# Analyse Qualitative 2D - Applications - corrigé

C. Wolf

avec les packages deSolve et PhaseR

## Exemple : Modèle de compétition

```
library(deSolve)
library(phaseR)
```

Nous allons maintenant faire l'étude qualtitative du modèle de compétition de lotka-Volterra à l'aide du package **PhaseR**

$$\begin{cases} \frac{dx}{dt} &= r_1 x - b_1 xy \\ \frac{dy}{dt} &= r_2 y + b_2 xy \end{cases}$$

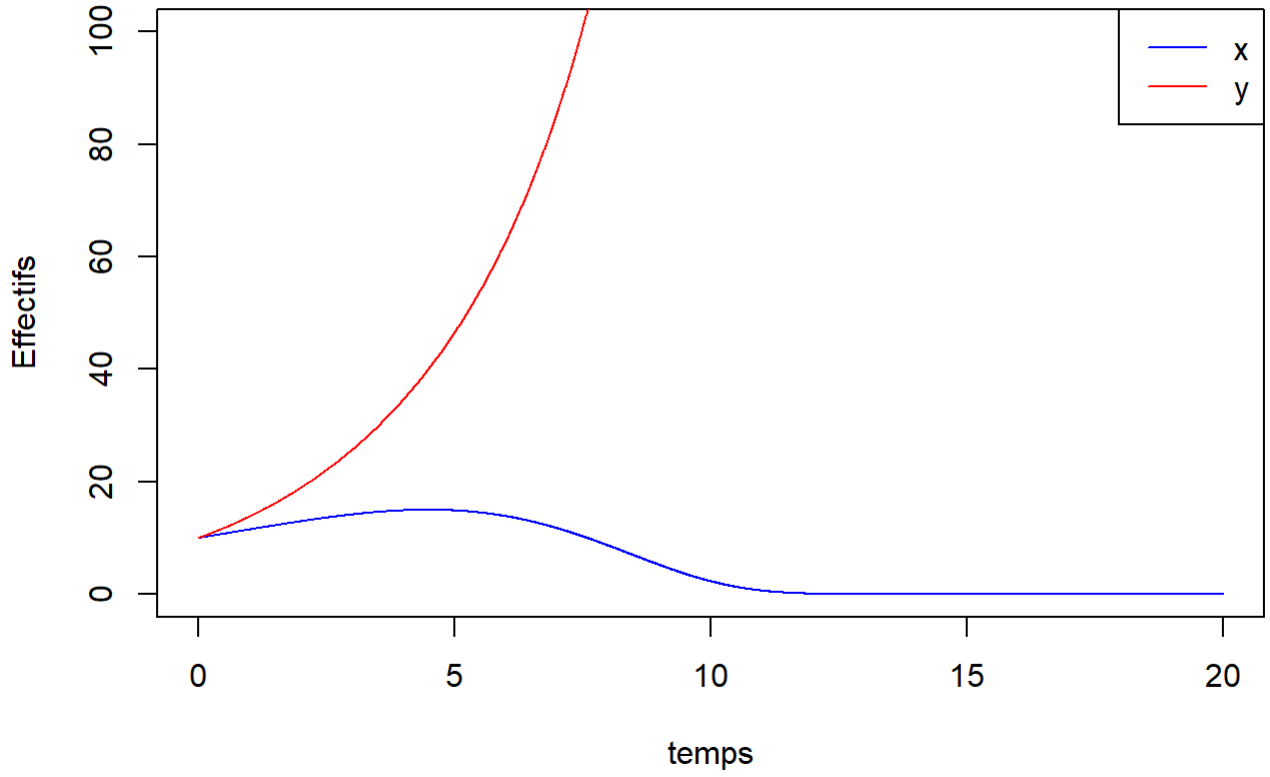
On rappelle que comme pour le package **deSolve** la fonction définissant le système doit respecter la syntaxe suivante :

```
Compet<- function(t, y, parameters) {
  x <- y[1]
  y <- y[2]
  r1 <- parameters[1]
  r2 <- parameters[2]
  c1 <- parameters[3]
  c2 <- parameters[4]
  dy <- numeric(2)
  dy[1] <- r1*x - c1*x*y
  dy[2] <- r2*y - c2*x*y
  list(dy)
}
```

Il est en particulier important que les 2 premiers arguments de la fonction soient le temps et un objet unique comportant les variables d'état, et qu'ensuite on ait un objet avec le(s) paramètre(s) éventuel(s).

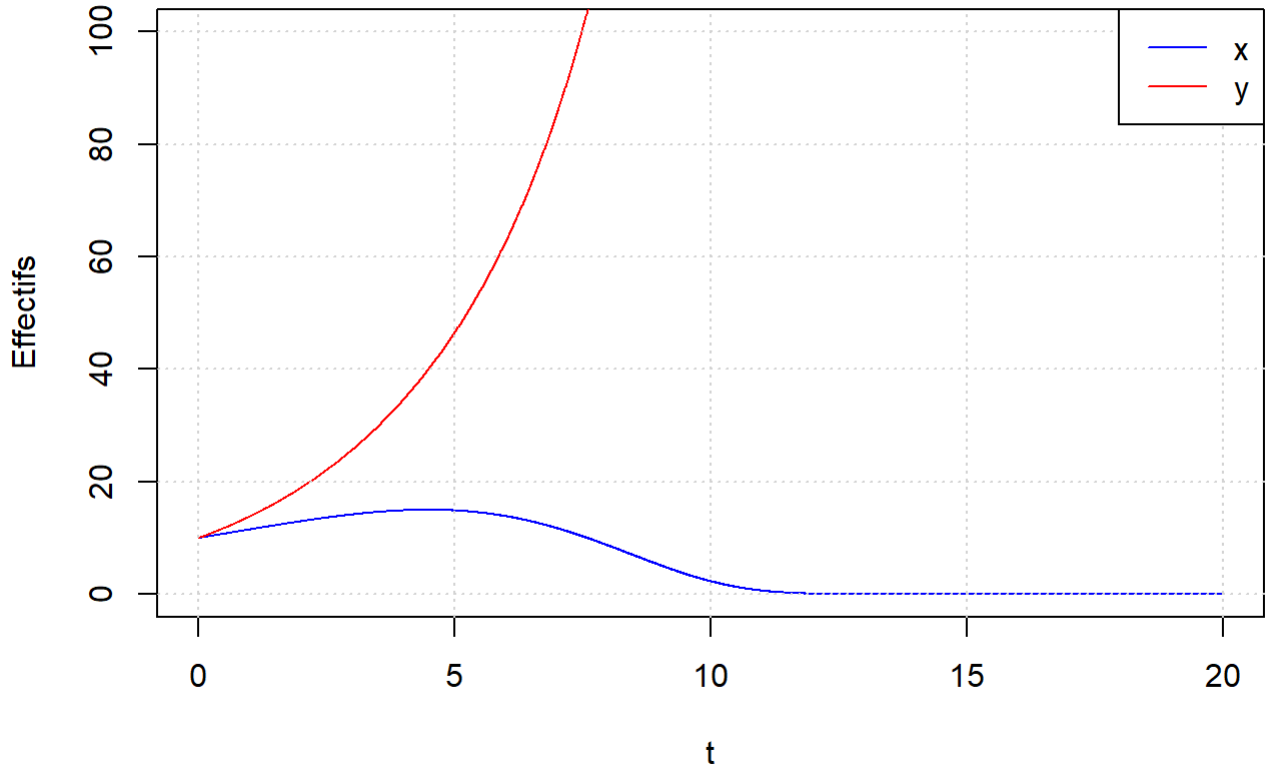
La fonction **ode()** du package solve s'utilise alors de la façon suivante :

```
parametres=c(r1=0.2,r2=0.4,c1=0.005,c2=0.007)
temps=seq(from = 0, to = 20, by = 0.01)
X0=c(10,10)
sol <- ode(y = X0, times = temps, func = Compet, parms = parametres)
plot(temps,sol[,2],col="blue",type="l",ylim=c(0,100),ylab="Effectifs")
lines(temps,sol[,3],col="red")
legend("topright",c("x","y"),col=c("blue","red"),lty=1)
```




Le package *PhaseR* permet aussi de simuler numériquement des trajectoires en fonction du temps :

```
numericalSolution(Compet, y0 = c(10,10), tlim =c(0, 20), type = "one", parameters = parametre
s, col = c("blue", "red"), ylab = "Effectifs", ylim = c(0, 100))
```




En réalité cette fonction fait elle-même appel à deSolve, mais produit directement la représentation graphique. A quoi sert l'argument type ici ?

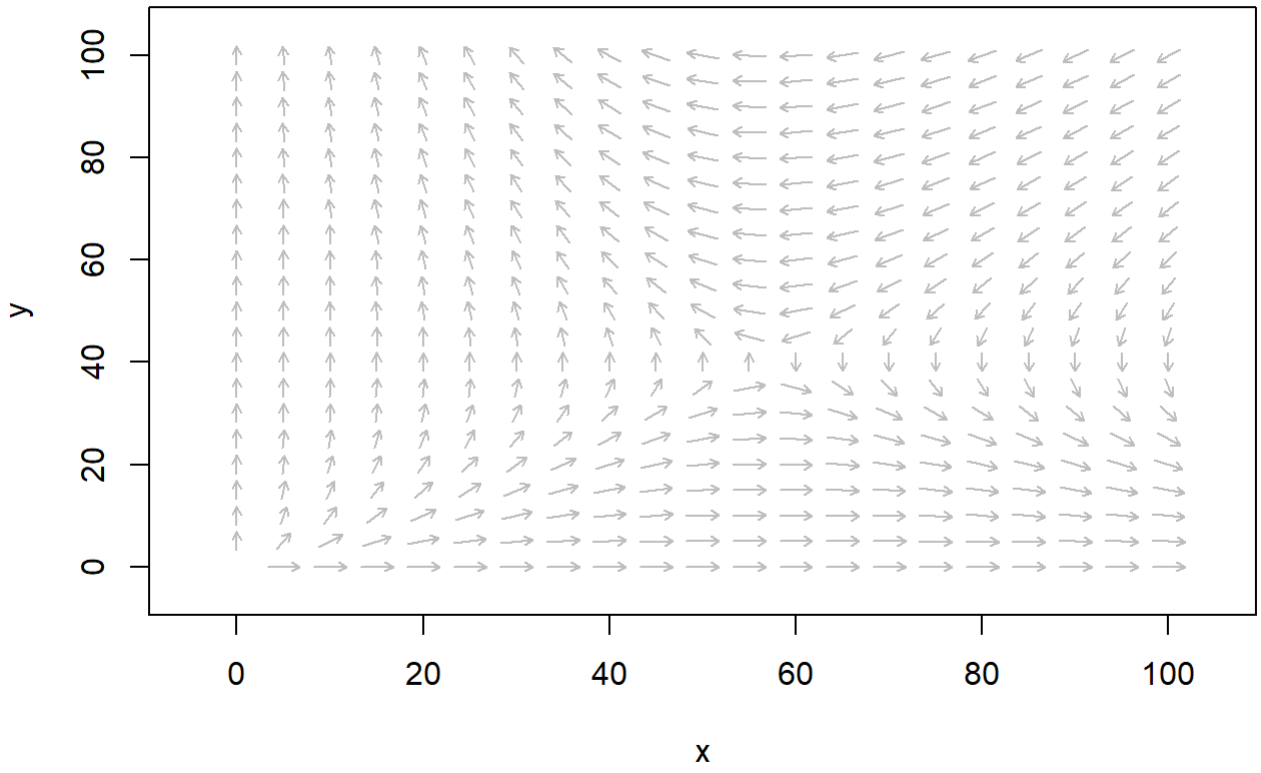
*Réponse : L'argument type sert à indiquer si l'on veut un seul graphe avec les 2 trajectoires ou si on les trace sur des graphes séparés*

*Rem : Par défaut elle "produit" également beaucoup de résultats qui prenent de la place dans la console et ne sont pas forcément souhaiter. Il suffit de donner un nom à cet objet pour éviter cet affichage (tout en pouvant accéder aux résultats ultérieurement si souhaité)*

1. ***En utilisant le package Phase R (et notamment les fonctions flowField() et nullclines() vues dans le cadre 1D), tracer le portrait de champ du modèle de compétition. On conservera les valeurs de paramètres ci-dessus***

