Feagin's Order 10, 12, and 14 Methods

Chris Rackauckas

July 1, 2020

DifferentialEquations.jl includes Feagin's explicit Runge-Kutta methods of orders 10/8, 12/10, and 14/12. These methods have such high order that it's pretty much required that one uses numbers with more precision than Float64. As a prerequisite reference on how to use arbitrary number systems (including higher precision) in the numerical solvers, please see the Solving Equations in With Chosen Number Types notebook.

0.1 Investigation of the Method's Error

We can use Feagin's order 16 method as follows. Let's use a two-dimensional linear ODE. Like in the Solving Equations in With Chosen Number Types notebook, we change the initial condition to BigFloats to tell the solver to use BigFloat types.

```
using DifferentialEquations
```

```
Error: ArgumentError: Package DifferentialEquations not found in current pa
th:
- Run `import Pkg; Pkg.add("DifferentialEquations")` to install the Differe
ntialEquations package.
```

```
const linear_bigα = big(1.01)
f(u,p,t) = (linear_bigα*u)

# Add analytical solution so that errors are checked
f_analytic(u0,p,t) = u0*exp(linear_bigα*t)
ff = ODEFunction(f,analytic=f_analytic)
```

```
Error: UndefVarError: ODEFunction not defined
```

```
prob = ODEProblem(ff,big(0.5),(0.0,1.0))
```

```
Error: UndefVarError: ODEProblem not defined
```

```
sol = solve(prob,Feagin14(),dt=1//16,adaptive=false);
```

```
Error: UndefVarError: Feagin14 not defined
```

```
println(sol.errors)
```

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

Compare that to machine ϵ for Float64:

```
eps(Float64)
```

```
2.220446049250313e-16
```

The error for Feagin's method when the stepsize is 1/16 is 8 orders of magnitude below machine ϵ! However, that is dependent on the stepsize. If we instead use adaptive timestepping with the default tolerances, we get

```
sol =solve(prob,Feagin14());
```

```
Error: UndefVarError: Feagin14 not defined
```

```
println(sol.errors); print("The length was $(length(sol))")
```

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

Notice that when the stepsize is much higher, the error goes up quickly as well. These super high order methods are best when used to gain really accurate approximations (using still modest timesteps). Some examples of where such precision is necessary is astrodynamics where the many-body problem is highly chaotic and thus sensitive to small errors.

0.2 Convergence Test

The Order 14 method is awesome, but we need to make sure it's really that awesome. The following convergence test is used in the package tests in order to make sure the implementation is correct. Note that all methods have such tests in place.

```
using DiffEqDevTools
dts = 1.0 ./ 2.0 .^(10:-1:4)
sim = test_convergence(dts,prob,Feagin14())
```

```
Error: UndefVarError: Feagin14 not defined
```

For a view of what's going on, let's plot the simulation results.

```
using Plots
gr()
plot(sim)
```

```
Error: UndefVarError: sim not defined
```

This is a clear trend indicating that the convergence is truly Order 14, which is the estimated slope.

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("ode_extras","02-feagin.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/ode_extras/Project.toml`
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.22.0
[961ee093-0014-501f-94e3-6117800e7a78] ModelingToolkit 3.11.0
[76087f3c-5699-56af-9a33-bf431cd00edd] NLopt 0.6.0
[2774e3e8-f4cf-5e23-947b-6d7e65073b56] NLsolve 4.4.0
[429524aa-4258-5aef-a3af-852621145aeb] Optim 0.22.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.41.0
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.5.1
[37e2e46d-f89d-539d-b4ee-838fcccc9c8e] LinearAlgebra
[2f01184e-e22b-5df5-ae63-d93ebab69eaf] SparseArrays
```