David Michel
david.michel@ens-rennes.fr
perso.eleves.ens-rennes.fr/people/David.Michel

ECE 1
Lycée Saint-Vincent
2016-2017

TP8 : Représentation graphique 1 – Instruction plot() – Correction

Q.1 L'instruction **linspace(-%pi,%pi,10)** crée un vecteur de taille 10, dont les coefficients sont répartis régulièrement entre $-\pi$ et π. Elle a le même effet que l'instruction **-%pi :(2*%pi/9) :%pi**.

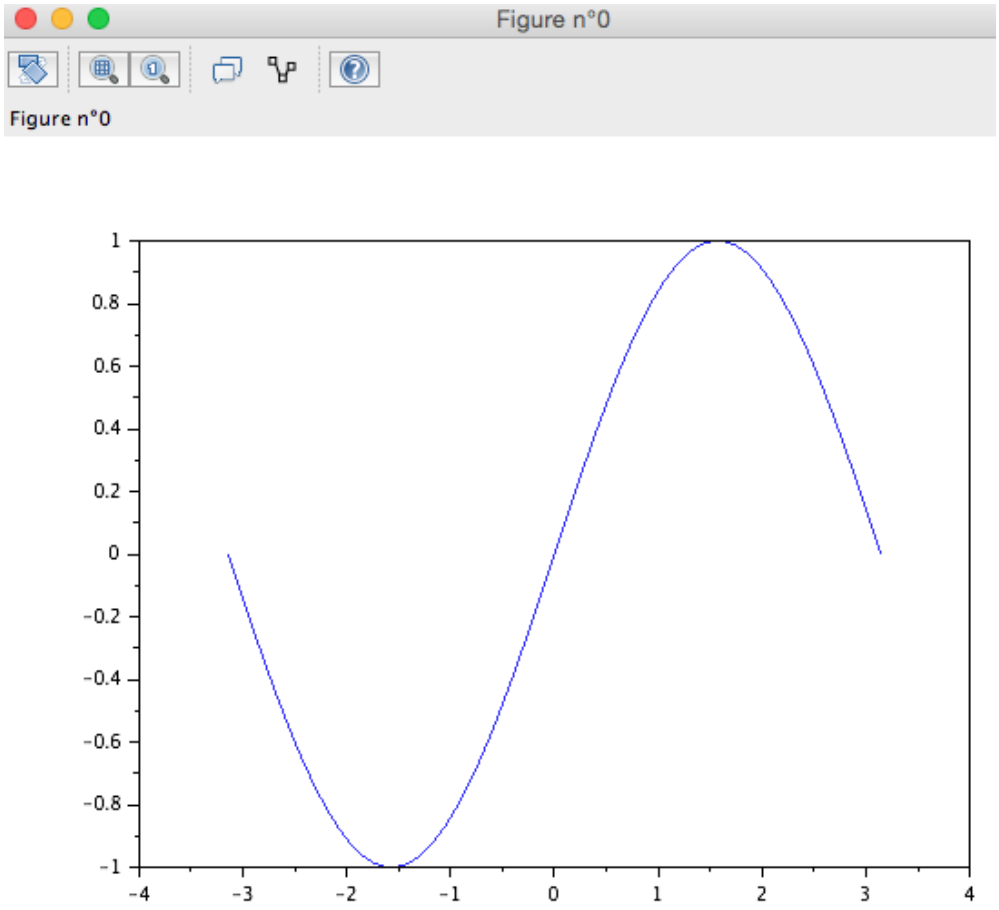
On utilisera l'instruction **linspace** lorsque ce qui nous intéresse est de diviser de façon régulière un intervalle $[a, b]$ donné en commençant par a et en finissant par b. Lorsque c'est la valeur précise des coefficients du vecteur qui nous intéresse, ou l'écart entre deux coefficients, alors on utilisera la syntaxe **a :h :b**.

Q.2 On va utiliser l'instruction **plot**. Il faut donc définir un vecteur d'abscisses et un vecteur d'ordonnées.

