# Equações diferenciais 2

*Ivette Luna*

*25 de junho de 2019*

## Contents

# Exercício 1 - sistema linear de equações diferenciais

A simulação de sistemas (lineares ou não lineares) segue o mesmo formato adotado para a simulação de equações. Para a resolução numérica:

1. Usamos o pacote *deSolve*;
   - Definimos uma função que especifique as equações para cada taxa de variação;
   - Inicializamos os parâmetros da simulação, incluindo o período de tempo;
   - Fazemos uso do comando *ode* para a resolução numérica.

Por exemplo, seja o sistema de duas variáveis, em que a matriz de coeficientes é dada por:

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

Carregamos o pacote *deSolve* e criamos a função de derivadas:

```
library(deSolve)
```

```
## Warning: package 'deSolve' was built under R version 3.4.4
```

```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
library(tidyr)
```

```
## Warning: package 'tidyr' was built under R version 3.4.4
```

```
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
```

Logo inicializamos os parâmetros:

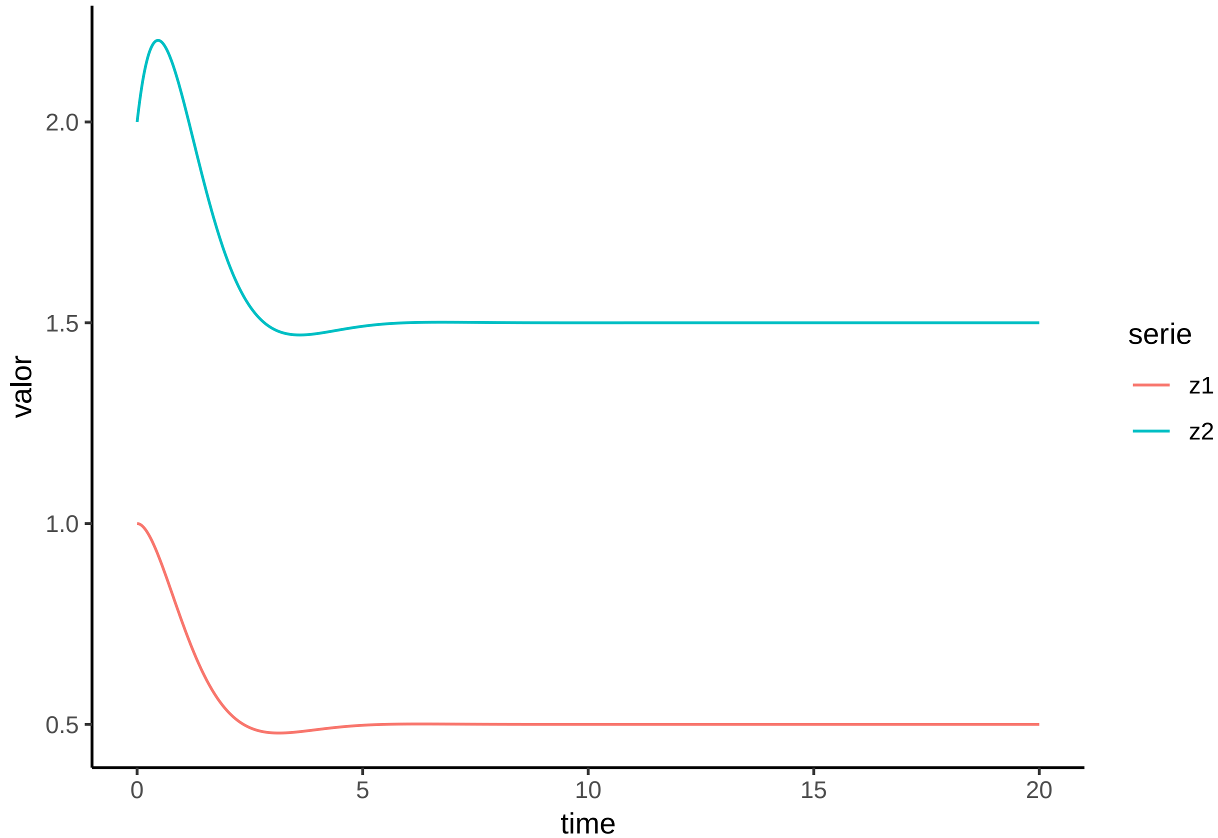
```
y_ini <- c(z1 = 1, z2 = 2)

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

E executamos o 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()
```




