

# Burgers Pseudospectral Methods Work-Precision Diagrams

HAO HAO

July 25, 2020

```
using ApproxFun, OrdinaryDiffEq, Sundials
using DiffEqDevTools
using LinearAlgebra
using Plots; gr()
```

```
Plots.GRBackend()
```

Here is the Burgers equation using Fourier spectral methods.

```
S = Fourier()
n = 512
x = points(S, n)
D2 = Derivative(S,2)[1:n,1:n]
D = (Derivative(S) → S)[1:n,1:n]
T = ApproxFun.plan_transform(S, n)
Ti = ApproxFun.plan_itransform(S, n)
```

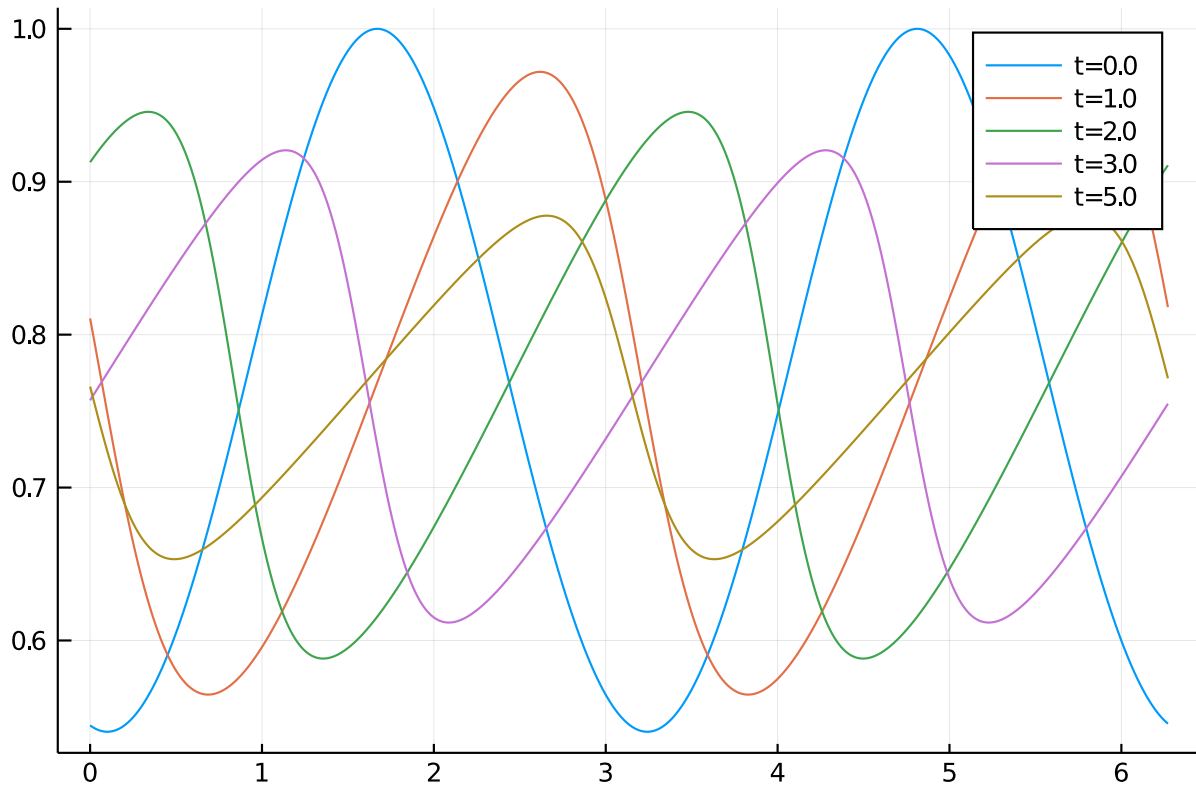
```
û_0 = T*cos.(cos.(x.-0.1))
A = 0.03*D2
tmp = similar(û_0)
p = (D,D2,T,Ti,tmp,similar(tmp))
function burgers_nl(dû,û,p,t)
    D,D2,T,Ti,u,tmp = p
    mul!(tmp, D, û)
    mul!(u, Ti, tmp)
    mul!(tmp, Ti, û)
    @. tmp = tmp*u
    mul!(u, T, tmp)
    @. dû = - u
end
```

```
burgers_nl (generic function with 1 method)
```

Reference solution using Rodas5 is below:

```
prob = SplitODEProblem(DiffEqArrayOperator(Diagonal(A)), burgers_nl, û_0, (0.0,5.0), p)
sol = solve(prob, Rodas5(autodiff=false); reltol=1e-12, abstol=1e-12)
test_sol = TestSolution(sol)

tslices=[0.0 1.0 2.0 3.0 5.0]
ys=hcata((Ti*sol(t) for t in tslices)...)
labels=["t=$t" for t in tslices]
plot(x,ys,label=labels)
```



## 0.1 High tolerances

```
diag_linsolve=LinSolveFactorize(W->let tmp = tmp
  for i in 1:size(W, 1)
    tmp[i] = W[i, i]
  end
  Diagonal(tmp)
end)
```

```
DiffEqBase.LinSolveFactorize{Main.##WeaveSandBox#313.var"#5#6"}(Main.##WeaveSandBox#313.var"#5#6"(), nothing)
```

## 0.2 In-family comparisons

1.IMEX methods (diagonal linear solver)

```
abstols = 0.1 .^ (5:8)
reltols = 0.1 .^ (1:4)
multipliers = 0.5 .^ (0:3)
setups = [Dict{:alg => IMEXEuler(linsolve=diag_linsolve), :dts => 1e-3 * multipliers},
  Dict{:alg => CNAB2(linsolve=diag_linsolve), :dts => 5e-3 * multipliers},
  Dict{:alg => CNLF2(linsolve=diag_linsolve), :dts => 5e-3 * multipliers},
  Dict{:alg => SBDF2(linsolve=diag_linsolve), :dts => 1e-3 * multipliers}]
labels = ["IMEXEuler" "CNAB2" "CNLF2" "SBDF2"]
@time wp1 = WorkPrecisionSet(prob,abstols,reltols,setups;
  print_names=true,names=labels,
  numruns=5,seconds=5,
  save_everystop=false,appxsol=test_sol,maxiters=Int(1e5));
```

```
IMEXEuler
Error: LinearAlgebra.SingularException(504)
```

```
plot(wp1,label=labels,markershape=:auto,title="IMEX methods, diagonal linsolve, low
order")
```

Error: UndefVarError: wp1 not defined

## 2. ExpRK methods

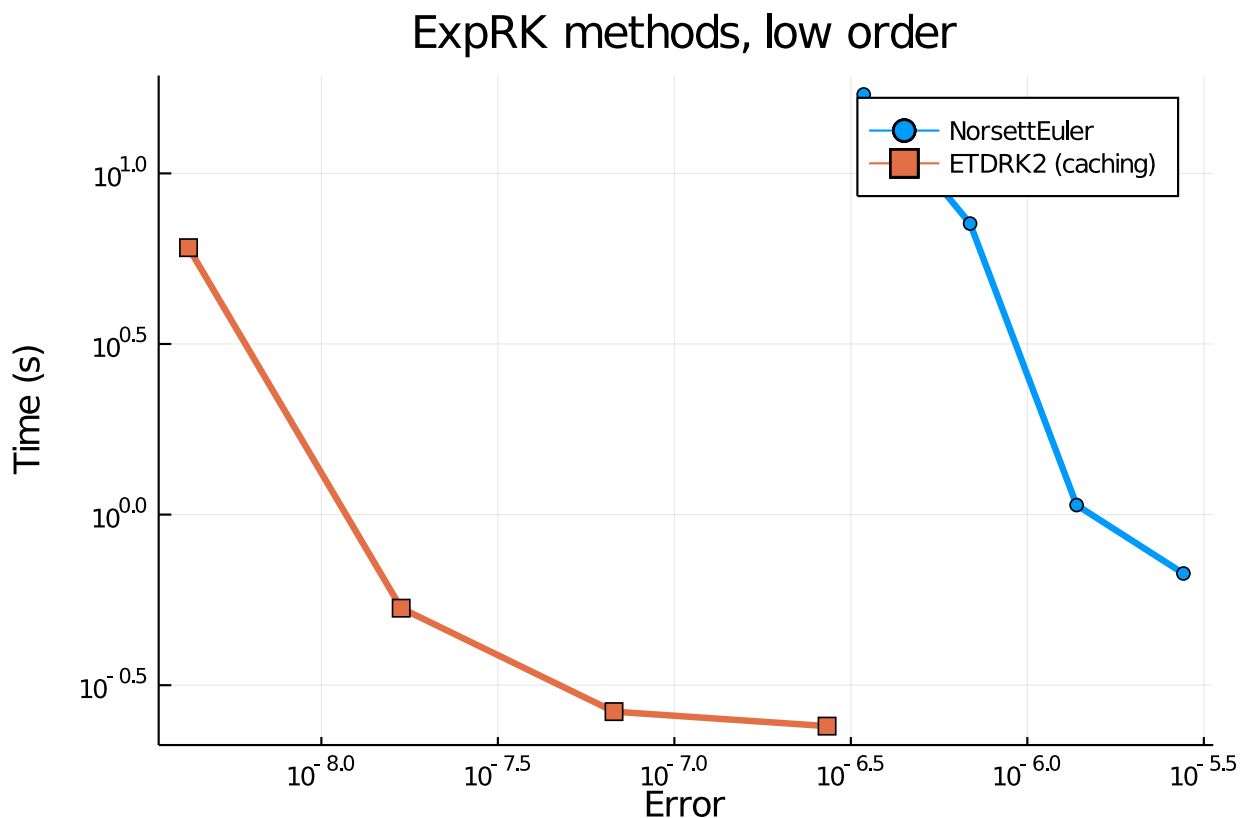
```
abstols = 0.1 .^ (5:8) # all fixed dt methods so these don't matter much
reltols = 0.1 .^ (1:4)
multipliers = 0.5 .^ (0:3)
setups = [Dict{:alg => NorsettEuler(), :dts => 1e-3 * multipliers},
          Dict{:alg => ETD RK2(), :dts => 1e-2 * multipliers}]
labels = hcat("NorsettEuler",
             "ETDRK2 (caching)")
@time wp2 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));
```

NorsettEuler

ETDRK2 (caching)

155.798635 seconds (55.98 M allocations: 4.763 GiB, 1.18% gc time)

```
plot(wp2, label=labels, markershape=:auto, title="ExpRK methods, low order")
```



## 0.3 Between family comparisons

```

abstols = 0.1 .^ (5:8) # all fixed dt methods so these don't matter much
reltols = 0.1 .^ (1:4)
multipliers = 0.5 .^ (0:3)
setup = [Dict(:alg => CNAB2(linsolve=diag_linsolve), :dts => 5e-3 * multipliers),
         Dict(:alg => ETDK2(), :dts => 1e-2 * multipliers)]
labels = ["CNAB2 (diagonal linsolve)" "ETDK2"]
@time wp3 = WorkPrecisionSet(prob,abstols,reltols,setup;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));

```

```

CNAB2 (diagonal linsolve)
Error: LinearAlgebra.SingularException(264)

```

```

plot(wp3, label=labels, markershape=:auto, title="Between family, low orders")

```

```

Error: UndefVarError: wp3 not defined

```

## 0.4 Low tolerances

## 0.5 In-family comparisons

1. IMEX methods (band linear solver)

```

abstols = 0.1 .^ (7:13)
reltols = 0.1 .^ (4:10)
setup = [Dict(:alg => ARKODE(Sundials.Implicit(), order=3, linear_solver=:Band,
                             jac_upper=1, jac_lower=1)),
         Dict(:alg => ARKODE(Sundials.Implicit(), order=4, linear_solver=:Band,
                             jac_upper=1, jac_lower=1)),
         Dict(:alg => ARKODE(Sundials.Implicit(), order=5, linear_solver=:Band,
                             jac_upper=1, jac_lower=1))]
labels = hcat("ARKODE3", "ARKODE4", "ARKODE5")
@time wp4 = WorkPrecisionSet(prob,abstols,reltols,setup;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));

```

```

ARKODE3
ARKODE4
ARKODE5
273.827022 seconds (72.73 M allocations: 6.911 GiB, 1.08% gc time)

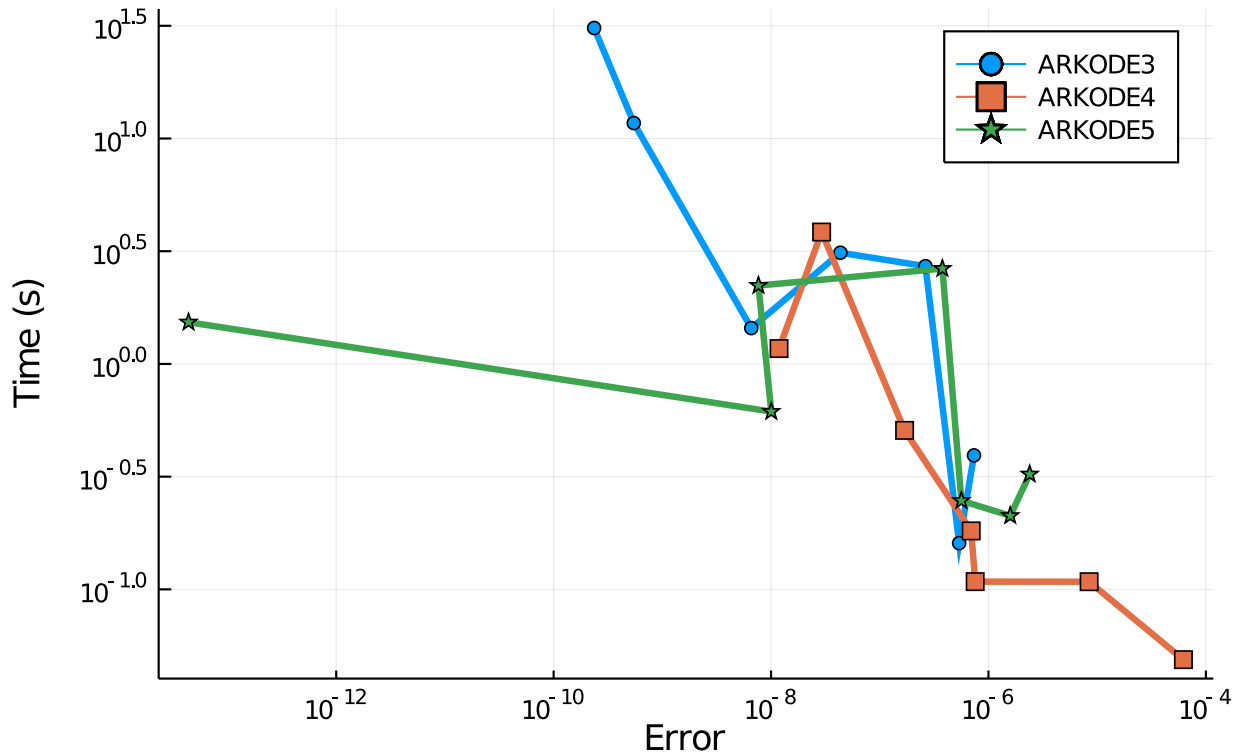
```

```

plot(wp4, label=labels, markershape=:auto, title="IMEX methods, band linsolve, medium
order")

```

## IMEX methods, band linsolve, medium order



## 2.ExprRK methods

```

abstols = 0.1 .^ (7:11) # all fixed dt methods so these don't matter much
reltols = 0.1 .^ (4:8)
multipliers = 0.5 .^ (0:4)
setups = [Dict(:alg => ETD RK3(), :dts => 1e-2 * multipliers),
          Dict(:alg => ETD RK4(), :dts => 1e-2 * multipliers),
          Dict(:alg => HochOst4(), :dts => 1e-2 * multipliers)]
labels = hcat("ETDRK3 (caching)", "ETDRK4 (caching)",
             "HochOst4 (caching)")
@time wp5 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));