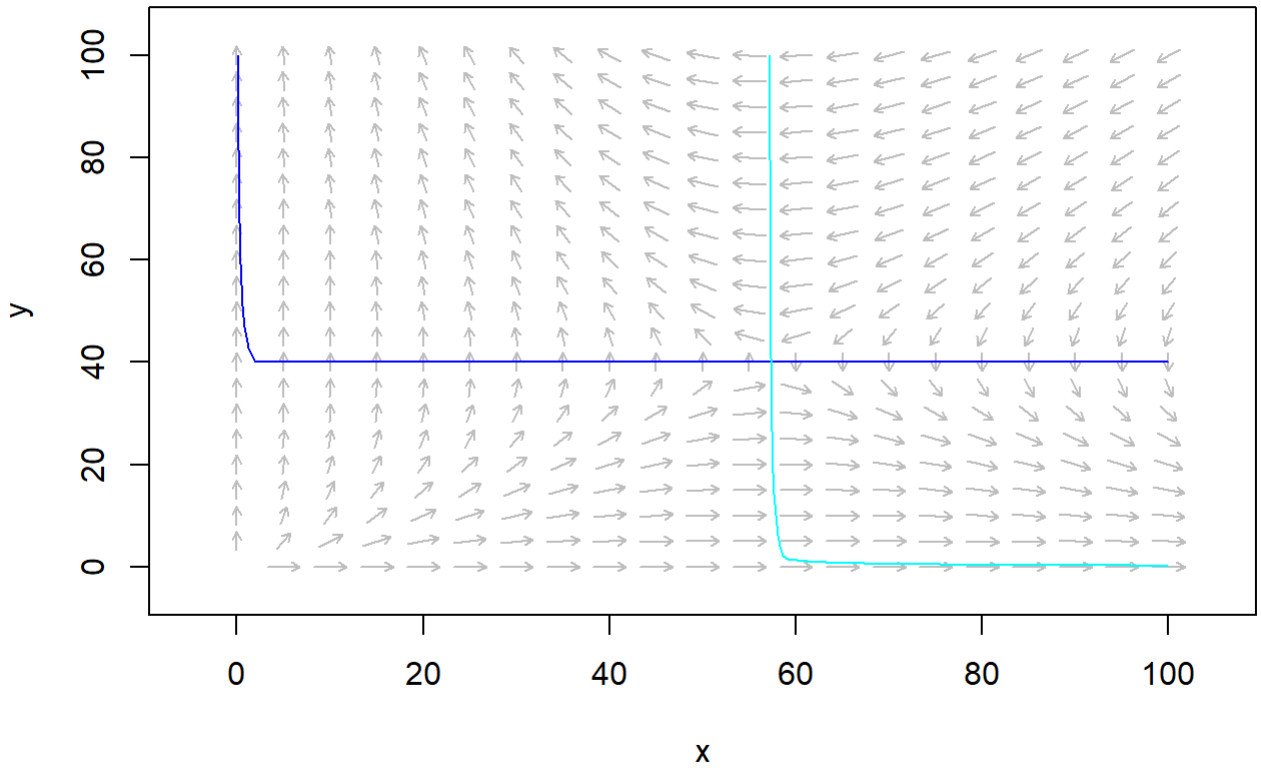
On commence avec la fonction flowField :

```
flowField(Compet, xlim = c(0, 100), ylim = c(0, 100),parameters = parametres,add = FALSE)
```



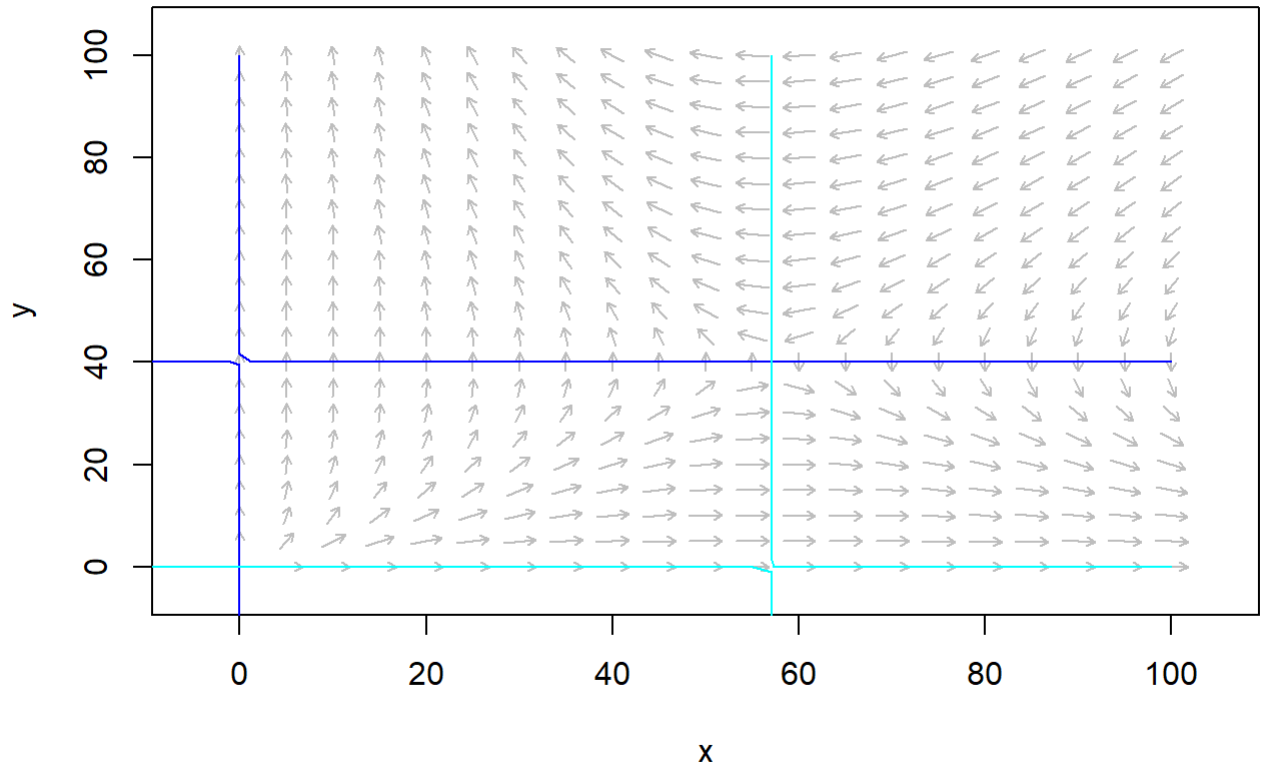

puis on ajoute les isoclines :

```
nullclines(Compet,xlim=c(0,100),ylim=c(0,100),parameters = parametres, points = 500,add.legend = FALSE)
```




Même avec 500 points (ce qui est assez fin, ce sont des coordonnées tireés au hasard), il ne “trouve” pas les isoclines en entier. En fait, seules les trajectoires simulées issues de ces points qui “croisent” une isocline permettent de les identifier.

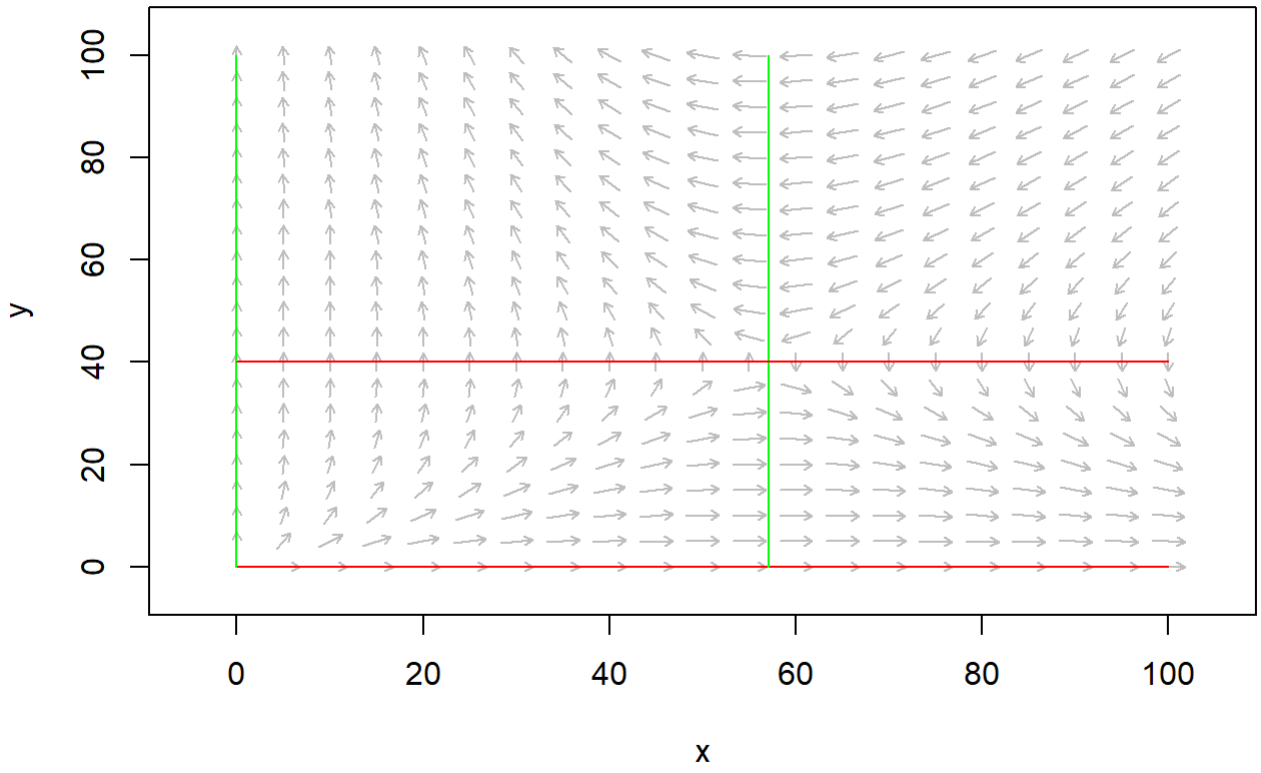
Parfois, on peut les prolonger en “explorant” une zone un peu plus large :




```
nullclines(Compet,xlim=c(-10,100),ylim=c(-10,100),parameters = parametres, points = 500,add.l
egend = FALSE)
```

Ou quand on les a déterminé à l'avance et qu'il s'agit "juste" de les tracer, cela peut être aussi simple de les tracer nous-même (Il suffit en pratique de tracer un segment entre 2 points de coordonnées (xA,yA) et (xB,yB) par la commande *lines(c(xA,xB),c(yA,yB))*:

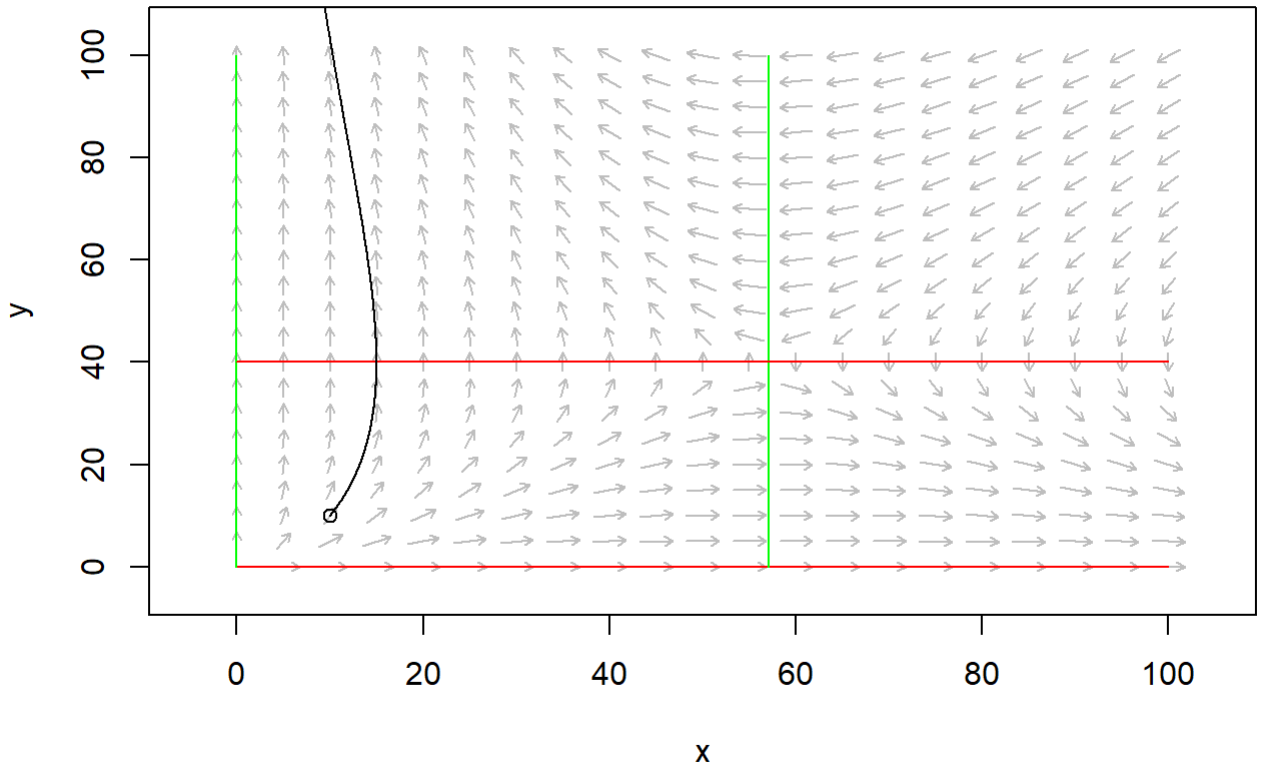
```
flowField(Compet, xlim = c(0, 100), ylim = c(0, 100),parameters = parametres,add = FALSE)
lines(c(0,100),c(0,0),col="red")
lines(c(0,0),c(0,100),col="green")
xstar=parametres[2]/parametres[4]
ystar=parametres[1]/parametres[3]
lines(c(xstar,xstar),c(0,100),col="green")
lines(c(0,100),c(ystar,ystar),col="red")
```