Pour les abscisses, ce qui nous intéresse est de recouvrir l'intervalle $[-\pi, \pi]$ avec 200 points. On utilise donc la commande **linspace**. Ensuite, on applique la fonction sinus à notre vecteur.

```
X=linspace(-%pi,%pi,200);
Y=sin(X);
plot(X,Y)
```

et on obtient :

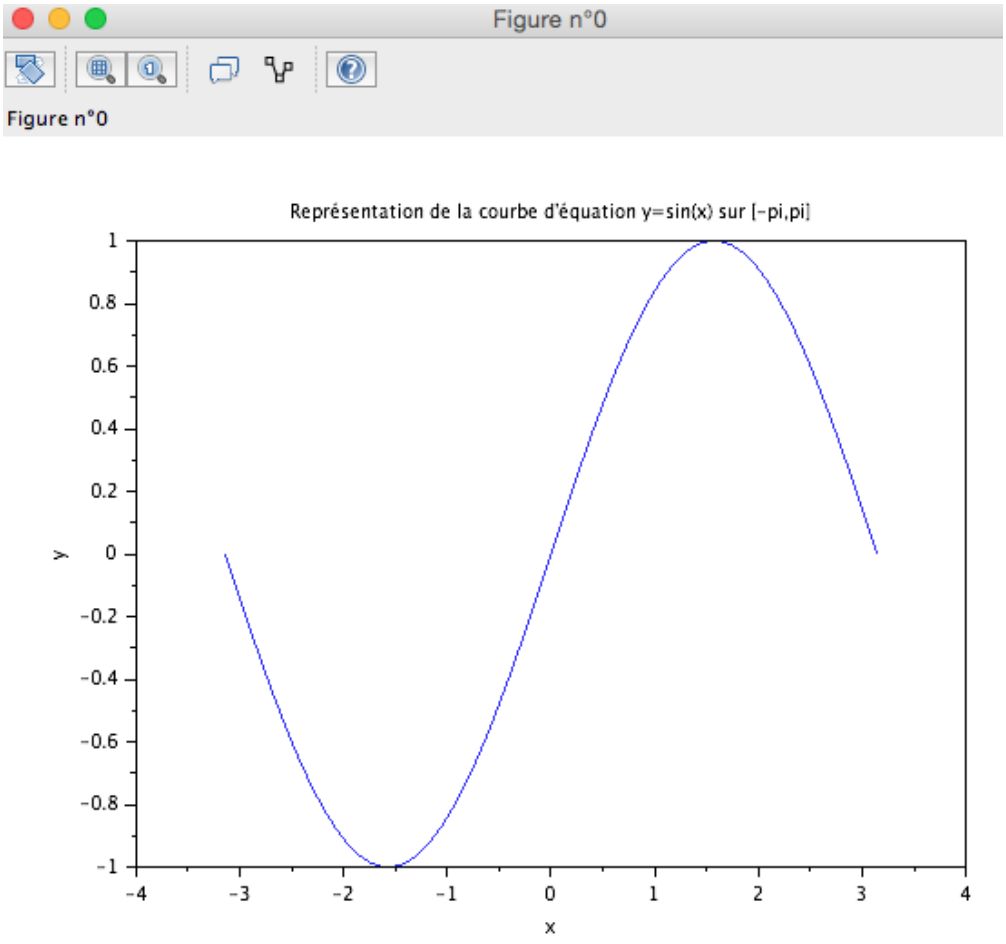


On remarquera les " ;" à la fin des lignes définissant **X** et **Y**. Sans eux, l'exécution de ces lignes dans la console aurait conduit à l'affichage de vecteurs de taille 200, ce qui est assez encombrant et inutile.

Remarque : On peut donner un titre à la figure et nommer les axes en rajoutant à la suite des instructions précédentes

```
xtitle("Représentation de la courbe d"'équation y=sin(x) sur [-pi,pi]","x","y")
```

et on obtient :



Q.3 On veut ici utiliser le même principe, en faisant varier le dernier paramètre donné à l'instruction **linspace**. On commence ouvrir une nouvelle fenêtre graphique.

