Autonomní systémy

Robert Mařík

27. listopadu 2010

1 Autonomní systém v rovině

Autonomní systém definujeme tak, že zadáme pravé strany, tj. funkce f a g z předpisu $\begin{cases} x' = f(x,y) \\ y' = g(x,y) \end{cases}$. Pro jednoduchost budeme funkci f označovat `de_x` a funkci g budeme označovat `de_y`. Současně s tím nadefinujeme v proměnných `XMIN`, `XMAX`, `YMIN` a `YMAX` meze pro kreslení.

Sage code

```
x,y,t = var('x y t')

# zapoznamkujte vsechny systemy krome toho, ktery chcete studovat

de_x , de_y, XMIN, XMAX, YMIN, YMAX = x*(1-2*x-y) , y*(2-3*x-3*y), 0, 1, 0, 1.5  # konkurence
# de_x , de_y, XMIN, XMAX, YMIN, YMAX = y , -x , -2, 2, -2, 2  # stred
[de_x, de_y]
```

$[-(2\,x+y-1)x, -(3\,x+3\,y-2)y]$

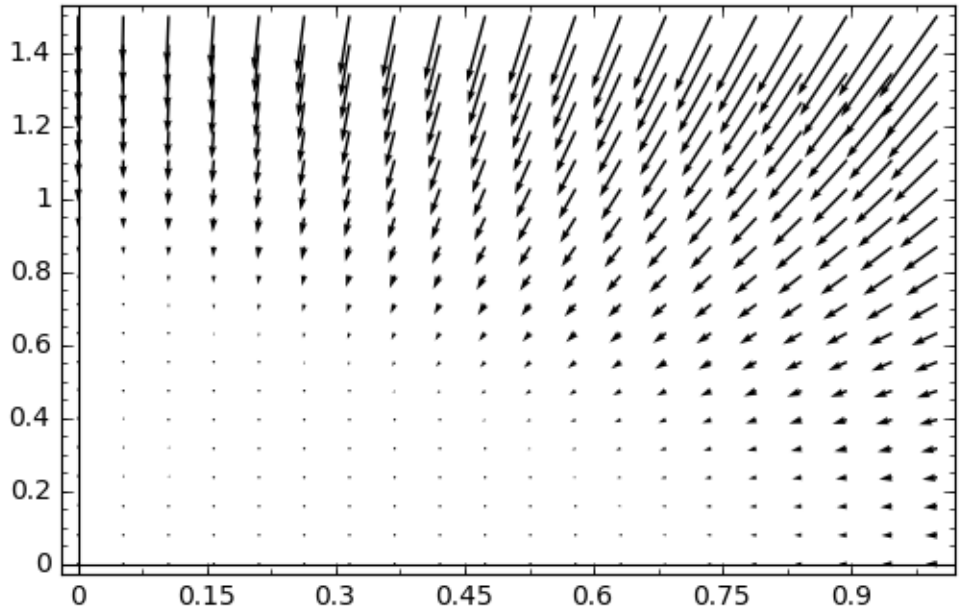
2 Numerické řešení a kreslení trajektorií

Směrové pole nakreslíme tak, že nakreslíme vektorové pole příkazem `plot_vector_field`.

Sage code

```
plot_vector_field((de_x,de_y),(x,XMIN,XMAX),(y,YMIN,YMAX))
```

[0]Podporováno grantem FRVŠ 131/2010.