2. ***A l'aide de la fonction trajectory(), tracer la trajectoire simulée précédement dans ce plan. Tracer ensuite plusieurs trajectoires (nombreuses) "partant" d'un peu partout***

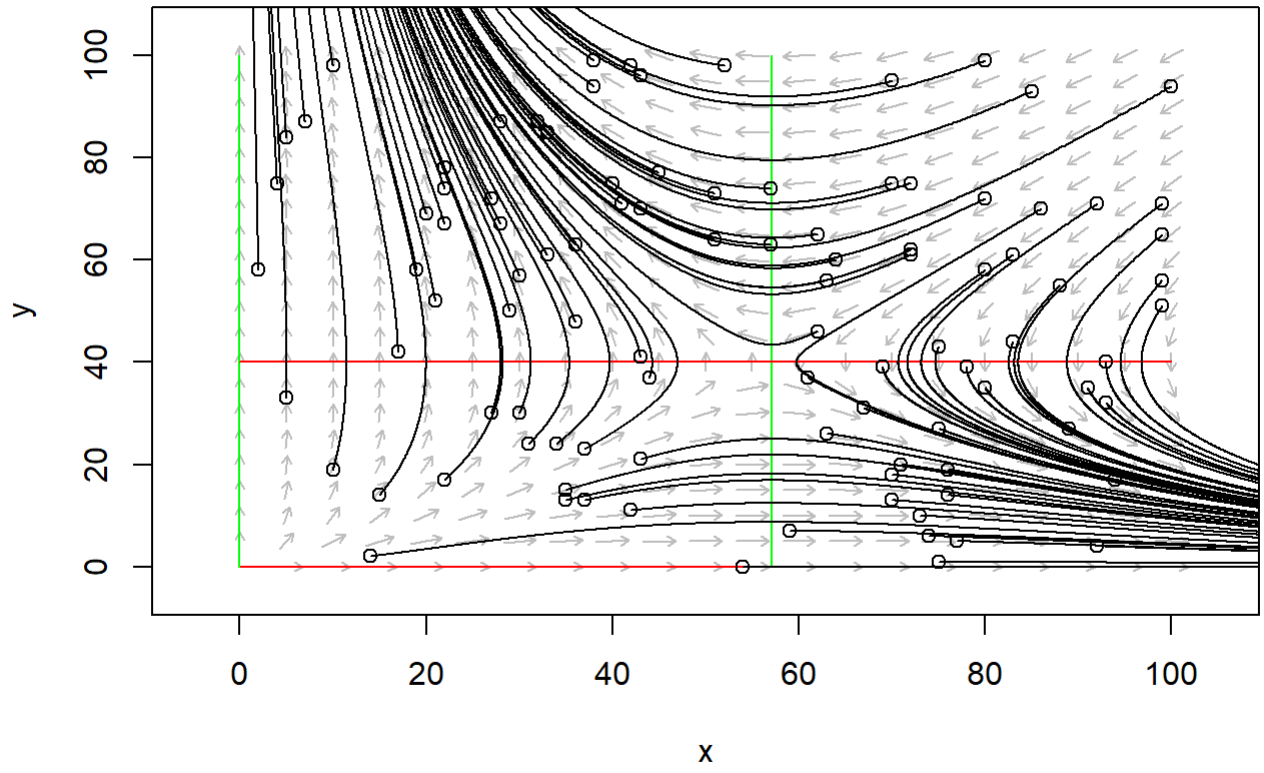
Traçons la trajectoire issue du point de coordonnée (10,10) :

```
X0=c(10,10)
trajectory(Compet,X0,tlim=c(0,20), parameters=parametres)
```




Pour en tracer plusieurs, par exemple (plein d'autres possibilités) :

```
for (i in 1:100){
  X0=sample(0:100,2,replace=TRUE)
  trajectory(Compet,X0,tlim=c(0,20), parameters=parametres)
}
```




3. ***Revenez à un tracé du portrait de champ sans les trajectoires. Utiliser la fonction stability() pour déterminer (en fait estimer) la stabilité du point fixe (0,0).***

Cela se fait avec la commande suivante :

```
stability(Compet, ystar = c(0, 0), parameters = parametres)
```

```
## tr = 0.6, Delta = 0.08, discriminant = 0.04, classification = Unstable node
```

4. ***Testez la fonction findEquilibrium() pour tenter de trouver d'autres points fixes du modèle. Quelle est leur stabilité ? On pourra tester également la fonction drawManifolds() sur ce(s) point(s) fixes. A quoi sert-elle ?***

Avec par exemple :

```
findEquilibrium(Compet,y0=NULL,parameters = parametres,plot.it = TRUE)
```

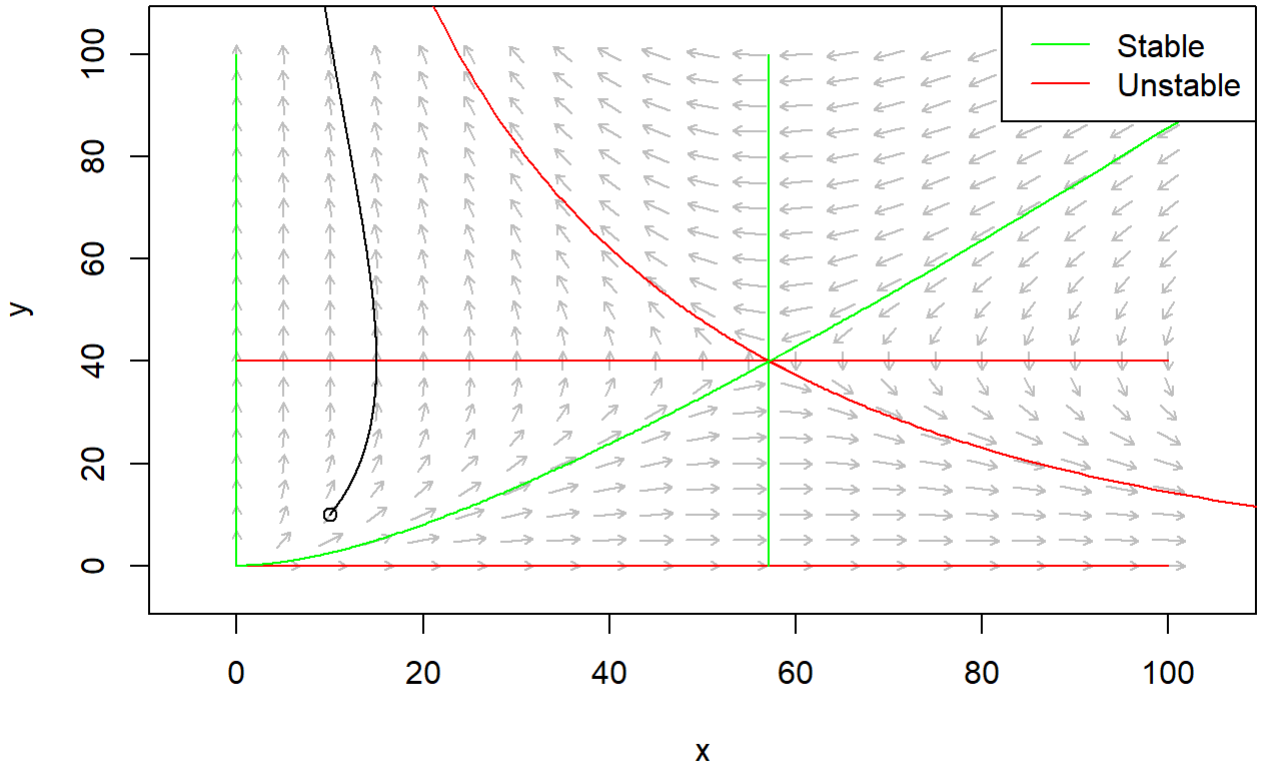
il est assez facile de tomber sur le point fixe (5.14 ; 40) - qui correspond bien à $\left(\frac{r_2}{c_2}; \frac{r_1}{c_1}\right)$ en cliquant sur le graphique. Rappelons que cette foncion va trouver (s'il y en a) un des points fixes du modèle, pas forcément le plus proche ni celui duquel la trajectoire passerait le plus près.

testons la fonction *drawMaifolds* comme demandé :

```
drawManifolds(Compet,y0=c(xstar,ystar),parameters = parametres)
```

```
## Warning in lsoda(y, times, func, parms, ...): Excessive precision requested.
## scale up `rtol' and `atol' e.g by the factor 10
```

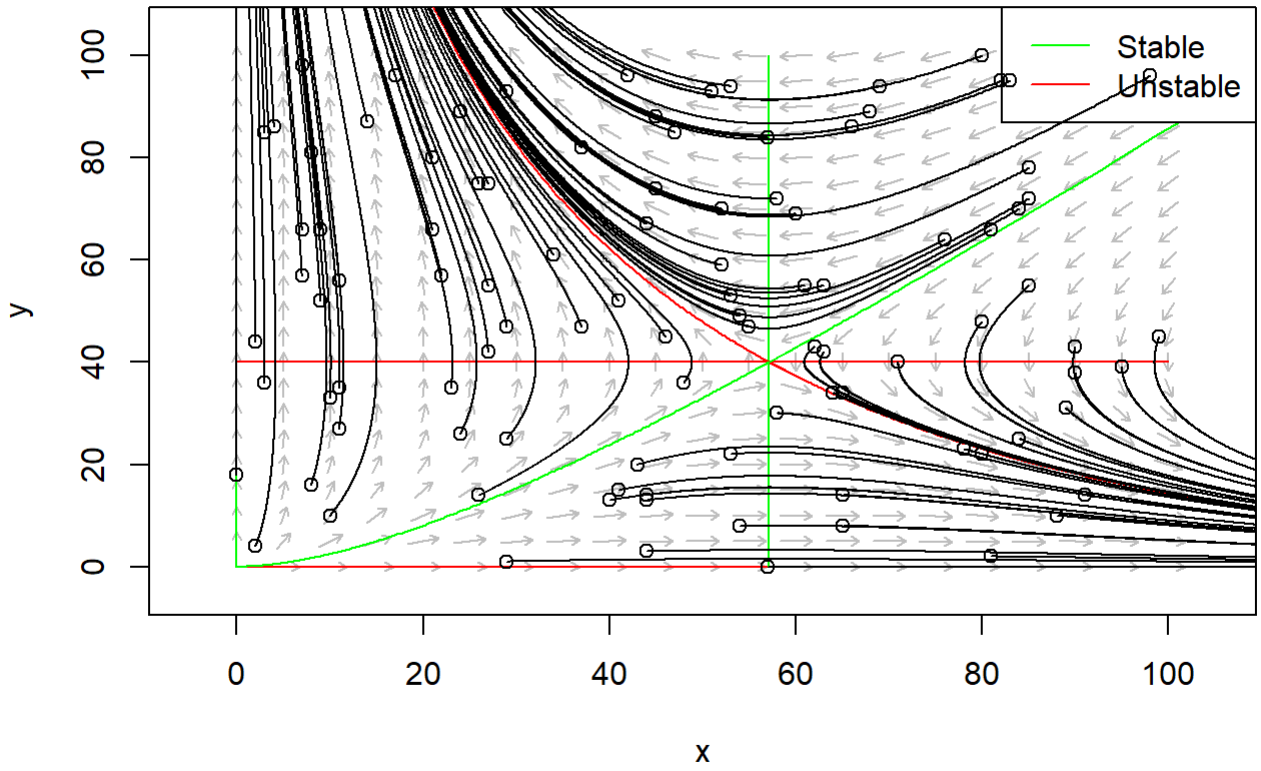
```
## Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
## accurate, as far as they go
```




Cette fonction permet de voir les “flux” entrants et sortants du point selle. Cela permet par la même occasion de visualiser les “frontières” de comportements ; si on les trace sur les 100 trajectoires simulées tout à l’heure, on obtient le graphe suivant :

```
## Warning in lsoda(y, times, func, parms, ...): Excessive precision requested.
## scale up `rtol' and `atol' e.g by the factor 10
```

```
## Warning in lsoda(y, times, func, parms, ...): Returning early. Results are
## accurate, as far as they go
```




# Autres modèles vus en cours

« Vérifiez » les comportements déterminés en cours par l'analyse qualitative pour les modèles suivants. On veillera à traiter les différents cas possibles et on traçera notamment à chaque fois le portrait de champs seul, puis en y traçant plusieurs trajectoires, et on tracera en fonction du temps des trajectoires types.

On considèrera (pour des graphiques représentatifs) des taux de croissance compris entre 0 et 1 et les autres coefficient de l'ordre de 10 à 100 fois plus petits (En particulier tous les coefficients sont donc positifs).

