FitzHugh-Nagumo Parameter Estimation Benchmarks

Vaibhav Dixit, Chris Rackauckas

March 16, 2019

1 Parameter estimation of FitzHugh-Nagumo model using optimisation methods

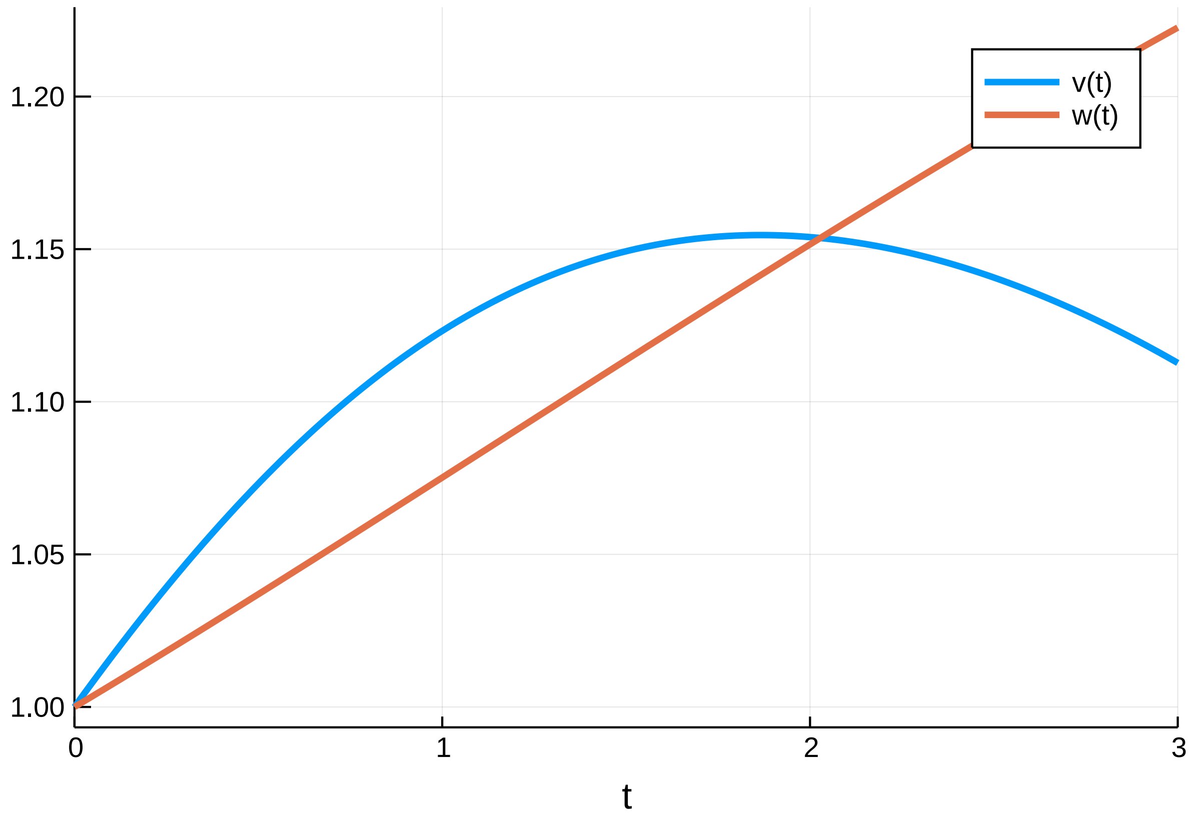
```julia
using ParameterizedFunctions, OrdinaryDiffEq, DiffEqParamEstim
using BlackBoxOptim, NLopt, Plots,QuadDIRECT
gr(fmt=:png)
```

```
Plots.GRBackend()
```

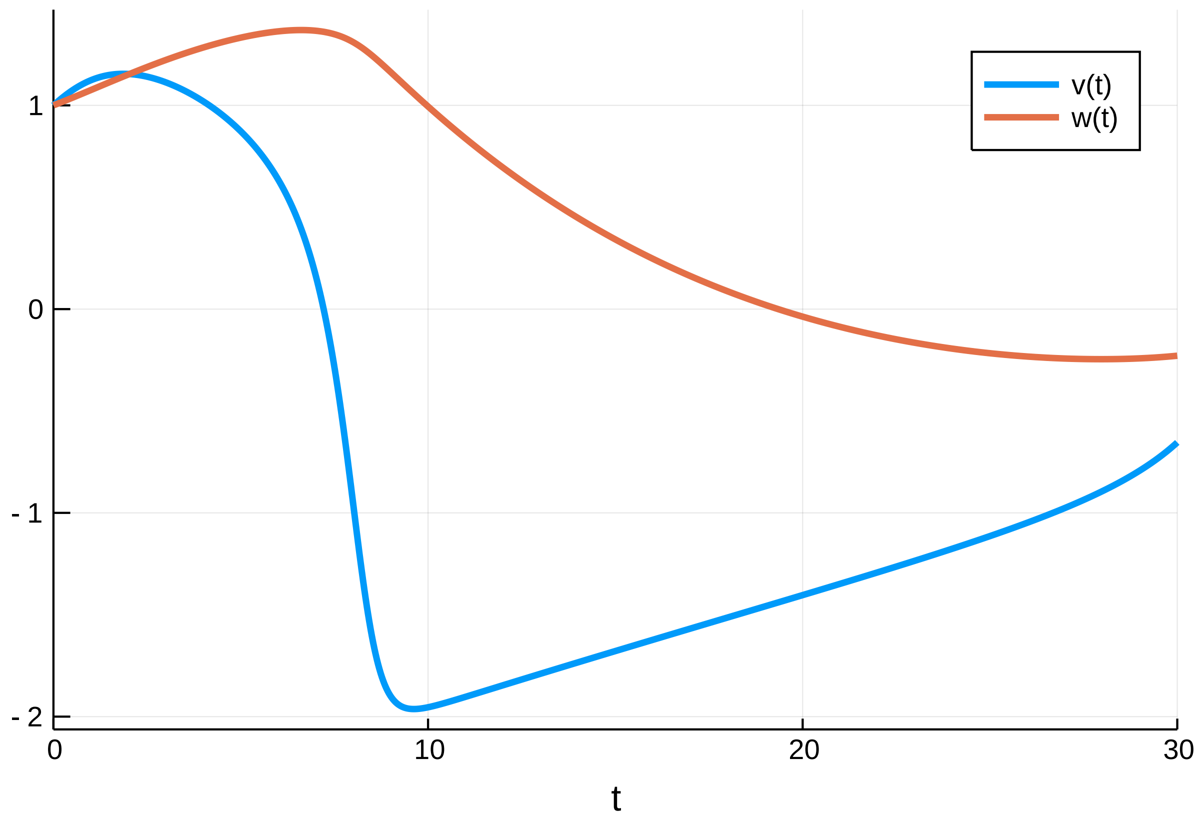
```julia
loc_bounds = Tuple{Float64,Float64}[(0, 1), (0, 1), (0, 1), (0, 1)]
glo_bounds = Tuple{Float64,Float64}[(0, 5), (0, 5), (0, 5), (0, 5)]
loc_init = [0.5,0.5,0.5,0.5]
glo_init = [2.5,2.5,2.5,2.5]

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

p = [0.7,0.8,0.08,0.5]              # Parameters used to construct the dataset
r0 = [1.0; 1.0]                     # initial value
tspan = (0.0, 30.0)                 # sample of 3000 observations over the (0,30)
    timespan
prob = ODEProblem(fitz, r0, tspan,p)
tspan2 = (0.0, 3.0)                 # sample of 300 observations with a timestep of 0.01
prob_short = ODEProblem(fitz, r0, tspan2,p)

dt = 30.0/3000
tf = 30.0
tinterval = 0:dt:tf
t  = collect(tinterval)

h = 0.01
M = 300
tstart = 0.0
tstop = tstart + M * h
tinterval_short = 0:h:tstop
t_short = collect(tinterval_short)

#Generate Data
data_sol_short = solve(prob_short,Vern9(),saveat=t_short,reltol=1e-9,abstol=1e-9)
data_short = convert(Array, data_sol_short) # This operation produces column major
    dataset obs as columns, equations as rows
data_sol = solve(prob,Vern9(),saveat=t,reltol=1e-9,abstol=1e-9)
data = convert(Array, data_sol)
```

Plot of the solution

```
plot(data_sol_short)
```



```
plot(data_sol)
```



1.1 Local Solution from the short data set

```
obj_short =
    build_loss_objective(prob_short,Tsit5(),L2Loss(t_short,data_short),tstops=t_short)
res1 = bboptimize(obj_short;SearchRange = glo_bounds, MaxSteps = 7e3)
```

```
Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.RangePerDi
mSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 1580 evals, 1450 steps, improv/step: 0.215 (last = 0.2152), fitn
ess=0.003747732
1.00 secs, 3290 evals, 3160 steps, improv/step: 0.154 (last = 0.1029), fitn
ess=0.003747732
1.50 secs, 4975 evals, 4845 steps, improv/step: 0.133 (last = 0.0920), fitn
ess=0.001792841
2.00 secs, 6804 evals, 6674 steps, improv/step: 0.126 (last = 0.1099), fitn
ess=0.000071607

Optimization stopped after 7001 steps and 2.0963449478149414 seconds
Termination reason: Max number of steps (7000) reached
Steps per second = 3339.62213961842
Function evals per second = 3401.634834683467
Improvements/step = 0.12614285714285714
Total function evaluations = 7131


Best candidate found: [0.57224, 0.78169, 0.0903249, 0.499753]
```

```
Fitness: 0.000019579

# Lower tolerance could lead to smaller fitness (more accuracy)

obj_short =
    build_loss_objective(prob_short,Tsit5(),L2Loss(t_short,data_short),tstops=t_short,reltol=1e-9)
res1 = bboptimize(obj_short;SearchRange = glo_bounds, MaxSteps = 7e3)

Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.RangePerDi
mSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 1771 evals, 1647 steps, improv/step: 0.234 (last = 0.2344), fitn
ess=0.122690564
1.00 secs, 3596 evals, 3472 steps, improv/step: 0.171 (last = 0.1145), fitn
ess=0.003434369
1.50 secs, 5202 evals, 5079 steps, improv/step: 0.152 (last = 0.1101), fitn
ess=0.000874396
2.00 secs, 6766 evals, 6643 steps, improv/step: 0.151 (last = 0.1483), fitn
ess=0.000161415

Optimization stopped after 7001 steps and 2.09977388381958 seconds
Termination reason: Max number of steps (7000) reached
Steps per second = 3334.168528310713
Function evals per second = 3392.7462642030455
Improvements/step = 0.152
Total function evaluations = 7124


Best candidate found: [0.217735, 0.724688, 0.139631, 0.499946]

Fitness: 0.000161415

# Change in tolerance makes it worse

obj_short =
    build_loss_objective(prob_short,Vern9(),L2Loss(t_short,data_short),tstops=t_short,reltol=1e-9,abst
res1 = bboptimize(obj_short;SearchRange = glo_bounds, MaxSteps = 7e3)

Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.RangePerDi
mSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 1336 evals, 1216 steps, improv/step: 0.285 (last = 0.2845), fitn
ess=0.099610322
1.00 secs, 2786 evals, 2666 steps, improv/step: 0.217 (last = 0.1607), fitn
ess=0.021569370
1.50 secs, 4107 evals, 3987 steps, improv/step: 0.183 (last = 0.1143), fitn
ess=0.007132159
2.00 secs, 5342 evals, 5222 steps, improv/step: 0.165 (last = 0.1069), fitn
ess=0.003526182
2.52 secs, 6411 evals, 6291 steps, improv/step: 0.159 (last = 0.1310), fitn
ess=0.001576092

Optimization stopped after 7001 steps and 2.8826050758361816 seconds
Termination reason: Max number of steps (7000) reached
Steps per second = 2428.7059155923953
```

```
Function evals per second = 2469.641110284568
Improvements/step = 0.15685714285714286
Total function evaluations = 7119


Best candidate found: [0.0586887, 0.774558, 0.22912, 0.499901]

Fitness: 0.000978444

# using the moe accurate Vern9() reduces the fitness marginally and leads to some
    increase in time taken
```