```

```

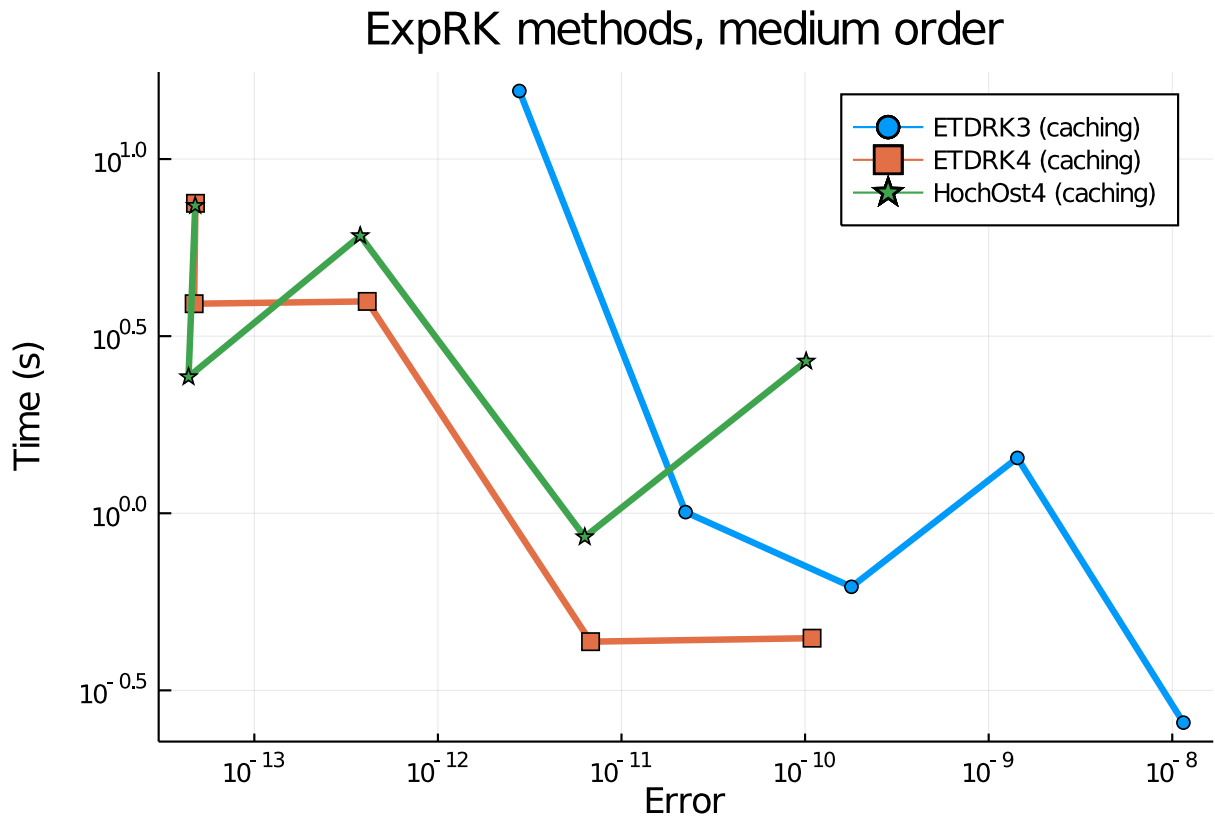
ETDRK3 (caching)
ETDRK4 (caching)
HochOst4 (caching)
253.057805 seconds (94.65 M allocations: 8.206 GiB, 1.28% gc time)

```

```

plot(wp5, label=labels, markershape=:auto, title="ExpRK methods, medium order")

