Sistemas1

April 11, 2020

1 Introdução

Apresentamos aqui simulações para dois sitemas bidimensionais. Em ambos, usaremos o Método Runge-Kutta Clássico (RK4) com passo fixo (i.e. sem controle adaptativo do tamanho do passo). Os pacotes usados serão *DifferentialEquations* e *Plots* (este último com o seu backend padrão GR())

[1]:
```
using DifferentialEquations
using Plots
gr(fmt=:png)  # use para poder gerar pdf via latex
```

[1]:
```
Plots.GRBackend()
```

2 O Sistema de Lotka-Volterra

O sistema de Lotka-Volterra é um modelo para uma espécie de presas de população x e uma espécie de predadores de população y. Neste modelo assume-se que a população de presas cresceria com uma taxa proporciona à sua quantidade, mas, devido ao contato com predadores, há uma taxa de decrescimento proporcional ao encontro das populações de presas e predadores. Para os predadores, a sua população cresceria a uma taxa proporcional ao encontro com as presas mas, não havendo alimento, decresceria com uma taxa proporcional à sua população. O sistema de EDOs para as populações é descrito por

$$\dot{x} = ax - bxy \tag{1}$$
$$\dot{y} = cxy - dy \tag{2}$$

onde a, b, c e d são parâmetros positivos. Definamos então a função a ser usada

[2]:
```
function lotka_volterra!(du,u,p,t)   # Explicar essa exclamação
    a,b,c,d = p                      # parâmetros
    du[1] = u[1]*(a - b*u[2])
    du[2] = (c*u[1] - d)*u[2]
end
```

[2]:
```
lotka_volterra! (generic function with 1 method)
```

Usaremos as condições iniciais $x_0 = 3$ e $y_0 = 1$, com parâmetros $a = 10$, $b = 10$, $c = 1$ e $d = 1$, no intervalo de tempo $[0, 5]$

[3]:
```
u0 = [3.0,1.0]
p = (10.0,10.0,1.0,1.0)
tspan = (0.0,5.0)
prob = ODEProblem(lotka_volterra!,u0,tspan,p)
```

[3]:
```
ODEProblem with uType Array{Float64,1} and tType
Float64. In-place: true
timespan: (0.0, 5.0)
u0: [3.0, 1.0]
```

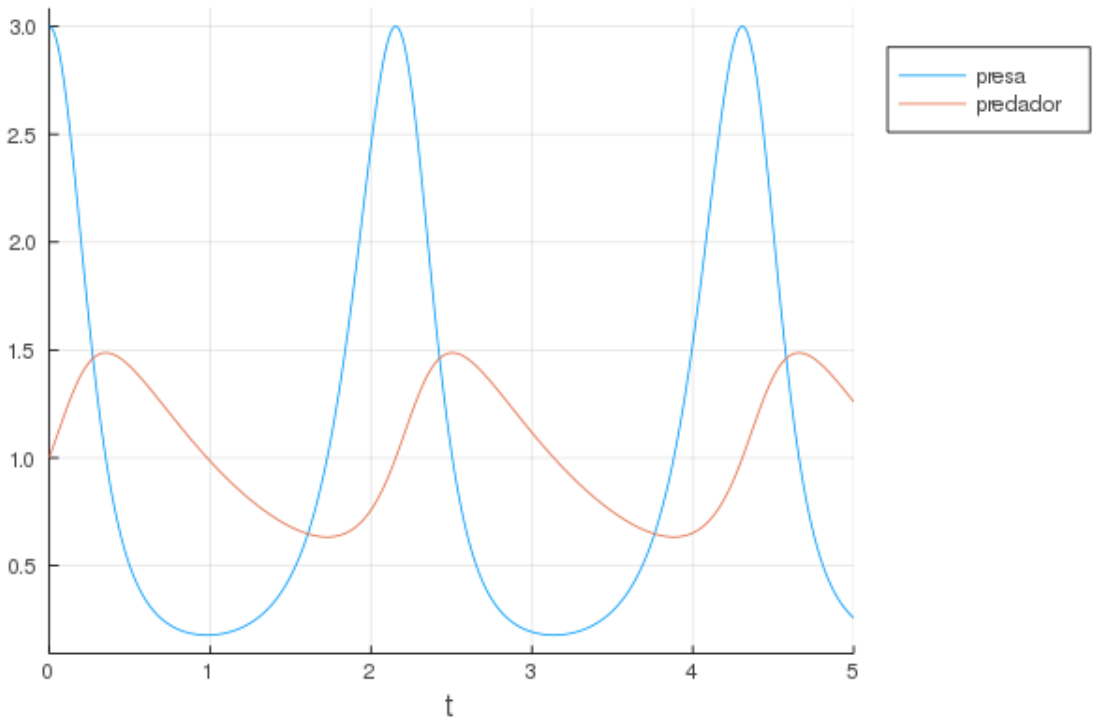
Como não conhecemos a solução exata, é preciso um critério para decidir se a aproximação está boa. Usaremos força bruta. Serão calculadas aproximações com passos h e $h/2$ até que o erro entre elas nos pontos comuns seja menor do que uma tolerância ϵ. No caso, $\epsilon = 10^{-6}$.