1.2 Using NLopt

```
obj_short =
    build_loss_objective(prob_short,Vern9(),L2Loss(t_short,data_short),tstops=t_short,reltol=1e-9,abst

opt = Opt(:GN_ORIG_DIRECT_L, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

3.359506 seconds (12.93 M allocations: 1.156 GiB, 11.17% gc time)
(0.11016600768053908, [0.192044, 1.13169, 1.11111, 0.509578], :XTOL_REACHED
)

opt = Opt(:GN_CRS2_LM, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

3.315937 seconds (13.41 M allocations: 1.199 GiB, 11.46% gc time)
(6.167287337059935e-14, [0.699996, 0.8, 0.0800003, 0.5], :MAXEVAL_REACHED)

opt = Opt(:GN_ISRES, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

3.363021 seconds (13.41 M allocations: 1.199 GiB, 11.39% gc time)
(0.001624385026981637, [0.0806233, 0.771639, 0.210907, 0.498457], :MAXEVAL_
REACHED)

opt = Opt(:GN_ESCH, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj_short.cost_function2)
```

```
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

3.232063 seconds (13.41 M allocations: 1.199 GiB, 11.50% gc time)
(0.0006025074709344539, [0.306754, 0.349008, 0.0723259, 0.498673], :MAXEVAL
_REACHED)
```

Now local optimization algorithms are used to check the global ones, these use the local constraints, different intial values and time step

```
opt = Opt(:LN_BOBYQA, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

0.778110 seconds (3.18 M allocations: 290.917 MiB, 11.15% gc time)
(5.28611876473971e-24, [0.7, 0.8, 0.08, 0.5], :SUCCESS)

opt = Opt(:LN_NELDERMEAD, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

0.311784 seconds (1.24 M allocations: 113.667 MiB, 12.72% gc time)
(8.965505337540987e-5, [1.0, 1.0, 0.0735509, 0.500405], :XTOL_REACHED)

opt = Opt(:LD_SLSQP, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

0.582239 seconds (2.33 M allocations: 210.018 MiB, 11.07% gc time)
(3.732246654524026e-14, [0.699986, 0.799998, 0.080001, 0.5], :XTOL_REACHED)

opt = Opt(:LN_COBYLA, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

3.647066 seconds (13.41 M allocations: 1.199 GiB, 10.97% gc time)
(0.0007533012926827143, [0.182327, 0.834403, 0.195088, 0.500362], :MAXEVAL_
REACHED)

opt = Opt(:LN_NEWUOA_BOUND, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
```

```
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

0.139464 seconds (311.13 k allocations: 28.479 MiB, 10.54% gc time)
(0.0003911847251197023, [0.320842, 0.439003, 0.078888, 0.499177], :SUCCESS)

opt = Opt(:LN_PRAXIS, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

0.330066 seconds (1.23 M allocations: 112.562 MiB, 11.58% gc time)
(4.78123269843345e-25, [0.7, 0.8, 0.08, 0.5], :XTOL_REACHED)

opt = Opt(:LN_SBPLX, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

3.811030 seconds (13.41 M allocations: 1.199 GiB, 10.96% gc time)
(8.350546444056411e-15, [0.700007, 0.800002, 0.0799996, 0.5], :MAXEVAL_REAC
HED)

opt = Opt(:LD_MMA, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)

32.941630 seconds (120.39 M allocations: 10.780 GiB, 11.05% gc time)
(9.930463767834086e-5, [0.231856, 0.706134, 0.130932, 0.49972], :MAXEVAL_RE
ACHED)
```