Modèle Proie-Prédateur de Lotka-Volterra :

$$\begin{cases} \frac{dN}{dt} &= r_1 N - b_1 NP \\ \frac{dP}{dt} &= -r_2 P + b_2 NP \end{cases}$$

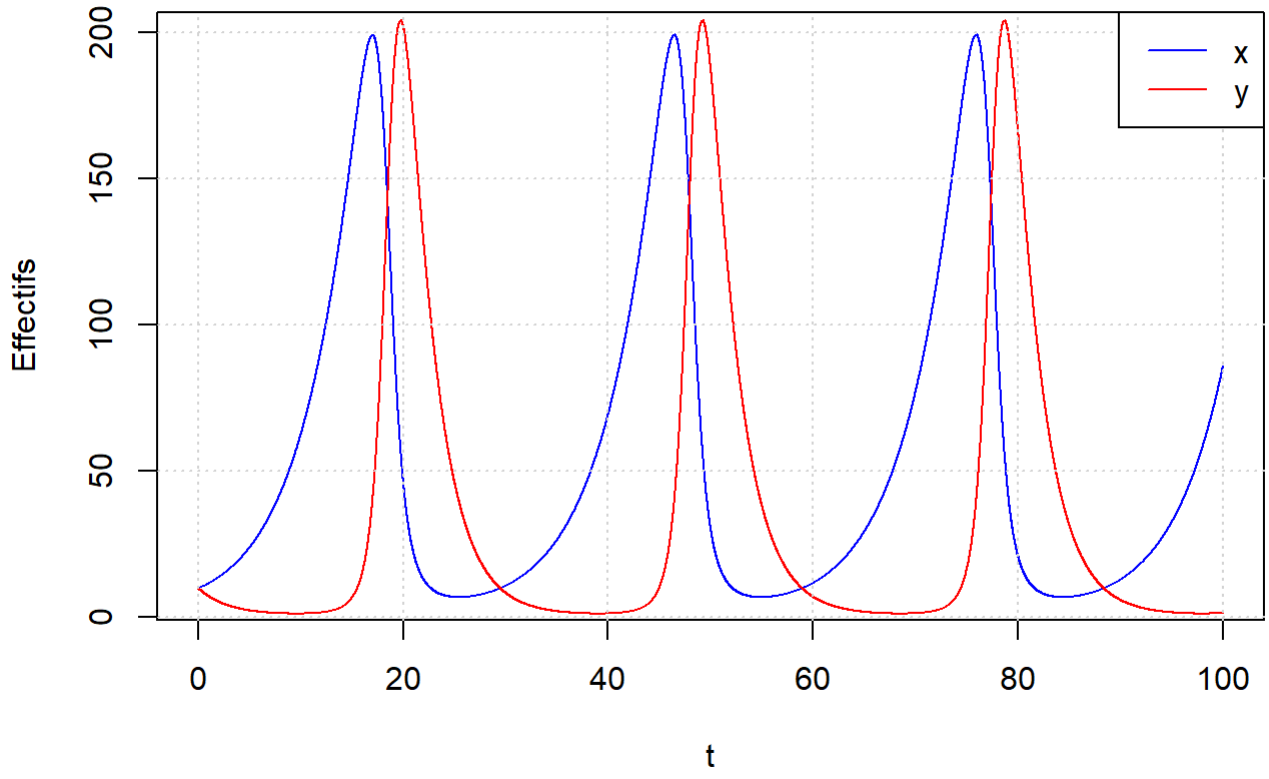
On rappelle (cf cours pour les détails) que ce modèle possède 2 points fixes, $(0,0)$ qui est un point selle et $\left(\frac{r_2}{b_2}; \frac{r_1}{b_1}\right)$ qui est un centre.

On commence par créer la fonction définissant le système :

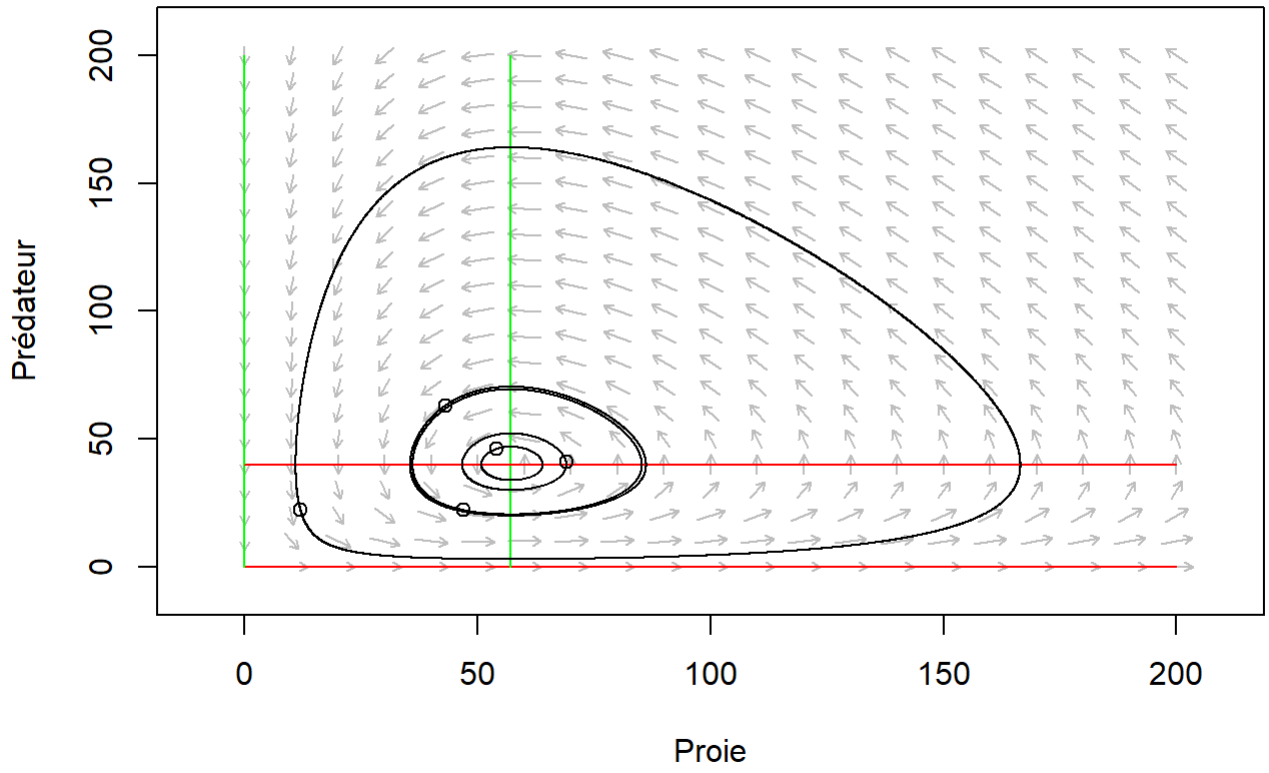
```
Proiepred<- function(t, y, parameters) {
  N <- y[1]
  P <- y[2]
  r1 <- parameters[1]
  r2 <- parameters[2]
  b1 <- parameters[3]
  b2 <- parameters[4]
  dPop <- numeric(2)
  dPop[1] <- r1*N - b1*N*P
  dPop[2] <- - r2*P + b2*N*P
  list(dPop)
}
```

Voici par exemple les trajectoires représentées en fonction du temps :

```
parametres=c(r1=0.2,r2=0.4,b1=0.005,b2=0.007)
numericalSolution(Proiepred, y0 = c(10,10), tlim =c(0, 100), type = "one", parameters = param
etres, col = c("blue", "red"), ylab = "Effectifs")
```




Et voici un portrait de champ pour ce modèle, avec plusieurs trajectoires :




Modèle Proie-Prédateur logistique de Lotka-Volterra :

$$\begin{cases} \frac{dN}{dt} &= r_1 N\left(1-\frac{N}{K}\right) - b_1 NP \\ \frac{dP}{dt} &= -r_2 P + b_2 NP \end{cases}$$

On rappelle (cf cours pour les détails) que ce modèle possède 3 points fixes, $(0,0)$ ; $(K,0)$ et $\left(\frac{r_2}{b_2}; \frac{r_1}{b_1}\left(1-\frac{r_2}{b_2 K}\right)\right)$. 2 cas se présentent suivant les valeurs des paramètres

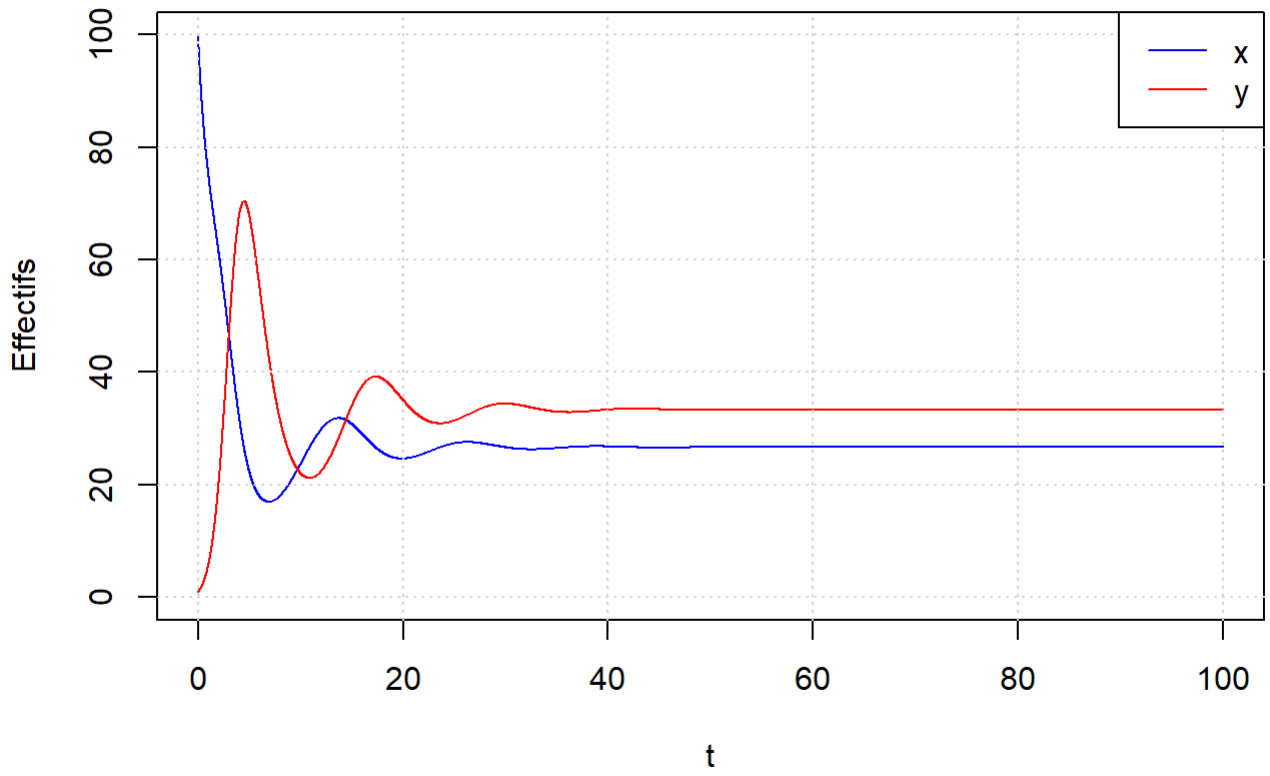
On commence par créer la fonction définissant le système :

```
Proiepredlog<- function(t, y, parameters) {
  N <- y[1]
  P <- y[2]
  r1 <- parameters[1]
  r2 <- parameters[2]
  b1 <- parameters[3]
  b2 <- parameters[4]
  K <- parameters[5]
  dPop <- numeric(2)
  dPop[1] <- r1*N*(1-N/K) - b1*N*P
  dPop[2] <- - r2*P + b2*N*P
  list(dPop)
}
```