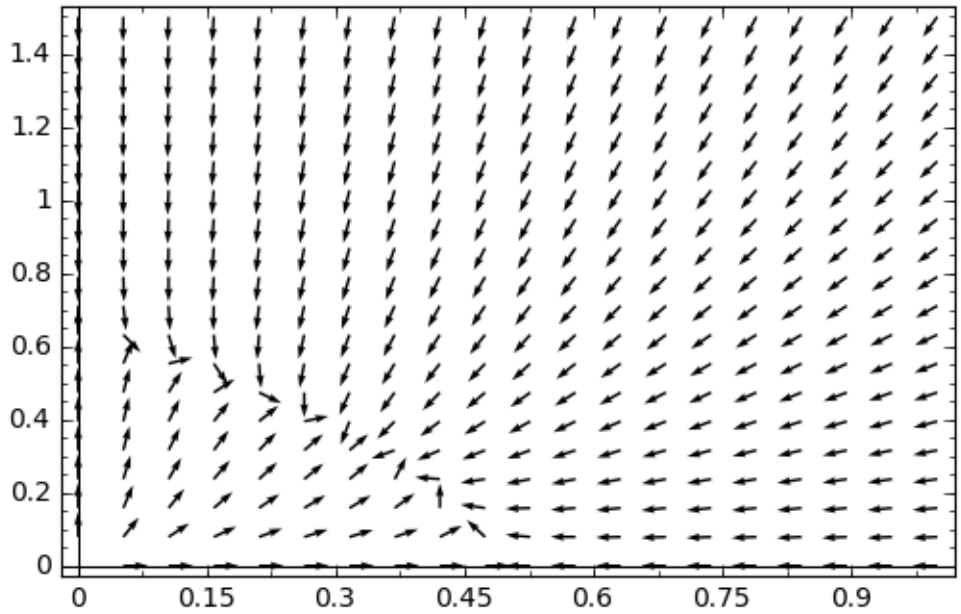
Vidíme, že v obrázku nemají všechny vektory stejnou délku, což působí rušivě. Toto je důsledek faktu, že funkce `plot_vector_field` je určena pro kreslení vektorových polí a ne pro kreslení směrových polí, kdy nám na délce vektoru nezáleží. Proto použijeme následující obrat: před kreslením vektorového pole všechny vektory vydělíme jejich normou tak, aby měly jednotkovou délku.

Sage code

```
def normalizuj(v):
    v_ = vector(v)  # prevod n-tice na vektor
    nv_ = norm(v_)  # delka vektoru
    if nv_ == 0:    # test zda nebudeme delit nulou
        return (0,0)
    else:
        return v_/nv_  # vektor vydeleny svou delkou
```

Sage code

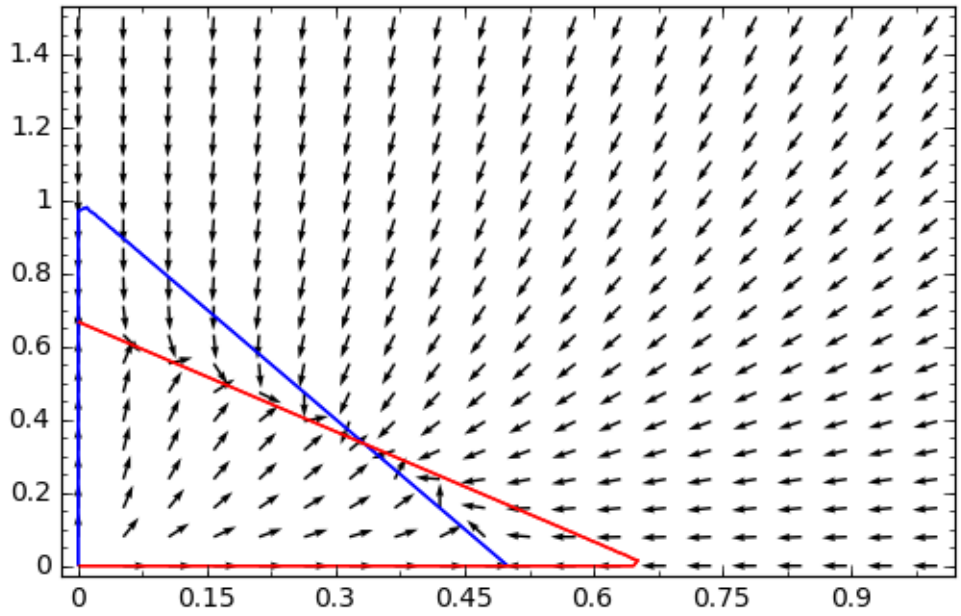
```
P = plot_vector_field(normalizuj((de_x,de_y)),(x,XMIN,XMAX),(y,YMIN,YMAX))
P
```



Přidáme k obrázku nulkliny. Je-li pravá strana první rovnice `de_x`, má x-nulklina rovnici `de_x == 0`. K nakreslení řešení takové rovnice slouží příkaz `implicit_plot`.

