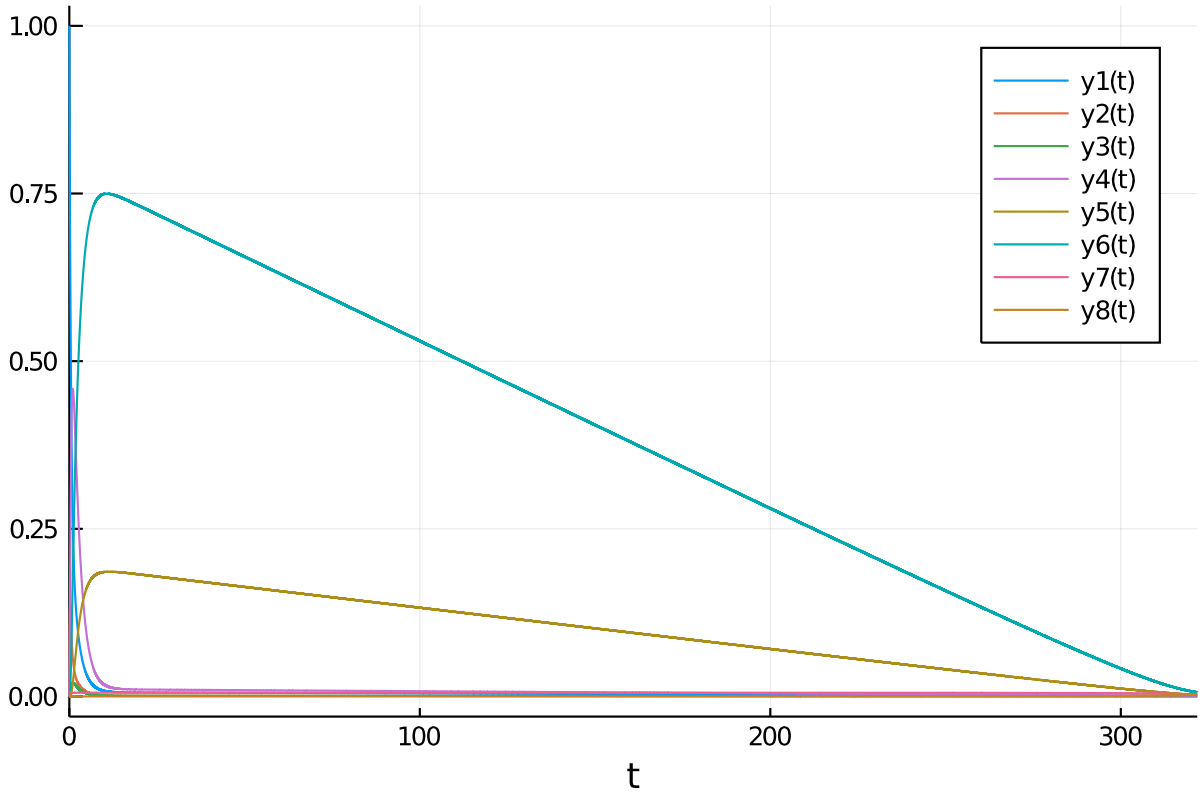
HIRES Work-Precision Diagrams

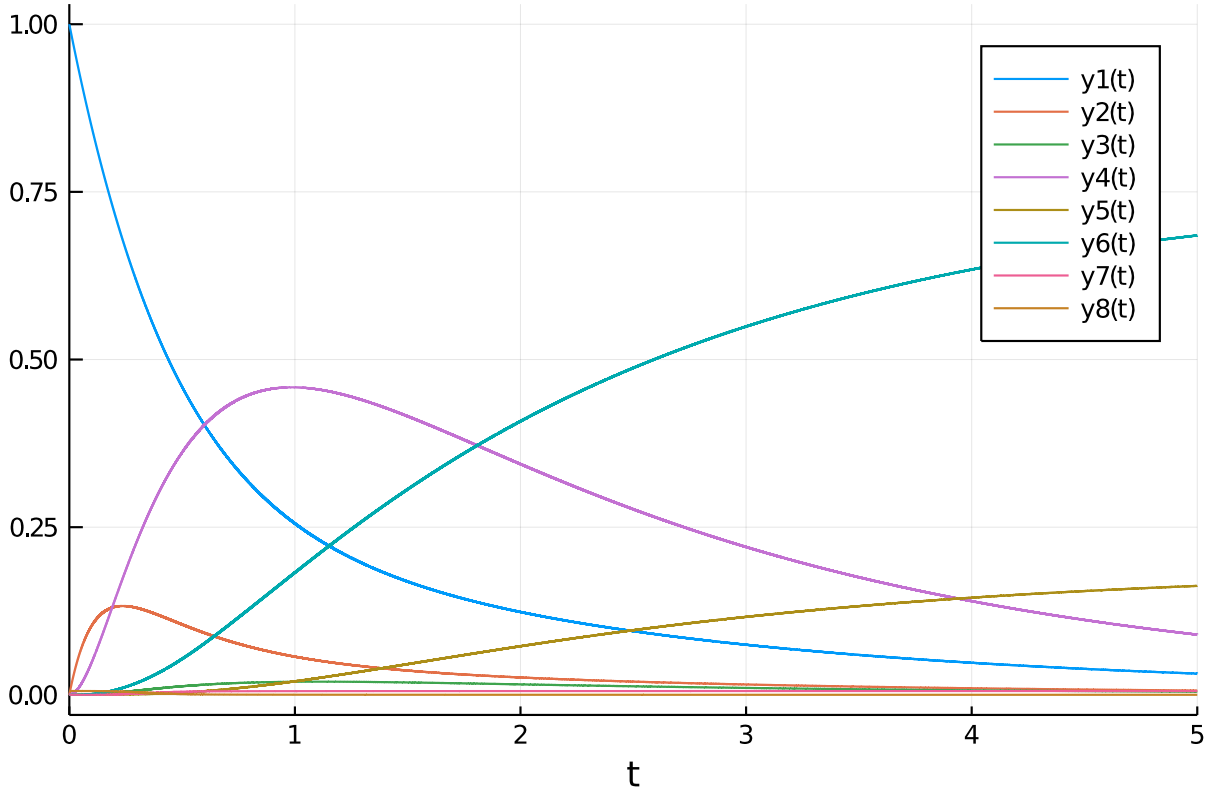
Chris Rackauckas

July 4, 2020

```
using OrdinaryDiffEq, ParameterizedFunctions, Plots, ODE, ODEInterfaceDiffEq, LSODA,
DiffEqDevTools, Sundials
using LinearAlgebra
LinearAlgebra.BLAS.set_num_threads(1)

gr() #gr(fmt=:png)

f = @ode_def Hires begin
  dy1 = -1.71*y1 + 0.43*y2 + 8.32*y3 + 0.0007
  dy2 = 1.71*y1 - 8.75*y2
  dy3 = -10.03*y3 + 0.43*y4 + 0.035*y5
  dy4 = 8.32*y2 + 1.71*y3 - 1.12*y4
  dy5 = -1.745*y5 + 0.43*y6 + 0.43*y7
  dy6 = -280.0*y6*y8 + 0.69*y4 + 1.71*y5 -
           0.43*y6 + 0.69*y7
  dy7 = 280.0*y6*y8 - 1.81*y7
  dy8 = -280.0*y6*y8 + 1.81*y7
end

u0 = zeros(8)
u0[1] = 1
u0[8] = 0.0057
prob = ODEProblem(f,u0,(0.0,321.8122))

sol = solve(prob,Rodas5(),abstol=1/10^14,reltol=1/10^14)
test_sol = TestSolution(sol)
abstols = 1.0 ./ 10.0 .^ (4:11)
reltols = 1.0 ./ 10.0 .^ (1:8);

8-element Array{Float64,1}:
 0.1
 0.01
 0.001
 0.0001
 1.0e-5
 1.0e-6
 1.0e-7
 1.0e-8

plot(sol)
```




```
plot(sol,tspan=(0.0,5.0))
```



0.1 Omissions

The following were omitted from the tests due to convergence failures. ODE.jl's adaptivity is not able to stabilize its algorithms, while GeometricIntegratorsDiffEq has not upgraded to Julia 1.0. GeometricIntegrators.jl's methods used to be either fail to converge at comparable dts (or on some computers errors due to type conversions).

```
#sol = solve(prob,ode23s()); println("Total ODE.jl steps: $(length(sol))")
#using GeometricIntegratorsDiffEq
#try
#    sol = solve(prob,GIRadIIA3(),dt=1/10)
#catch e
#    println(e)
#end
```

The stabilized explicit methods are not stable enough to handle this problem well. While they don't diverge, they are really slow.

```
setups = [
#Dict(:alg=>ROCK2()),
#Dict(:alg=>ROCK4())
#Dict(:alg=>ESERK5())
]
```

```
0-element Array{Any,1}
```

0.2 High Tolerances

This is the speed when you just want the answer.

```
abstols = 1.0 ./ 10.0 .^ (5:8)
reltols = 1.0 ./ 10.0 .^ (1:4);
setups = [Dict(:alg=>Rosenbrock23()),
          Dict(:alg=>Rodas3()),
          Dict(:alg=>TRBDF2()),
          Dict(:alg=>CVODE_BDF()),
          Dict(:alg=>rodas()),
          Dict(:alg=>radau()),
          Dict(:alg=>RadauIIA5()),
          Dict(:alg=>ROS34PW1a()),
          Dict(:alg=>lsoda()),
          ]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      save_everystep=false,appxsol=test_sol,maxiters=Int(1e5),numruns=10)
```

```
Error: Cannot find method(s) for rodas! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

```
plot(wp)
```

```
Error: UndefVarError: wp not defined
```

```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;dense = false,verbose=false,
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:l2)
```

```
Error: Cannot find method(s) for rodas! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

```
plot(wp)
```

```
Error: UndefVarError: wp not defined
```

```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:L2)
```

```
Error: Cannot find method(s) for rodas! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

```
plot(wp)
```

```
Error: UndefVarError: wp not defined
```

```
setups = [Dict(:alg=>Rosenbrock23()),
          Dict(:alg=>Kvaerno3()),
          Dict(:alg=>CVODE_BDF()),
          Dict(:alg=>KenCarp4()),
          Dict(:alg=>TRBDF2()),
          Dict(:alg=>KenCarp3()),
    # Dict(:alg=>SDIRK2()), # Removed because it's bad
          Dict(:alg=>radau())]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
```

```
Error: Cannot find method(s) for radau! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

```
plot(wp)
```

```
Error: UndefVarError: wp not defined
```

```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;dense = false,verbose=false,
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:l2)
```

```
Error: Cannot find method(s) for radau! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

```
plot(wp)
```

```
Error: UndefVarError: wp not defined
```

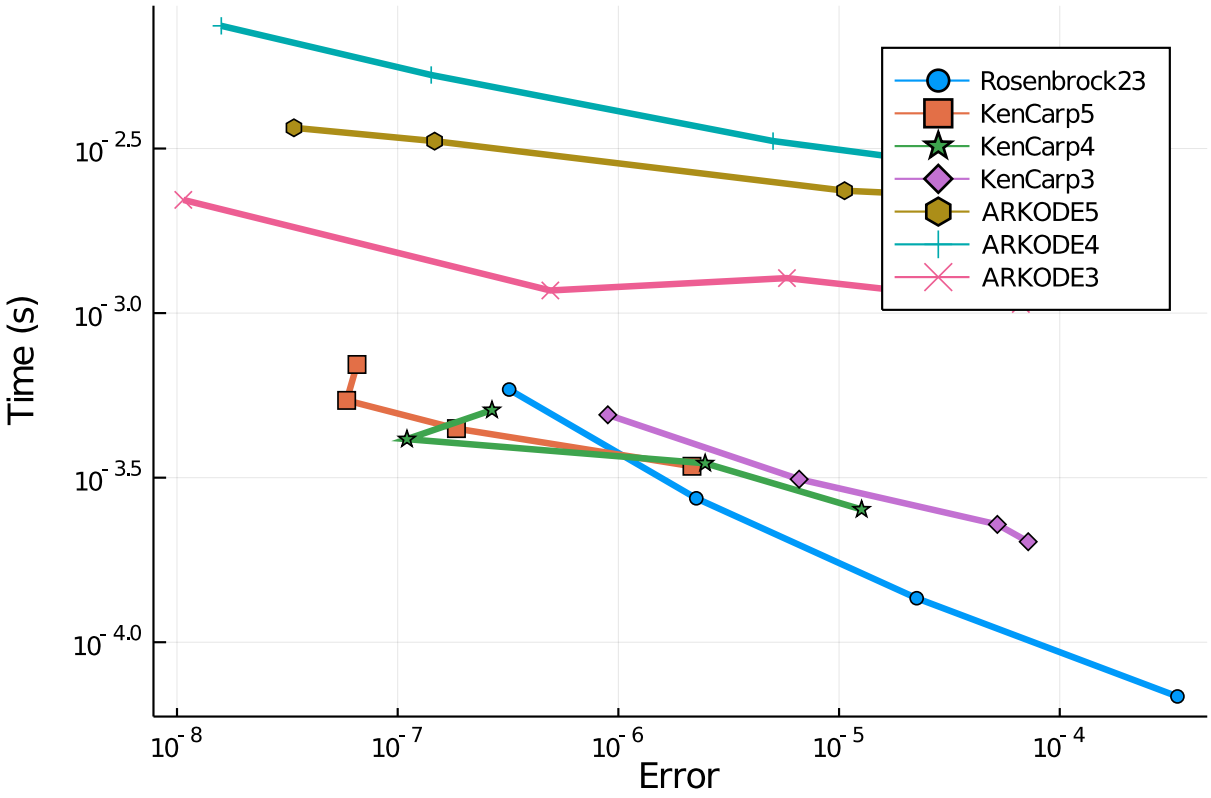
```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:L2)
```

```
Error: Cannot find method(s) for radau! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

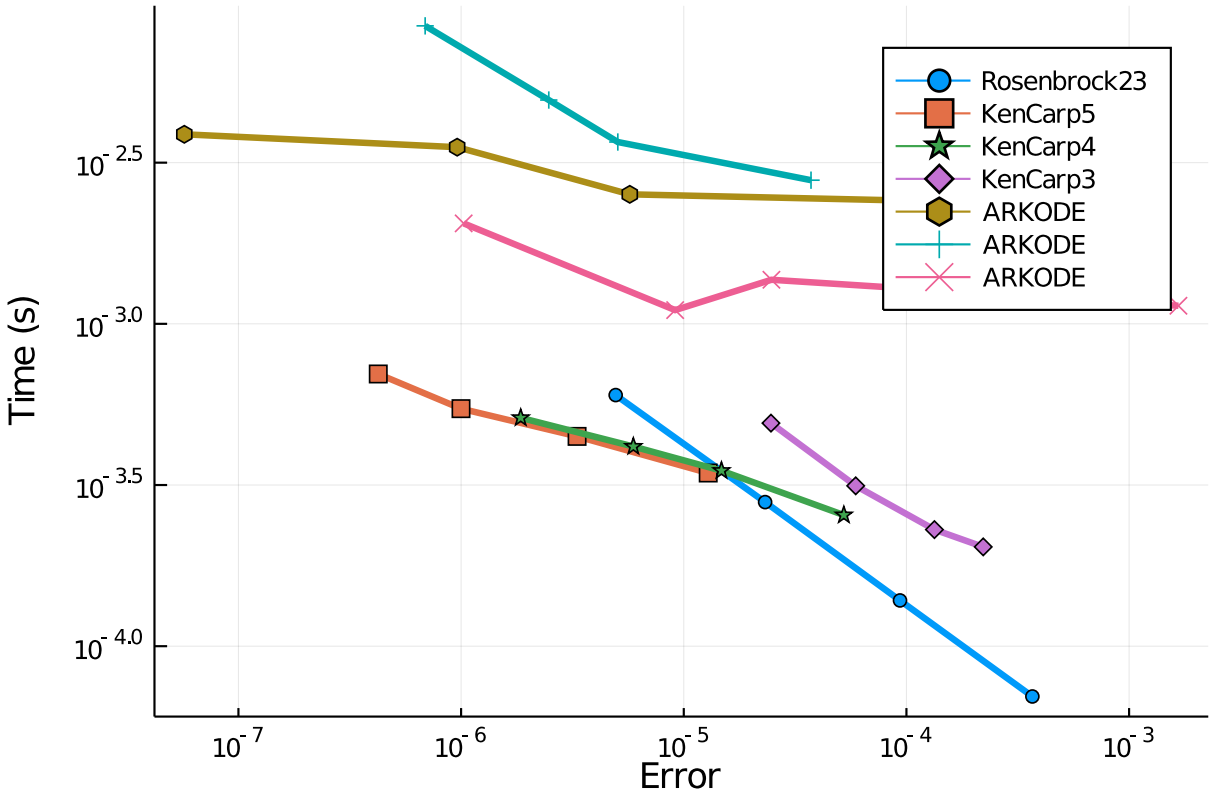
```
plot(wp)
```

```
Error: UndefVarError: wp not defined
```

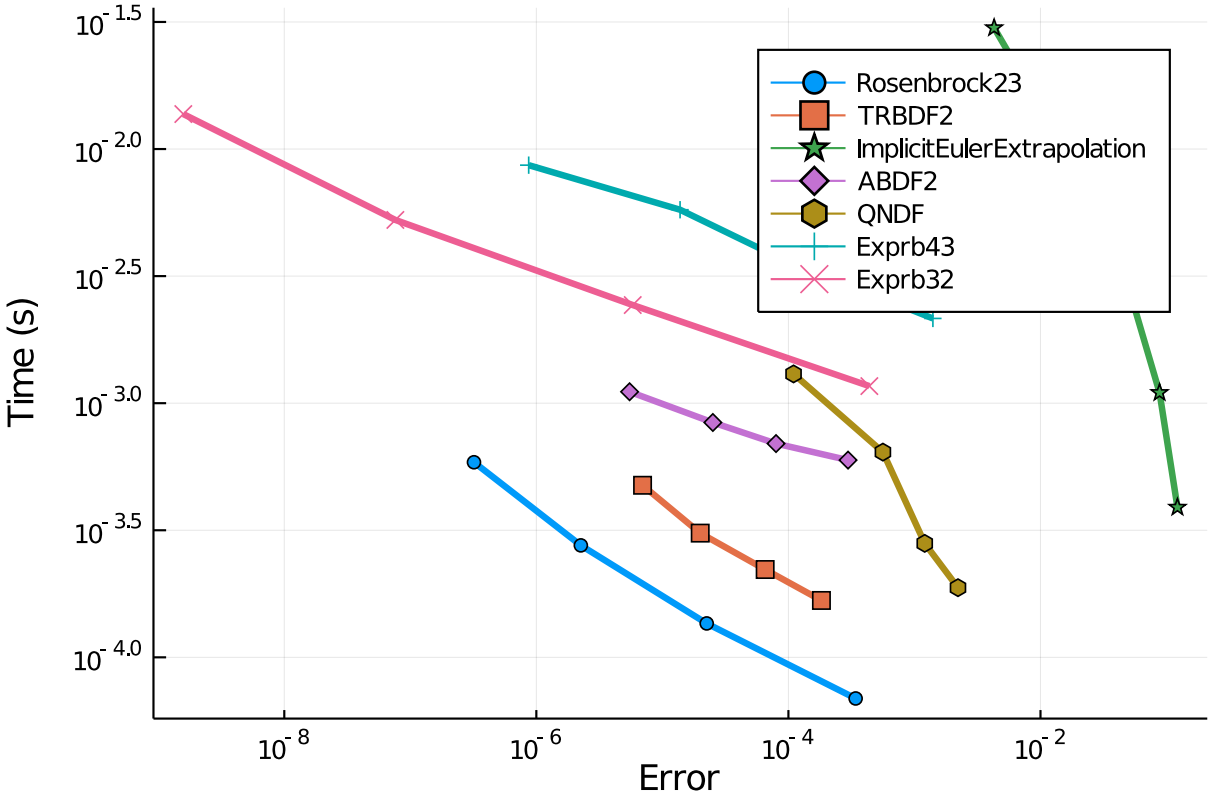
```
setups = [Dict(:alg=>Rosenbrock23()),
          Dict(:alg=>KenCarp5()),
          Dict(:alg=>KenCarp4()),
          Dict(:alg=>KenCarp3()),
          Dict(:alg=>ARKODE(order=5)),
          Dict(:alg=>ARKODE()),
          Dict(:alg=>ARKODE(order=3))]
names = ["Rosenbrock23" "KenCarp5" "KenCarp4" "KenCarp3" "ARKODE5" "ARKODE4" "ARKODE3"]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;

names=names,save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
plot(wp)
```




```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;dense = false,verbose=false,
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:l2)
plot(wp)
```




```
setups = [Dict(:alg=>Rosenbrock23()),
          Dict(:alg=>TRBDF2()),
          Dict(:alg=>ImplicitEulerExtrapolation()),
          #Dict(:alg=>ImplicitDeuflhardExtrapolation()), # Diverges
          #Dict(:alg=>ImplicitHairerWannerExtrapolation()), # Diverges
          Dict(:alg=>ABDF2()),
          Dict(:alg=>QNDF()),
          Dict(:alg=>Exprb43()),
          Dict(:alg=>Exprb32()),
]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
plot(wp)
```



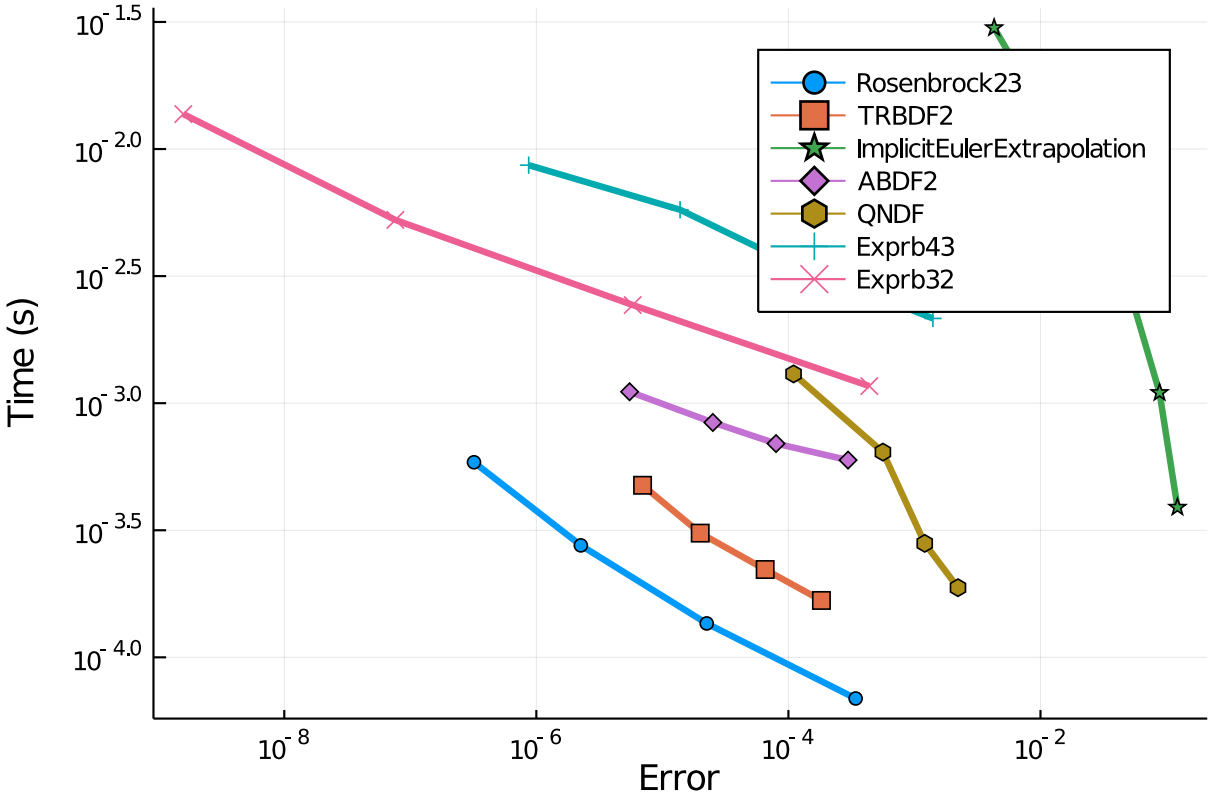
0.2.1 Low Tolerances

This is the speed at lower tolerances, measuring what's good when accuracy is needed.

```
abstols = 1.0 ./ 10.0 .^ (7:13)
reltols = 1.0 ./ 10.0 .^ (4:10)

setups = [Dict(:alg=>GRK4A()),
          Dict(:alg=>Rodas4P()),
          Dict(:alg=>CVODE_BDF()),
          Dict(:alg=>ddebdf()),
          Dict(:alg=>Rodas5()),
          Dict(:alg=>rodas()),
          Dict(:alg=>radau()),
          Dict(:alg=>lsoda()),
          Dict(:alg=>RadauIIA5()),
]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
```

```
Error: Cannot find method(s) for ddebdf! I've tried to loadODESolvers(), bu
t it didn't work. Please check ODEInterface.help_solversupport() and call l
oadODESolvers and check also this output. For further information see also
ODEInterface.help_install.
```

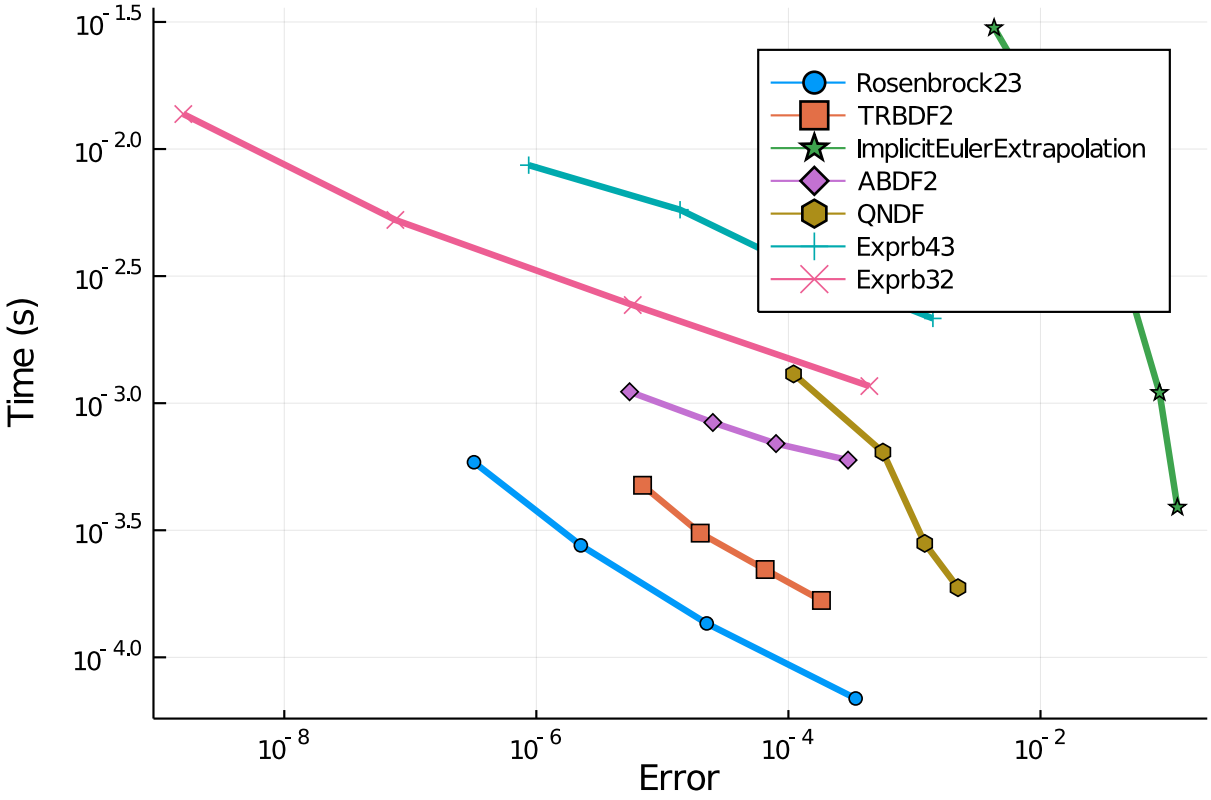
```
plot(wp)
```




```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;verbose=false,
                      dense=false,appxsol=test_sol,maxiters=Int(1e5),error_estimate=:l2)
```

```
Error: Cannot find method(s) for ddebdf! I've tried to loadODESolvers(), bu
t it didn't work. Please check ODEInterface.help_solversupport() and call l
oadODESolvers and check also this output. For further information see also
ODEInterface.help_install.
```

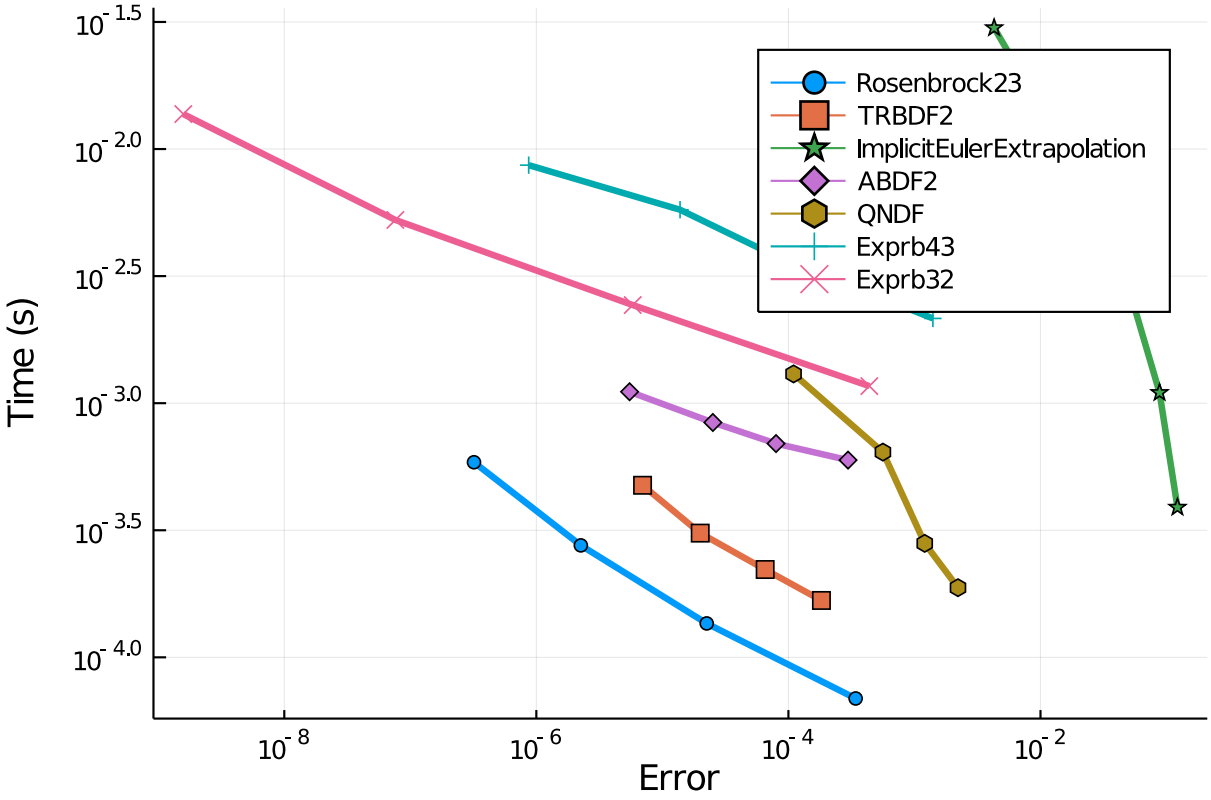
```
plot(wp)
```




```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:L2)
```

```
Error: Cannot find method(s) for ddebdf! I've tried to loadODESolvers(), bu
t it didn't work. Please check ODEInterface.help_solversupport() and call l
oadODESolvers and check also this output. For further information see also
ODEInterface.help_install.
```

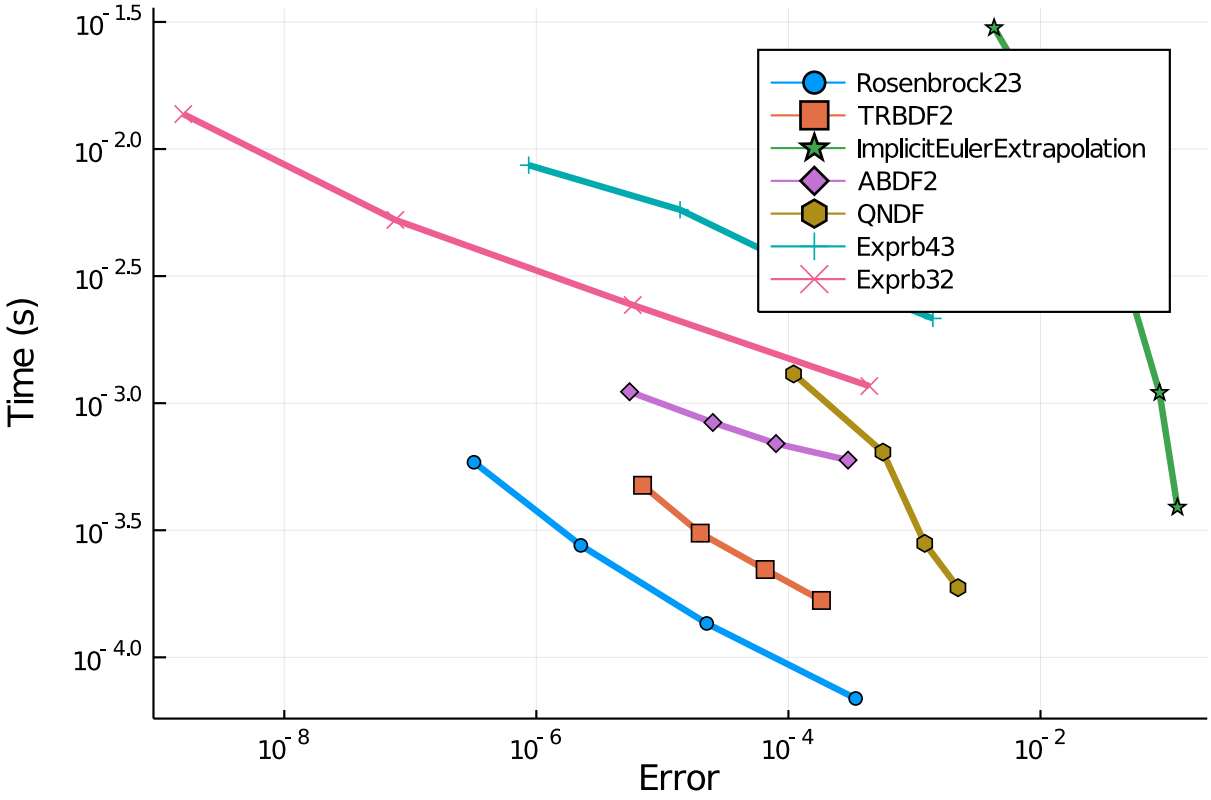
```
plot(wp)
```




```
setups = [
          Dict(:alg=>Rodas5()),
          Dict(:alg=>Kvaerno5()),
          Dict(:alg=>CVODE_BDF()),
          Dict(:alg=>KenCarp4()),
          Dict(:alg=>Rodas4()),
          Dict(:alg=>radau())]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
```

```
Error: Cannot find method(s) for radau! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

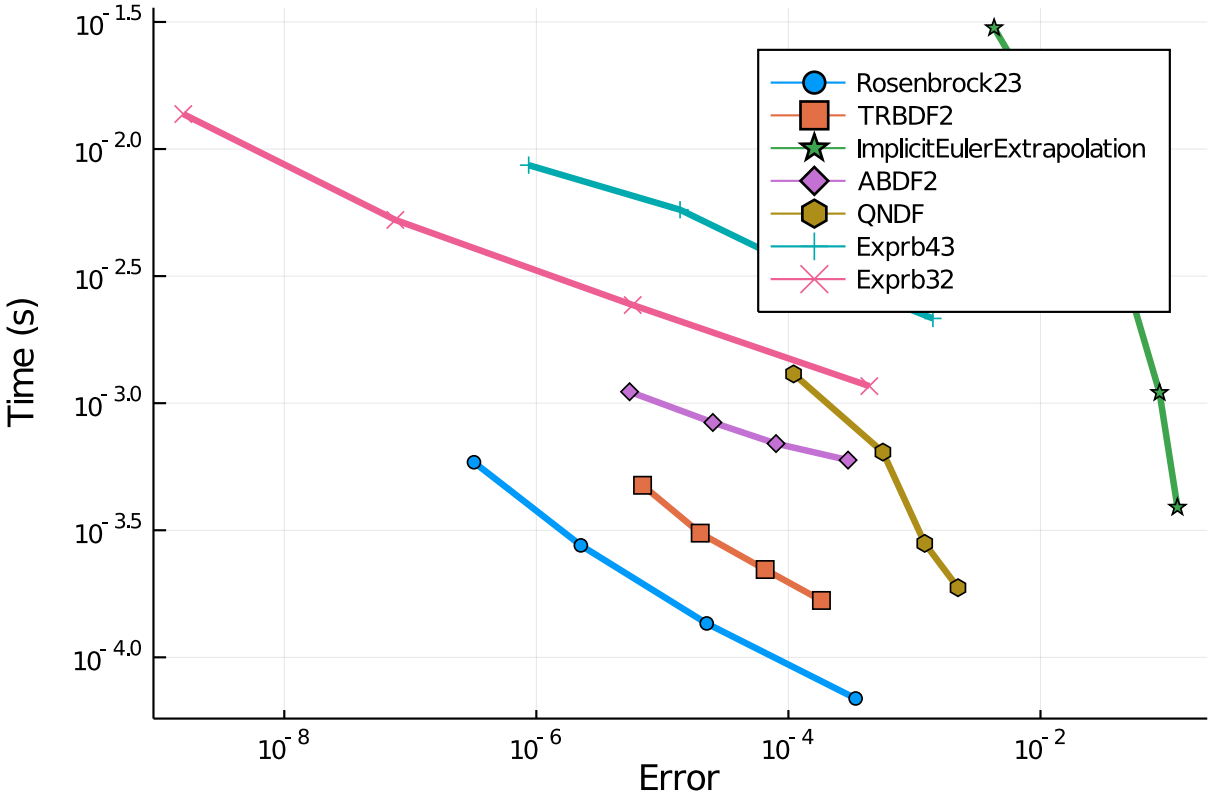
```
plot(wp)
```