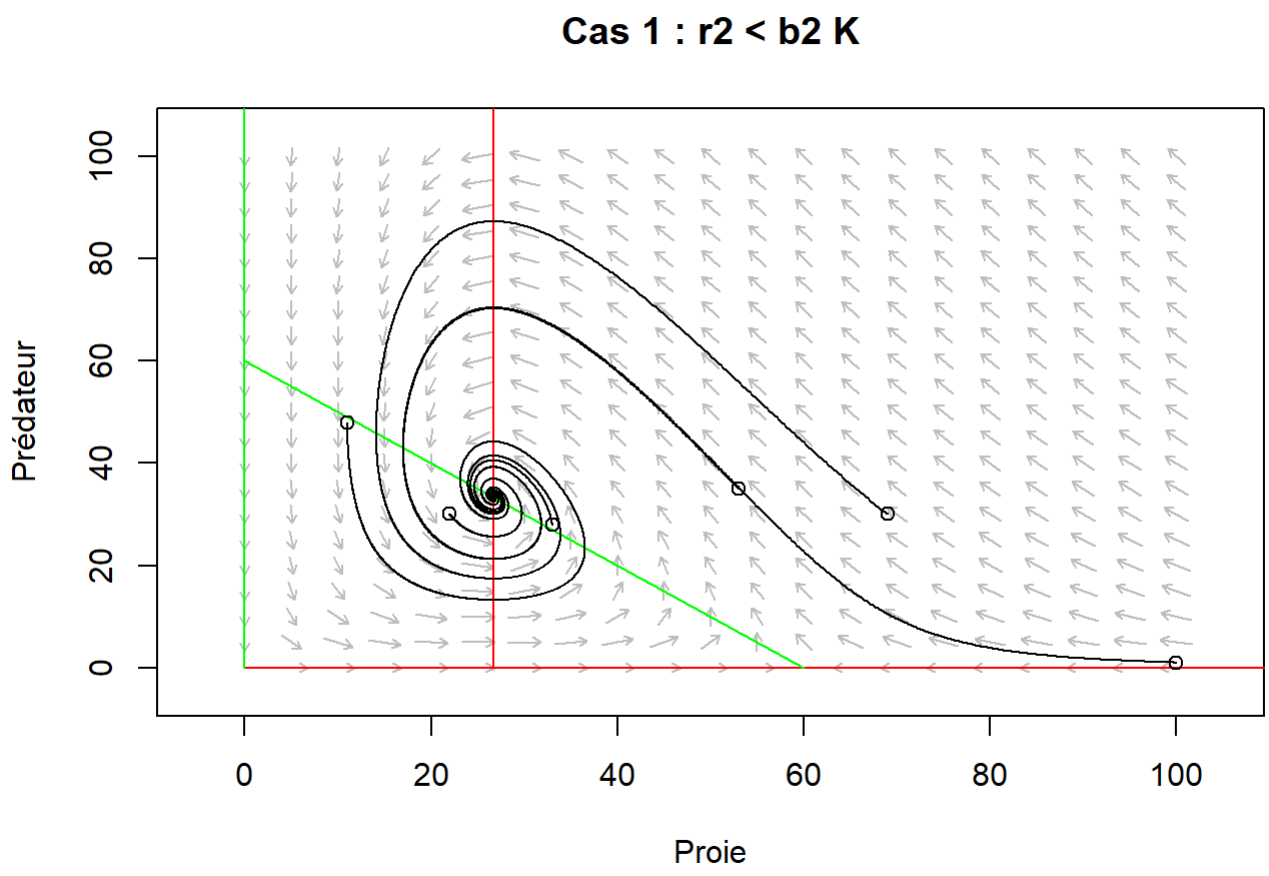
Cas 1 : r2 < b2 K : Les points points fixes sont alors dans le cadran positifs; (0,0) et (K,0) sont point selle et $\left(\frac{r_2}{b_2}; \frac{r_1}{b_1}\left(1-\frac{r_2}{b_2 K}\right)\right)$ est LAS. Toutes les trajectoires (partant de conditions initiales >0) vont converger vers ce dernier point fixe

Voici par exemple les trajectoires représentées en fonction du temps :

```
parametres=c(r1=0.6,r2=0.8,b1=0.01,b2=0.03,K=60)
numericalSolution(Proiepredlog, y0 = c(100,1), tlim =c(0, 100), type = "one", parameters = pa
rametres, col = c("blue", "red"), ylab = "Effectifs",ylim=c(0,100))
```



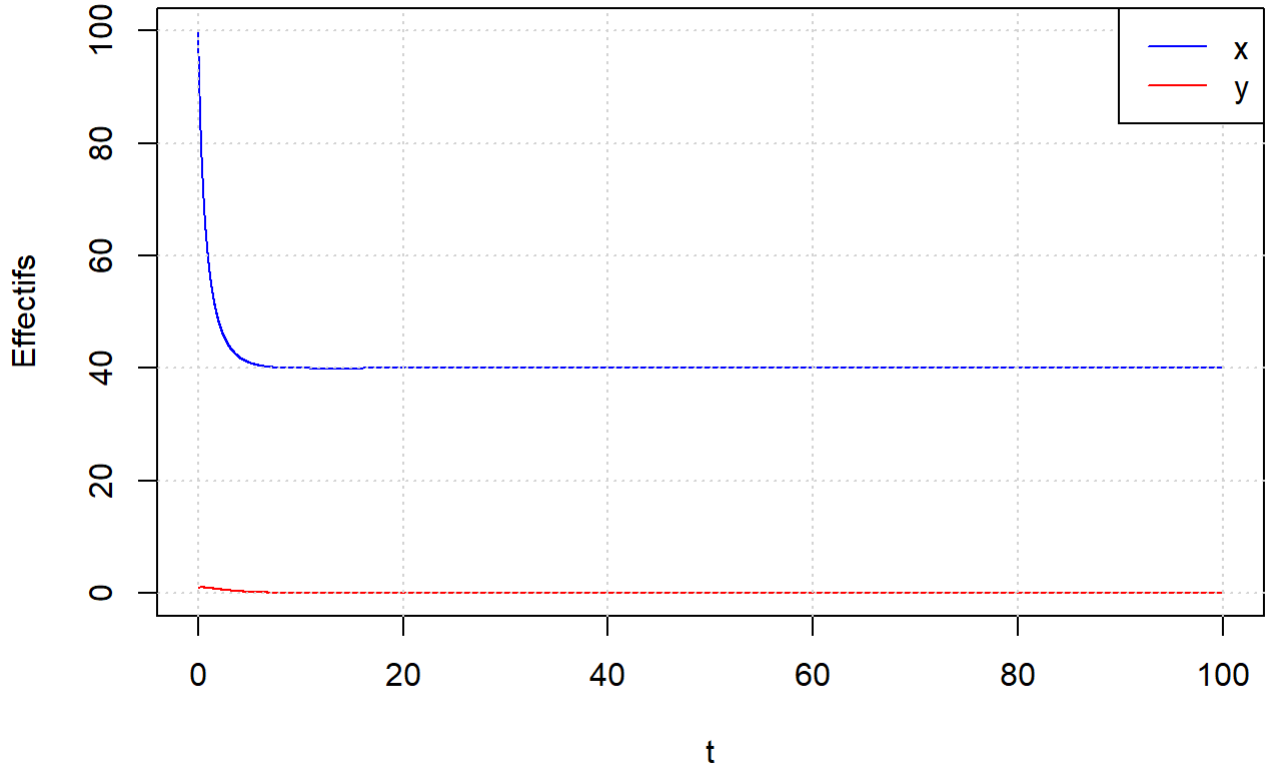

Et voici un portrait de champ pour ce modèle, avec plusieurs trajectoires :



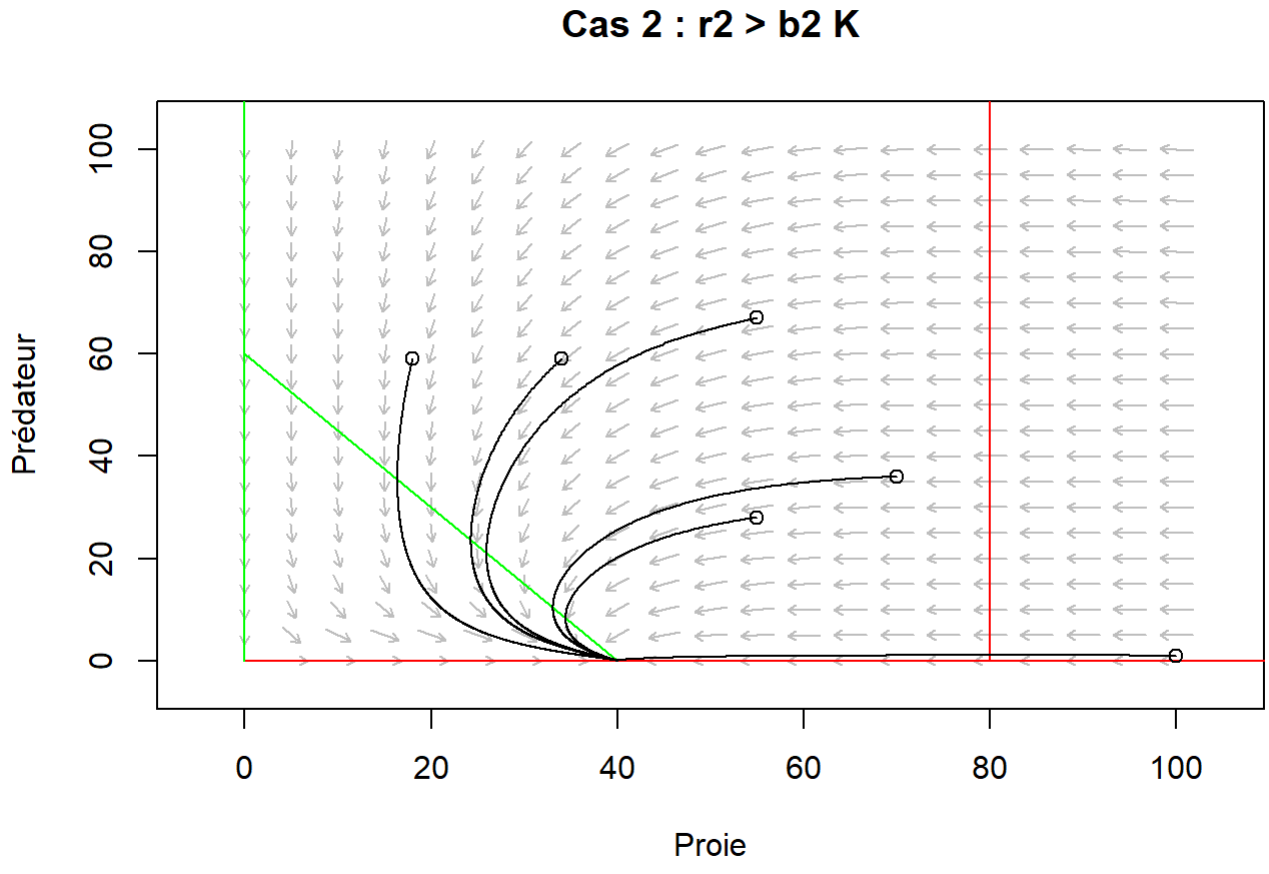

Cas 2 : r2 > b2 K : $\left(\frac{r_2}{b_2}; \frac{r_1}{b_1}\left(1-\frac{r_2}{b_2 K}\right)\right)$ n'est plus dans le cadran positif, donc ne nous intéresse pas dans un cadre de populations (D'autant que l'on sait qu'avec une condition initialle avec N et P positifs, nous resterons positifs puisque les axes sont des trajectoires). Le point fixe (0,0) est point selle et (K,0) est LAS. Toutes les trajectoires (partant de conditions initiales >0) vont converger vers ce dernier point fixe.

Voici par exemple les trajectoires représentées en fonction du temps :

```
parametres=c(r1=0.6,r2=0.8,b1=0.01,b2=0.01,K=40)
numericalSolution(Proiepredlog, y0 = c(100,1), tlim =c(0, 100), type = "one", parameters = pa
rametres, col = c("blue", "red"), ylab = "Effectifs",ylim=c(0,100))
```




Et voici un portrait de champ pour ce modèle, avec plusieurs trajectoires :




Modèle de compétition avec croissance logistique :

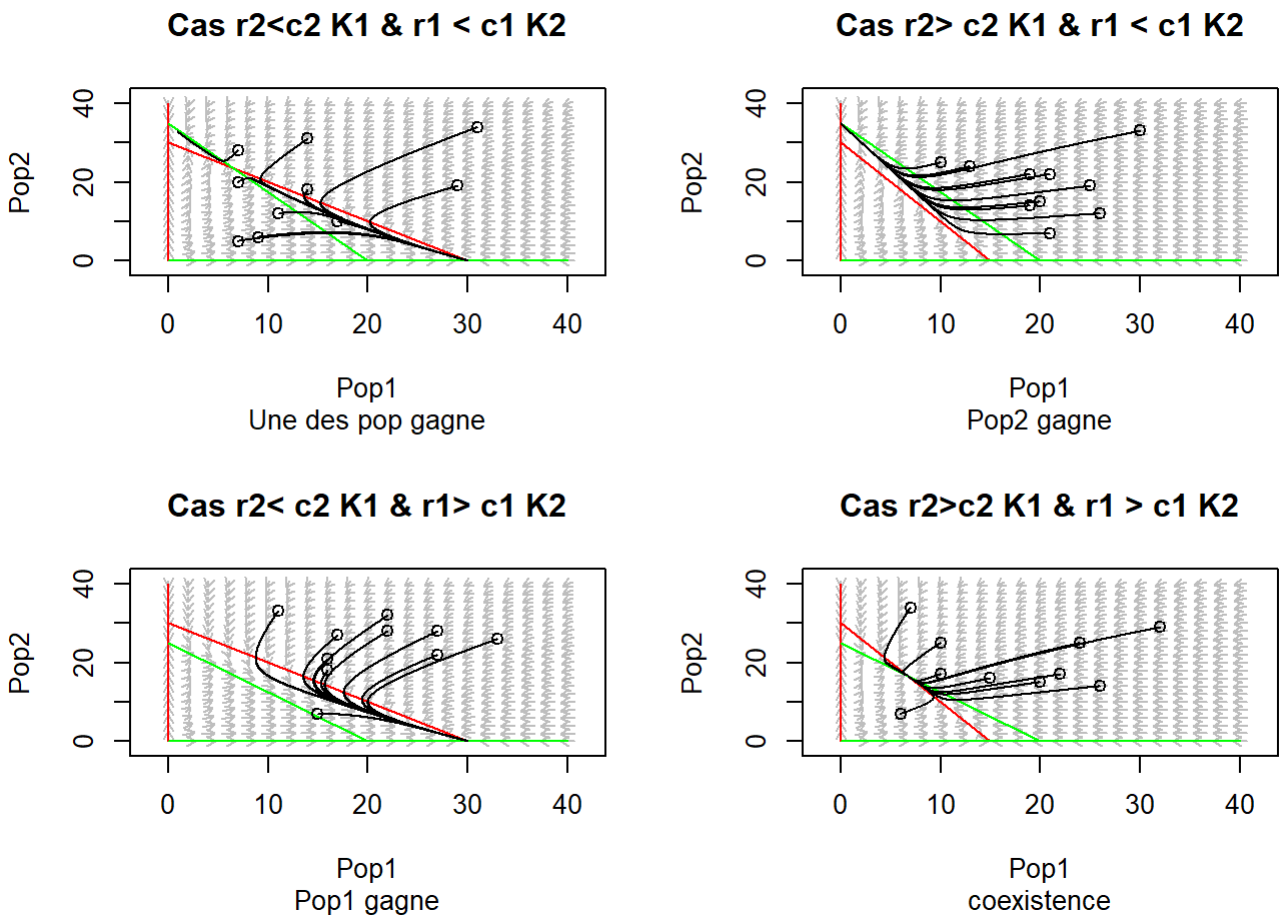
$$\begin{cases} \frac{dx}{dt} &= r_1 x \left(1 - \frac{x}{K_1}\right) - b_1 xy \\ \frac{dy}{dt} &= r_2 y \left(1 - \frac{y}{K_2}\right) - b_2 xy \end{cases}$$