1.2.1 Now the longer problem is solved for a global solution

Vern9 solver with reltol=1e-9 and abstol=1e-9 is used and the dataset is increased to 3000 observations per variable with the same integration time step of 0.01.

```
obj = build_loss_objective(prob,Vern9(),L2Loss(t,data),tstops=t,reltol=1e-9,abstol=1e-9)
res1 = bboptimize(obj;SearchRange = glo_bounds, MaxSteps = 4e3)

Starting optimization with optimizer BlackBoxOptim.DiffEvoOpt{BlackBoxOptim
.FitPopulation{Float64},BlackBoxOptim.RadiusLimitedSelector,BlackBoxOptim.A
daptiveDiffEvoRandBin{3},BlackBoxOptim.RandomBound{BlackBoxOptim.RangePerDi
mSearchSpace}}
0.00 secs, 0 evals, 0 steps
0.50 secs, 150 evals, 85 steps, improv/step: 0.447 (last = 0.4471), fitness
```

```
=1242.652944781
1.00 secs, 290 evals, 196 steps, improv/step: 0.388 (last = 0.3423), fitnes
s=1242.652944781
1.50 secs, 433 evals, 329 steps, improv/step: 0.362 (last = 0.3233), fitnes
s=1242.652944781
2.00 secs, 581 evals, 465 steps, improv/step: 0.342 (last = 0.2941), fitnes
s=1242.652944781
2.50 secs, 737 evals, 617 steps, improv/step: 0.323 (last = 0.2632), fitnes
s=980.136433884
3.01 secs, 877 evals, 752 steps, improv/step: 0.305 (last = 0.2222), fitnes
s=450.872624587
3.51 secs, 1002 evals, 874 steps, improv/step: 0.284 (last = 0.1557), fitne
ss=450.872624587
4.01 secs, 1126 evals, 998 steps, improv/step: 0.266 (last = 0.1371), fitne
ss=450.872624587
4.51 secs, 1239 evals, 1111 steps, improv/step: 0.248 (last = 0.0973), fitn
ess=450.872624587
5.02 secs, 1352 evals, 1224 steps, improv/step: 0.235 (last = 0.1062), fitn
ess=450.872624587
5.52 secs, 1480 evals, 1352 steps, improv/step: 0.226 (last = 0.1328), fitn
ess=159.370416451
6.02 secs, 1623 evals, 1495 steps, improv/step: 0.213 (last = 0.0909), fitn
ess=159.370416451
6.52 secs, 1762 evals, 1634 steps, improv/step: 0.204 (last = 0.1079), fitn
ess=159.370416451
7.02 secs, 1904 evals, 1776 steps, improv/step: 0.195 (last = 0.0986), fitn
ess=159.370416451
7.52 secs, 2049 evals, 1921 steps, improv/step: 0.186 (last = 0.0759), fitn
ess=159.370416451
8.03 secs, 2197 evals, 2069 steps, improv/step: 0.182 (last = 0.1284), fitn
ess=159.370416451
8.53 secs, 2345 evals, 2217 steps, improv/step: 0.178 (last = 0.1216), fitn
ess=159.370416451
9.04 secs, 2474 evals, 2346 steps, improv/step: 0.172 (last = 0.0620), fitn
ess=159.370416451
9.54 secs, 2617 evals, 2489 steps, improv/step: 0.166 (last = 0.0699), fitn
ess=159.370416451
10.04 secs, 2758 evals, 2630 steps, improv/step: 0.162 (last = 0.0922), fit
ness=159.370416451
10.54 secs, 2900 evals, 2772 steps, improv/step: 0.161 (last = 0.1338), fit
ness=159.370416451
11.04 secs, 3034 evals, 2906 steps, improv/step: 0.161 (last = 0.1642), fit
ness=159.370416451
11.55 secs, 3170 evals, 3042 steps, improv/step: 0.157 (last = 0.0809), fit
ness=159.370416451
12.05 secs, 3306 evals, 3178 steps, improv/step: 0.156 (last = 0.1250), fit
ness=159.370416451
12.56 secs, 3446 evals, 3318 steps, improv/step: 0.154 (last = 0.1143), fit
ness=159.370416451
13.06 secs, 3583 evals, 3455 steps, improv/step: 0.155 (last = 0.1898), fit
ness=20.468198132
13.57 secs, 3712 evals, 3584 steps, improv/step: 0.154 (last = 0.1085), fit
ness=20.468198132
14.07 secs, 3829 evals, 3701 steps, improv/step: 0.154 (last = 0.1709), fit
ness=20.468198132
14.57 secs, 3962 evals, 3834 steps, improv/step: 0.155 (last = 0.1880), fit
ness=20.468198132
15.08 secs, 4096 evals, 3968 steps, improv/step: 0.155 (last = 0.1343), fit
ness=20.468198132
```

```
Optimization stopped after 4001 steps and 15.192729949951172 seconds
Termination reason: Max number of steps (4000) reached
Steps per second = 263.34964243953135
Function evals per second = 271.7747247270245
Improvements/step = 0.1545
Total function evaluations = 4129


Best candidate found: [0.791468, 0.847595, 0.0848849, 0.543134]

Fitness: 20.468198132
```

```
opt = Opt(:GN_ORIG_DIRECT_L, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
28.545046 seconds (98.95 M allocations: 8.149 GiB, 9.97% gc time)
(81.06091854762182, [1.11111, 1.11111, 0.100594, 0.576132], :XTOL_REACHED)
```

```
opt = Opt(:GN_CRS2_LM, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 20000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
28.251575 seconds (95.17 M allocations: 7.837 GiB, 9.96% gc time)
(8.245888484957745e-19, [0.7, 0.8, 0.08, 0.5], :XTOL_REACHED)
```

```
opt = Opt(:GN_ISRES, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 50000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
172.675152 seconds (607.55 M allocations: 50.034 GiB, 9.99% gc time)
(1.0320122178245084e-15, [0.7, 0.8, 0.08, 0.5], :MAXEVAL_REACHED)
```

```
opt = Opt(:GN_ESCH, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[5.0,5.0,5.0,5.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 20000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
63.944282 seconds (243.02 M allocations: 20.014 GiB, 10.04% gc time)
(154.4828405156447, [0.826252, 1.23927, 0.101523, 0.422621], :MAXEVAL_REACH
ED)
```

```
opt = Opt(:LN_BOBYQA, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)
```

```
1.251427 seconds (4.78 M allocations: 402.706 MiB, 9.78% gc time)
(8.090325025142272e-19, [0.7, 0.8, 0.08, 0.5], :XTOL_REACHED)
```

```
opt = Opt(:LN_NELDERMEAD, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-9)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)
```

```
1.698981 seconds (6.49 M allocations: 547.189 MiB, 10.00% gc time)
(3160.405522281567, [1.0, 1.0, 1.0, 0.865699], :XTOL_REACHED)
```

```
opt = Opt(:LD_SLSQP, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[1.0,1.0,1.0,1.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)
```

```
0.815602 seconds (3.10 M allocations: 261.275 MiB, 10.00% gc time)
(3160.4056697694823, [1.0, 1.0, 1.0, 0.865765], :XTOL_REACHED)
```