```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;verbose=false,
                      dense=false,appxsol=test_sol,maxiters=Int(1e5),error_estimate=:l2)
```

```
Error: Cannot find method(s) for radau! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

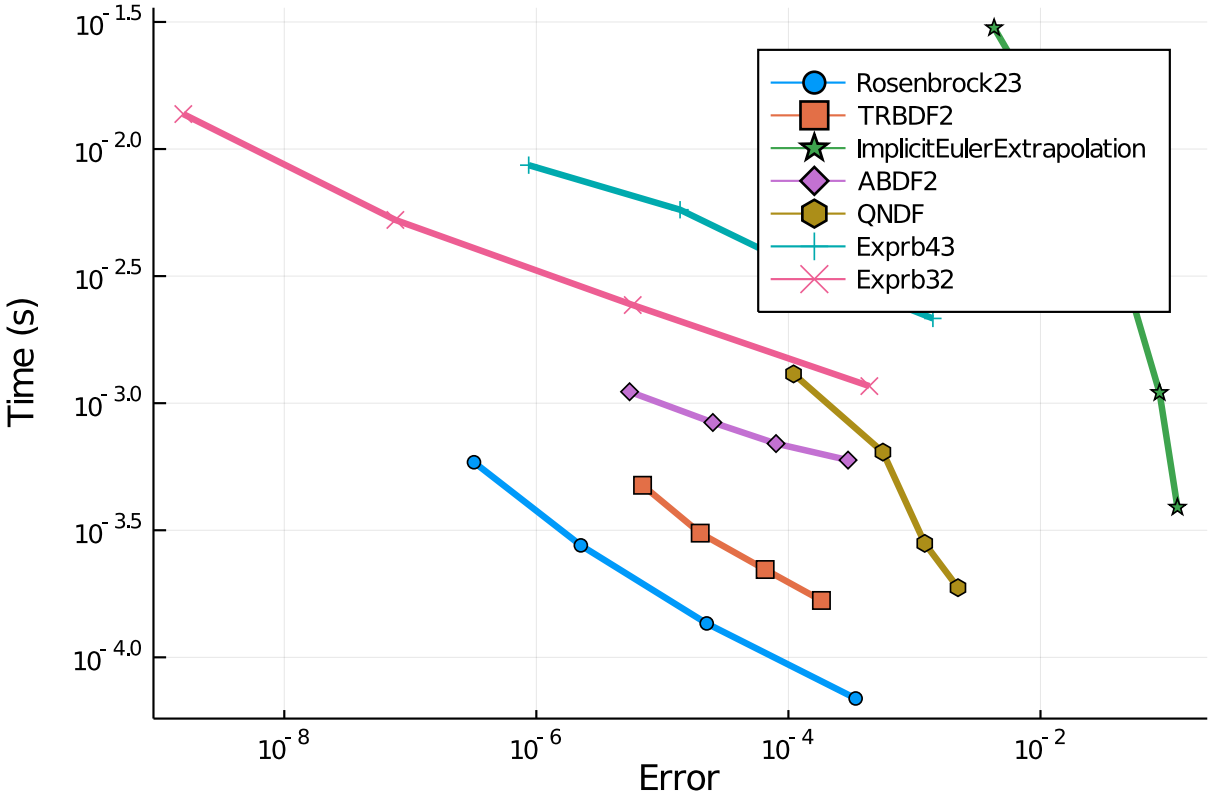
```
plot(wp)
```



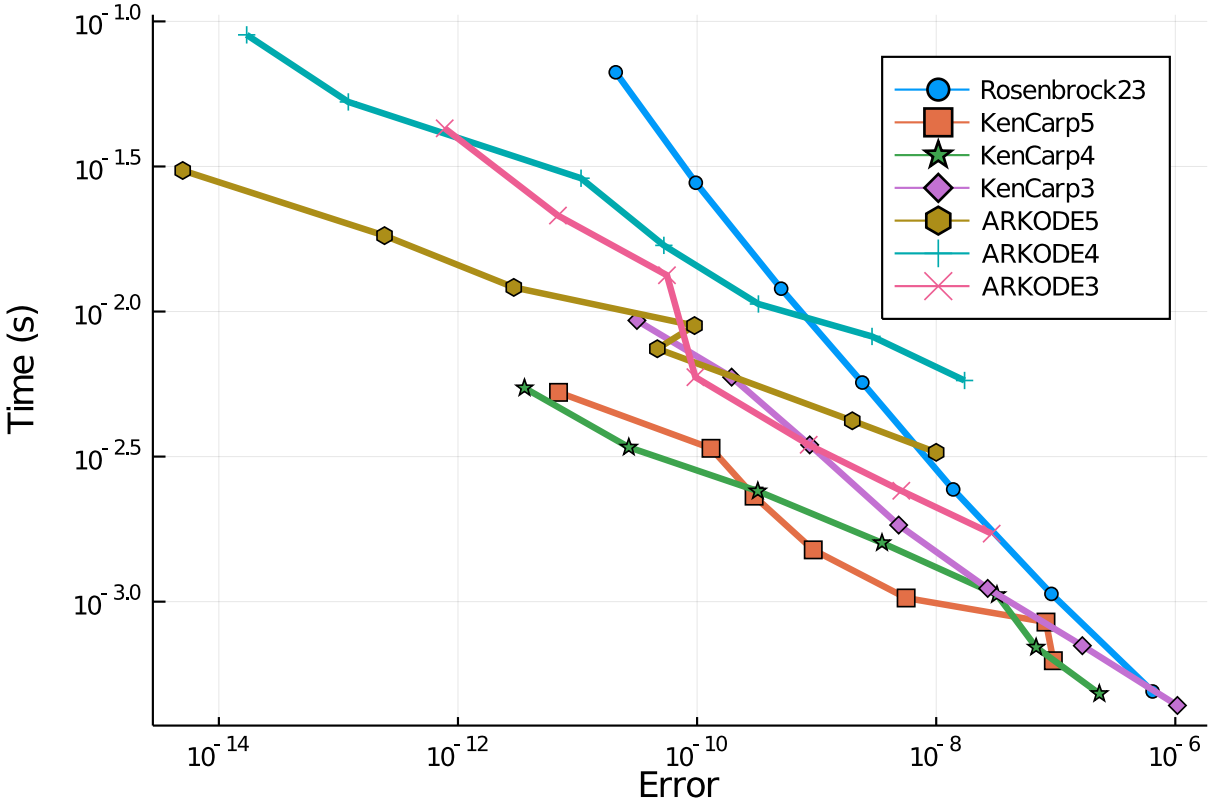

```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;
                      appxsol=test_sol,maxiters=Int(1e5),error_estimate=:L2)
```

```
Error: Cannot find method(s) for radau! I've tried to loadODESolvers(), but
 it didn't work. Please check ODEInterface.help_solversupport() and call lo
adODESolvers and check also this output. For further information see also O
DEInterface.help_install.
```

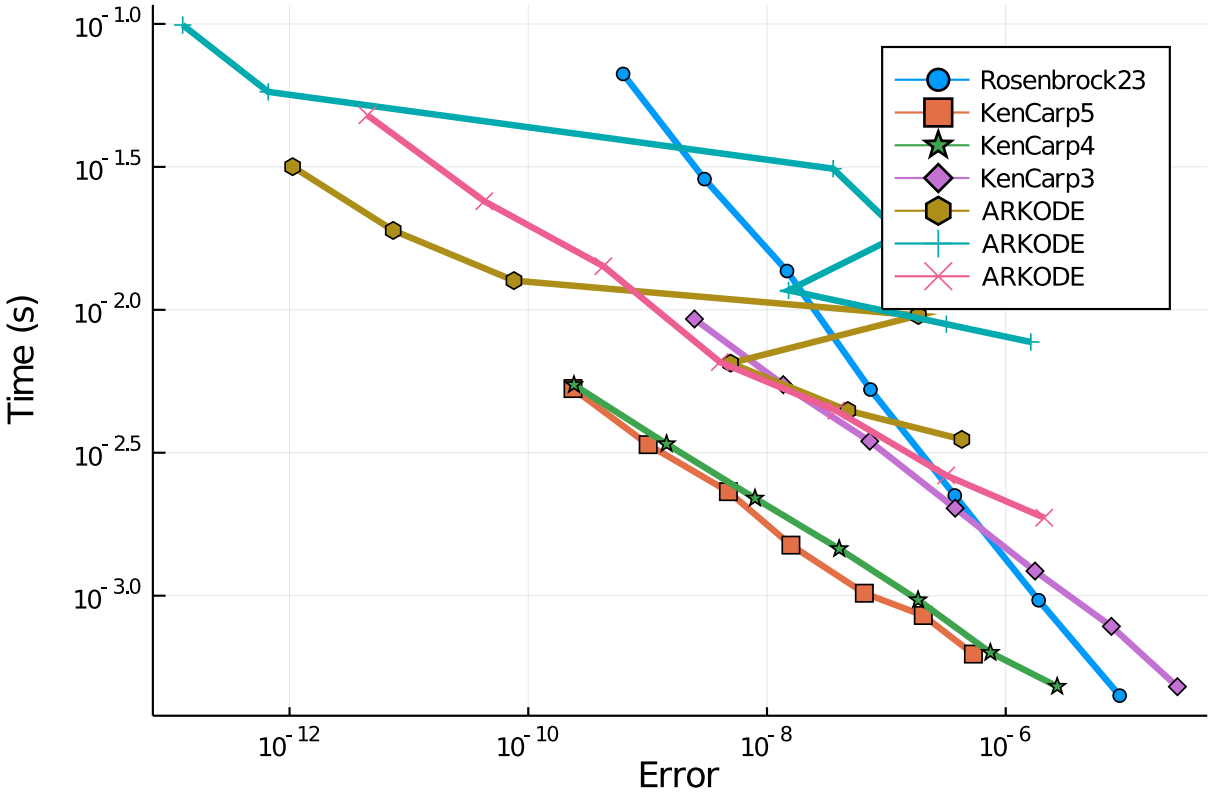
```
plot(wp)
```




```
setups = [Dict(:alg=>Rosenbrock23()),
          Dict(:alg=>KenCarp5()),
          Dict(:alg=>KenCarp4()),
          Dict(:alg=>KenCarp3()),
          Dict(:alg=>ARKODE(order=5)),
          Dict(:alg=>ARKODE()),
          Dict(:alg=>ARKODE(order=3))]
names = ["Rosenbrock23" "KenCarp5" "KenCarp4" "KenCarp3" "ARKODE5" "ARKODE4" "ARKODE3"]
wp = WorkPrecisionSet(prob,abstols,reltols,setups;

names=names,save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
plot(wp)
```




```
wp = WorkPrecisionSet(prob,abstols,reltols,setups;verbose=false,
                      dense=false,appxsol=test_sol,maxiters=Int(1e5),error_estimate=:l2)
plot(wp)
```



The following algorithms were removed since they failed.

```
#setups = [#Dict(:alg=>Hairer4()),
           #Dict(:alg=>Hairer42()),
           #Dict(:alg=>Rodas3()),
           #Dict(:alg=>Kvaerno4()),
           #Dict(:alg=>KenCarp5()),
           #Dict(:alg=>Cash4())
#]
#wp = WorkPrecisionSet(prob,abstols,reltols,setups;
#                      save_everystep=false,appxsol=test_sol,maxiters=Int(1e5))
#plot(wp)
```

0.2.2 Conclusion

At high tolerances, `Rosenbrock23` and `lsoda` hits the the error estimates and are fast. At lower tolerances and normal user tolerances, `Rodas5` is extremely fast. When you get down to `reltol=1e-10` `radau` begins to become as efficient as `Rodas4`, and it continues to do well below that.

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

0.3 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("StiffODE","Hires.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/DiffEqBenchmarks.jl/.julia
  JULIA_CUDA_MEMORY_LIMIT = 2147483648
  JULIA_PROJECT = @.
  JULIA_NUM_THREADS = 4
```

Package Information:

```
Status: `/builds/JuliaGPU/DiffEqBenchmarks.jl/benchmarks/StiffODE/Project.toml`
[eb300fae-53e8-50a0-950c-e21f52c2b7e0] DiffEqBiological 4.3.0
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.22.0
[5a33fad7-5ce4-5983-9f5d-5f26ceab5c96] GeometricIntegratorsDiffEq 0.1.0
[7f56f5a3-f504-529b-bc02-0b1fe5e64312] LSODA 0.6.1
[c030b06c-0b6d-57c2-b091-7029874bd033] ODE 2.5.0
[09606e27-ecf5-54fc-bb29-004bd9f985bf] ODEInterfaceDiffEq 3.7.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.41.0
[65888b18-ceab-5e60-b2b9-181511a3b968] ParameterizedFunctions 5.3.0
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.5.3
[b4db0fb7-de2a-5028-82bf-5021f5cfa881] ReactionNetworkImporters 0.1.5
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 4.2.5
[a759f4b9-e2f1-59dc-863e-4aeb61b1ea8f] TimerOutputs 0.5.6
[37e2e46d-f89d-539d-b4ee-838fcccc9c8e] LinearAlgebra
```