Voir le document annexe pour l'étude qualitative détaillée de ce modèle. Ci-dessous un script permettant de représenter graphiquement le portrait de champ (+ quelques trajectoires) dans les 4 cas possibles :

```
Competlog<- function(t, y, parameters) {
  x <- y[1]
  y <- y[2]
  r1 <- parameters[1]
  r2 <- parameters[2]
  c1 <- parameters[3]
  c2 <- parameters[4]
  K1 <- parameters[5]
  K2 <- parameters[6]
  dy <- numeric(2)
  dy[1] <- r1*x*(1-x/K1) - c1*x*y
  dy[2] <- r2*y*(1-y/K2) - c2*x*y
  list(dy)
}

parametres1=c(r1=0.3,r2=0.2,c1=0.01,c2=0.01,K1=30,K2=35)
parametres2=c(r1=0.3,r2=0.2,c1=0.01,c2=0.01,K1=15,K2=35)
parametres3=c(r1=0.3,r2=0.2,c1=0.01,c2=0.01,K1=30,K2=25)
parametres4=c(r1=0.3,r2=0.2,c1=0.01,c2=0.01,K1=15,K2=25)
Param=data.frame(parametres1,parametres2,parametres3,parametres4)
cas=c("Cas r2<c2 K1 & r1 < c1 K2","Cas r2> c2 K1 & r1 < c1 K2","Cas r2< c2 K1 & r1> c1 K2","C
as r2>c2 K1 & r1 > c1 K2")
comp=c("Une des pop gagne","Pop2 gagne","Pop1 gagne","coexistence")
Nmax=40
Tmax=100

par(mfrow=c(2,2))
for (i in 1:4){
  parametres=Param[,i]
flowField(Competlog, xlim = c(0, Nmax), ylim = c(0, Nmax),parameters = parametres,add = FALS
E,state.names=c("Pop1","Pop2"))
lines(c(0,0),c(0,Nmax),col="red") #Isocline N1=0 (pour N1)
lines(c(parametres[5],0),c(0,parametres[1]/parametres[3]),col="red") #Isocline N2=r1/c1(1-N1/
K1) (pour N1)
lines(c(0,Nmax),c(0,0),col="green") #Isocline N2=0 (pour N2)
lines(c(parametres[2]/parametres[4],0),c(0,parametres[6]),col="green") #Isocline N1=r2/c2(1-N
2/K2) (pour N2)
title(cas[i],sub=comp[i])
for (i in 1:10){
  X0=sample(5:35,2)
  trajectory(Competlog,X0,tlim=c(0,100),parameters=parametres)
  }
}
```




# Un modèle épidémiologique

Faire l'analyse qualitative et étudier numériquement le système suivant :

$$\begin{cases} \frac{dS}{dt} &= r_1 S - \beta SI + \gamma I \\ \frac{dI}{dt} &= \beta SI - \gamma I - \delta I \end{cases}$$

avec :

$r_1$ - Taux de croissance de la population

$\beta$ - Taux de transmission du virus

$\gamma$ - Taux de guérison (sans immunité)

$\delta$ - Mortalité dûe au virus

Une base de départ pour vos études (pour une visualisation correcte) sera de considérer que les paramètres sont compris entre 0 et 1, et que la zone d'étude couvre de 0 à quelques centaines d'individus.

Voici ce que que donne l'analyse qualitative du modèle.

Points fixes ?

$$\begin{cases} r_1 S-\beta SI+\gamma I=0 \\ \beta SI-\gamma I-\delta I=0 \end{cases} \Leftrightarrow \begin{cases} r_1 S-\beta SI+\gamma I=0 \\ I(\beta S-\gamma-\delta)=0 \end{cases} \Leftrightarrow \begin{cases} r_1 S=0 \\ I=0 \end{cases} \quad ou \quad \begin{cases} r_1 S=(\beta S-\gamma) I \\ S=\frac{\gamma+\delta}{\beta} \end{cases}$$

$$\Leftrightarrow \begin{cases} S=0 \\ I=0 \end{cases} \quad ou \quad \begin{cases} r_1 S=\delta I \\ S=\frac{\gamma+\delta}{\beta} \end{cases} \Leftrightarrow \begin{cases} S=0 \\ I=0 \end{cases} \quad ou \quad \begin{cases} I=r_1 \frac{\gamma+\delta}{\beta\delta} \\ S=\frac{\gamma+\delta}{\beta} \end{cases}$$

On a donc 2 points fixes.

Graphiquement ?

Isoclines I : $\beta SI-\gamma I-\delta I=0 \Leftrightarrow I=0 \quad ou \quad S=\frac{\gamma+\delta}{\beta}$ : axe des abscisses et droite verticale

Isoclines S : $r_1 S-\beta SI+\gamma I=0 \Leftrightarrow r_1 S-(\beta S-\gamma) I=0 \Leftrightarrow I=\frac{r_1 S}{\beta S-\gamma}$ : hyperbole

Stabilité ?

$$Jac(S,I)=\begin{pmatrix} r_1-\beta I & \gamma-\beta S \\ \beta I & \beta S-\gamma-\delta \end{pmatrix}$$

donc $Jac(0,0)=\begin{pmatrix} r_1 & \gamma \\ 0 & -\gamma-\delta \end{pmatrix}$ : Les valeurs propres sont $r_1>0$ et $-\gamma-\delta<0$ donc (0,0) est point selle

$$Jac(S^*,I^*)=\begin{pmatrix} r_1-r_1\frac{\gamma+\delta}{\delta} & -\delta \\ r_1\frac{\gamma+\delta}{\delta} & 0 \end{pmatrix} \cdot \quad Trace=-r_1\frac{\gamma}{\delta}<0 \quad \text{et} \quad Det=r_1(\gamma+\delta)>0$$

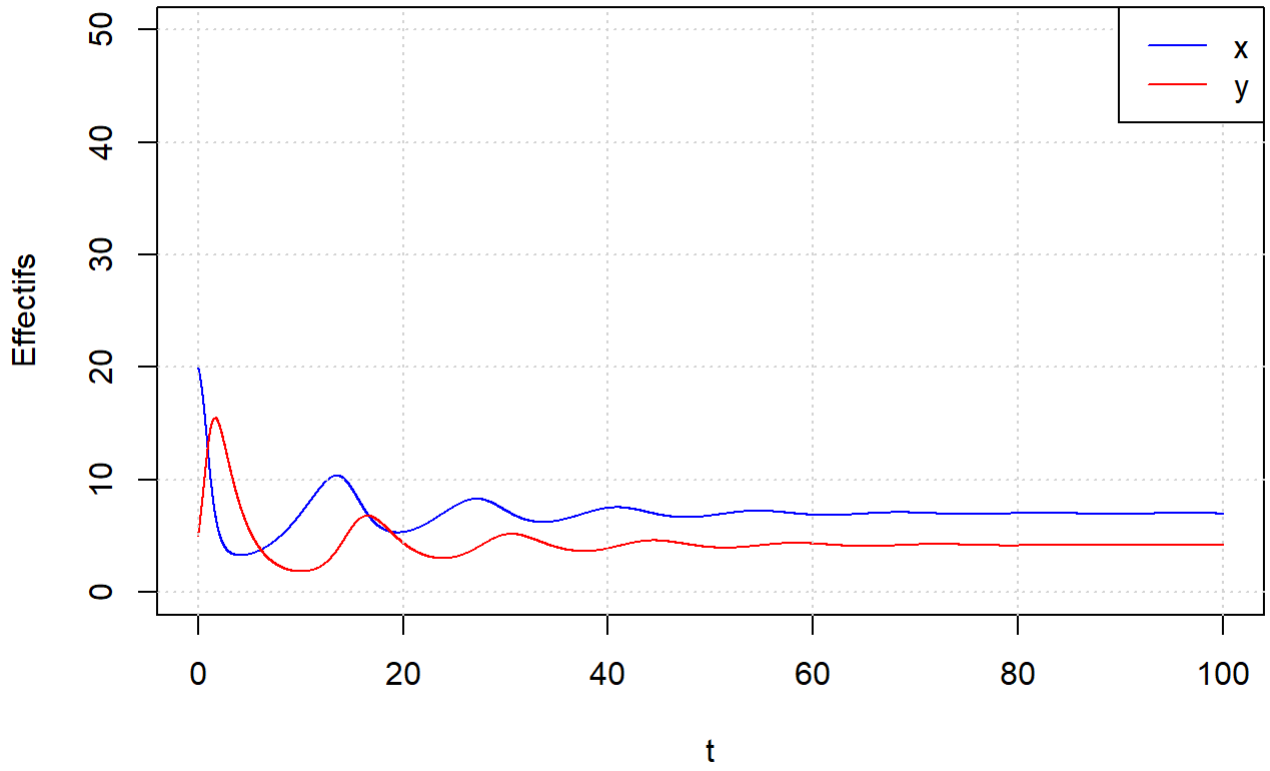
donc le point fixe de coexistence est LAS ; on va tendre vers lui

