Systems of differential equations using the Scatterplot3d

Henri Makika[*]

Junho 26, 2019

Contents

[*]University of Campinas, São Paulo. Email : hd.makika@gmail.com

Système linéaire d'équations différentielles

La simulation de systèmes (linéaire ou non linéaire) suit le même format que celui adopté pour la simulation d'équations différentielles. Les excercices que nous traitons ici sont tirés de Gandolfo, 1991 chapitres 11, 12, 14 et 18 et dans Shone, 2002 chapitre 2. Pour la résolution numérique:

1. Nous utilisons le paquet *deSolve*;

i. Nous définissons une fonction qui spécifie les équations pour chaque taux de changement;
ii. Ensuite, nous initialisons les paramètres de la simulation, y compris la période; iii.Enfin, nous utilisons la commande *ode* pour la résolution numérique.

Par exemple, il s'agit du système à deux variables, où la matrice de coefficients est donnée par:

$$A = \begin{bmatrix} 1 & -1 \\ 5 & -3 \end{bmatrix}; \qquad Z(0) = \begin{bmatrix} x(0) \\ y(0) \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

2. Chargement de paquet *deSolve* et création de la fonction dérivée :

```
library(deSolve)
library(ggplot2)
library(tidyr)

linsis <- function (time, y, parms) {

                dy1 <- 1*y[1] +-1*y[2] + 1
                dy2 <- 5*y[1] +-3*y[2] + 2
                list( c(dy1, dy2)  )

} # end
```

Nous initialisons ensuite les paramètres:

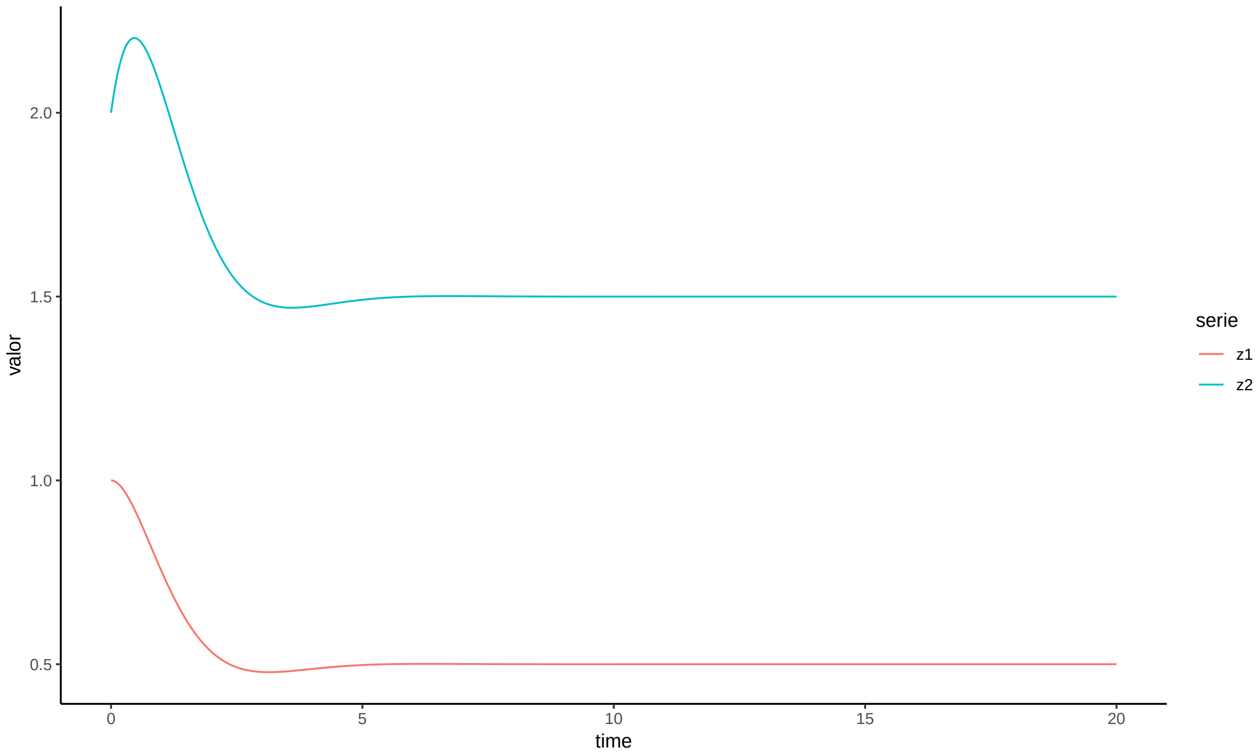
```
y_ini <- c(z1 = 1, z2 = 2)

times <- seq(0, 20, .01)
```

