Conditional Dosing Pharmacometric Example

Chris Rackauckas

March 5, 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.

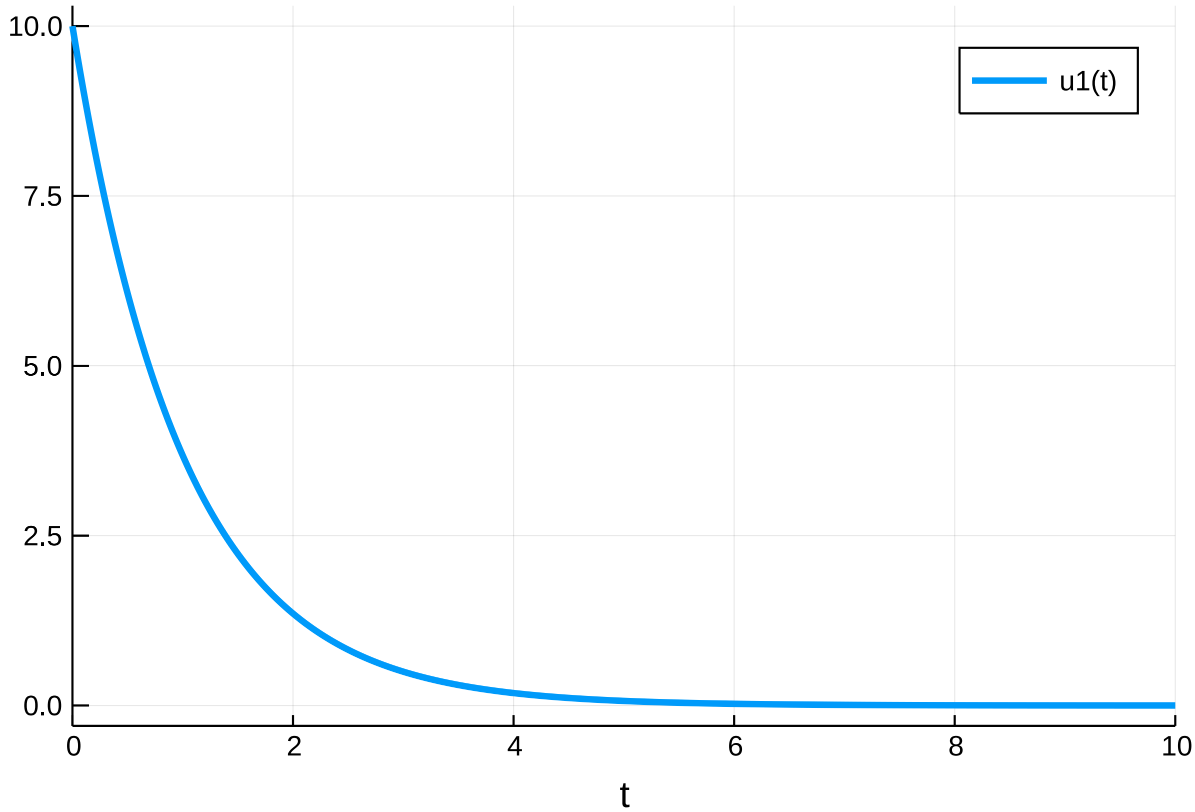
For our model, we will use the simple decay equation. We will write this in the in-place form to make it easy to extend to more complicated examples:

```julia
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.