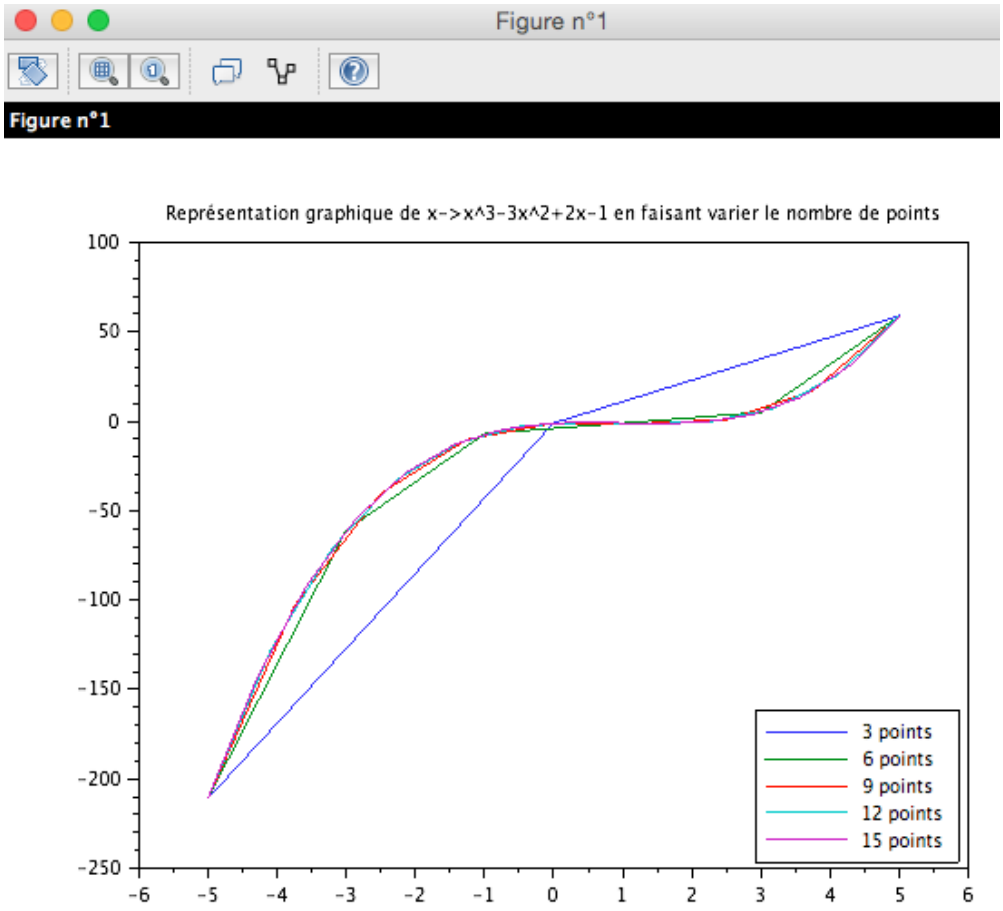
```
scf()

X1=linspace(-5,5,3);
Y1=X1.^3-3*X1.^2+2*X1-1;

X2=linspace(-5,5,6);
Y2=X2.^3-3*X2.^2+2*X2-1;
```

```
X3=linspace(-5,5,9);
Y3=X3.^3-3*X3.^2+2*X3-1;

X4=linspace(-5,5,12);
Y4=X4.^3-3*X4.^2+2*X4-1;

X5=linspace(-5,5,15);
Y5=X5.^3-3*X5.^2+2*X5-1;

plot(X1,Y1,X2,Y2,X3,Y3,X4,Y4,X5,Y5)
```

On agrémente la figure obtenue en ajoutant :

```
title("Représentation graphique de x->x^3-3x^2+2x-1 en faisant varier le nombre de points")
legend(["3 points","6 points","9 points","12 points","15 points"],4);
```

On obtient :



On observe que plus on utilise de points, plus la courbe est "lisse", et correspond mieux à la courbe représentative de la fonction.