[4]:
```
h = 0.1
sol = solve(prob,RK4(),adaptive=false,dt=h)
erro = 1.0
while erro > 10.0^(-6.0)
    x = sol[1,:]
    y = sol[2,:]
    vt = sol.t
    h = 0.5*h
    sol = solve(prob,RK4(),adaptive=false,dt=h)
    vu = sol(vt)
    erro = maximum([maximum(abs.(vu[1,:]-x)) maximum(abs.(vu[2,:]-y))])
end
println("h = $h e erro = $erro")
```

```
h = 0.00625 e erro = 4.0140615209516284e-7
```

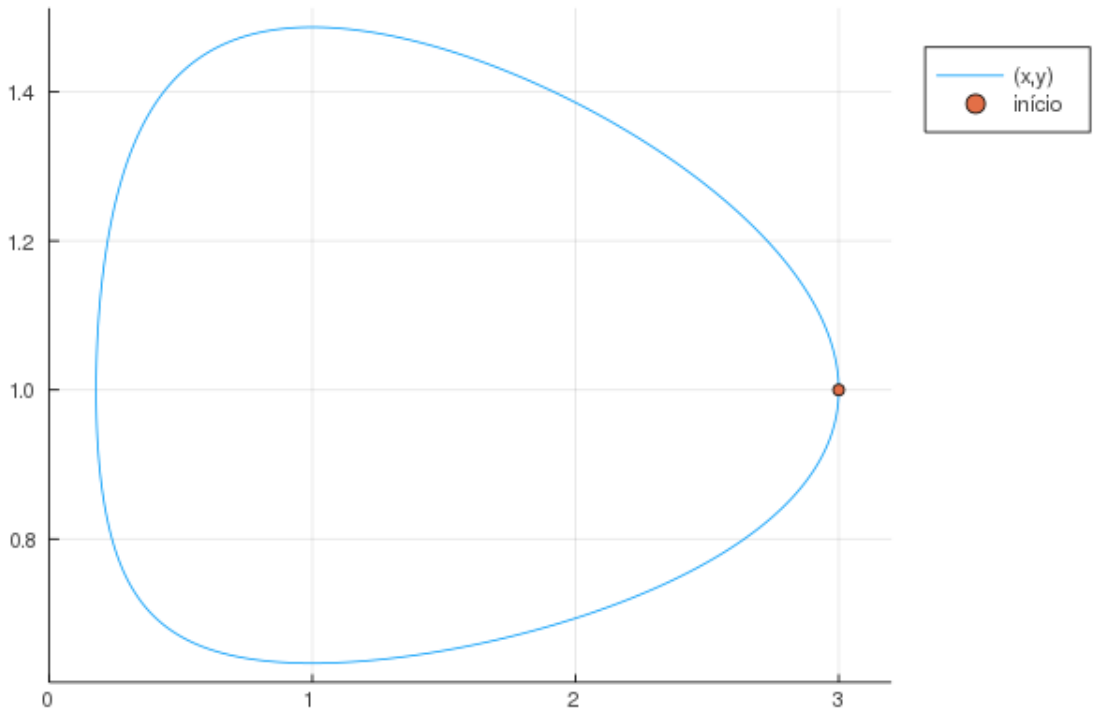
[5]:
```
plot(sol,legend=:outertopright,label=["presa" "predador"])
```

[5]:



[6]:
```
plot(sol,vars=(1,2),xlim=(0,3.2),label = "(x,y)")
 ↪                  # Espaço de fase
scatter!(u0[1,:],u0[2,:],legend=:outertopright,label="início")
```