A présent, nous exécutons le solver :

```
out <- ode (times = times, y = y_ini, func = linsis, parms = NULL)

dados = as.data.frame(out)
head(dados)
```

```
##   time        z1       z2
## 1 0.00 1.0000000 2.000000
## 2 0.01 0.9999505 2.009851
## 3 0.02 0.9998030 2.019406
## 4 0.03 0.9995594 2.028669
## 5 0.04 0.9992215 2.037643
## 6 0.05 0.9987915 2.046334
```

```
dados_tidy = gather(dados, -time, key = "serie", value = "valor")

ggplot(dados_tidy, aes(x = time, y = valor, color = serie)) +

  geom_line() +

  theme_classic()
```



Système non linéaire d'équations différentielles

L'exemple est tiré de système de Rossler :

$$y_1' = -y_2 - y_3$$
$$y_2' = y_1 + a * y_2$$
$$y_3' = b + y_3 * (y_1 - c)$$

Pour $y_1 = y_2 = y_3 = 1$; paramètres $a = -.2,\ b = 0.2,\ c = 5$ e $t \in [0, 100]$

```
nlinsis <- function (time, y, parms) {

        a = parms[1]
        b = parms[2]
        c = parms[3]

        dy1 = -y[2] - y[3]
        dy2 = y[1] + a*y[2]
        dy3 = b + y[3]*(y[1]-c)

        dy = c(dy1, dy2, dy3)

        list( dy  )


} # end
```

Ainsi, nous définisons les paramètres du modèle :

```r
y_ini <- c(y1 = 1, y2 = 1.5, y3 = 1)

times <- seq(0, 100, .1)

parametros <- c(a = 0.2, b = 0.2, c = 5)
```

