Unit Checked Arithmetic via Unitful.jl

Chris Rackauckas

November 7, 2020

Units and dimensional analysis are standard tools across the sciences for checking the correctness of your equation. However, most ODE solvers only allow for the equation to be in dimensionless form, leaving it up to the user to both convert the equation to a dimensionless form, punch in the equations, and hopefully not make an error along the way.

DifferentialEquations.jl allows for one to use Unitful.jl to have unit-checked arithmetic natively in the solvers. Given the dispatch implementation of the Unitful, this has little overhead.

0.1 Using Unitful

To use Unitful, you need to have the package installed. Then you can add units to your variables. For example:

```
using Unitful
t = 1.0u"s"
```

```
1.0 s
```

Notice that `t` is a variable with units in seconds. If we make another value with seconds, they can add

```
t2 = 1.02u"s"
t+t2
```

```
2.02 s
```

and they can multiply:

```
t*t2
```

```
1.02 s^2
```

You can even do rational roots:

```
sqrt(t)
```

```
1.0 s^1/2
```

Many operations work. These operations will check to make sure units are correct, and will throw an error for incorrect operations:

```
t + sqrt(t)
```

```
Error: DimensionError: 1.0 s and 1.0 s^1/2 are not dimensionally compatible.
```

0.2 Using Unitful with DifferentialEquations.jl

Just like with other number systems, you can choose the units for your numbers by simply specifying the units of the initial condition and the timestep. For example, to solve the linear ODE where the variable has units of Newton's and `t` is in Seconds, we would use:

```
using DifferentialEquations
f = (y,p,t) -> 0.5*y
u0 = 1.5u"N"
prob = ODEProblem(f,u0,(0.0u"s",1.0u"s"))
sol = solve(prob,Tsit5())
```

```
Error: DimensionError: N s^-1 and 0.75 N are not dimensionally compatible.
```

Notice that we recieved a unit mismatch error. This is correctly so! Remember that for an ODE:

$$\frac{dy}{dt} = f(t,y)$$

we must have that `f` is a rate, i.e. `f` is a change in `y` per unit time. So we need to fix the units of `f` in our example to be `N/s`. Notice that we then do not receive an error if we do the following:

```
f = (y,p,t) -> 0.5*y/3.0u"s"
prob = ODEProblem(f,u0,(0.0u"s",1.0u"s"))
sol = solve(prob,Tsit5())
```

```
retcode: Success
Interpolation: specialized 4th order "free" interpolation
t: 3-element Array{Unitful.Quantity{Float64,T@*(,Unitful.FreeUnits(*@{(s,),
T@*(,nothing(*@}},1}:
                 0.0 s
 0.14311598261241779 s
                 1.0 s
u: 3-element Array{Unitful.Quantity{Float64,L@*( (*@M@*(
(*@T@*((*@^-2,Unitful.FreeUnits{(N,),L@*( (*@M@*( (*@T@*((*@^-2,nothing}},1}:
                1.5 N
 1.5362091208988309 N
 1.7720406194871123 N
```