Sage code

```
P = P + implicit_plot(de_x,(x,XMIN,XMAX),(y,YMIN,YMAX), plot_points=100, cmap=['blue'])
P = P + implicit_plot(de_y,(x,XMIN,XMAX),(y,YMIN,YMAX), plot_points=100, cmap=['red'])
P
```



Vypočítáme numericky řešení odpovídající dvěma počátečním podmínkám a nakreslíme trajektorie. Řešením je uspořádaná trojice $[t, x(t), y(t)]$, při kreslení trajketorií používáme jenom poslední dvě komponenty. Numerické řešení zajišťuje funkce `desolve_system_rk4`.

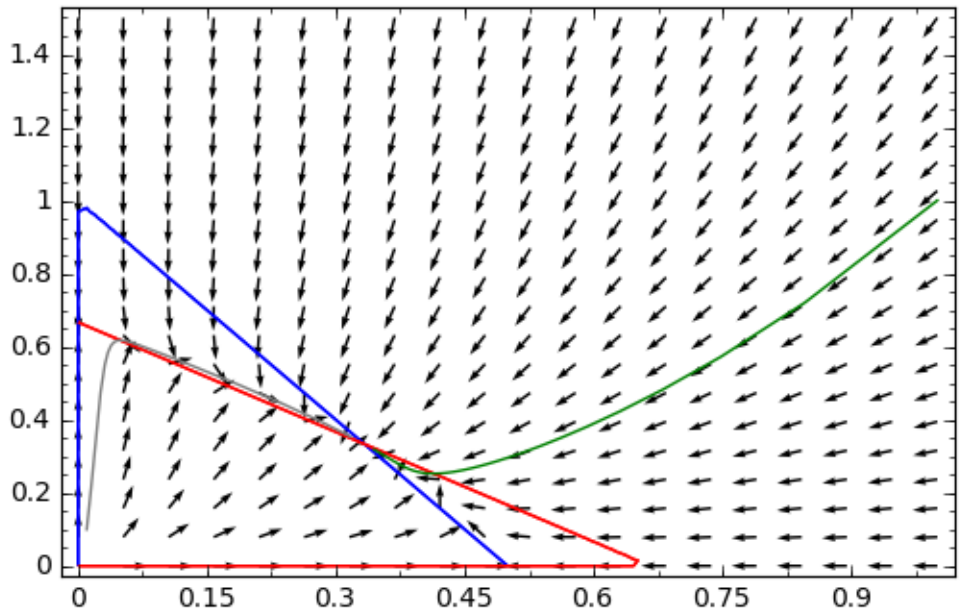
Sage code

```
P = P + list_plot([ [j,k] for i,j,k in desolve_system_rk4([de_x,de_y],[x,y],\
 ics=[0,1,1],ivar=t)], plotjoined=True,rgbcolor='green')
P = P + list_plot([ [j,k] for i,j,k in desolve_system_rk4([de_x,de_y],[x,y],\
 ics=[0,.01,.1],ivar=t,end_points=20)], plotjoined=True,rgbcolor='gray')
```

Zobrazíme obrázek se směrovým polem, nulklinami a trajektoriemi.

Sage code

```
P
```



3 Stacionární body a chování autonomního systému v okolí těchto bodů

Stacionární body vypočteme tak, že řeším soustavu rovnic, kdy se pravé strany autonomního systému rovnají nule.

Sage code

```
stacionarni_body=solve([de_x,de_y], [x,y], solution_dict=True)
sb=[]
for s in stacionarni_body: sb.append([x==s[x],y==s[y]])
sb
```

$$\left[[x=0, y=0], \left[x=\left(\frac{1}{2}\right), y=0\right], \left[x=0, y=\left(\frac{2}{3}\right)\right], \left[x=\left(\frac{1}{3}\right), y=\left(\frac{1}{3}\right)\right]\right]$$

Jacobiho matici určíme podle definice výpočtem čtyř parciálních derivací. V programu Sage na to máme přímo příkaz `jacobian`.

