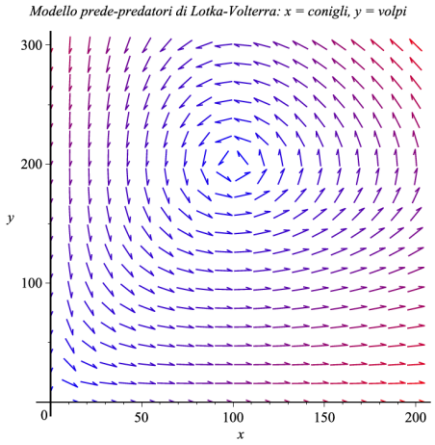
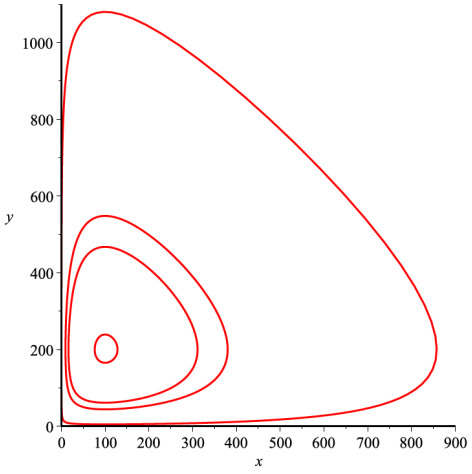
Risolviamo il problema 7. Per soddisfare le prime richieste, diamo i comandi:

```
restart:
with(DEtools):
a:=2: b:=0.01: c:=1: d:=0.01:
LVpp := diff(x(t), t) = x(t)*(a-b*y(t)), diff(y(t), t) = y(t)*(-c+d*x(t)):
vars := x(t), y(t):
DEplot({LVpp}, {vars}, t = 0..8, x = 0..200, y = 0..300, title =
  `Modello prede-predatori di Lotka-Volterra: x = conigli, y = volpi`,
  colour = magnitude);
```




```
ics := [x(0)=100, y(0)=5], [x(0)=80, y(0)=180], [x(0)=50, y(0)=50],
       [x(0)=300, y(0)=150], [x(0)=100, y(0)=200]:
DEplot({LVpp}, {vars}, t = 0..8, x = 0..900, y = 0..1100, [ics], animate =
  true, numframes = 400, arrows = none, thickness = 1, linecolour = red);
```

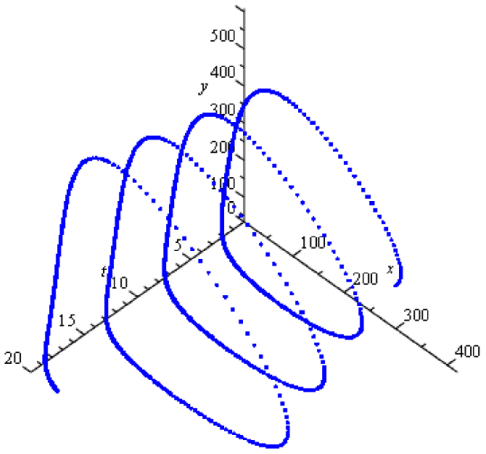


Si noti che nessuna traiettoria parte dallo stato (100, 200), che è un punto di equilibrio (*centro*). Ricordiamo la possibilità di ottenere lo stesso risultato anche col comando:

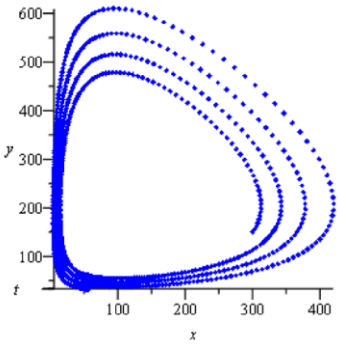
```
phaseportrait({LVpp}, {vars}, t = 0..8, [ics], x = 0..900, y = 0..1100,
      stepsize = 0.02, arrows = none, thickness = 1, linecolour = red);
```

Passiamo quindi a scrivere la procedura di calcolo numerico per ottenere, col metodo di Eulero (diretto), una soluzione approssimata a partire dallo stato (300, 150); prima però dobbiamo esplicitare le equazioni (6), adattate al caso del nostro sistema, rispetto ai "nuovi" valori delle variabili di stato (cioè quelli al passo $n+1$, da calcolare) in funzione dei "vecchi" valori (cioè quelli al passo n, già noti). In conclusione, scriveremo:

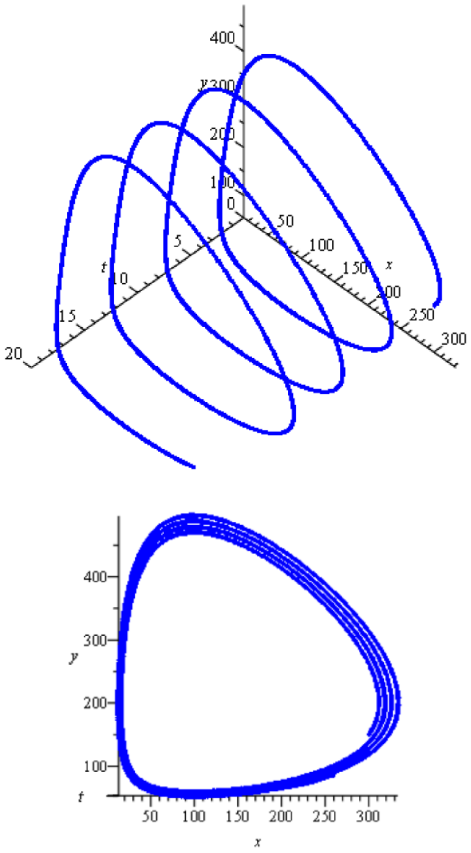
```
with(plots):
Digits:=15:
t[0]:=0: x[0]:=300: y[0]:=150:
v[0]:=[t[0], x[0], y[0]]:
h:=0.02: N:=1000:
for n from 0 to N-1 do
  x[n+1] := x[n] + h*x[n]*(a-b*y[n]);
  y[n+1] := y[n] + h*y[n]*(-c+d*x[n]);
  t[n+1] := t[n] + h;
  v[n+1] := [t[n+1], x[n+1], y[n+1]];
od:
pointplot3d([seq(v[i], i=0..N)], colour=blue, axes=normal, labels=[t,x,y]);
```




```
pointplot3d([seq(v[i], i=0..N)], colour=blue, axes=normal, labels=[t,x,y],
            orientation = [0, 90]);
```



Cambiamo i due valori di `h` e `N`, con le istruzioni `h:=0.005: N:=4000:`, e quindi ripetiamo tutti i comandi evidenziati in giallo; otteniamo le due figure che seguono.



Si noti che, avendo ridotto il passo, la traiettoria tende ad allargarsi più lentamente; ma ciò non è sufficiente a indicarci che, in realtà, l'allontanamento dall'orbita su cui sta lo stato iniziale (300, 150) è dovuto all'accumulo di errori di approssimazione insiti nel procedimento di calcolo eseguito… Dai risultati di questa procedura si potrebbe arguire, erroneamente, che le traiettorie abbiano una forma a spirale − come se il punto di equilibrio fosse un fuoco instabile anziché un centro!

In effetti, ad ogni successiva iterazione, si verifica un errore intrinseco, dovuto sia alla scelta dell'algoritmo e dell'ampiezza *h* del passo, sia all'uso di un numero finito di cifre per rappresentare i valori numerici reali. All'atto pratico, con l'algoritmo di Eulero (importante concettualmente e storicamente, ma di scarsa utilità), si dovrebbe dimezzare l'ampiezza del passo, raddoppiando il numero di iterazioni, fino a quando la differenza tra i risultati non è più apprezzabile; d'altra parte, *h* non può diventare arbitrariamente piccolo, a causa degli errori introdotti dal numero di cifre usate…

Si ruotino ancora le figure sopra ottenute, in entrambi i casi previsti per *h*, al fine di visualizzare l'andamento nel tempo delle due variabili di stato… E si osservi che, essendo tracciato per punti (uno per ogni passo nel tempo), il grafico permette pure di vedere dove il sistema si muove più velocemente lungo la traiettoria: ciò avviene in corrispondenza dei punti più distanziati. Volendo unire i punti del grafico, si usi il comando `spacecurve` al posto di `pointplot3d`, sugli stessi argomenti.