Conditional Dosing Pharmacometric Example

Chris Rackauckas

March 1, 2019

In this example we will show how to model a conditional dosing using the `DiscreteCallbacks`. The problem is as follows. The patient has a drug `A(t)` in their system. The concentration of the drug is given as `C(t)=A(t)/V` for some volume constant `V`. At `t=4`, the patient goes to the clinic and is checked. If the concentration of the drug in their body is below `4`, then they will receive a new dose.

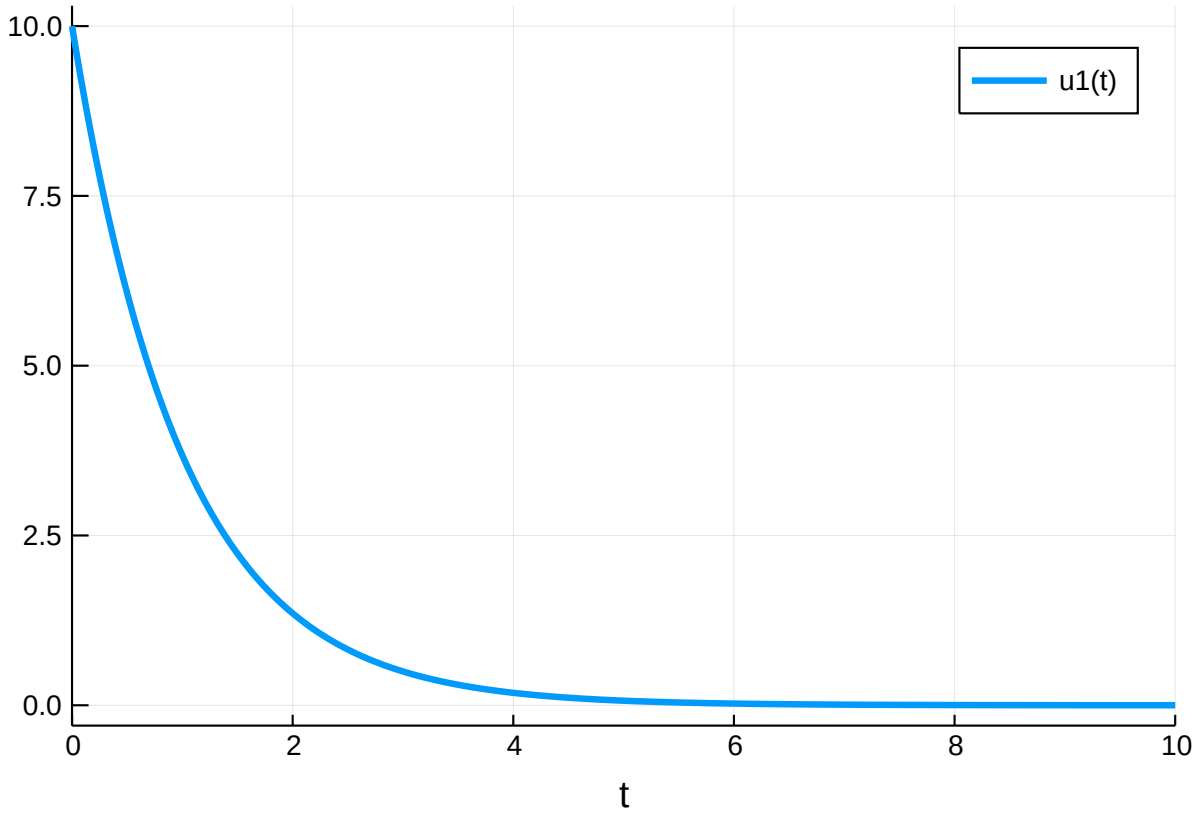
For our model, we will use the simple decay equation. We will write this in the in-place form to make it easy to extend to more complicated examples:

```
using DifferentialEquations
function f(du,u,p,t)
    du[1] = -u[1]
end
u0 = [10.0]
const V = 1
prob = ODEProblem(f,u0,(0.0,10.0))
```

```
ODEProblem with uType Array{Float64,1} and tType Float64. In-place: true
timespan: (0.0, 10.0)
u0: [10.0]
```

Let's see what the solution looks like without any events.

```
sol = solve(prob,Tsit5())
using Plots; gr()
plot(sol)
```