Si on avait utilisé cinq fois d'affilée la commande plot, plutôt que de ne l'utiliser qu'une seule fois, les courbes n'auraient pas été (par défaut) de différentes couleurs.

Q.4 Dans cette question on étudie le système de Lotka-Volterra :

$$\begin{cases} x'(t) = \alpha x(t) - \beta x(t)y(t) \\ y'(t) = -\gamma y(t) + \delta x(t)y(t) \end{cases}$$

Donnons une interprétation de ces équations. Le paramètre t représente le temps. On considère un environnement composé de deux populations : des proies, d'effectif $x(t)$ à l'instant t ; des prédateurs, d'effectif $y(t)$ à l'instant t. Les dérivées $x'(t)$ et $y'(t)$ représentent les variations de ces populations. Les paramètres $\alpha, \beta, \gamma, \delta$ décrivent les interactions entre les espèces :

- α est le taux de reproduction des proies. On considère que les proies ont une source illimitée de nourriture et se reproduisent de façon exponentielle. Ce comportement se traduit par le terme $\alpha x(t)$ dans la première équation.
- β est le taux de mortalité des proies du fait de la rencontre avec un prédateur. La prédation est alors représentée dans la première équation par le terme $-\beta x(t)y(t)$: la population tend à diminuer à la rencontre avec des prédateurs.
- γ est le taux de mortalité des prédateurs. La mort naturelle est traduite dans la deuxième équation par le terme $-\gamma y(t)$.
- δ est le taux de reproduction des prédateurs en fonction des proies rencontrées et mangées. Le terme $\delta x(t)y(t)$ traduit la croissance de la population de prédateurs lorsqu'elle peut se nourrir.

Ce modèle produit des résultats en adéquation avec les données obtenues par la Compagnie de la baie d'Hudson au XIX$^{\text{e}}$ siècle sur les populations de lynx et de lièvres des neiges.

Les quelques paragraphes qui suivent présentent le cadre mathématique de cet exercice. Ils sont présents à titre culturel et doivent être **ignorés** en première lecture. On passera donc au paragraphe intitulé "Première lecture".

Dans la suite, on prendra les constantes égales à 1 de sorte qu'on considère le système

$$\begin{cases} x'(t) = x(t) - x(t)y(t) \\ y'(t) = -y(t) + x(t)y(t) \end{cases}$$

et on fixe des conditions initiales $x(0) = x_0$ et $y(0) = y_0$ où $x_0, y_0 \in \mathbb{R}_+$. On se sait pas résoudre analytiquement ce système d'équations différentielles. On va tenter d'en calculer une solution approchée.

La méthode repose sur le rappel suivant :

$$x'(t) = \lim_{\Delta t \to 0} \frac{x(t+\Delta t) - x(t)}{\Delta t}, \qquad y'(t) = \lim_{\Delta t \to 0} \frac{y(t+\Delta t) - y(t)}{\Delta t}.$$

On se donne un temps $T > 0$ jusqu'auquel on voudrait connaître l'évolution de x et y. On découpe l'intervalle $[0, T]$ en sous-intervalles $[n\Delta t, (n+1)\Delta t]$ de longueur $\Delta t > 0$, pour $n = 0, \dots, N = \frac{T}{\Delta t}$.

Plutôt que de calculer $x(t)$, $\forall t \in [0, T]$, on va tenter d'approcher $x(n\Delta t)$, $\forall n \in [\![0, n]\!]$. Ceci permettra d'approcher $x(t)$ par une fonction affine sur chaque sous-intervalle $[n\Delta t, (n+1)\Delta t]$.