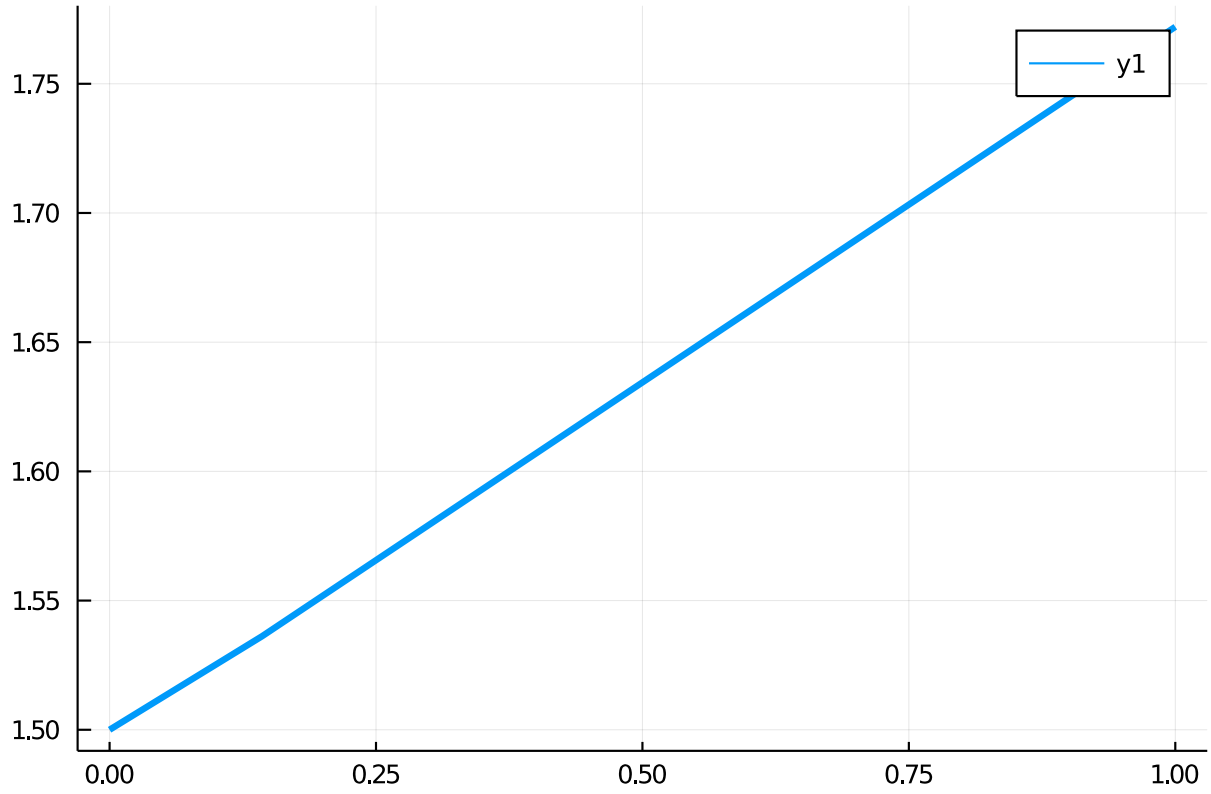
This gives a a normal solution object. Notice that the values are all with the correct units:

```
print(sol[:])
```

```
Unitful.Quantity{Float64,L@*( (*@M@*( (*@T@*((*@^-2,Unitful.FreeUnits{(N,),L@*(
(*@M@*( (*@T@*((*@^-2,nothing}}[1.5 N, 1.5362091208988309 N, 1.7720406194871123 N]
```

We can plot the solution by removing the units:

```
using Plots
gr()
plot(ustrip(sol.t),ustrip(sol[:]),lw=3)
```



0.3 Appendix

This tutorial is part of the SciMLTutorials.jl repository, found at: https://github.com/SciML/SciMLTutor
For more information on doing scientific machine learning (SciML) with open source software,
check out https://sciml.ai/.

To locally run this tutorial, do the following commands:

```
using SciMLTutorials
SciMLTutorials.weave_file("type_handling","03-unitful.jmd")
```

Computer Information:

```
Julia Version 1.4.2
Commit 44fa15b150* (2020-05-23 18:35 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: Intel(R) Core(TM) i7-9700K CPU @ 3.60GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
  LLVM: libLLVM-8.0.1 (ORCJIT, skylake)
Environment:
  JULIA_LOAD_PATH = /builds/JuliaGPU/DiffEqTutorials.jl:
  JULIA_DEPOT_PATH = /builds/JuliaGPU/DiffEqTutorials.jl/.julia
  JULIA_CUDA_MEMORY_LIMIT = 2147483648
  JULIA_NUM_THREADS = 8
```

Package Information:

```
Status `/builds/JuliaGPU/DiffEqTutorials.jl/tutorials/type_handling/Project.toml`
[7e558dbc-694d-5a72-987c-6f4ebed21442] ArbNumerics 1.2.1
[55939f99-70c6-5e9b-8bb0-5071ed7d61fd] DecFP 1.0.0
[abce61dc-4473-55a0-ba07-351d65e31d42] Decimals 0.4.1
[0c46a032-eb83-5123-abaf-570d42b7fbaa] DifferentialEquations 6.15.0
[497a8b3b-efae-58df-a0af-a86822472b78] DoubleFloats 1.1.13
[eff96d63-e80a-5855-80a2-b1b0885c5ab7] Measurements 2.3.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.44.0
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.6.12
[1986cc42-f94f-5a68-af5c-568840ba703d] Unitful 1.5.0
```