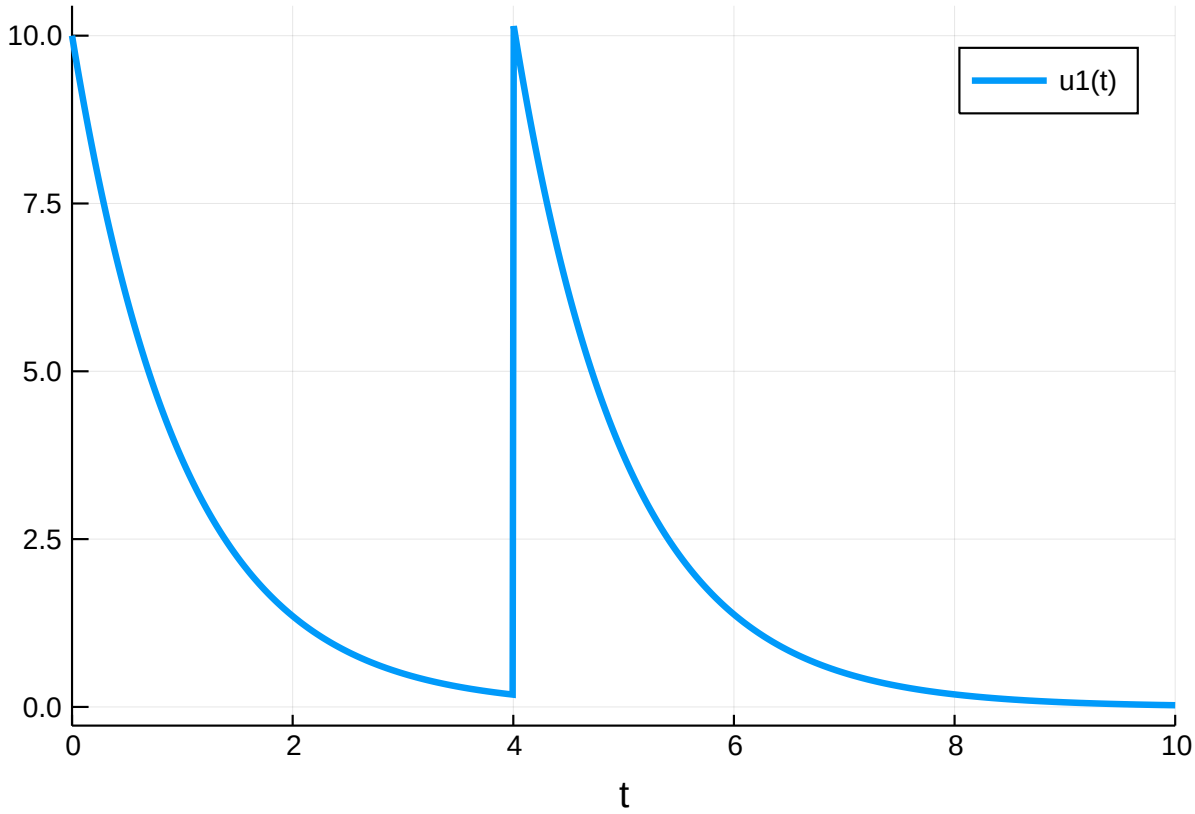
We see that at time `t=4`, the patient should receive a dose. Let's code up that event. We need to check at `t=4` if the concentration `u[1]/4` is `<4`, and if so, add `10` to `u[1]`. We do this with the following:

```
condition(u,t,integrator) = t==4 && u[1]/V<4
affect!(integrator) = integrator.u[1] += 10
cb = DiscreteCallback(condition,affect!)
```

```
DiffEqBase.DiscreteCallback{typeof(Main.WeaveSandBox17.condition),typeof(Ma
in.WeaveSandBox17.affect!),typeof(DiffEqBase.INITIALIZE_DEFAULT)}(Main.Weav
eSandBox17.condition, Main.WeaveSandBox17.affect!, DiffEqBase.INITIALIZE_DE
FAULT, Bool[true, true])
```

Now we will give this callback to the solver, and tell it to stop at `t=4` so that way the condition can be checked:

```
sol = solve(prob,Tsit5(),tstops=[4.0],callback=cb)
using Plots; gr()
plot(sol)
```