## Caso 2 - sistema não linear

Como exemplo, podemos usar o sistema de Rossler:

$$
\begin{aligned}
y_1' &= -y_2 - y_3 \\
y_2' &= y_1 + a * y_2 \\
y_3' &= b + y_3 * (y_1 - c)
\end{aligned}
$$

Para $y_1 = y_2 = y_3 = 1$; parâmetros $a = -.2,\ b = 0.2,\ c = 5$ 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
```

Definimos os parâmetros do modelo:

```
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)
```

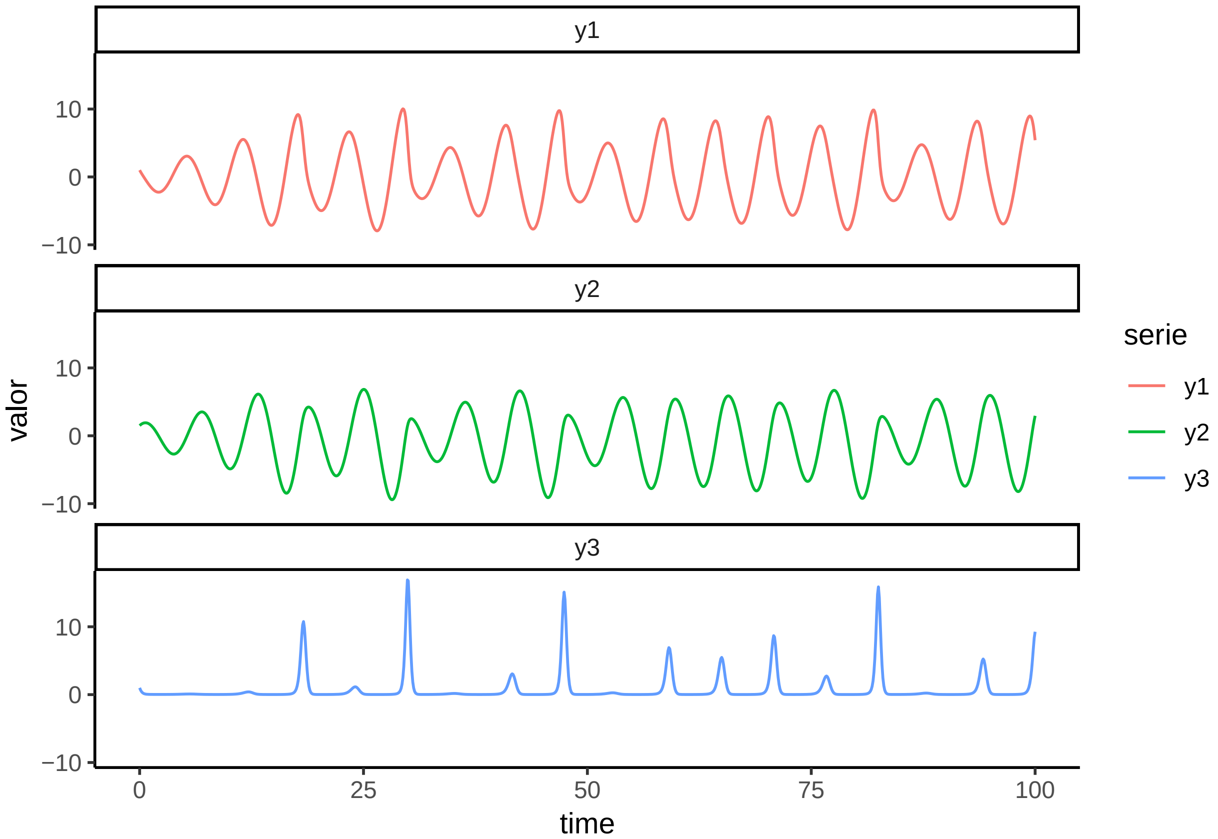
E simulamos por `ODE`:;

```
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
```

para visualizar as séries geradas:

