Unit Checked Arithmetic via Unitful.jl

Chris Rackauckas

July 1, 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
```

```
Error: ArgumentError: Package Unitful not found in current path:
- Run `import Pkg; Pkg.add("Unitful")` to install the Unitful package.
```

```
t = 1.0u"s"
```

```
Error: LoadError: UndefVarError: @u_str not defined
in expression starting at none:1
```

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

```
t2 = 1.02u"s"
```

```
Error: LoadError: UndefVarError: @u_str not defined
in expression starting at none:1
```

```
t+t2
```

```
Error: UndefVarError: t not defined
```

and they can multiply:

```
t*t2
```

```
Error: UndefVarError: t not defined
```

You can even do rational roots:

```
sqrt(t)
```

```
Error: UndefVarError: t not defined
```

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: UndefVarError: t not defined
```

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"
```

```
Error: LoadError: UndefVarError: @u_str not defined
in expression starting at none:1
```

```
prob = ODEProblem(f,u0,(0.0u"s",1.0u"s"))
```

```
Error: LoadError: UndefVarError: @u_str not defined
in expression starting at none:1
```

```
sol = solve(prob,Tsit5())
```

```
Error: UndefVarError: prob not defined
```

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"
```

```
Error: LoadError: UndefVarError: @u_str not defined
in expression starting at none:1
```

```
prob = ODEProblem(f,u0,(0.0u"s",1.0u"s"))
```

```
Error: LoadError: UndefVarError: @u_str not defined
in expression starting at none:1
```

```
sol = solve(prob,Tsit5())
```

```
Error: UndefVarError: prob not defined
```

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

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

```
Error: UndefVarError: sol not defined
```

We can plot the solution by removing the units:

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

```
Error: UndefVarError: sol not defined
```

0.3 Appendix

This tutorial is part of the DiffEqTutorials.jl repository, found at: https://github.com/JuliaDiffEq/DiffEq

To locally run this tutorial, do the following commands:

```
using DiffEqTutorials
DiffEqTutorials.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_DEPOT_PATH = /builds/JuliaGPU/DiffEqTutorials.jl/.julia
  JULIA_CUDA_MEMORY_LIMIT = 536870912
  JULIA_PROJECT = @.
  JULIA_NUM_THREADS = 4
```

Package Information:

```
Status `/builds/JuliaGPU/DiffEqTutorials.jl/tutorials/type_handling/Project.toml`
[7e558dbc-694d-5a72-987c-6f4ebed21442] ArbNumerics 1.0.5
[55939f99-70c6-5e9b-8bb0-5071ed7d61fd] DecFP 1.0.0
[abce61dc-4473-55a0-ba07-351d65e31d42] Decimals 0.4.1
[0c46a032-eb83-5123-abaf-570d42b7fbaa] DifferentialEquations 6.14.0
[497a8b3b-efae-58df-a0af-a86822472b78] DoubleFloats 1.1.12
[eff96d63-e80a-5855-80a2-b1b0885c5ab7] Measurements 2.2.1
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.41.0
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.5.0
```