Let's show that it actually added 10 instead of setting the value to 10. We could have set the value using `affect!(integrator) = integrator.u[1] = 10`

```
println(sol(4.00000))
```

```
[0.183164]
```

```
println(sol(4.000000000001))
```

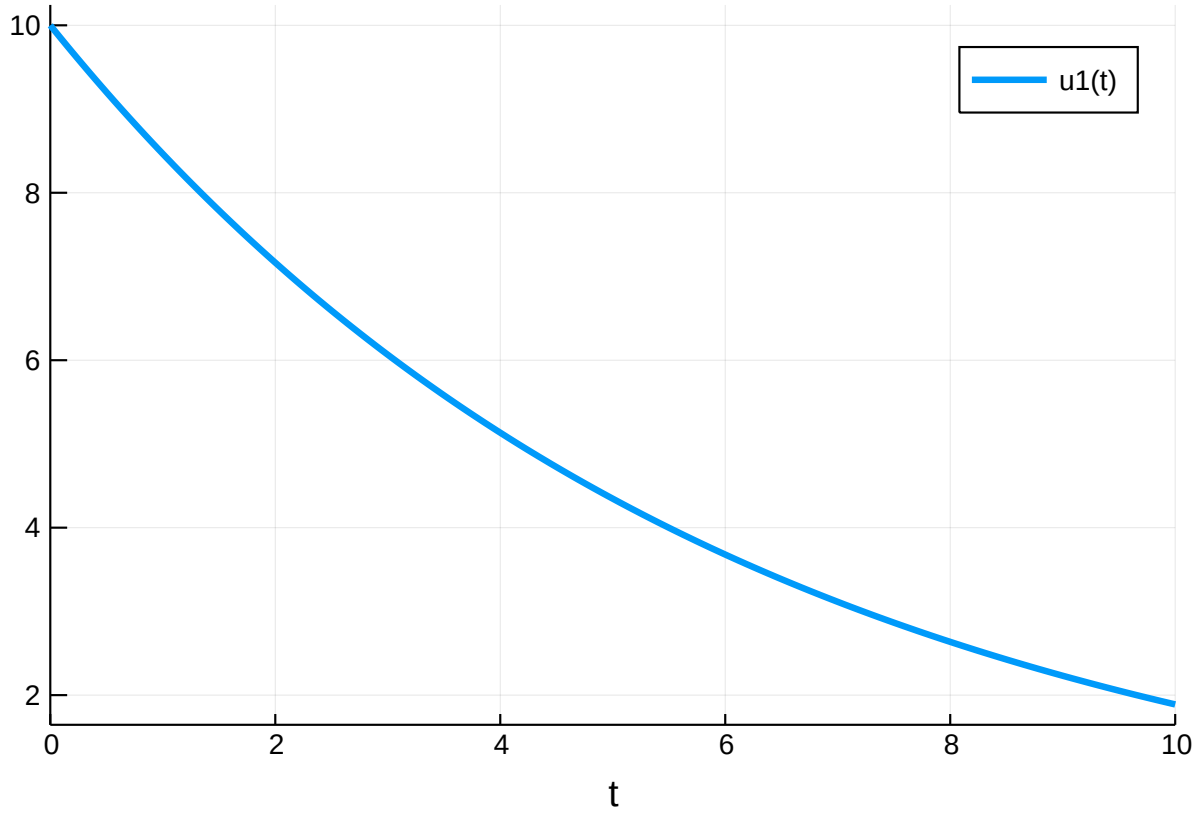
```
[10.1832]
```

Now let's model a patient whose decay rate for the drug is lower:

```
function f(du,u,p,t)
    du[1] = -u[1]/6
end
u0 = [10.0]
const V = 1
prob = ODEProblem(f,u0,(0.0,10.0))
```