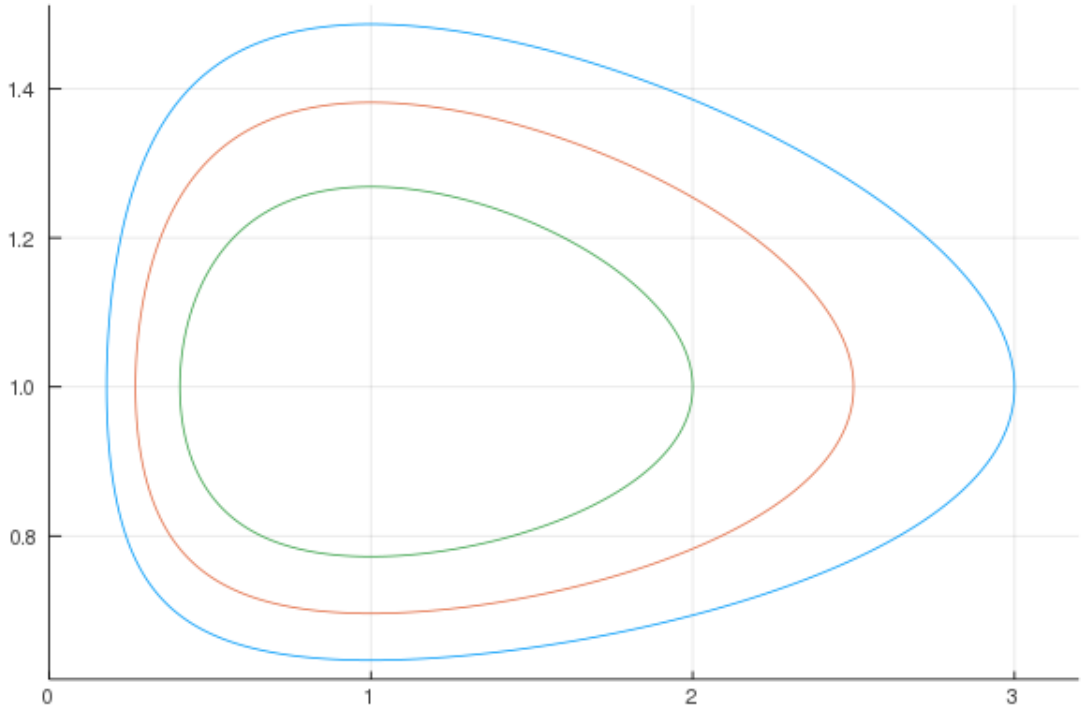
[6]:



Observe que as soluções são periódicas e portanto a órbita é fechada. Pode-se demonstrar teoricamente que, qualquer que seja a condição inicial no primeiro quadrante, a órbita está contida no primeiro quadrante e é fechada. Vejamos algumas órbitas

[7]:
```
u0 = [3.0,1.0]
prob = ODEProblem(lotka_volterra!,u0,tspan,p)
sol = solve(prob,RK4(),adaptive=false,dt=h)
fig = plot(sol,vars=(1,2),xlim=(0,3.2),legend=false)
u0 = [2.5,1.0]
prob = ODEProblem(lotka_volterra!,u0,tspan,p)
sol = solve(prob,RK4(),adaptive=false,dt=h)
fig = plot!(sol,vars=(1,2),xlim=(0,3.2))
u0 = [2.0,1.0]
prob = ODEProblem(lotka_volterra!,u0,tspan,p)
sol = solve(prob,RK4(),adaptive=false,dt=h)
fig = plot!(sol,vars=(1,2),xlim=(0,3.2))
```

[7]:



3 Um outro Modelo Presa-Predador

O modelo de Lotka-Volterra pode ser modificado para descrever outras situações. Usaremos o seguinte sistema de EDOs, onde novamente x é a população de presas e y é a população de predadores:

$$\dot{x} = x(1.2 - x) - \frac{xy}{x + 0.2} \tag{3}$$

$$\dot{y} = \frac{1.5xy}{x + 0.2} - y \tag{4}$$

Tente interpretar as equações.

[8]:
```
function presa_predador!(du,u,p,t)
    du[1] = u[1]*(1.2 - u[1] - u[2]/(u[1]+0.2))
    du[2] = u[2]*(1.5*u[1]/(u[1]+0.2) -1.0)
end
```

[8]:
```
presa_predador! (generic function with 1 method)
```

Serão feitas duas simulações no intervalo de tempo $[0, 40]$: uma com as condições iniciais $x_0 = 1.0$, $y_0 = 0.75$, e a outra com $x_0 = 0.75$, $y_0 = 0.25$.

