Fitzhugh-Nagumo Work-Precision Diagrams

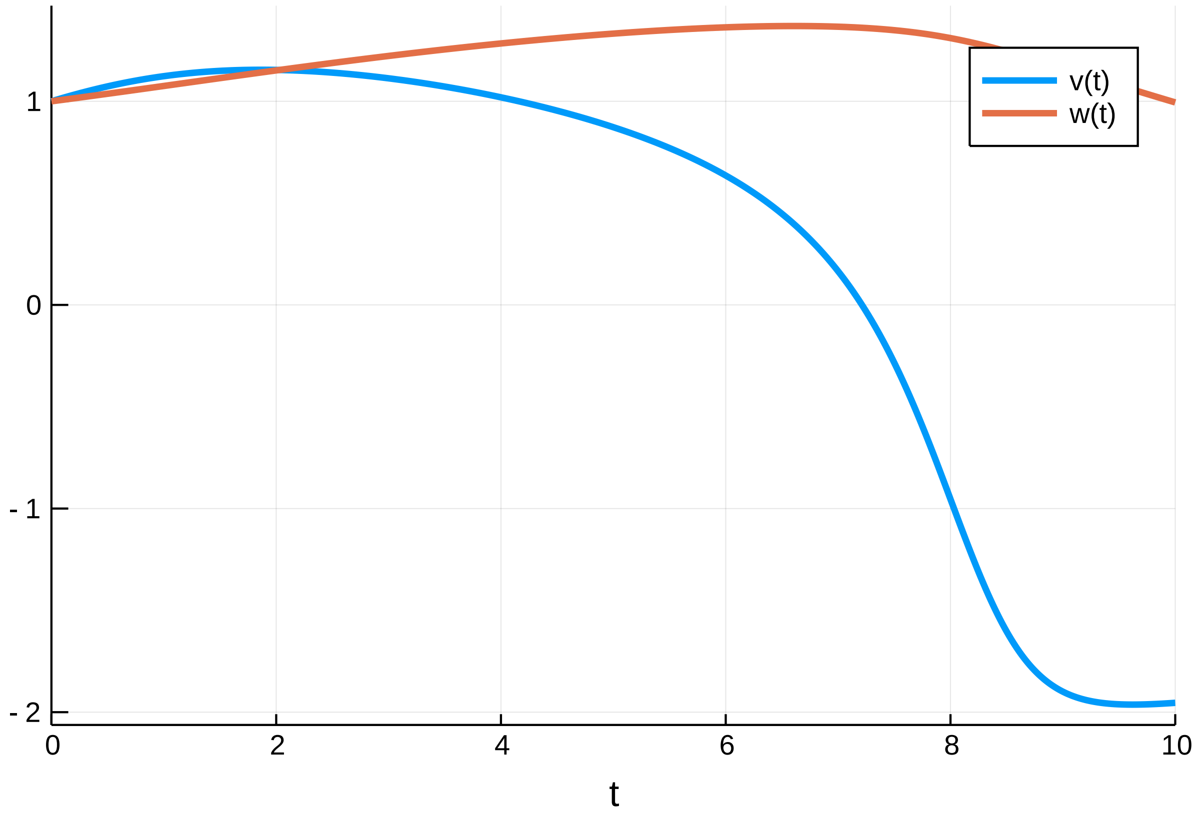
Chris Rackauckas

March 2, 2019

1 Fitzhugh-Nagumo

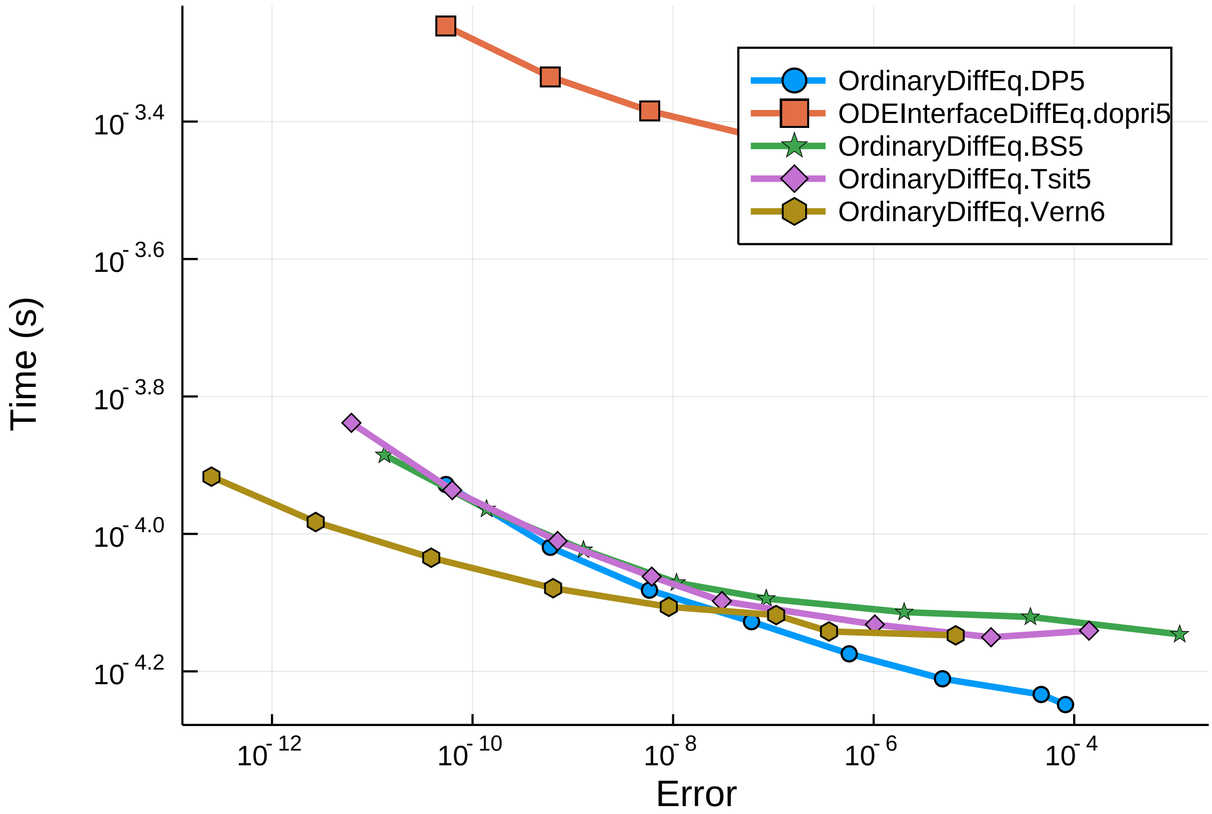
The purpose of this is to see how the errors scale on a standard nonlinear problem.

```
using OrdinaryDiffEq, ParameterizedFunctions, ODE, ODEInterfaceDiffEq,
      LSODA, Sundials, DiffEqDevTools

f = @ode_def FitzhughNagumo begin
  dv = v - v^3/3 -w + l
  dw = τinv*(v +  a - b*w)
end a b τinv l

p = [0.7,0.8,1/12.5,0.5]
prob = ODEProblem(f,[1.0;1.0],(0.0,10.0),p)

abstols = 1.0 ./ 10.0 .^ (6:13)
reltols = 1.0 ./ 10.0 .^ (3:10);

sol = solve(prob,Vern7(),abstol=1/10^14,reltol=1/10^14)
test_sol = TestSolution(sol)
using Plots; gr()

plot(sol)
```