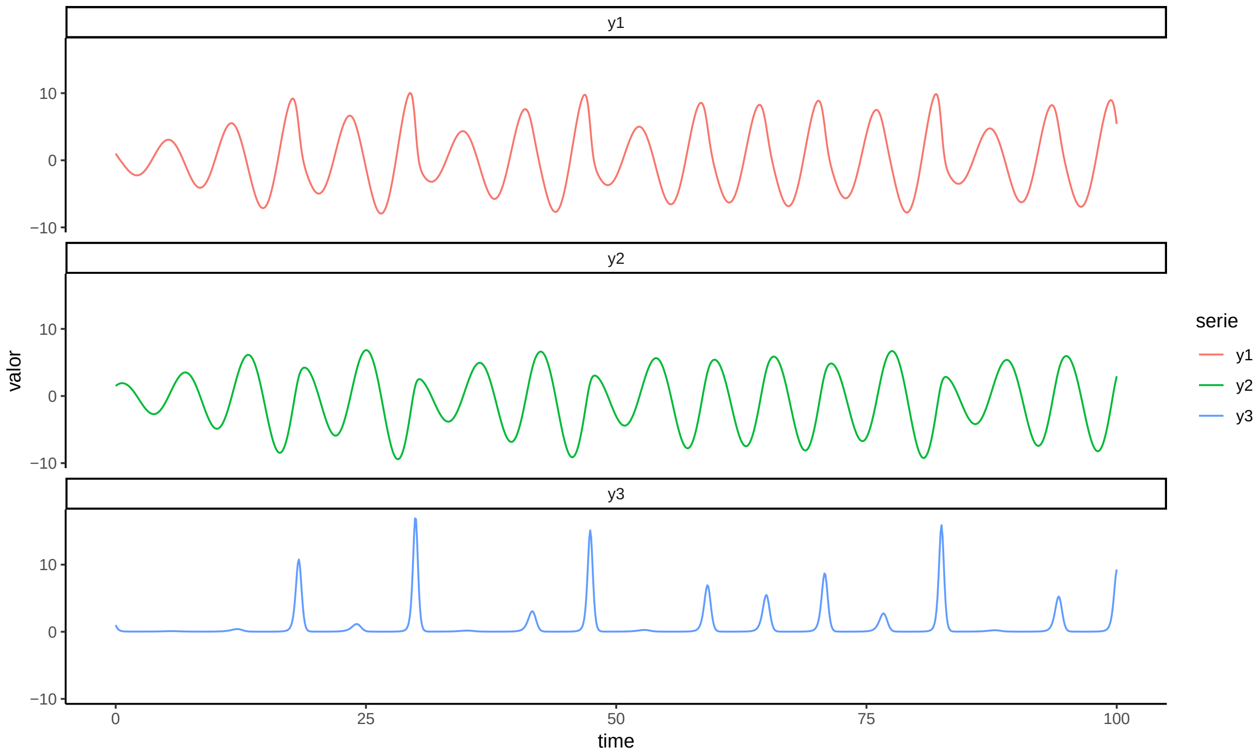
Simulation pour `ODE`:;

```r
out2 <- ode (times = times, y =y_ini, func = nlinsis, parms=parametros)

head(out2)
```

```
##      time          y1       y2        y3
## [1,]  0.0  1.0000000 1.500000 1.0000000
## [2,]  0.1  0.7608725 1.619102 0.6786016
## [3,]  0.2  0.5378868 1.717335 0.4553487
## [4,]  0.3  0.3245981 1.795560 0.3043619
## [5,]  0.4  0.1168547 1.854129 0.2045893
## [6,]  0.5 -0.0876603 1.893072 0.1399901
```

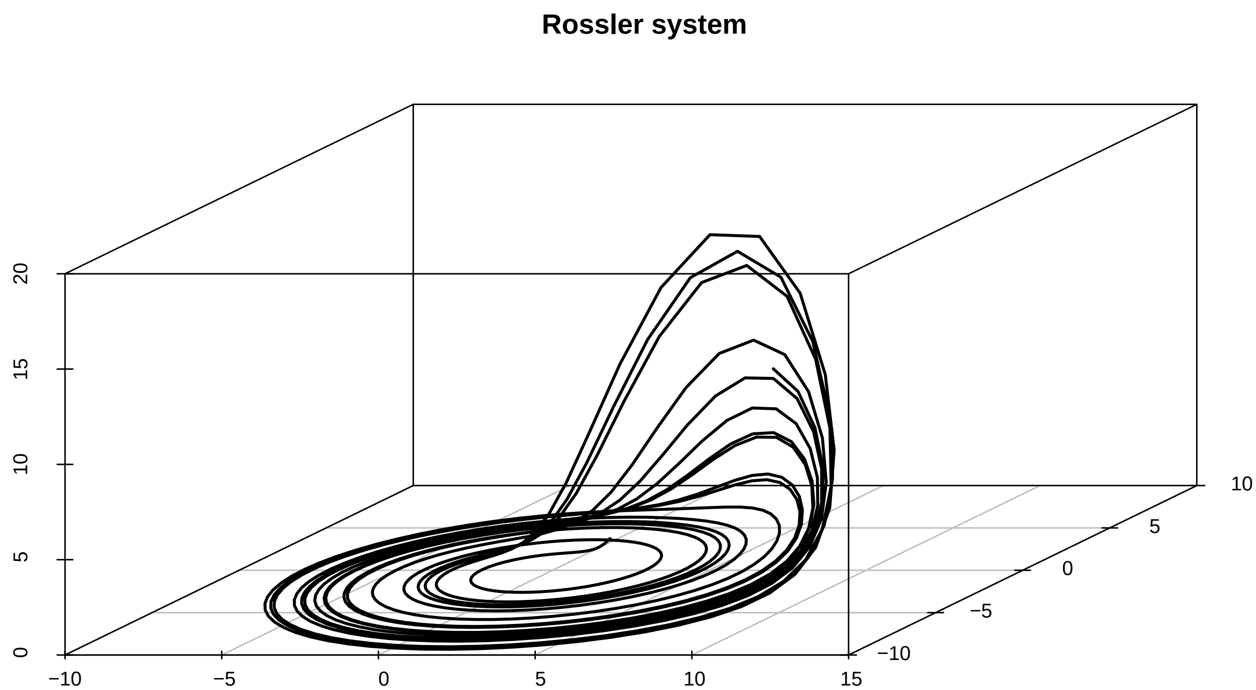
Visualisons notre série en utilisant la fonction ggplot :

```r
dados2 = as.data.frame(out2)

dados_tidy2 = gather(dados2, -time, key = "serie", value = "valor")

ggplot(dados_tidy2, aes(x = time, y = valor, color = serie)) +

  geom_line() +

  facet_wrap(~serie, nrow=3) +

  theme_classic()
```



The Rössler attractor peut être représenté sous forme :

```
library(scatterplot3d)


scatterplot3d(out2[,-1], type = "l", lwd = 2, xlab = "",
              ylab = "", zlab = "", lty.hide = 1, main = "Rossler system")
```



Diagrammes de stabilité et diagramme de phase

Cas d'une variable continue

Lorsque nous avons une équation différentielle autonome d'ordre 1, l'image de phase est automatique car nous n'avons qu'une fonction et $y' = f(y)$ à tracer.

Pour les équations d'ordre supérieur, nous devons transformer l'équation en un système à deux variables. Par exemple, soit l'équation différentielle :

$$x'' + x' + x = 0$$

Pour analyser la stabilité du système, nous pouvons suivre la suggestion de Gandolfo et réécrire l'équation différentielle telle que :

$$x'' + x' + x = 0 \Rightarrow x'' + f(x, x') = 0 \tag{1}$$

avec $f(x, x') = x' + x$. Ou soit,

$$x'' = -f(x, x') \tag{2}$$

Seja

$$\begin{aligned} x_1 &= x \\ x_2 &= x' \end{aligned}$$

Notez que le portrait de phase sera donné par les coordonnées $(x_1,\ x_2)$. Par conséquent, dériver et utiliser (2):

$$\begin{aligned} x_1' &= x' = x_2 \\ x_2' &= x'' = -f(x_1, x_2) \end{aligned}$$

Finalement, on utilise (1) dans le système intérieur :