As expected from other problems the longer sample proves to be extremely challenging for some of the global optimizers. A few give the accurate values, while others seem to struggle with accuracy a lot.

```
obj_short =
    build_loss_objective(prob_short,Tsit5(),L2Loss(t_short,data_short),tstops=t_short)
lower = [0,0,0,0]
upper = [1,1,1,1]
splits = ([0,0.3,0.7],[0,0.3,0.7],[0,0.3,0.7],[0,0.3,0.7])
@time root, x0 = analyze(obj_short,splits,lower,upper)
```

```
1.894278 seconds (5.71 M allocations: 400.681 MiB, 16.17% gc time)
(BoxRoot@[NaN, NaN, NaN, NaN], [0.3, 0.3, 0.3, 0.3])
```

```
minimum(root)
```

```
Box0.0006293336660765138@[0.595183, 0.480868, 0.0638946, 0.500513]
```

```
obj = build_loss_objective(prob,Vern9(),L2Loss(t,data),tstops=t,reltol=1e-9,abstol=1e-9)
lower = [0,0,0,0]
upper = [5,5,5,5]
splits = ([0,0.5,1],[0,0.5,1],[0,0.5,1],[0,0.5,1])
@time root, x0 = analyze(obj_short,splits,lower,upper)
```

```
0.175610 seconds (927.04 k allocations: 80.222 MiB, 10.61% gc time)
(BoxRoot@[NaN, NaN, NaN, NaN], [0.5, 0.5, 0.5, 0.5])
```

```
minimum(root)
```

```
Box0.012650644127007372@[0.166695, 0.999023, 0.389981, 0.499926]
```

2 Conclusion

It is observed that lower tolerance lead to higher accuracy but too low tolerance could affect the convergance time drastically. Also fitting a shorter timespan seems to be easier in comparision (quite intutively). NLOpt methods seem to give great accuracy in the shorter problem with a lot of the algorithms giving 0 fitness, BBO performs very well on it with marginal change with tol values. In case of global optimization of the longer problem there is some difference in the perfomance amongst the algorithms with :LN*BOBYQA giving accurate results for the local optimization and :GN*ISRES :GN*CRS2*LM in case of the global give the highest accuracy. BBO also fails to perform too well in the case of the longer problem. QuadDIRECT performs well in case of the shorter problem but fails to give good results in the longer version.