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



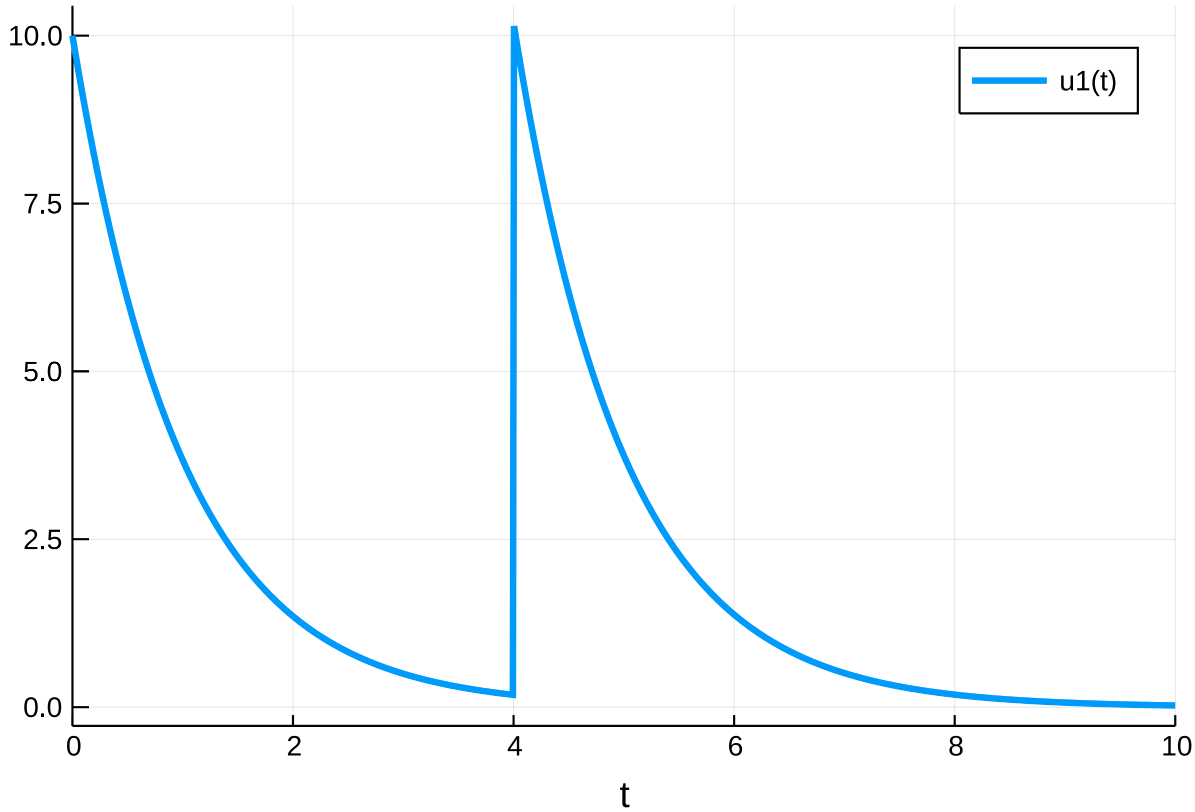
We see that at time `t=4`, the patient should receive a dose. Let's code up that event. We need to check at `t=4` if the concentration `u[1]/4` is `<4`, and if so, add `10` to `u[1]`. We do this with the following:

```
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))
```

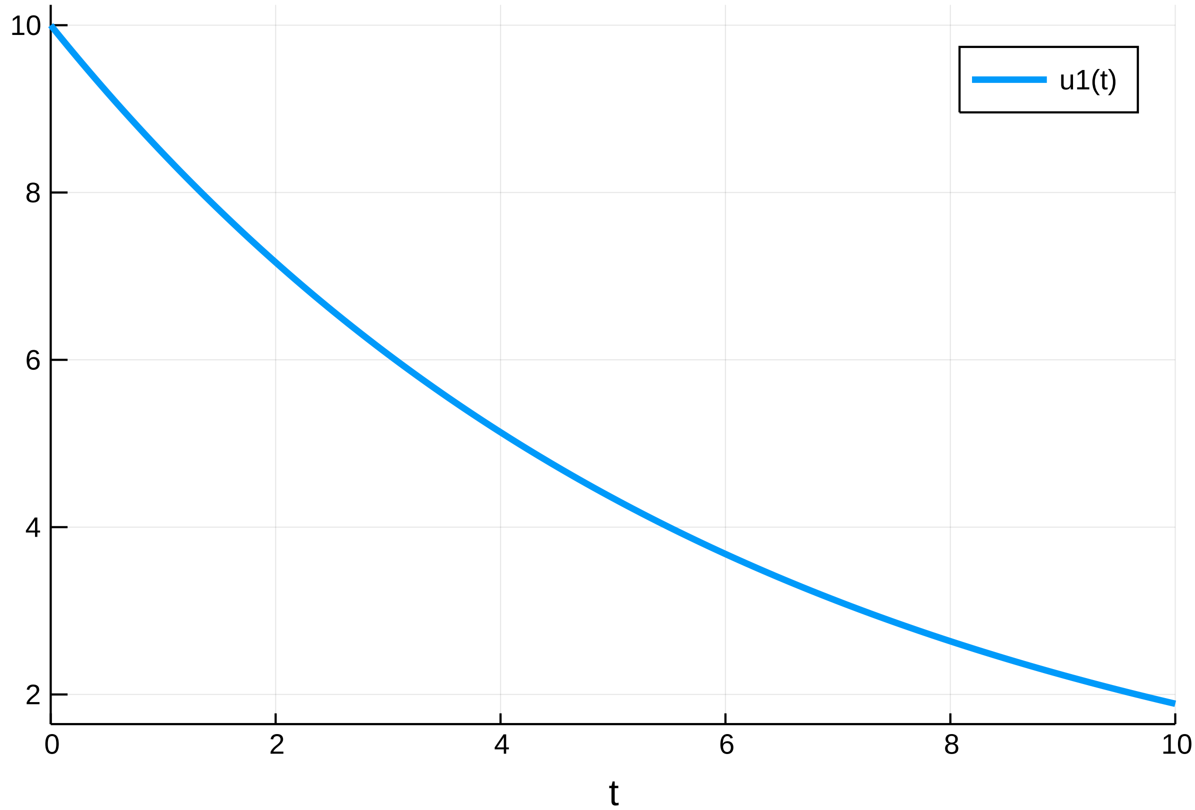
```
[10.1832]
```

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))
```

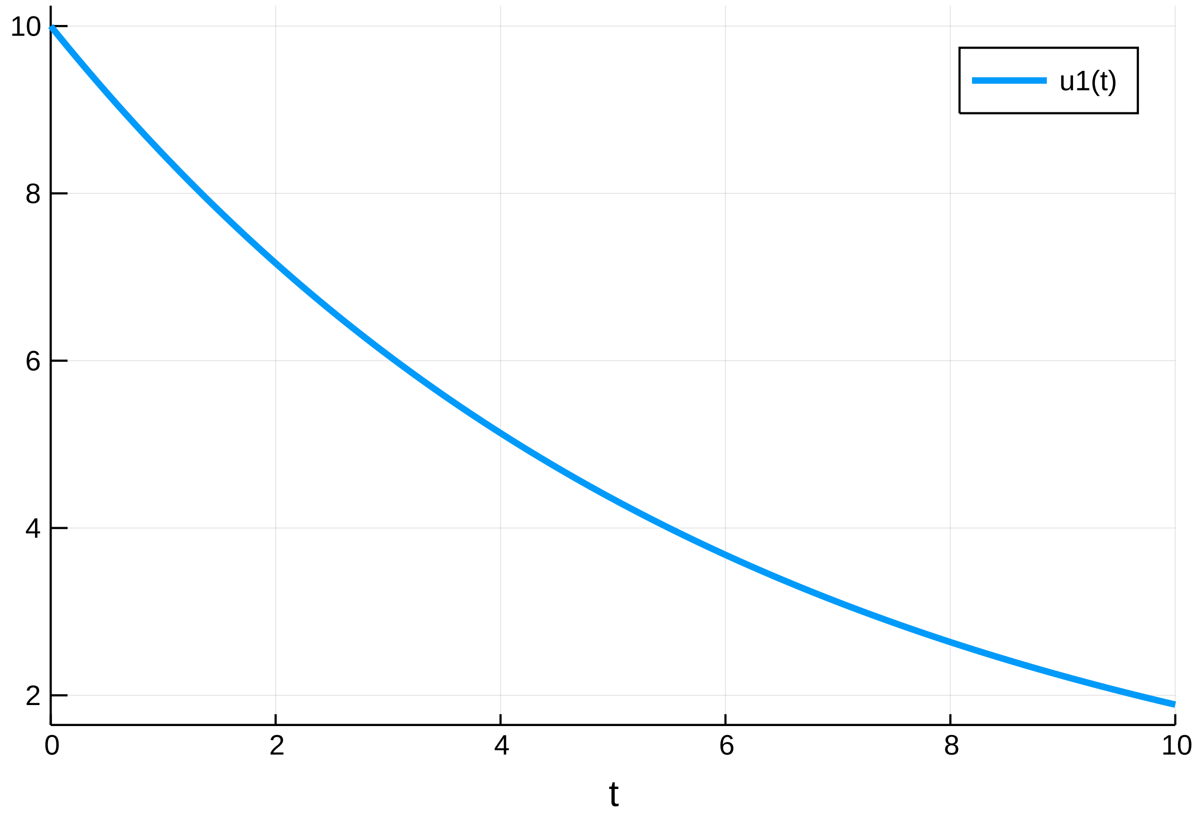
```
ODEProblem with uType Array{Float64,1} and tType Float64. In-place: true
timespan: (0.0, 10.0)
u0: [10.0]
```

```
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\Project.toml`
[7e558dbc-694d-5a72-987c-6f4ebed21442] ArbNumerics 0.3.6