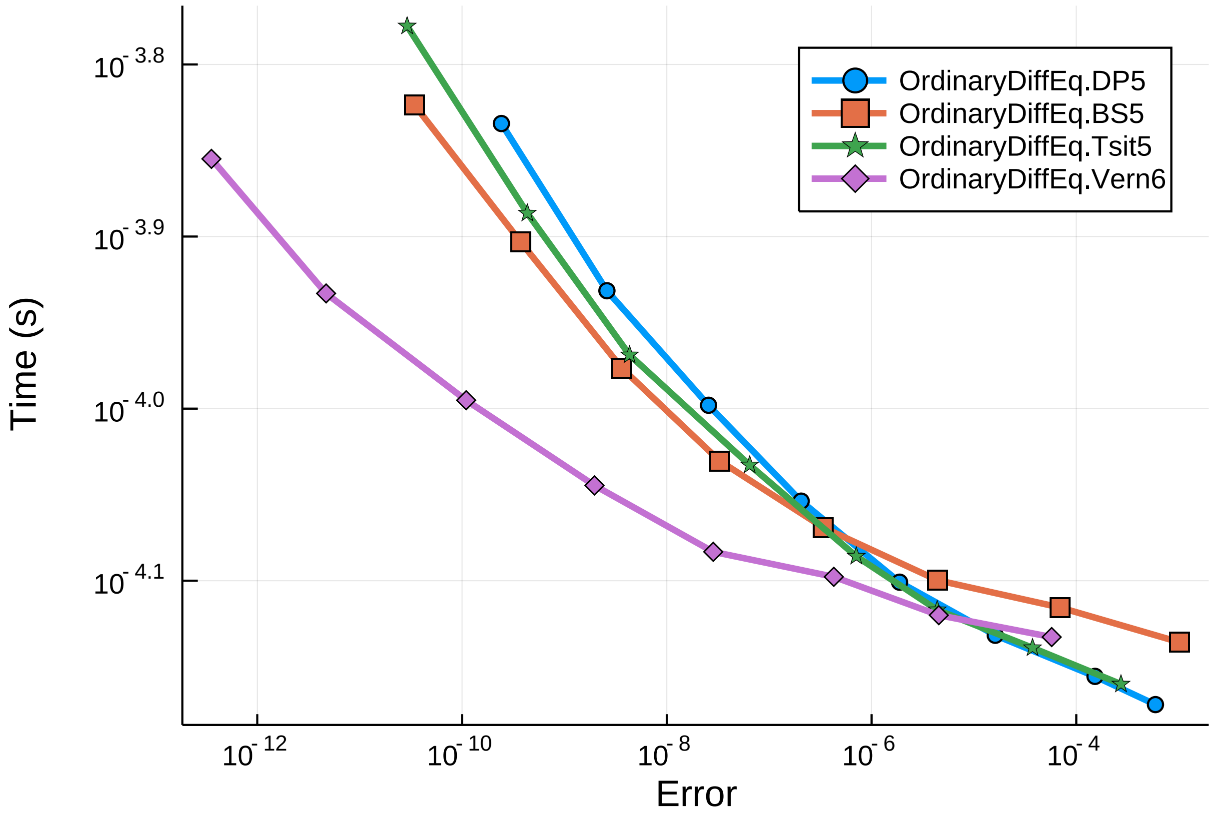
1.1 Low Order

```
setups = [Dict(:alg=>DP5())
          #Dict(:alg=>ode45()) #fails
          Dict(:alg=>dopri5())
          Dict(:alg=>BS5())
          Dict(:alg=>Tsit5())
          Dict(:alg=>Vern6())
]
wp =
    WorkPrecisionSet(prob,abstols,reltols,setups;appxsol=test_sol,save_everystep=false,numruns=100,max
plot(wp)
```