Commençons par créer la fonction définissant le système

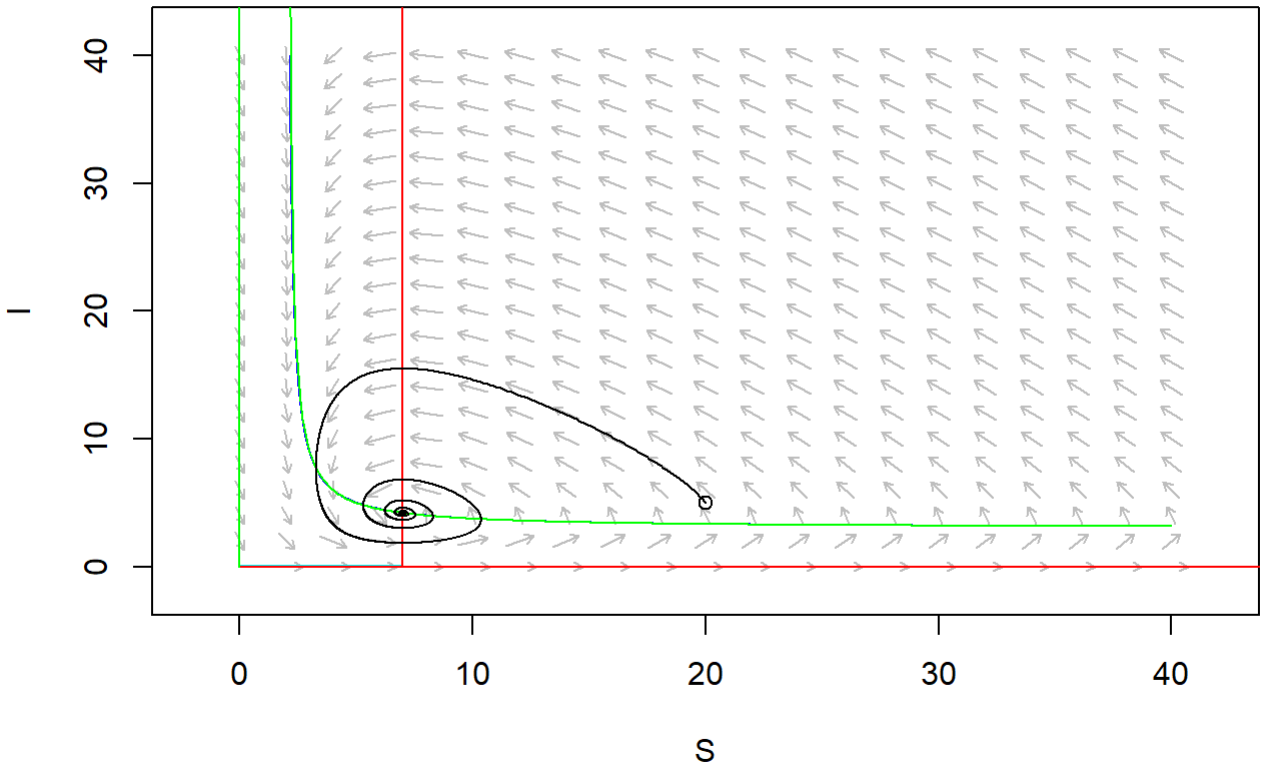
Une “nouveauté” par rapport aux modèles précédement vus est que nous avons une isocline qui n’est pas une droite mais une hyperbole (un morceau d’hyperbole pour le cadran positif), ce qui va compliquer un peu son tracé à la main. La fonction *nullclines()* la trouve bien, mais j’ai quand même inclu dans le code ci-dessous comment faire pour la tracer “seul”

```
Epidemio<- function(t, y, parameters) {
  S <- y[1]
  I <- y[2]
  r1 <- parameters[1]
  beta <- parameters[2]
  gamma <- parameters[3]
  delta <- parameters[4]
  dy <- numeric(2)
  dy[1] <- r1*S - beta*S*I+ gamma*I
  dy[2] <- beta*S*I- gamma*I - delta*I
  list(dy)
}

parametres=c(r1=0.3,beta=0.1,gamma=0.2,delta=0.5)
numericalSolution(Epidemio, y0 = c(20,5), tlim =c(0, 100), type = "one", parameters = paramet
res, col = c("blue", "red"), ylab = "Effectifs",ylim=c(0,50))
```




```
flowField(Epidemio, xlim = c(0, 40), ylim = c(0, 40),parameters = parametres,add = FALSE,stat
e.names=c("S","I"))
nullclines(Epidemio,xlim=c(0,40),ylim=c(0,40),parameters = parametres, points = 500,add.legen
d = FALSE)
lines(c(0,200),c(0,0),col="red") #Isocline I
lines(c(0,0),c(0,200),col="green") #Isocline S
xstar=(parametres[3]+parametres[4])/parametres[2]
lines(c(xstar,xstar),c(0,200),col="red") #Isocline I
x=seq(2,40,0.1)
lines(x,parametres[1]*x/(parametres[2]*x-parametres[3]),col="green")
trajectory(Epidemio,c(20,5),tlim=c(0,100),parameters=parametres)
```




# Et avec 3 équations ?

Mobiliser vos connaissances et compétences pour étudier le système suivant, correspondant à un système Proie-Prédateur-Superprédateur

$$\begin{cases} \frac{dN}{dt} &= r_1 N - b_1 NP \\ \frac{dP}{dt} &= -r_2 P + b_2 NP - c_1 PS \\ \frac{dS}{dt} &= -r_3 S + c_2 PS \end{cases}$$

Voilà ce que donne l'analyse qualitative:


Il y a donc 2 cas. Nous allons tracer ici pour chacun des 2 cas : - Les trajectoires en fonction du temps - Les projections sur chacun des plan (N,P), (N,S) et (P,S) de l'espace des phases (en 3D)

Il serait possible de faire des représentations en 3D, en utilisant des packages dédiés, que nous ne développerons pas ici

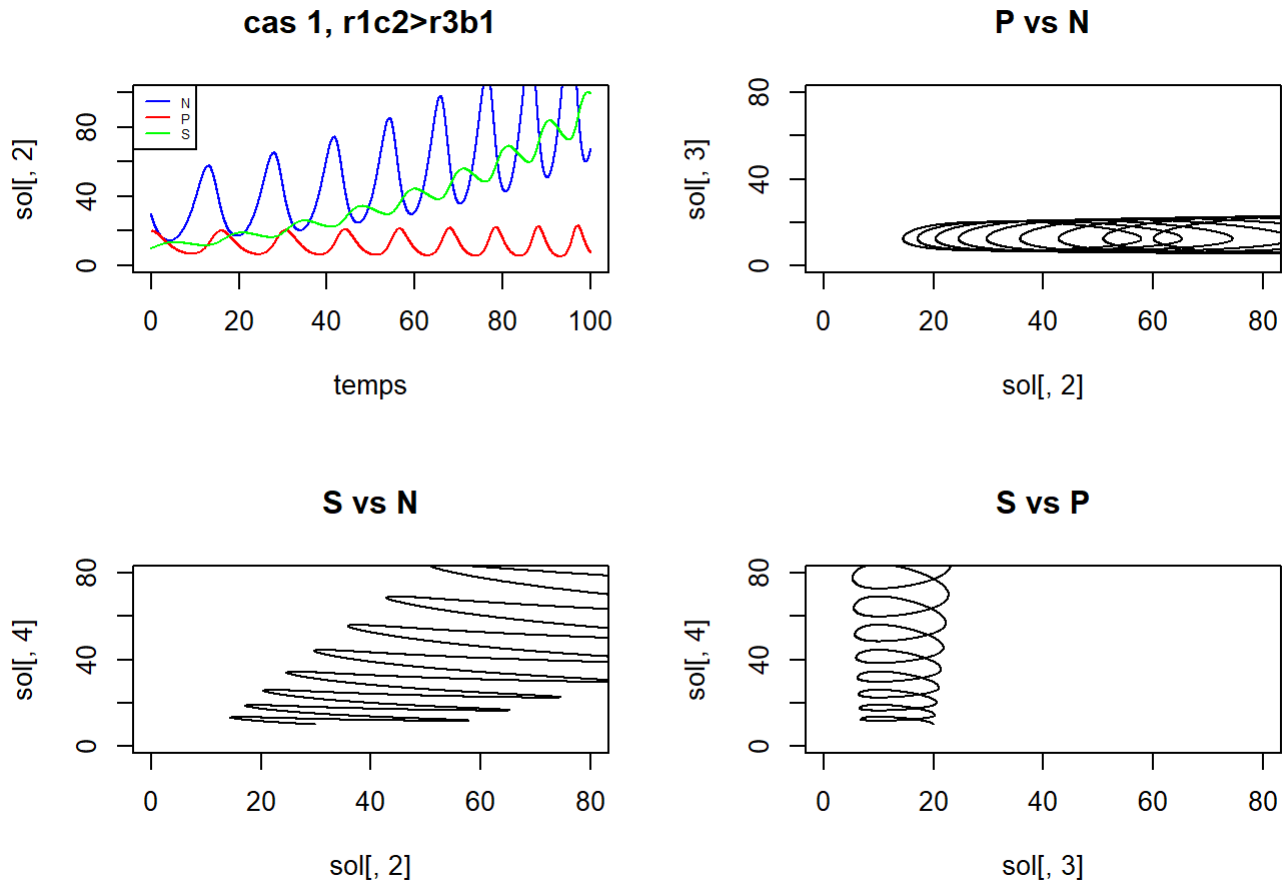
Pour l'étude numérique sous R, nous ne pouvons plus utiliser le package **phaseR** qui ne traite que une ou deux équations ! On peut toujours en revanche exploiter **deSolve**

Commençons par définir la fonction définissant le système :

```
Super<- function(t, y, parameters) {
  N <- y[1]
  P <- y[2]
  S <- y[3]
  r1 <- parameters[1]
  r2 <- parameters[2]
  r3 <- parameters[3]
  b1 <- parameters[4]
  b2 <- parameters[5]
  c1 <- parameters[6]
  c2 <- parameters[7]
  dy <- numeric(3)
  dy[1] <- r1*N - b1*N*P
  dy[2] <- -r2*P+b2*N*P - c1*P*S
  dy[3] <- -r3*S + c2*P*S
  list(dy)
}
```

Représentations dans le cas 1 (lorsque $r_1 c_2 > r_3 b_1$)

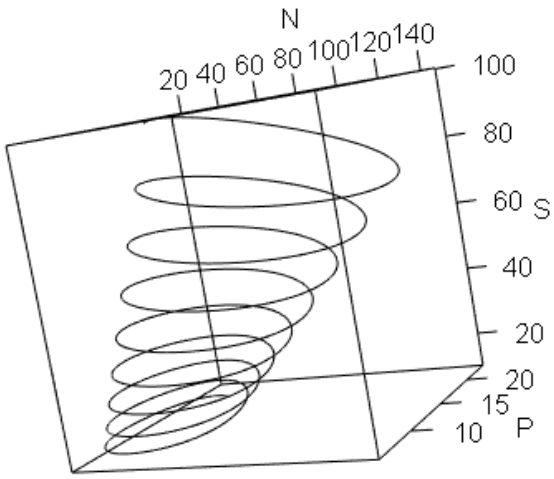
```
par(mfrow=c(2,2))
parametres=c(r1=0.5,r2=0.2,r3=0.1,b1=0.04,b2=0.01,c1=0.009,c2=0.01)
temps=seq(from = 0, to = 100, by = 0.01)
y0=c(30,20,10)
sol <- ode(y = y0, times = temps, func = Super, parms = parametres)
plot(temps,sol[,2],type="l",col="blue",ylim=c(0,100),main="cas 1, r1c2>r3b1")
lines(temps,sol[,3],col="red")
lines(temps,sol[,4],col="green")
legend('topleft',legend=c("N","P","S"),col=c("blue","red","green"),lty=1,cex=0.5)
plot(sol[,2],sol[,3],type="l",xlim=c(0,80),ylim=c(0,80),main="P vs N")
plot(sol[,2],sol[,4],type="l",xlim=c(0,80),ylim=c(0,80),main="S vs N")
plot(sol[,3],sol[,4],type="l",xlim=c(0,80),ylim=c(0,80),main="S vs P")
```




On peut aussi envisager une représentation 3d, par exemple avec la fonction **plot3d()** du package *rgl* :

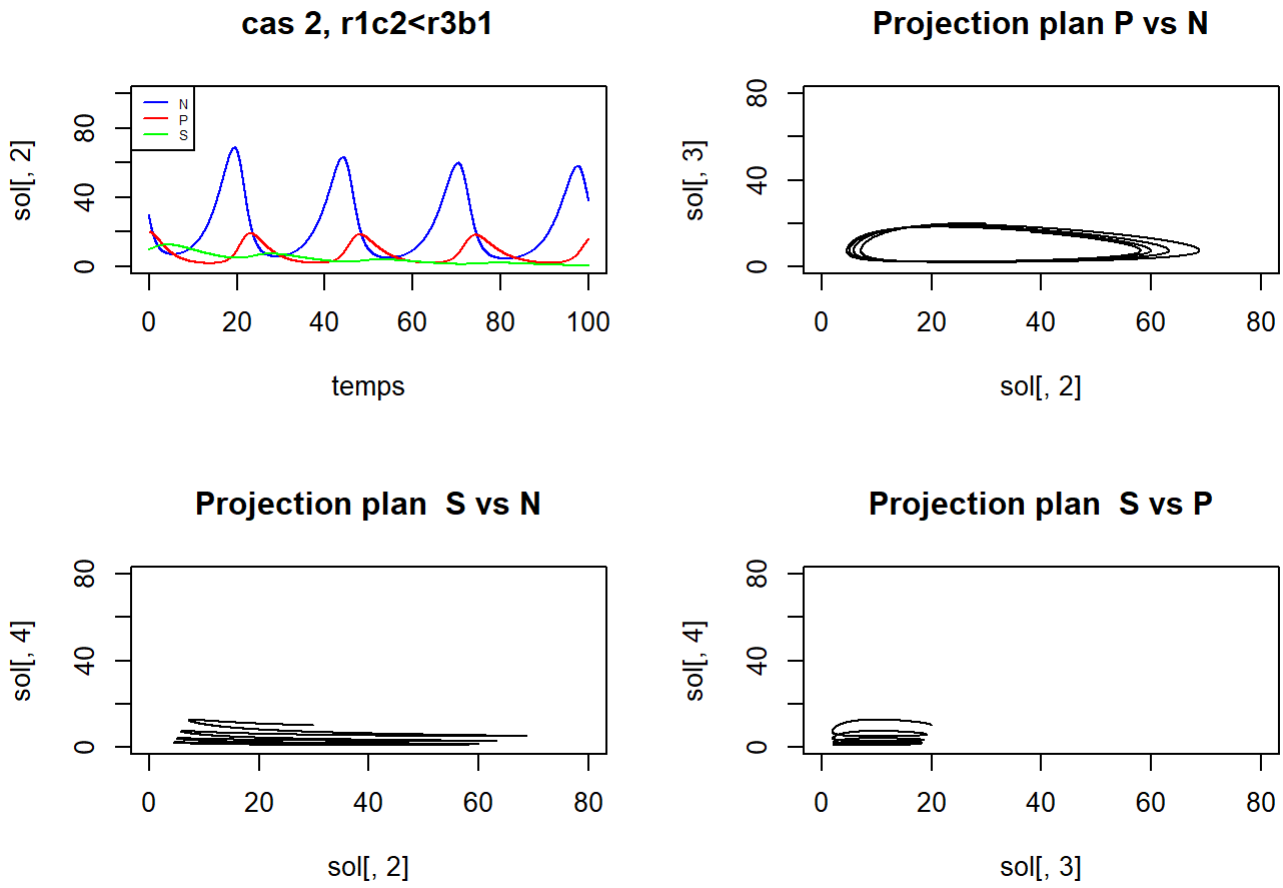
```
library(rgl)
plot3d(sol[,2],sol[,3],sol[,4],type="l",xlab="N",ylab="P",zlab="S")
```