```



## 0.6 Between family comparisons

```

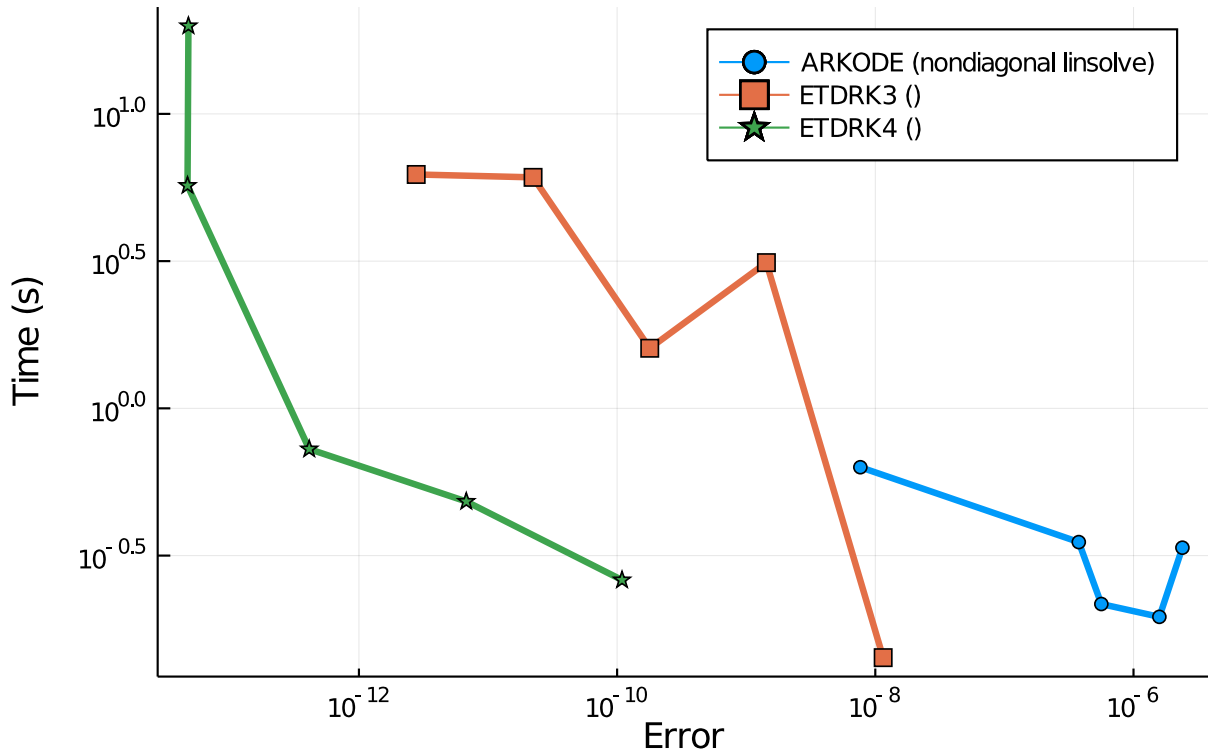
abstols = 0.1 .^ (7:11)
reltols = 0.1 .^ (4:8)
multipliers = 0.5 .^ (0:4)
setups = [Dict(:alg => ARKODE(Sundials.Implicit(), order=5, linear_solver=:Band,
jac_upper=1, jac_lower=1)),
          Dict(:alg => ETD RK3(), :dts => 1e-2 * multipliers),
          Dict(:alg => ETD RK4(), :dts => 1e-2 * multipliers)]
labels = hcat("ARKODE (nondiagonal linsolve)", "ETDRK3 ()", "ETDRK4 ()")
          #"ARKODE (Krylov linsolve)")
@time wp6 = WorkPrecisionSet(prob,abstols,reltols,setups;
                             print_names=true, names=labels,
                             numruns=5, error_estimate=:l2,
                             save_everystep=false, appxsol=test_sol, maxiters=Int(1e5));

ARKODE (nondiagonal linsolve)
ETDRK3 ()
ETDRK4 ()
191.437298 seconds (46.87 M allocations: 4.677 GiB, 1.10% gc time)

plot(wp6, label=labels, markershape=:auto, title="Between family, medium order")

```

## Between family, medium order



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

## 0.7 Appendix

These benchmarks are a part of the SciMLBenchmarks.jl repository, found at: <https://github.com/SciML/SciMLBenchmarks.jl>. For more information on high-performance scientific machine learning, check out the SciML Open Source Software Organization <https://sciml.ai>.

To locally run this benchmark, do the following commands:

```
using SciMLBenchmarks
SciMLBenchmarks.weave_file("MOLPDE","burgers_spectral_wpd.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/DiffEqBenchmarks.jl:
  JULIA_DEPOT_PATH = /builds/JuliaGPU/DiffEqBenchmarks.jl/.julia
```

```
JULIA_CUDA_MEMORY_LIMIT = 2147483648
JULIA_NUM_THREADS = 8
```

Package Information:

```
Status: `~/builds/JuliaGPU/DiffEqBenchmarks.jl/benchmarks/MOLPDE/Project.toml`
[28f2ccd6-bb30-5033-b560-165f7b14dc2f] ApproxFun 0.11.14
[f3b72e0c-5b89-59e1-b016-84e28bfd966d] DiffEqDevTools 2.24.0
[7f56f5a3-f504-529b-bc02-0b1fe5e64312] LSODA 0.6.1
[09606e27-ecf5-54fc-bb29-004bd9f985bf] ODEInterfaceDiffEq 3.7.0
[1dea7af3-3e70-54e6-95c3-0bf5283fa5ed] OrdinaryDiffEq 5.41.0
[91a5bcdd-55d7-5caf-9e0b-520d859cae80] Plots 1.5.6
[c3572dad-4567-51f8-b174-8c6c989267f4] Sundials 4.2.5
[37e2e46d-f89d-539d-b4ee-838fcccc9c8e] LinearAlgebra
```