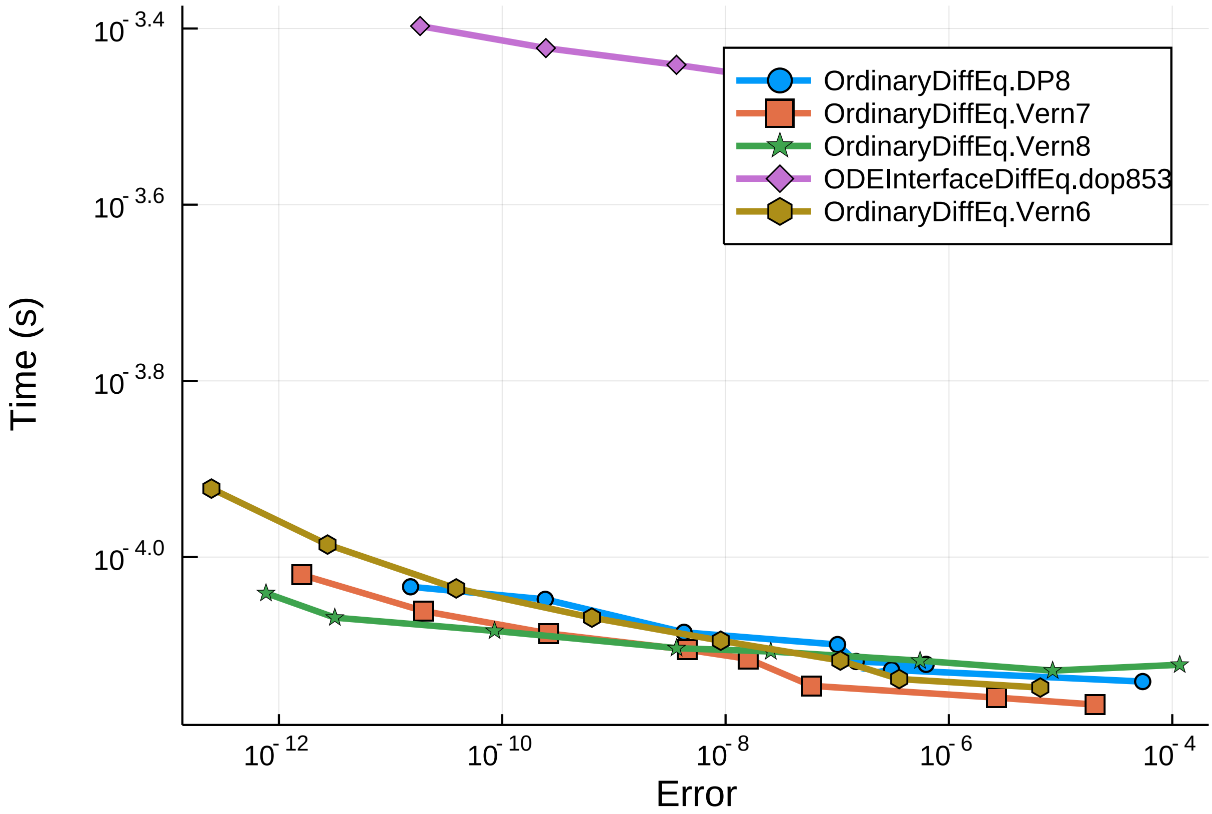
1.1.1 Interpolation

```
setups = [Dict(:alg=>DP5())
          #Dict(:alg=>ode45()) # fails
          Dict(:alg=>BS5())
          Dict(:alg=>Tsit5())
          Dict(:alg=>Vern6())
]
wp =
    WorkPrecisionSet(prob,abstols,reltols,setups;appxsol=test_sol,numruns=100,maxiters=10000,error_est
plot(wp)
```

