibru $(0, 0)$ obtinem:

In [33]:

```
J(x=0,y=0)
```

Out[33]:

```
[ 1  0]
[ 0 -1]
```

In [34]:

```
J(x=0,y=0).eigenvalues()
```

Out[34]:

```
[-1, 1]
```

In cazul punctului de echilibru $(0, 0)$ avem $\lambda_1 = -1$ si $\lambda_2 = 1$, cum $\lambda_2 > 0$ atunci $(0, 0)$ este *instabil*, el fiind de tip *sa* (si in cazul sistemelor neliniare se pastreaza clasificarea lui (0,0) din cazul sistemelor liniare).

Pentru al doilea punct de echilibru obtinem:

In [35]:

```
J(x=2,y=0)
```

Out[35]:

```
[-1 -2]
[ 0  1]
```

In [36]:

```
J(x=2,y=0).eigenvalues()
```

Out[36]:

```
[-1, 1]
```

Observam ca avem aceleasi valori proprii ca in cazul punctului $(0, 0)$, deci si punctul de echilibru $(2, 0)$ este *instabil* de tip *sa*.

Pentru cel de al treilea punct de echilibru obtinem:

In [37]:

```
J(x=0,y=-2)
```

Out[37]:

```
[ 3  0]
[-2  1]
```

In [38]:

```
J(x=0,y=-2).eigenvalues()
```

Out[38]:

```
[1, 3]
```

Pentru $(0, -2)$ avem $\lambda_1 = 1$ si $\lambda_2 = 3$, ambele valori proprii $\lambda_{1,2} > 0$ deci $(0, -2)$ este *instabil* de tip *nod*.

Pentru ultimul punct de echilibru obtinem:

In [39]:

```
J(x=6/5,y=2/5)
```

Out[39]:

```
[-3/5 -6/5]
[ 2/5 -1/5]
```

In [40]:

```
J(x=6/5,y=2/5).eigenvalues()
```

Out[40]:

```
[-1/5*I*sqrt(11) - 2/5, 1/5*I*sqrt(11) - 2/5]
```

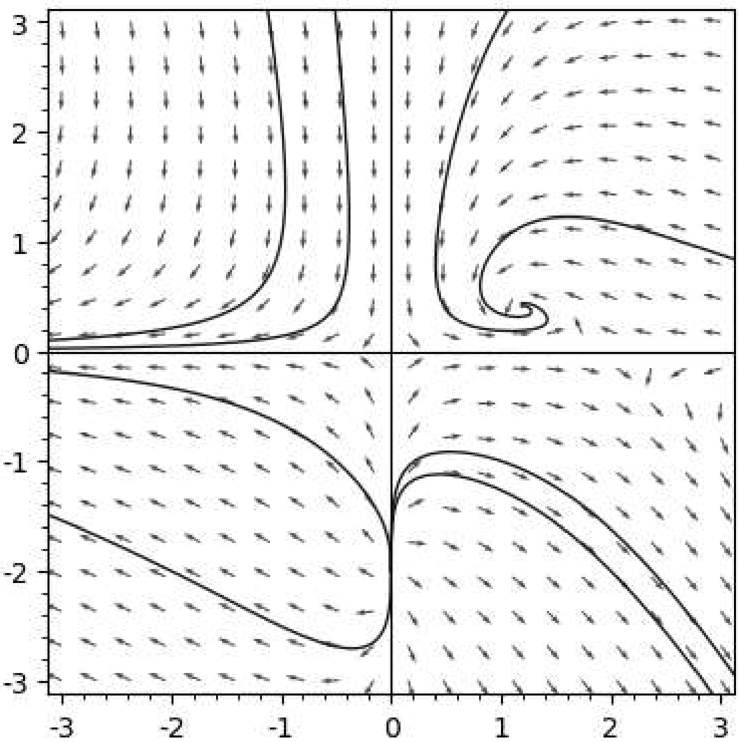
Pentru punctul de echilibru $(6/5, 2/5)$ avem $\lambda_{1,2} = -\frac{2}{5} \mp i\frac{\sqrt{11}}{5}$ cu $\mathrm{Re}(\lambda_{1,2}) = -\frac{2}{5} < 0$, deci punctul de echilibru $(6/5, 2/5)$ este *local asimptotic stabil* de tip *focus*.

Pentru reprezentarea portretului fazic trebuie aleasa o fereastra de reprezentare $[a, b] \times [c, d]$ care sa contina toate punctele de echilibru si trebuiesc alese cateva orbite aflate in vecinatatea acestor puncte de echilibru.

In cazul sistemului considerat putem considera fereastra de reprezentare $[-3, 3] \times [-3, 3]$

In [41]:

```
n=sqrt(f1(x,y)^2+f2(x,y)^2)
g=plot_vector_field( [f1(x,y)/n,f2(x,y)/n], [x,-3,3], [y,-3,3],color='red' ,aspect_ratio =
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,1,1], ivar = t, end_points = [-10,1
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,1,3], ivar = t, end_points = [-10,1
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,-1,1], ivar = t, end_points = [-10,
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,-0.5,3], ivar = t, end_points = [-1
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,-0.5,-1], ivar = t, end_points = [-
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,-2,-2], ivar = t, end_points = [-10
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,1,-1], ivar = t, end_points = [-10,
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
P=desolve_system_rk4([f1(x,y),f2(x,y)], [x,y], ics = [0,2,-2], ivar = t, end_points = [-10,
XY=[ [j, k] for i,j,k in P]
g=g+list_plot(XY,plotjoined=True, color='blue')
g.show(xmin=-3,xmax=3,ymin=-3,ymax=3)
```



In [ ]:

```

```

In [ ]:

```

```