$$\begin{aligned} x_1' &= x_2 \\ x_2' &= -x_2 - x_1 \end{aligned}$$

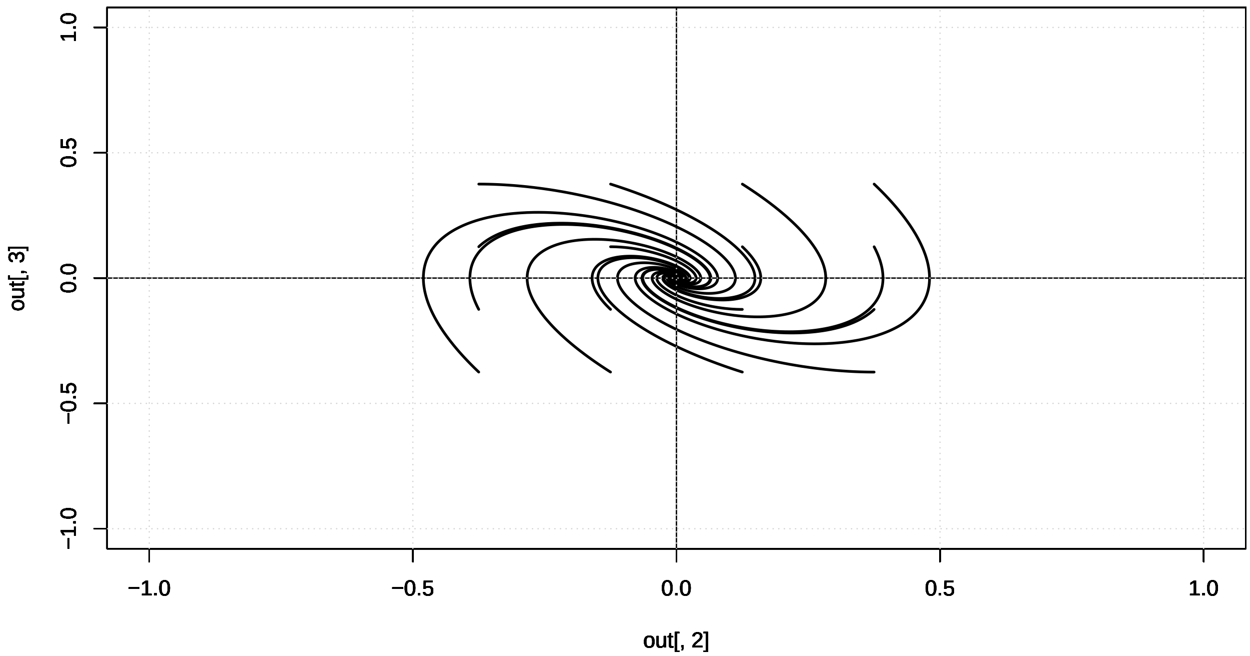
Tout d'abord, nous définissons la fonction système et initialisons les paramètres:

```r
# définition du système
linsis <- function (t, y, parameters) {

                dy1 <-                   parameters[3]*y[2]
                dy2 <- parameters[2]*y[1]  + parameters[4]*y[2]
                list( c(dy1, dy2 ))

} # end
```

```
# conditions initiales
n = 4  # Numéro de conditions initiales à considérer égal à n^2
# Valeurs maximum à considérer dans le plan de phase pour les deux variables x1 = x(t) et x2 = dx/dt

ymax = 1

Dymax = 1

control = TRUE  # variable auxiliaire

times <- seq(0, 10, .01)

parameters = c(0, -1, 1, -1)
```

Ensuite, nous générons les points à analyser dans l'espace euclidien:

```
for(i in 1:n){

        for (j in 1:n){

                y1 = (i-(n+1)/2)*ymax/n

                y2 = (j-(n+1)/2)*Dymax/n  # Vélocité initiale

                y_ini = c(y1, y2) # vecteur de conditions initiales

                out <- ode (times = times, y = y_ini, func = linsis, parms = parameters)


                plot( out[,2], out[,3], type = "l", lwd = 2, xlim = c(-ymax, ymax),
                               ylim = c(-Dymax, Dymax))
                par(new = TRUE) # para indicar que ha mais a plotar no mesmo grafico


        } # end for j

} # end for i

abline(h = 0, v = 0) # lignes horizontale et verticale
par(new = F)
grid()
```



Le diagramme de phase ne se croise pas. Le graphique nous le démontre parfaitement. Toute les lignes du diagramme de phase cheminent vers un seul point.