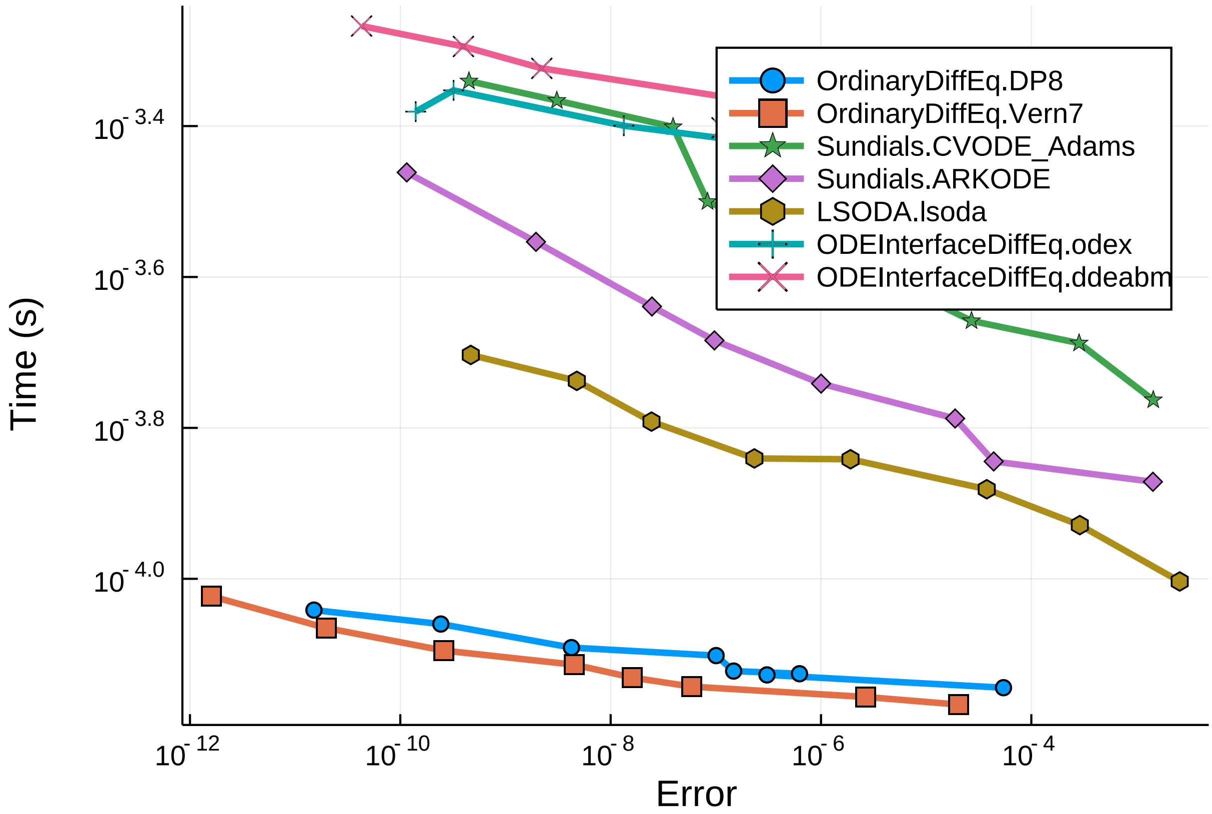


1.2 Higher Order

```
setups = [Dict(:alg=>DP8())
          #Dict(:alg=>ode78()) # fails
          Dict(:alg=>Vern7())
          Dict(:alg=>Vern8())
          Dict(:alg=>dop853())
          Dict(:alg=>Vern6())
]
wp =
    WorkPrecisionSet(prob,abstols,reltols,setups;appxsol=test_sol,save_everystep=false,numruns=100,max
plot(wp)
```

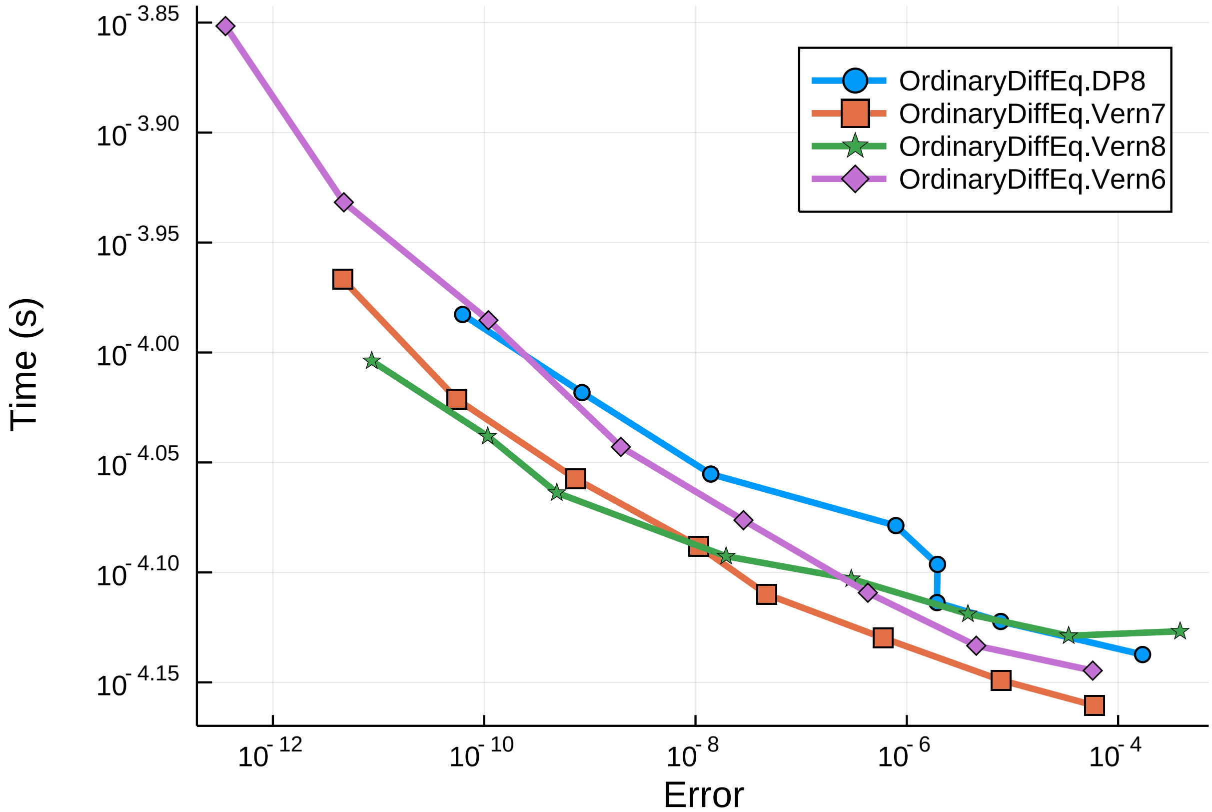



```
setups = [Dict(:alg=>DP8())
          Dict(:alg=>Vern7())
          Dict(:alg=>CVODE_Adams())
          Dict(:alg=>ARKODE(Sundials.Explicit(),order=6))
          Dict(:alg=>lsoda())
          Dict(:alg=>odex())
          Dict(:alg=>ddeabm())
]
wp =
    WorkPrecisionSet(prob,abstols,reltols,setups;appxsol=test_sol,save_everystep=false,numruns=100,max
plot(wp)
```