Sage code

```
jacobiho_matice=jacobian((de_x,de_y),(x,y))
jacobiho_matice
```

$$\begin{pmatrix} -4\,x-y+1 & -x \\ -3\,y & -3\,x-6\,y+2 \end{pmatrix}$$

V cyklu probereme jednotlivé stacionární body a v kadém bodě samostatně určíme Jacobiho matici a její vlastní čísla. Tím získáme informaci o typu stacionárních bodů.

Sage code

```
A=[["Stacionární bod","Jacobiho matice","Vlastní hodnoty"]]
for s in stacionarni_body:
    A.append([[s[x],s[y]], jacobiho_matice(s),\
    [a.real().n(5)+I*a.imag().n(5) for a in jacobiho_matice(s).eigenvalues()]])
html.table(A,header=True)
```

Stacionární bod	Jacobiho matice	Vlastní hodnoty
$[0, 0]$	$\begin{pmatrix} 1 & 0 \\ 0 & 2 \end{pmatrix}$	$[1.0, 2.0]$
$\left[\frac{1}{2}, 0\right]$	$\begin{pmatrix} -1 & -\frac{1}{2} \\ 0 & \frac{1}{2} \end{pmatrix}$	$[0.50, -1.0]$
$\left[0, \frac{2}{3}\right]$	$\begin{pmatrix} \frac{1}{3} & 0 \\ -2 & -2 \end{pmatrix}$	$[0.33, -2.0]$
$\left[\frac{1}{3}, \frac{1}{3}\right]$	$\begin{pmatrix} -\frac{2}{3} & -\frac{1}{3} \\ -1 & -1 \end{pmatrix}$	$[-1.4, -0.25]$

Pokusíme se výpočet automatizovat. Nadefinujeme funkci, do které stačí vložit pravé strany autonomních systémů a tato funkce nalezne automaticky stacionární body, Jakobiho matici v těchto bodech a vlastní čísla. Funkci navíc vylepšíme tak, aby autoamticky pracovala pouze s těmi řešeními rovnice, které jsou reálnými čísly.

Sage code

```
def as_tabulka(de_x,de_y):
    # Tisk zadani pro kontrolu
    html(r"Autonomní systém $\left\{\begin{aligned}x'&=%s \\y'&=%s \end{aligned}\right.$."\
       %(latex(de_x),latex(de_y)))

    # nalezeni stacionernich bodu a jakobianu
```

```
    stacionarni_body=solve([de_x,de_y], [x,y], solution_dict=True)
    jacobiho_matice=jacobian((de_x,de_y),(x,y))

    A=[["Stacionární bod","Jacobiho matice","Vlastní hodnoty"]]

    # cyklus pres vsechny stacionarni body
    for s in stacionarni_body:
        # budeme pracovat pouze s resenimi, ktera jsou realna cisla,
        # tj ktera maji nulou imaginarni cast
        if s[x].imag() == 0 and s[y].imag() == 0:
            A.append([[s[x],s[y]], jacobiho_matice(s),\
            [a.real().n(20)+I*a.imag().n(20) for a in jacobiho_matice(s).eigenvalues()]])

    # vytiskneme tabulku
    html.table(A,header=True)
```

Nyní je snadné prozkoumat například sacionární systém $\begin{cases} x' = x + 2y \\ y' = (y+2)(x+1)y \end{cases}$.

Sage code

```
as_tabulka(x+2*y,y*(y+2)*(x+1))
```

Autonomní systém $\begin{cases} x' = x + 2\,y \\ y' = (y+2)(x+1)y \end{cases}$.

Stacionární bod	Jacobiho matice	Vlastní hodnoty
$[0, 0]$	$\begin{pmatrix} 1 & 2 \\ 0 & 2 \end{pmatrix}$	$[1.0000, 2.0000]$
$\left[-1, \frac{1}{2}\right]$	$\begin{pmatrix} 1 & 2 \\ \frac{5}{4} & 0 \end{pmatrix}$	$[-1.1583, 2.1583]$
$[4, -2]$	$\begin{pmatrix} 1 & 2 \\ 0 & -10 \end{pmatrix}$	$[-10.000, 1.0000]$