[6e4b80f9-dd63-53aa-95a3-0cdb28fa8baf] BenchmarkTools 0.4.2
[be33ccc6-a3ff-5ff2-a52e-74243cff1e17] CUDAnative 1.0.1
[3a865a2d-5b23-5a0f-bc46-62713ec82fae] CuArrays 0.9.1
[55939f99-70c6-5e9b-8bb0-5071ed7d61fd] DecFP 0.4.8
[abce61dc-4473-55a0-ba07-351d65e31d42] Decimals 0.4.0
[459566f4-90b8-5000-8ac3-15dfb0a30def] DiffEqCallbacks 2.5.2
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.6.1
[1130ab10-4a5a-5621-a13d-e4788d82bd4c] DiffEqParamEstim 1.6.0
[055956cb-9e8b-5191-98cc-73ae4a59e68a] DiffEqPhysics 3.1.0
[0c46a032-eb83-5123-abaf-570d42b7fbaa] DifferentialEquations 6.3.0
[497a8b3b-efae-58df-a0af-a86822472b78] DoubleFloats 0.7.5
[f6369f11-7733-5829-9624-2563aa707210] ForwardDiff 0.10.3
[7073ff75-c697-5162-941a-fcdaad2a7d2a] IJulia 1.17.0
[eff96d63-e80a-5855-80a2-b1b0885c5ab7] Measurements 2.0.0
[429524aa-4258-5aef-a3af-852621145aeb] Optim 0.17.2
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.3.0
[65888b18-ceab-5e60-b2b9-181511a3b968] ParameterizedFunctions 4.1.1
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 0.23.1
[731186ca-8d62-57ce-b412-fbd966d074cd] RecursiveArrayTools 0.20.0
[90137ffa-7385-5640-81b9-e52037218182] StaticArrays 0.10.3
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 3.1.0
[1986cc42-f94f-5a68-af5c-568840ba703d] Unitful 0.14.0
[2a06ce6d-1589-592b-9c33-f37faeaed826] UnitfulPlots 0.0.0
[44d3d7a6-8a23-5bf8-98c5-b353f8df5ec9] Weave 0.7.2
[b77e0a4c-d291-57a0-90e8-8db25a27a240] InteractiveUtils
[37e2e46d-f89d-539d-b4ee-838fcccc9c8e] LinearAlgebra
[44cfe95a-1eb2-52ea-b672-e2afdf69b78f] Pkg
```