1.2.1 Interpolation

```
setups = [Dict(:alg=>DP8())
          #Dict(:alg=>ode78()) # fails
          Dict(:alg=>Vern7())
          Dict(:alg=>Vern8())
          Dict(:alg=>Vern6())
]
wp =
    WorkPrecisionSet(prob,abstols,reltols,setups;appxsol=test_sol,numruns=100,maxiters=1000,error_esti
plot(wp)
```



1.3 Conclusion

As expected, the algorithms are all pretty matched on time for this problem. However, you can clearly see the OrdinaryDiffEq.jl algorithms solving to a much higher accuracy and still faster, especially when the interpolations are involved.

```
using DiffEqBenchmarks
DiffEqBenchmarks.bench_footer(WEAVE_ARGS[:folder],WEAVE_ARGS[:file])
```

1.4 Appendix

These benchmarks are a part of the DiffEqBenchmarks.jl repository, found at: https://github.com/JuliaDi

To locally run this tutorial, do the following commands:

```
using DiffEqBenchmarks
DiffEqBenchmarks.weave_file("NonStiffODE","FitzhughNagumo_wpd.jmd")
```

Computer Information:

```
Julia Version 1.1.0
Commit 80516ca202 (2019-01-21 21:24 UTC)
Platform Info:
  OS: Linux (x86_64-pc-linux-gnu)
  CPU: Intel(R) Xeon(R) CPU E5-2680 v4 @ 2.40GHz
  WORD_SIZE: 64
  LIBM: libopenlibm
```

```
  LLVM: libLLVM-6.0.1 (ORCJIT, haswell)
```

Package Information:

```
Status: `/home/yingboma/.julia/dev/DiffEqBenchmarks/Project.toml`
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.7.0
[7073ff75-c697-5162-941a-fcdaad2a7d2a] IJulia 1.17.0
[7f56f5a3-f504-529b-bc02-0b1fe5e64312] LSODA 0.4.0
[c030b06c-0b6d-57c2-b091-7029874bd033] ODE 2.4.0
[09606e27-ecf5-54fc-bb29-004bd9f985bf] ODEInterfaceDiffEq 3.0.0
[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
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 3.1.0
[44d3d7a6-8a23-5bf8-98c5-b353f8df5ec9] Weave 0.7.2
[b77e0a4c-d291-57a0-90e8-8db25a27a240] InteractiveUtils
[d6f4376e-aef5-505a-96c1-9c027394607a] Markdown
[44cfe95a-1eb2-52ea-b672-e2afdf69b78f] Pkg
[9a3f8284-a2c9-5f02-9a11-845980a1fd5c] Random
```