[9]:
```
tspan = (0.0, 40.0);
```

[10]:
```
u0_a = [1.0,0.75]                                    # Primeira condição␣
 ↪inicial
prob = ODEProblem(presa_predador!,u0_a,tspan)
```

[10]:
```
ODEProblem with uType Array{Float64,1} and tType
Float64. In-place: true
timespan: (0.0, 40.0)
u0: [1.0, 0.75]
```

[11]:
```
h = 0.1
sol_a = solve(prob,RK4(),adaptive=false,dt=h)
erro = 1.0
while erro > 10.0^(-6.0)
    x = sol_a[1,:]
    y = sol_a[2,:]
    vt = sol_a.t
    h = 0.5*h
    sol_a = solve(prob,RK4(),adaptive=false,dt=h)
    vu = sol_a(vt)
    erro = maximum([maximum(abs.(vu[1,:]-x)) maximum(abs.(vu[2,:]-y))])
end
println("h = $h e erro = $erro")
```

```
h = 0.05 e erro = 2.7942284264170 78e-7
```

[12]:
```
u0_b = [0.75,0.25] # Segunda condição inicial
prob = ODEProblem(presa_predador!,u0_b,tspan)
```

[12]:
```
ODEProblem with uType Array{Float64,1} and tType
Float64. In-place: true
timespan: (0.0, 40.0)
u0: [0.75, 0.25]
```

[13]:
```
h = 0.1
sol_b = solve(prob,RK4(),adaptive=false,dt=h)
erro = 1.0
while erro > 10.0^(-6.0)
    x = sol_b[1,:]
    y = sol_b[2,:]
    vt = sol_b.t
    h = 0.5*h
    sol_b = solve(prob,RK4(),adaptive=false,dt=h)
    vu = sol_b(vt)
    erro = maximum([maximum(abs.(vu[1,:]-x)) maximum(abs.(vu[2,:]-y))])
end
```

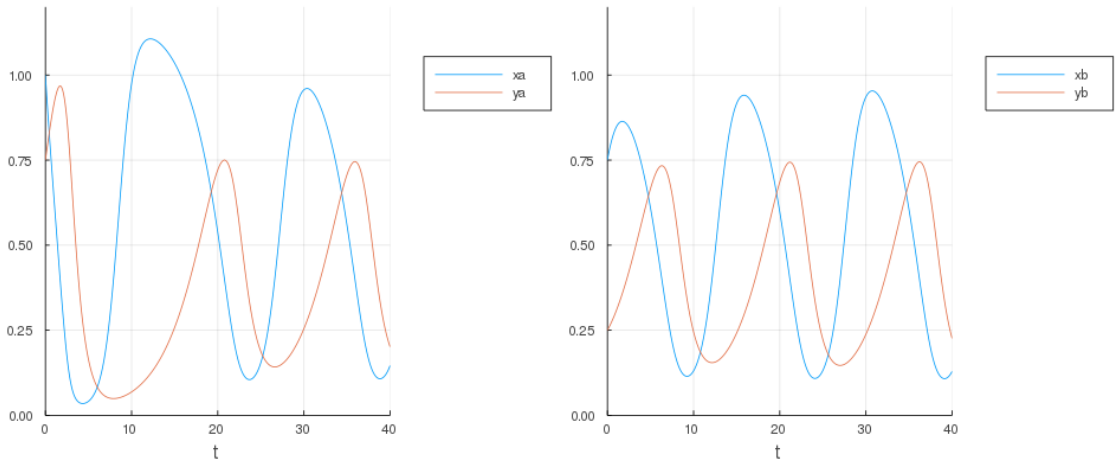
```
println("h = $h e erro = $erro")
```

```
h = 0.05 e erro = 7.09813053667574e-7
```

Vejamos alguns gráficos. Interprete o que você observa.

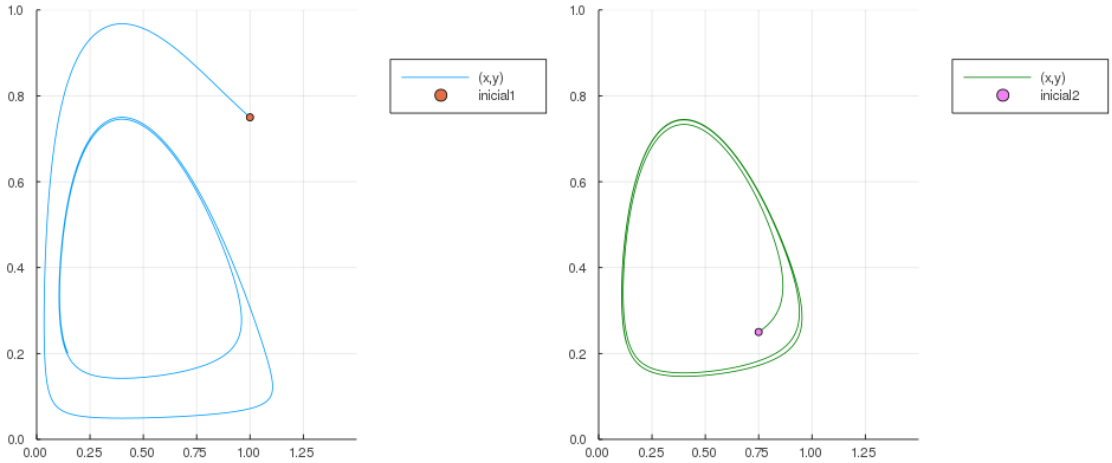
[14]:
```
pa = plot(sol_a,ylim=(0,1.2),legend = :outertopright,label = ["xa" "ya"])
pb = plot(sol_b,ylim=(0,1.2),legend = :outertopright,label = ["xb" "yb"])
plot(pa,pb,size=(950,400))
```