Cas de deux variables continues

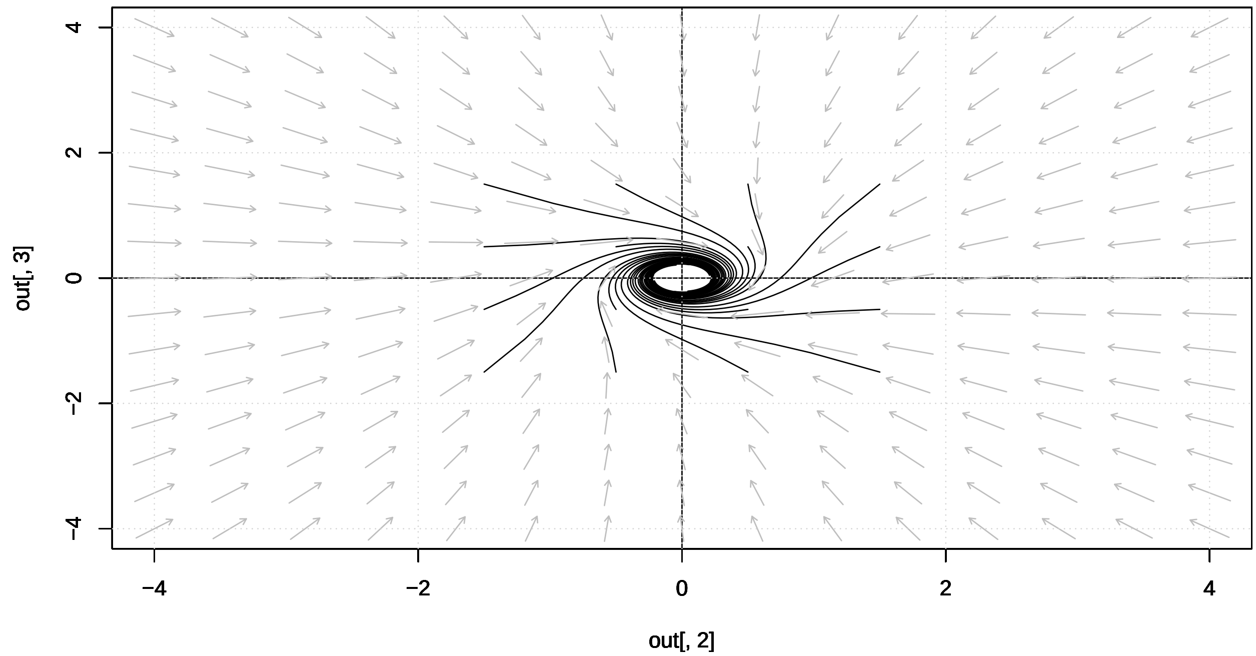
Analysons la stabilité du système :

$$x_1' = x_2 - x_1(x_1^2 + x_2^2)$$
$$x_2' = -x_1 - x_2(x_1^2 + x_2^2)$$

Par simulation, nous pouvons créer le diagramme de phase de la manière :

```
library(phaseR)

nlinsis2 <- function (t, y, parameters) {

        dy1 <- y[2] - y[1]*(y[1]^2 + y[2]^2)
        dy2 <- -y[1]  -y[2]*(y[1]^2 + y[2]^2)
        list( c(dy1, dy2 )  )

} # end


n = 4  # Numéro de conditions initiales à considérer
# Valeurs maximum à considérer dans le plan de phase pour les deux variables x1 = x(t) et x2 = dx/dt

ymax = 4
Dymax = 4

times <- seq(0, 10, .1)
```

```r
for(i in 1:n){

        for (j in 1:n){


                y1 = (i-(n+1)/2)*ymax/n

                y2 = (j-(n+1)/2)*Dymax/n  # Vélocité initiale

                y_ini = c(y1, y2) # vecteur de conditions initiales

                out <- ode (times = times, y = y_ini, func = nlinsis2, parms = NULL)



                plot( out[,2], out[,3], type = "l", lwd = 1, xlim = c(-ymax, ymax),
                      ylim = c(-Dymax, Dymax))
                par(new = TRUE)


        }


}

abline(h = 0, v = 0)

linsis.flowField <- flowField(nlinsis2,
                              xlim = c(-ymax, ymax),
                              ylim = c(-Dymax, Dymax),
                              parameters = NULL,
                              points = 15,
                              add = TRUE,
                              system = "two.dim")

grid()
```