Comment construire cette approximation ? On part de l'instant initial, où l'on sait que $x(0) = x_0$ et $y(0) = y_0$. Pour approcher $x(\Delta t)$, on se souvient que si Δt est petit, on peut approcher $x'(0)$ par $\frac{x(\Delta t) - x(0)}{\Delta t}$. On pose donc

$$x_1 = x_0 + \Delta t x'(0)$$

et on espère que c'est une bonne approximation de $x(\Delta t)$. Mais on connait $x'(0)$ grâce à l'équation, donc

$$x_1 = x_0 + \Delta t(x(0) - x(0)y(0)) = x_0 + \Delta t(x_0 - x_0 y_0).$$

De même, on pose

$$y_1 = y_0 + \Delta t(-y_0 + x_0 y_0).$$

Pour l'instant suivant, on approche $x(2\Delta t)$ par $x(\Delta t)+\Delta t x'(\Delta t) = x(\Delta t)+\Delta t(x(\Delta t)-x(\Delta t)y(\Delta t))$. Or on a déjà défini nos approximations de $x(\Delta t)$ et $y(\Delta t)$: ce sont x_1 et y_1. Donc on pose

$$\begin{cases} x_2 = x_1 + \Delta t(x_1 - x_1 y_1) \\ y_2 = y_2 + \Delta t(x_2 - x_2 y_2). \end{cases}$$

De proche en proche, on définit donc les suites $(x_n)_{0\leq n\leq N}$ et $(y_n)_{0\leq n\leq N}$ par

$$\forall n \in [\![0, N-1]\!], \qquad \begin{cases} x_{n+1} = x_n + \Delta t(x_n - x_n y_n) \\ y_{n+1} = y_n + \Delta t(-y_n + x_n y_n) \end{cases}$$

où x_0, y_0 sont connus. On peut désormais résoudre ce système, en espérant que pour tout $n \in [\![0, N]\!]$, x_n et y_n soient des approximations de $x(n\Delta t)$ et $y(n\Delta t)$.

Première lecture On peut tout à fait oublier le cadre précédent et ne se concentrer que sur l'exercice, qui consiste à calculer les termes des suites $(x_n)_{0\leq n\leq N}$ et $(y_n)_{0\leq n\leq N}$.

$\star$ La fonction **Evolution** doit permettre de calculer x_{n+1} et y_{n+1} en connaissant x_n et y_n. On écrit donc les relations satisfaites par ces quantités.

```
function [u,v]=Evolution(x,y,DeltaT)
   u=x+DeltaT*(x-x*y)
   v=y+DeltaT*(-y+x*y)
endfunction
```

$\star$ La fonction **Population** doit calculer tous les termes des suites (x_n) et (y_n) pour $n \leq N$. On va donc créer deux vecteurs **X** et **Y** qui vont contenir les termes successifs de la suite. Présentons deux méthodes. La première présente une façon "d'élargir" des vecteurs. Noter que l'indexation des vecteurs dans Scilab débute à 1 (et non à 0 comme pour nous), ce qui conduit à un décalage dans la numérotation...

```
function [X,Y]=Population(x0,y0,N,DeltaT)
   X=[x0]
   Y=[y0]
   for n=1:N
      [u,v]=Evolution(X(n),Y(n),DeltaT)     //Passage de x_{n-1},y_{n-1} à x_n,y_n
      X=[X;u]         //On ajoute une ligne sous le vecteur X et
                       on y met la valeur de la variable u (x_n)
      Y=[Y;v]         //On ajoute une ligne sous le vecteur Y et
                       on y met la valeur de la variable v (y_n)
   end
endfunction
```

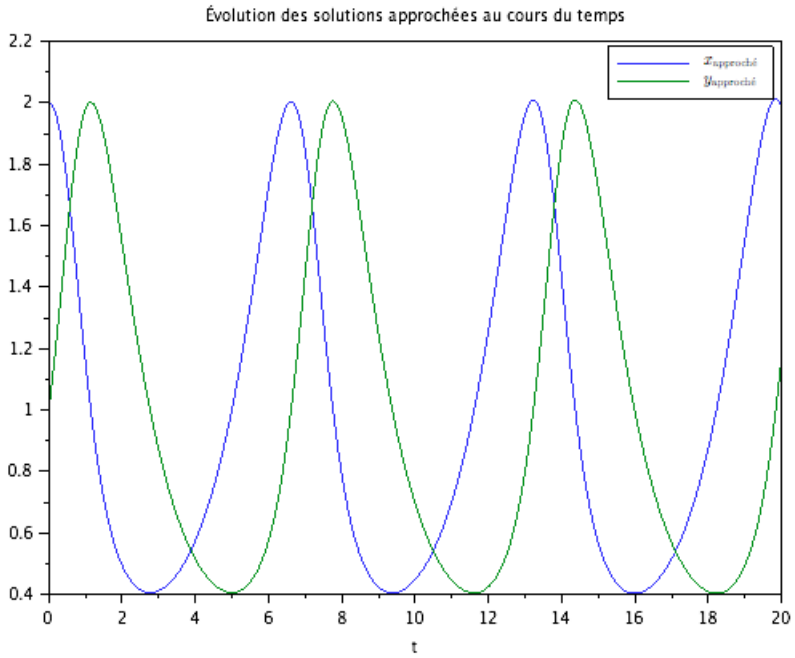
La deuxième méthode consiste à pré-définir deux vecteurs de taille $N+1$ et de les remplir au fur et à mesure. On notera, comme à l'exemple précédent, que l'indexation des coefficients des vecteurs dans Scilab débute à 1 (alors que nous manipulons nous des indices nuls (x_0, y_0)). On n'utilisera pas la fonction **Evolution**, bien que ce soit possible (exercice : comment ?).

```
function [X,Y]=Population(x0,y0,N,DeltaT)
   X=zeros(N+1,1)
   Y=zeros(N+1,1)
   X(1)=x0
   Y(1)=y0
   for n=1:N
      X(n+1)=X(n)+DeltaT*(X(n)-X(n)*Y(n))
      Y(n+1)=Y(n)+DeltaT*(-Y(n)+X(n)*Y(n))
   end
endfunction
```

⋆ On fait tourner notre programme avec les valeurs prescrites par l'énoncé. Pour la première représentation graphique, les abscisses des points à représenter sont $\{0, \Delta t, 2\Delta t, 3\Delta t, \ldots, N\Delta t\}$ donc on utilise la syntaxe **0 :DeltaT :N*DeltaT**.

```
x0=2;
y0=1;
DeltaT=10^-3;
N=2*10^4;
[X,Y]=Population(x0,y0,N,DeltaT);
plot(0:DeltaT:N*DeltaT,X')
plot(0:DeltaT:N*DeltaT,Y')
```