On peut changer l'orientation (avec la souris) pour mieux voir les trajectoires



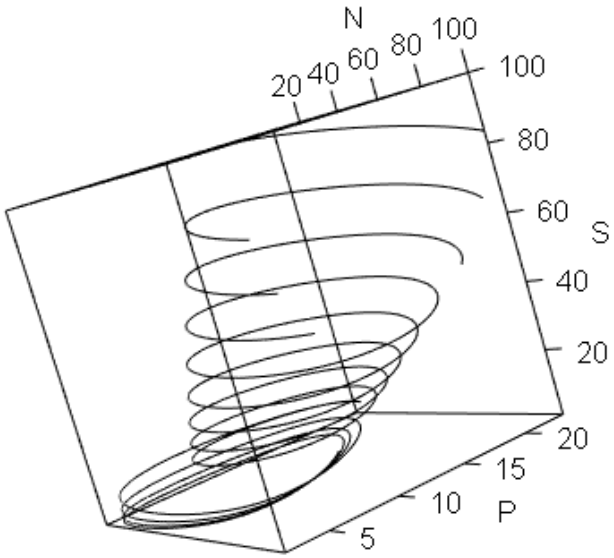

Représentations dans le cas 2 (lorsque $r_1 c_2 < r_3 b_1$)

```
par(mfrow=c(2,2))
parametres=c(r1=0.3,r2=0.2,r3=0.1,b1=0.04,b2=0.01,c1=0.009,c2=0.01)
temps=seq(from = 0, to = 100, by = 0.01)
y0=c(30,20,10)
sol <- ode(y = y0, times = temps, func = Super, parms = parametres)
plot(temps,sol[,2],type="l",col="blue",ylim=c(0,100),main="cas 2, r1c2<r3b1")
lines(temps,sol[,3],col="red")
lines(temps,sol[,4],col="green")
legend('topleft',legend=c("N","P","S"),col=c("blue","red","green"),lty=1,cex=0.5)
plot(sol[,2],sol[,3],type="l",xlim=c(0,80),ylim=c(0,80),main="Projection plan P vs N")
plot(sol[,2],sol[,4],type="l",xlim=c(0,80),ylim=c(0,80),main="Projection plan  S vs N")
plot(sol[,3],sol[,4],type="l",xlim=c(0,80),ylim=c(0,80),main="Projection plan  S vs P")
```




et en 3D :

```
plot3d(sol[,2],sol[,3],sol[,4],type="l",xlab="N",ylab="P",zlab="S")
```




Pour tracer les champs de vecteurs, c'est plus laborieux. Formellement pour connaitre la direction (dans l'espace) en un point, ce n'est pas compliqué, il suffit d'appliquer la fonction définissant le système en ce point.

Par exemple, la commande suivante donnera la direction prise par la trajectoire (=pente dans chaque direction) au point de coordonnées (30,20,10) (on se fiche du premier argument, qui est le temps car notre modèle n'a pas de paramètre dépendant du temps) :

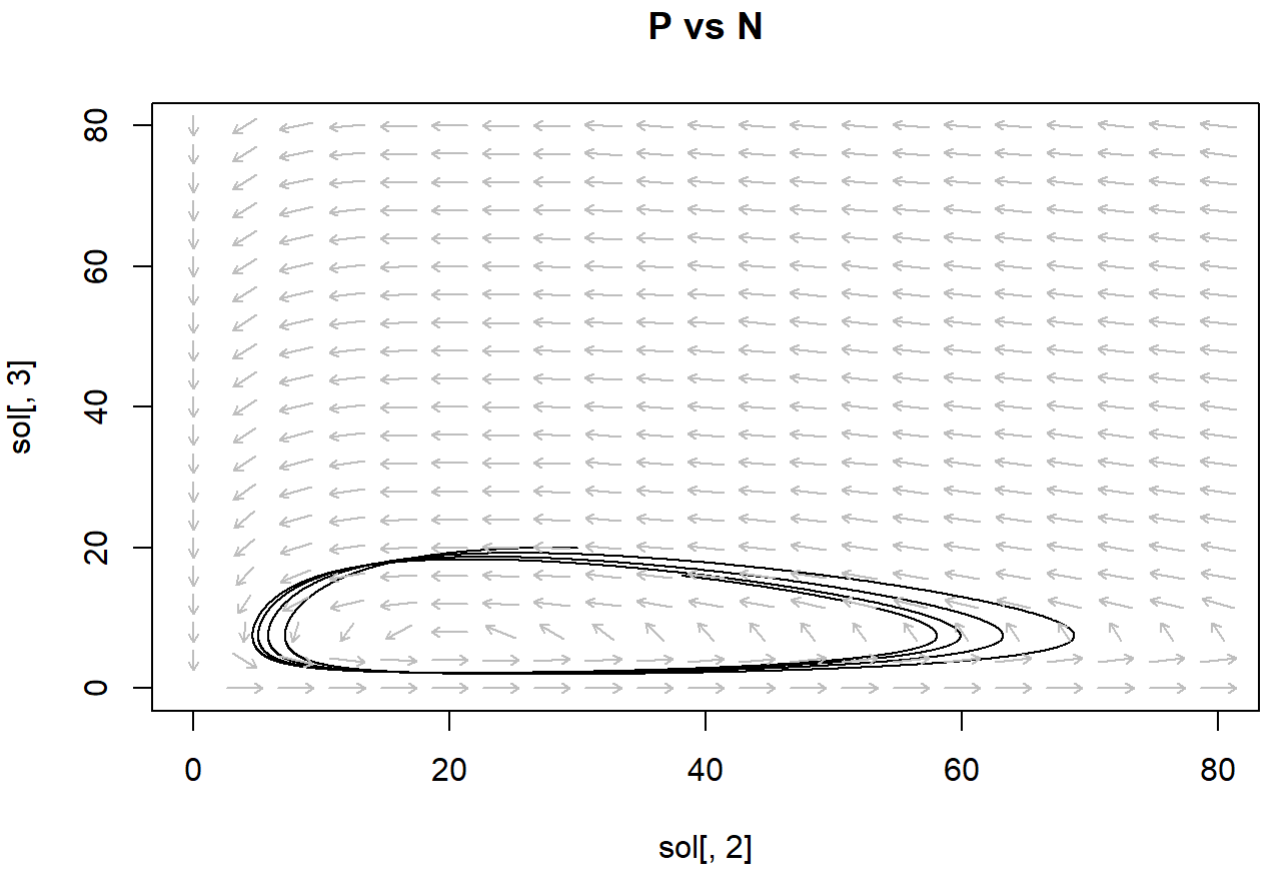
```
Super(0,c(30,20,10),parametres)
```

```
## [[1]]
## [1] -15.0   0.2   1.0
```

En 3 dimensions, on pourrait donc tracer les flèches (encore que l'étape consistant à homogénéiser les longueurs n'est pas triviale), mais la projection sur un plan ne peut que être "moche" car on projetterai à un même point plein de flèches (toutes celles pour toutes les valeurs de la 3ème coordonnée) d'orientations différentes.

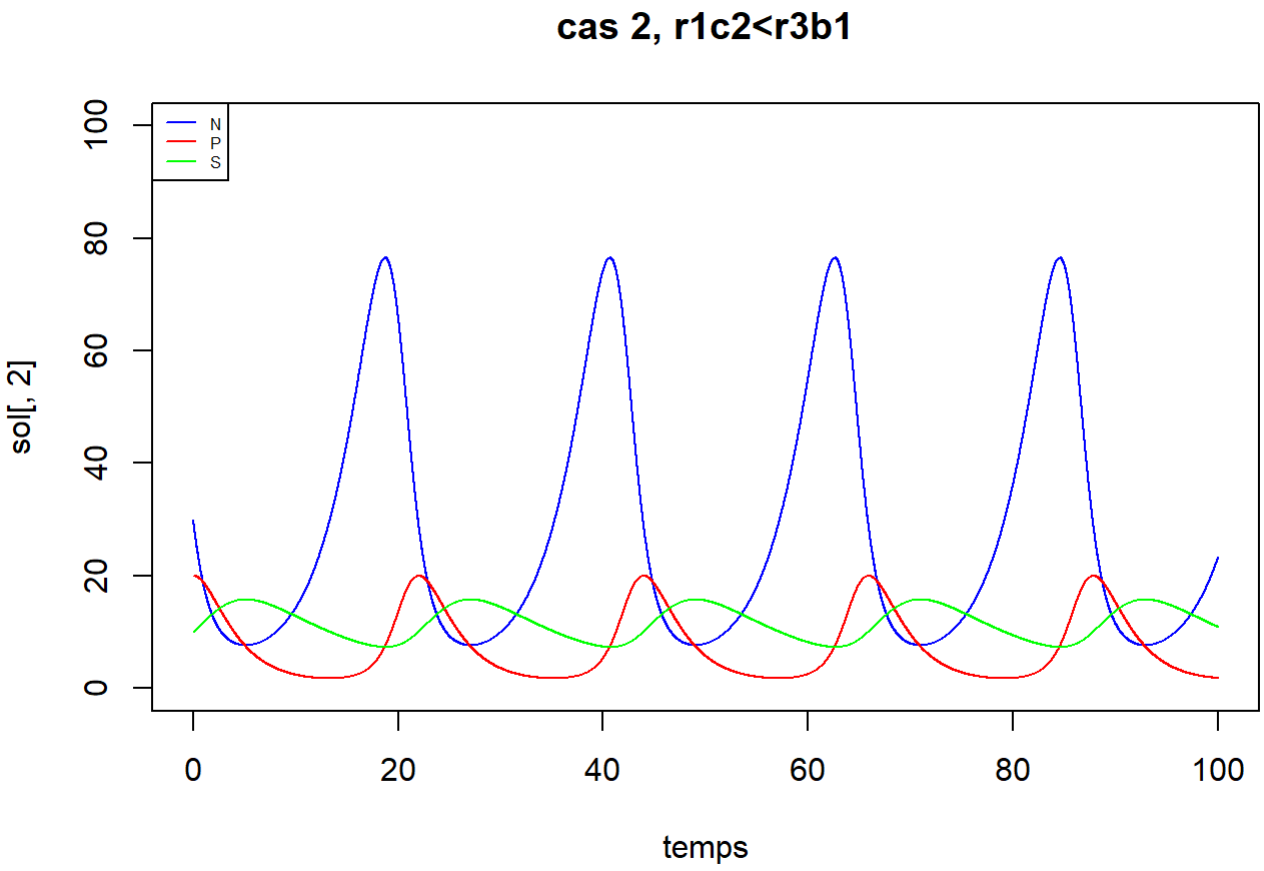
On peut toutefois se servir des fonctions vues précédement pour le tracer dans des cas particulier, notamment en fixant une des valeurs. Par exemple, l'étude qualitative montre que (dans le cas 2), on a une disparition de S. On peut donc s'intéresser notamment à ce qu'il se passe quand la population S sera éteinte : on se retrouve alors avec une équation 2D (en l'occurence le modèle proie-prédateur déjà vu !). On peut donc s'en servir pour tracer le champ de vecteur dans ce plan :

```
plot(sol[,2],sol[,3],type="l",xlim=c(0,80),ylim=c(0,80),main="P vs N")
flowField(Proiepred, xlim = c(0, 80), ylim = c(0, 80),parameters = parametres[c(1,2,4,5)],add
= TRUE)
```




*Rem* Dans le cas spécifique ou (lorsque $r_1 c_2 = r_3 b_1$), nous avons un cycle (en 3 dimensions)

```
parametres=c(r1=0.3,r2=0.2,r3=0.1,b1=0.04,b2=0.01,c1=0.009,c2=0.1*0.04/0.3)
temps=seq(from = 0, to = 100, by = 0.01)
y0=c(30,20,10)
sol <- ode(y = y0, times = temps, func = Super, parms = parametres)
plot(temps,sol[,2],type="l",col="blue",ylim=c(0,100),main="cas 2, r1c2<r3b1")
lines(temps,sol[,3],col="red")
lines(temps,sol[,4],col="green")
legend('topleft',legend=c("N","P","S"),col=c("blue","red","green"),lty=1,cex=0.5)
```




```
plot3d(sol[,2],sol[,3],sol[,4],type="l",xlab="N",ylab="P",zlab="S")
```