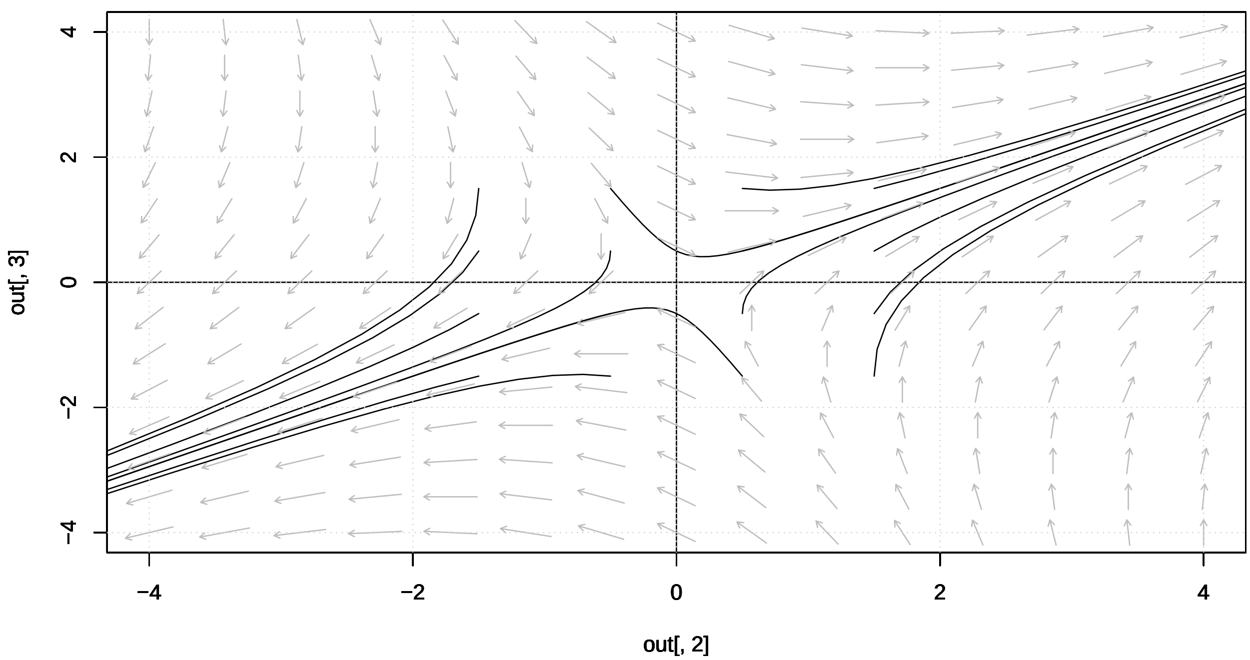

```
par(new = F)
```

Pour améliorer l'analyse, nous avons utilisé le paquet `phaseR`, qui trace le champ de puissance du système.

Dans le cas d'un système linéaire

```
linsis3 <- function (t, y, parameters) {

        dy1 <- y[1]  + y[2]
        dy2 <- 2*y[1] - y[2]
        list( c(dy1, dy2 ))

} # end


n = 4  # Numéro de conditions initiales à considérer
# Valeurs maximum à considérer dans le plan de phase pour les deux variables x1 = x(t) et x2 = dx/dt

ymax = 4
Dymax = 4

times <- seq(0, 10, .1)

# générer une solution pour différentes conditions initiales

for(i in 1:n){

        for (j in 1:n){


                y1 = (i-(n + 1)/2)*ymax/n
```

```
            y2 = (j-(n+1)/2)*Dymax/n  # Vélocité initiale

            y_ini=c(y1, y2) # Vecteur de conditions initiales

            out <- ode (times = times, y = y_ini, func = linsis3, parms = NULL)



            plot( out[,2], out[,3], type = "l", lwd = 1, xlim = c(-ymax, ymax),
                  ylim = c(-Dymax, Dymax))
            par(new = TRUE)


        }


}

abline(h = 0, v = 0)

linsis.flowField <- flowField(linsis3,
                              xlim = c(-ymax, ymax),
                              ylim = c(-Dymax, Dymax),
                              parameters = NULL,
                              points = 15,
                              add = TRUE,
                              system = "two.dim")
grid()
```




```
par(new = F)
```

Vérification des autovaleurs

```r
A = matrix(c(1, 2, 1, -1), nrow = 2)

eigen(A)$values
```

```
## [1]  1.732051 -1.732051
```