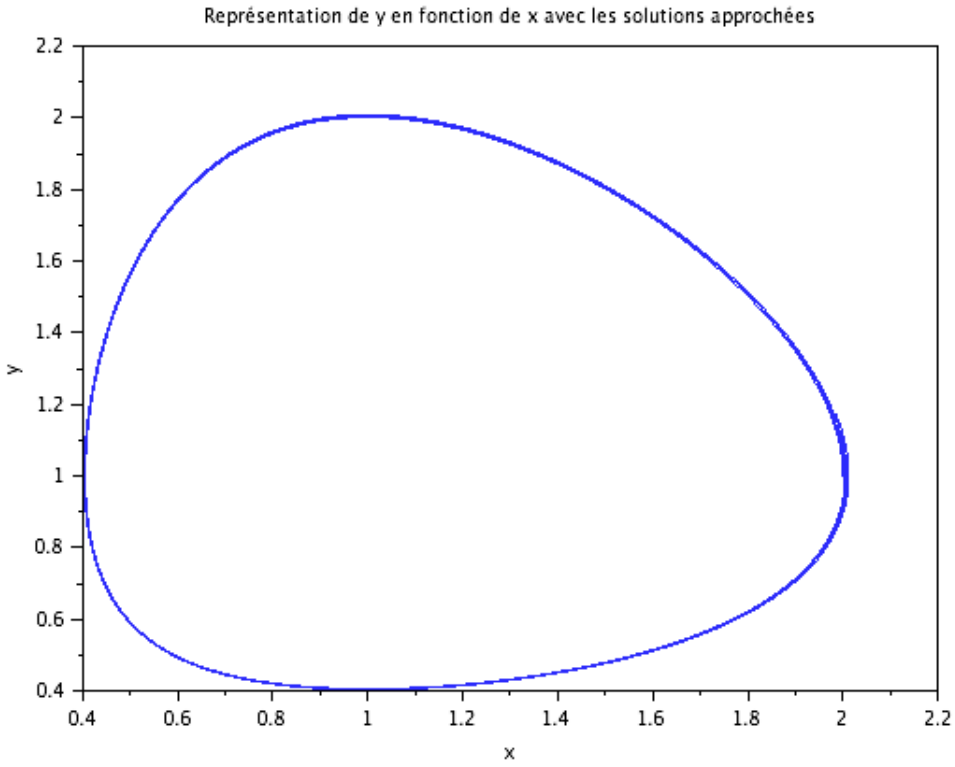
Noter la transposition des vecteurs X et Y pour bien avoir des vecteurs ligne. Ce n'est pas absolument nécessaire car Scilab aurait pris l'initiative d'effectuer la transposition si on ne l'avait pas fait nous-même. Comme on n'a pas demandé à changer de fenêtre graphique, nos deux représentations se superposent et on obtient (après avoir indiqué la légende) :



Nos solutions semblent être périodiques : "un motif semble se répéter".
Pour mieux voir les interactions entre x et y, on peut tracer y et fonction de x.

```
scf()
plot(X',Y')
```

On obtient



Revenons au problème initial : x représente le nombre de proies (une approximation) et y le nombre de prédateurs. On part de $x_0 = 2$ et $y_0 = 1$ (droite de la boucle) donc d'après l'équation différentielle, $y'(0) = 1 > 0$ donc y commence par croître : on parcourt la boucle dans le sens anti-horaire. On commence donc par une situation où il y a beaucoup de proies, donc les prédateurs peuvent manger et se reproduire tandis que la population de proies diminue. Nos solutions approchées sont donc cohérentes, de ce point de vue là, avec notre modèle initial. La population de prédateurs continue d'augmenter jusqu'à ce qu'ils soient trop nombreux par rapport aux proies (haut de la boucle) et la population de prédateurs commence à diminuer : ils ne peuvent pas tous manger. Cependant, les prédateurs survivants se nourrissent et la population de proies continue de diminuer. Jusqu'à atteindre un minimum non nul (gauche de la boucle). Il y a trop peu de proies : elles peuvent se cacher et les prédateurs ne les mangent pas ! Leur nombre recommence donc à augmenter tandis que la population de prédateurs continue de diminuer. Jusqu'à atteindre un minimum (bas de la boucle) pour lequel il y a suffisamment peu de prédateurs et suffisamment de proies pour que beaucoup de prédateurs trouvent à manger et se reproduisent : la population augmente, tandis que la population de proies continue d'augmenter (moins vite). Et on arrive à nouveau au point de départ : le cycle reprend.

Remarque : Il semble que la courbe obtenue soit une boucle. Cela traduit la périodicité de $(x(t), y(t))$. On peut en fait démontrer rigoureusement (sans approximation) cette périodicité.

Mais, en zoomant un peu, on se rend compte que la courbe que l'on a tracée n'est pas réellement

une boucle : $(x_{\text{approché}}, y_{\text{approché}})$ n'est pas vraiment périodique. Notre approximation (dont on peut montrer qu'elle est convenable, c'est-à-dire que lorsque Δt est de plus en plus petit, l'approximation est de plus en plus proche de la solution réelle) ne conserve pas la périodicité de la solution...