```
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()
```

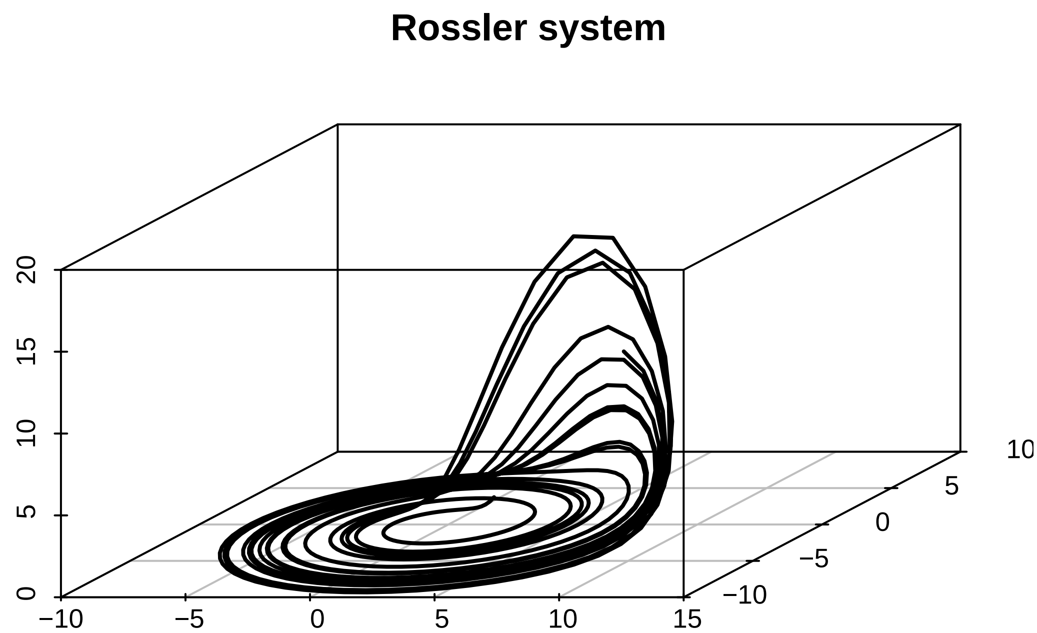



Adicionalmente:

```
library(scatterplot3d)
```

```
## Warning: package 'scatterplot3d' was built under R version 3.4.4
```

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




## Estabilidade e diagramas de fase - caso contínuo de uma variável

Quando temos uma equação autônoma diferencial de ordem 1, o retrato de fase é automático pois temos apenas uma função $y' = f(y)$ a plotar.

Para equações de ordens superiores, precisamos de transformar a equação a um sistema de duas variáveis. Por exemplo, seja a equação diferencial

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

Para analisar a estabilidade do sistema podemos seguir a sugestão do Gandolfo e reescrever a equação diferencial tal que

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

com $f(x, x') = x' + x$. Ou seja,

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

Seja

$$x_1 = x$$
$$x_2 = x'$$

Note que o retrato de fase será dado pelas coordenadas $(x_1,\ x_2)$. Logo, derivando e usando (2):

$$x_1' = x' = x_2$$
$$x_2' = x'' = -f(x_1, x_2)$$

Finalmente, usando (1) no sistema anterior:

$$x_1' = x_2$$
$$x_2' = -x_2 - x_1$$

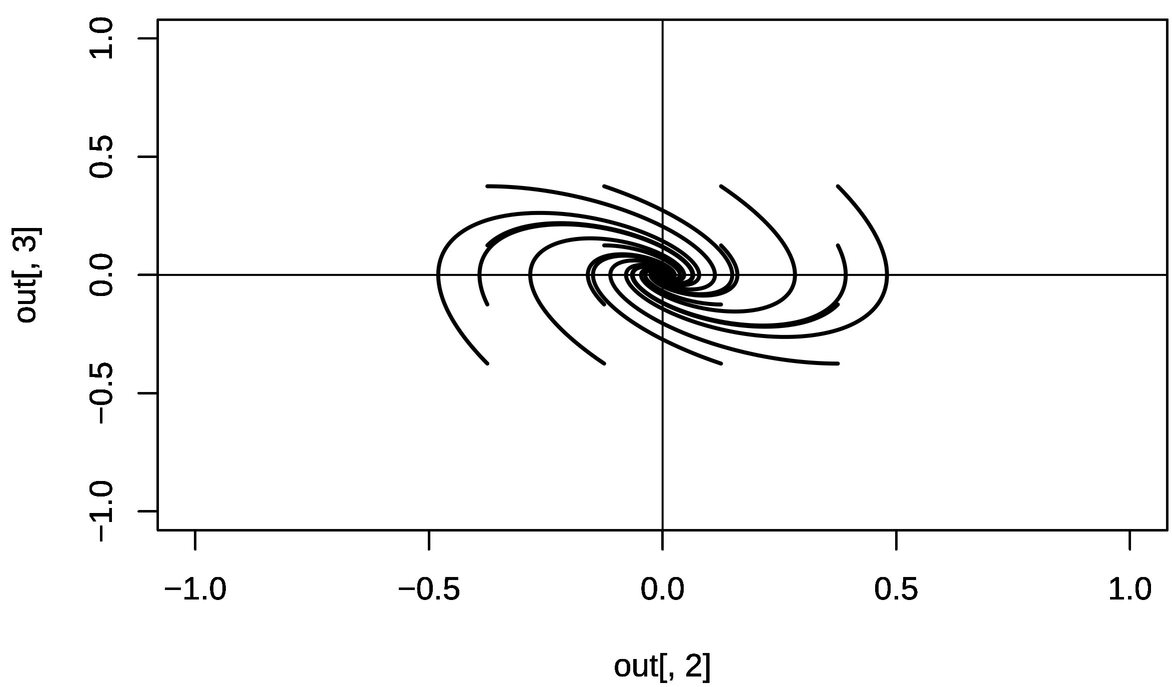
Primeiro, definimos a função do sistema e inicializammos os parâmetros:

```
# definicao do sistema
linsis <- function (t, y, parameters) {

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




          } # end

# condicoes iniciais
n = 4  #Numero de condicoes iniciais a considerar igual a n^2
# Valores maximos a considerar no plano de fase para as duas variaveis x1=x(t) e x2=dx/dt
ymax = 1

Dymax = 1

control = TRUE  # variavel auxiliar

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

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

A seguir, geramos os pontos a analisar no espaço euclidiano:

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

        for (j in 1:n){

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

                y2=(j-(n+1)/2)*Dymax/n  # Velocidade inicial

```

```
            y_ini=c(y1, y2) #vetor de condicoes iniciais

            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) # linhas horizontal e vertical
```




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

## Caso de duas variáveis

Analise a estabilidade do sistema:

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

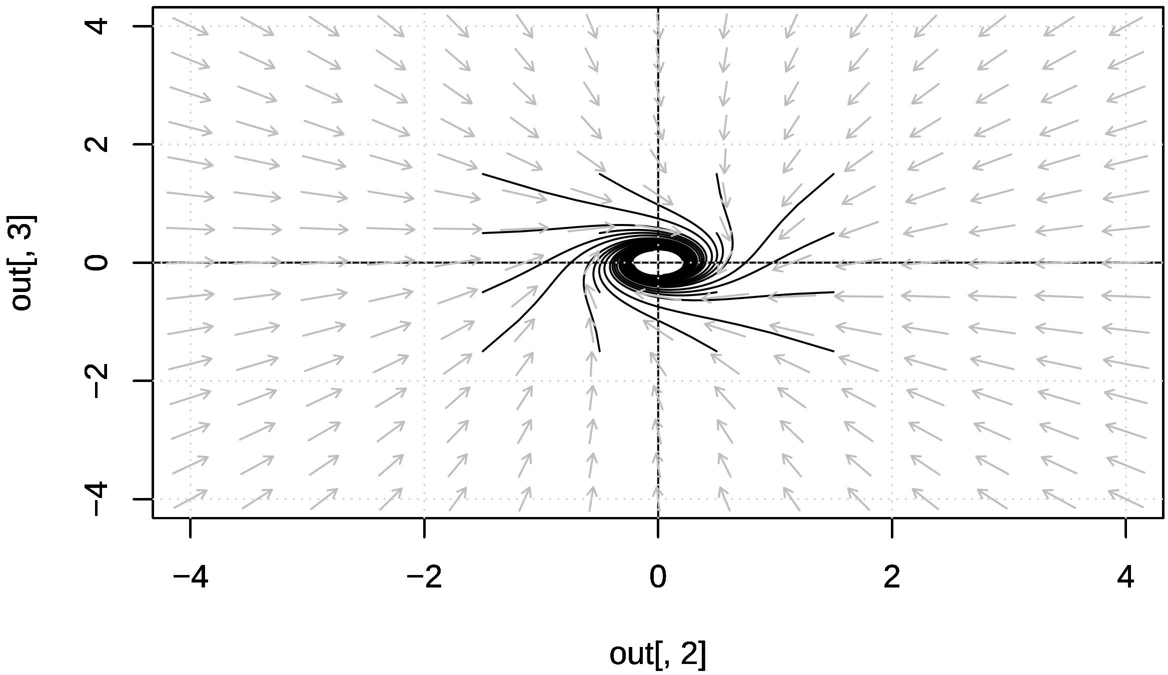
Por simulação podemos construir o diagrama de fase:

```r
library(phaseR)
```

```
## Warning: package 'phaseR' was built under R version 3.4.4
```

```r
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  #Numero de condicoes iniciais a considerar
# Valores maximos a considerar no plano de fase para as duas variaveis x1=x(t) e x2=dx/dt
ymax = 4
Dymax = 4

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


for(i in 1:n){

        for (j in 1:n){


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

                y2=(j-(n+1)/2)*Dymax/n  # Velocidade inicial

                y_ini=c(y1, y2) #vetor de condicoes iniciais

                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()
```



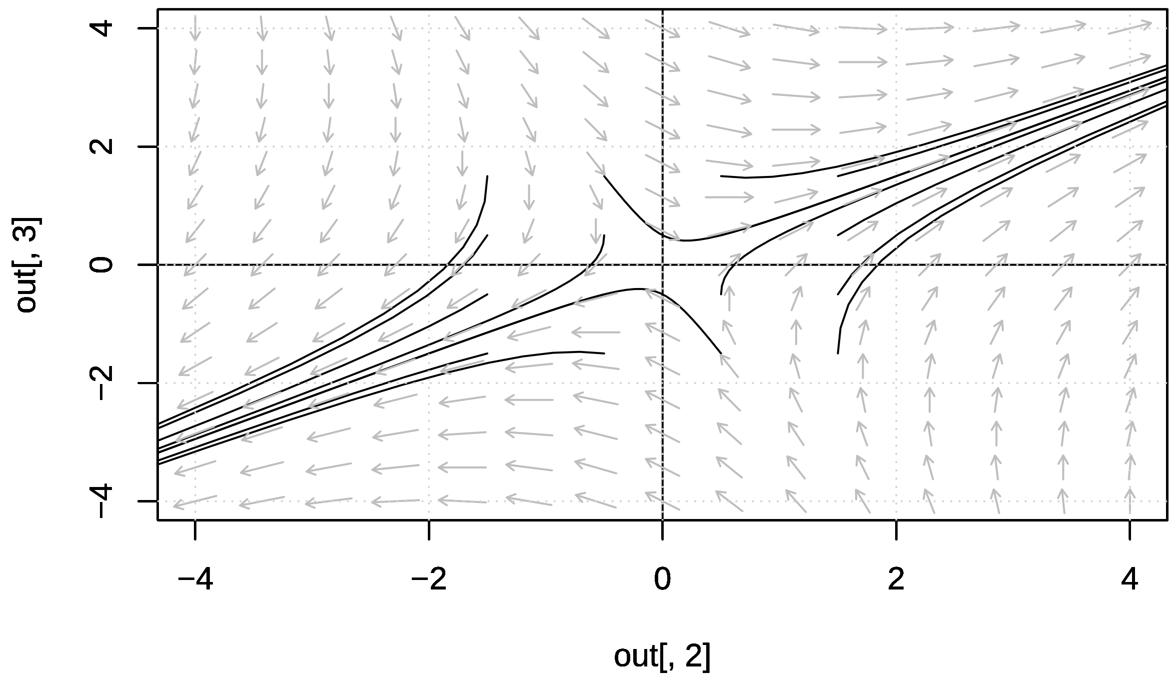

```
par(new=F)
```

Para aprimorar a análise, fizemos uso do pacote `phaseR`, que plota o campo de força do sistema.

## No caso de um sistema linear

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

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

} # end


n=4  #Numero de condicoes iniciais a considerar
# Valores maximos a considerar no plano de fase para as duas variaveis x1=x(t) e x2=dx/dt
ymax=4
Dymax=4
```

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

# gerando uma solucao para diferentes condicoes iniciais
for(i in 1:n){

        for (j in 1:n){


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

                y2=(j-(n+1)/2)*Dymax/n  # Velocidade inicial

                y_ini=c(y1, y2) #vetor de condicoes iniciais

                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()
```




```r
par(new=F)

# verificando os autovalores

A = matrix(c(1, 2, 1, -1), nrow=2)
eigen(A)$values
```

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