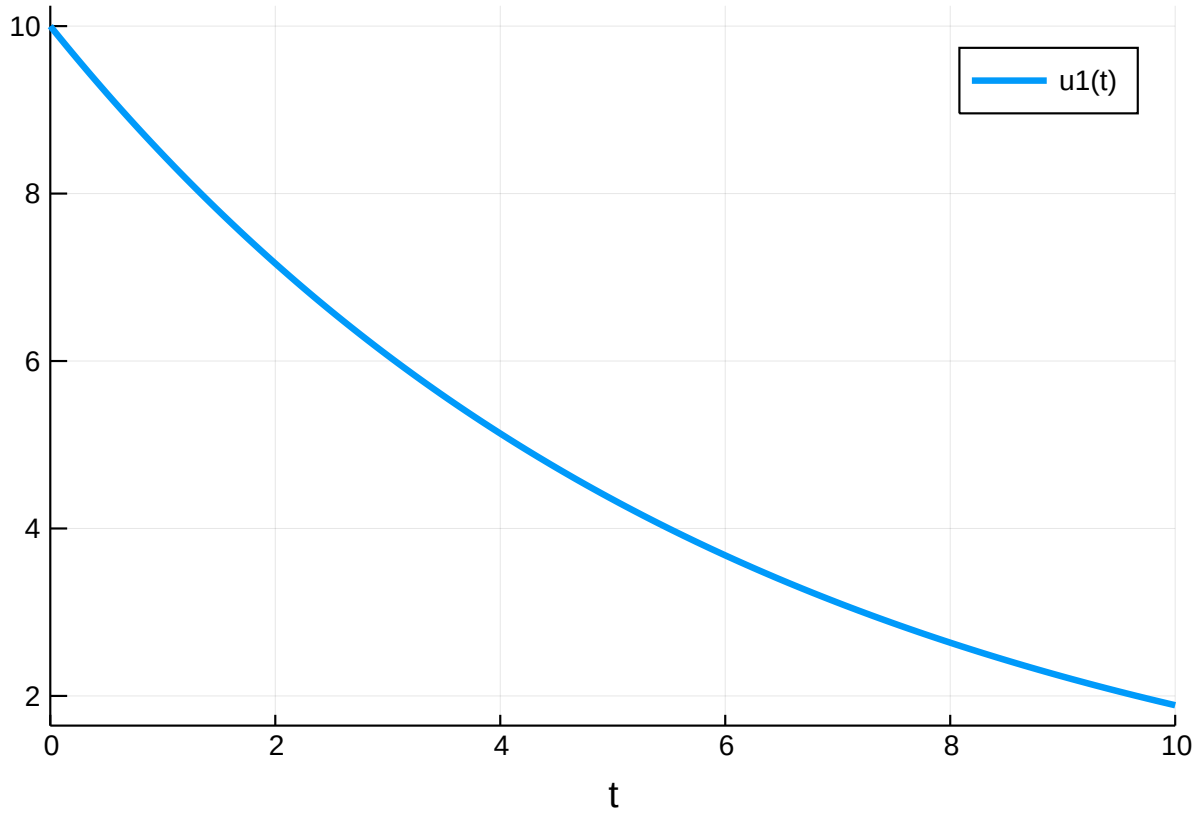
```
ODEProblem with uType Array{Float64,1} and tType Float64. In-place: true
timespan: (0.0, 10.0)
u0: [10.0]
```

```
sol = solve(prob,Tsit5())
using Plots; gr()
plot(sol)
```



Under the same criteria, with the same event, this patient will not receive a second dose:

```
sol = solve(prob,Tsit5(),tstops=[4.0],callback=cb)
using Plots; gr()
plot(sol)
```




```
using DiffEqTutorials
DiffEqTutorials.tutorial_footer(WEAVE_ARGS[:folder],WEAVE_ARGS[:file])
```

0.1 Appendix

These benchmarks are part of the DiffEqTutorials.jl repository, found at: https://github.com/JuliaDiffEq/

To locally run this tutorial, do the following commands:

```
using DiffEqTutorials
DiffEqTutorials.weave_file("models","conditional_dosing.jmd")
```

Computer Information:

```
Julia Version 1.1.0
Commit 80516ca202 (2019-01-21 21:24 UTC)
Platform Info:
  OS: Windows (x86_64-w64-mingw32)
  CPU: Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-6.0.1 (ORCJIT, skylake)
Environment:
  JULIA_EDITOR = "C:\Users\accou\AppData\Local\atom\app-1.34.0\atom.exe" -a
  JULIA_NUM_THREADS = 6
```

Package Information:

```
Status `C:\Users\accou\.julia\external\DiffEqTutorials.jl\src\Project.toml`
```