[14]:



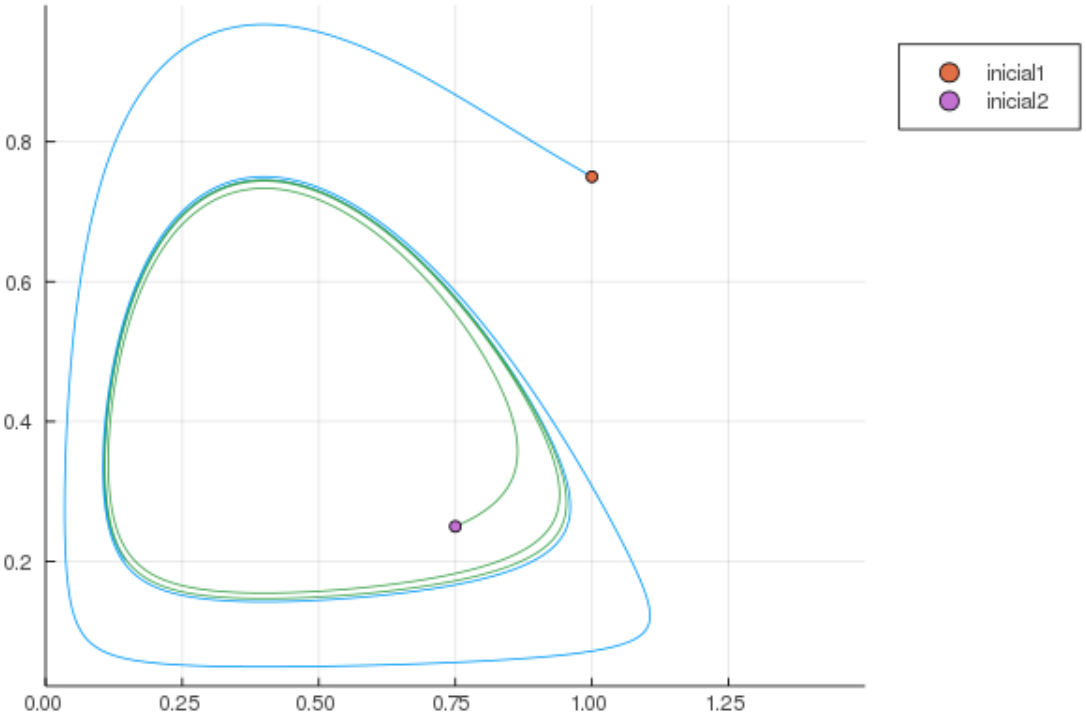
[15]:
```
pa = plot(sol_a, vars = (1, 2),label = "(x,y)")                    #␣
 ↪Espaço de fase para as duas condições
pa = scatter!(u0_a[1,:],u0_a[2,:],label = "inicial1")              #␣
 ↪iniciais. Figuras indivduais
pa = plot(pa, xlim=(0.0,1.5),ylim = (0.0,1.0), legend = :outertopright)
pb = plot(sol_b, vars = (1, 2), color = :green,label = "(x,y)")
pb = scatter!(u0_b[1,:],u0_b[2,:], color = :violet, label = "inicial2")
pb = plot(pb, xlim=(0.0,1.5),ylim = (0.0, 1.0),legend = :outertopright)
plot(pa, pb, size = (950,400))
```

[15]:



[16]:
```
plot(sol_a, vars = (1,2), legend = :outertopright, label = "")  # Espaço de␣
 ↪fase com as duas condições iniciais
scatter!(u0_a[1,:], u0_a[2,:], label = "inicial1")              # juntas.
plot!(sol_b, vars = (1,2), xlim = (0.0,1.5), label = "")
scatter!(u0_b[1,:], u0_b[2,:], label = "inicial2")
```

[16]:



[ ]: