Lotka-Volterra Parameter Estimation Benchmarks

Vaibhav Dixit, Chris Rackauckas

March 23, 2019

1 Parameter estimation of Lotka Volterra model using optimisation methods

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

Plots.GRBackend()

loc_bounds = Tuple{Float64, Float64}[(0, 5), (0, 5), (0, 5), (0, 5)]
glo_bounds = Tuple{Float64, Float64}[(0, 10), (0, 10), (0, 10), (0, 10)]
loc_init = [1,0.5,3.5,1.5]
glo_init = [5,5,5,5]

f = @ode_def LotkaVolterraTest begin
    dx = a*x - b*x*y
    dy = -c*y + d*x*y
end a b c d

u0 = [1.0,1.0]                          #initial values
tspan = (0.0,10.0)
p = [1.5,1.0,3.0,1,0]                   #parameters used, these need to be estimated from
    the data
tspan = (0.0, 30.0)                     # sample of 3000 observations over the (0,30)
    timespan
prob = ODEProblem(f, u0, tspan,p)
tspan2 = (0.0, 3.0)                     # sample of 3000 observations over the (0,30)
    timespan
prob_short = ODEProblem(f, u0, 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,Tsit5(),saveat=t_short,reltol=1e-9,abstol=1e-9)
data_short = convert(Array, data_sol_short)
data_sol = solve(prob,Tsit5(),saveat=t,reltol=1e-9,abstol=1e-9)
data = convert(Array, data_sol)
```

Plot of the solution

```
p1 = plot(data_sol_short)
```

```
p2 = plot(data_sol)
```

1.0.1 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, 1731 evals, 1611 steps, improv/step: 0.207 (last = 0.2073), fitn
ess=473.417248053
1.00 secs, 3431 evals, 3311 steps, improv/step: 0.187 (last = 0.1676), fitn
ess=23.836368235
1.50 secs, 5011 evals, 4892 steps, improv/step: 0.182 (last = 0.1714), fitn
ess=0.418052273
2.01 secs, 6866 evals, 6748 steps, improv/step: 0.178 (last = 0.1659), fitn
ess=0.002191742

Optimization stopped after 7001 steps and 2.0797221660614014 seconds
Termination reason: Max number of steps (7000) reached
Steps per second = 3366.315036810212
Function evals per second = 3423.0533848095843
Improvements/step = 0.17714285714285713
Total function evaluations = 7119


Best candidate found: [1.49953, 0.999482, 2.9994, 0.999937]

Fitness: 0.002191742
```

```
# 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, 1451 evals, 1309 steps, improv/step: 0.226 (last = 0.2261), fitn
ess=332.185301961
1.00 secs, 2503 evals, 2361 steps, improv/step: 0.197 (last = 0.1606), fitn
ess=30.629629319
1.50 secs, 3695 evals, 3554 steps, improv/step: 0.191 (last = 0.1802), fitn
ess=0.815254080
2.00 secs, 4855 evals, 4715 steps, improv/step: 0.188 (last = 0.1774), fitn
ess=0.124566083
2.50 secs, 6028 evals, 5888 steps, improv/step: 0.183 (last = 0.1628), fitn
ess=0.005911620

Optimization stopped after 7001 steps and 2.832087993621826 seconds
Termination reason: Max number of steps (7000) reached
Steps per second = 2472.027710921067
Function evals per second = 2521.461203212018
Improvements/step = 0.17885714285714285
Total function evaluations = 7141


Best candidate found: [1.49941, 0.999183, 3.00094, 1.00067]

Fitness: 0.001120227
```

```
# 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, 1503 evals, 1383 steps, improv/step: 0.196 (last = 0.1960), fitn
ess=143.389210244
1.00 secs, 3032 evals, 2913 steps, improv/step: 0.180 (last = 0.1647), fitn
ess=42.042827056
1.50 secs, 4596 evals, 4477 steps, improv/step: 0.174 (last = 0.1624), fitn
ess=3.947360391
2.00 secs, 6202 evals, 6084 steps, improv/step: 0.177 (last = 0.1848), fitn
ess=0.038355669

Optimization stopped after 7001 steps and 2.28381609916687 seconds
Termination reason: Max number of steps (7000) reached
Steps per second = 3065.4832508422837
Function evals per second = 3117.1511587982027
Improvements/step = 0.17885714285714285
Total function evaluations = 7119


Best candidate found: [1.49866, 0.999441, 3.00277, 1.00165]

Fitness: 0.005501799
```

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

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,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

1.221955 seconds (5.24 M allocations: 479.339 MiB, 11.75% gc time)
(368.38768828452805, [1.7284, 2.22222, 3.58025, 1.11721], :XTOL_REACHED)

opt = Opt(:GN_CRS2_LM, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

2.424252 seconds (10.60 M allocations: 970.285 MiB, 11.29% gc time)
(1.6661757787670773e-16, [1.5, 1.0, 3.0, 1.0], :XTOL_REACHED)

opt = Opt(:GN_ISRES, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

3.105609 seconds (13.41 M allocations: 1.199 GiB, 11.04% gc time)
(27.222074640940818, [1.43438, 0.887965, 3.1579, 1.09803], :MAXEVAL_REACHED
)

opt = Opt(:GN_ESCH, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj_short.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)

3.063588 seconds (13.41 M allocations: 1.199 GiB, 11.30% gc time)
(72.51887160056941, [1.25791, 0.957151, 4.81087, 1.58266], :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,[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,loc_init)
```

```
0.103867 seconds (427.79 k allocations: 39.158 MiB, 12.79% gc time)
(1.6661750310234297e-16, [1.5, 1.0, 3.0, 1.0], :SUCCESS)
```

```
opt = Opt(:LN_NELDERMEAD, 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,loc_init)
```

```
0.161857 seconds (671.86 k allocations: 61.498 MiB, 12.43% gc time)
(1.6661765163255584e-16, [1.5, 1.0, 3.0, 1.0], :XTOL_REACHED)
```

```
opt = Opt(:LD_SLSQP, 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,loc_init)
```

```
0.128522 seconds (463.94 k allocations: 38.670 MiB, 14.51% gc time)
(4.1924574575045983e-16, [1.5, 1.0, 3.0, 1.0], :XTOL_REACHED)
```

```
opt = Opt(:LN_COBYLA, 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,loc_init)
```

```
3.092641 seconds (13.41 M allocations: 1.199 GiB, 11.65% gc time)
(4.3894137711062486e-10, [1.5, 1.0, 3.0, 1.0], :MAXEVAL_REACHED)
```

```
opt = Opt(:LN_NEWUOA_BOUND, 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,loc_init)
```

```
0.226251 seconds (382.20 k allocations: 34.984 MiB, 5.75% gc time)
(1.6746576191653103e-9, [1.5, 1.0, 3.0, 0.999998], :SUCCESS)
```

```
opt = Opt(:LN_PRAXIS, 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,loc_init)
```

```
0.065307 seconds (286.99 k allocations: 26.269 MiB, 9.74% gc time)
(1.6667640468718725e-16, [1.5, 1.0, 3.0, 1.0], :SUCCESS)
```

```
opt = Opt(:LN_SBPLX, 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,loc_init)
```

```
3.067597 seconds (13.41 M allocations: 1.199 GiB, 11.43% gc time)
(3.857624965591935e-12, [1.5, 1.0, 3.0, 1.0], :MAXEVAL_REACHED)
```

```
opt = Opt(:LD_MMA, 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,loc_init)
```

```
5.920582 seconds (26.04 M allocations: 2.332 GiB, 11.46% gc time)
(4.632433645015572e-15, [1.5, 1.0, 3.0, 1.0], :XTOL_REACHED)
```

```
opt = Opt(:LD_TNEWTON_PRECOND_RESTART, 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,loc_init)
```

```
0.110489 seconds (469.54 k allocations: 43.051 MiB, 11.52% gc time)
(4.192581262191412e-16, [1.5, 1.0, 3.0, 1.0], :SUCCESS)
```

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, 162 evals, 96 steps, improv/step: 0.354 (last = 0.3542), fitness
=24728.216300682
1.01 secs, 327 evals, 222 steps, improv/step: 0.365 (last = 0.3730), fitnes
s=24550.304445423
1.51 secs, 489 evals, 375 steps, improv/step: 0.323 (last = 0.2614), fitnes
s=24205.454165712
2.01 secs, 652 evals, 535 steps, improv/step: 0.305 (last = 0.2625), fitnes
```

```
s=17668.498240448
2.51 secs, 818 evals, 701 steps, improv/step: 0.282 (last = 0.2108), fitnes
s=17668.498240448
3.01 secs, 983 evals, 866 steps, improv/step: 0.266 (last = 0.1939), fitnes
s=17668.498240448
3.51 secs, 1145 evals, 1028 steps, improv/step: 0.250 (last = 0.1667), fitn
ess=17668.498240448
4.02 secs, 1311 evals, 1194 steps, improv/step: 0.229 (last = 0.1024), fitn
ess=17668.498240448
4.52 secs, 1475 evals, 1358 steps, improv/step: 0.225 (last = 0.1951), fitn
ess=17668.498240448
5.02 secs, 1636 evals, 1519 steps, improv/step: 0.210 (last = 0.0807), fitn
ess=14124.375288830
5.52 secs, 1799 evals, 1682 steps, improv/step: 0.197 (last = 0.0798), fitn
ess=14124.375288830
6.03 secs, 1956 evals, 1839 steps, improv/step: 0.188 (last = 0.0892), fitn
ess=14124.375288830
6.53 secs, 2121 evals, 2004 steps, improv/step: 0.177 (last = 0.0545), fitn
ess=14124.375288830
7.03 secs, 2283 evals, 2166 steps, improv/step: 0.168 (last = 0.0556), fitn
ess=14124.375288830
7.53 secs, 2446 evals, 2329 steps, improv/step: 0.161 (last = 0.0736), fitn
ess=14124.375288830
8.03 secs, 2613 evals, 2496 steps, improv/step: 0.157 (last = 0.1018), fitn
ess=14124.375288830
8.53 secs, 2778 evals, 2661 steps, improv/step: 0.152 (last = 0.0727), fitn
ess=14124.375288830
9.04 secs, 2943 evals, 2826 steps, improv/step: 0.150 (last = 0.1212), fitn
ess=10278.389984085
9.54 secs, 3107 evals, 2990 steps, improv/step: 0.149 (last = 0.1341), fitn
ess=10278.389984085
10.04 secs, 3274 evals, 3157 steps, improv/step: 0.150 (last = 0.1497), fit
ness=9132.930254673
10.54 secs, 3439 evals, 3322 steps, improv/step: 0.149 (last = 0.1333), fit
ness=9132.930254673
11.05 secs, 3605 evals, 3488 steps, improv/step: 0.147 (last = 0.1024), fit
ness=9132.930254673
11.55 secs, 3770 evals, 3653 steps, improv/step: 0.144 (last = 0.0970), fit
ness=9132.930254673
12.05 secs, 3936 evals, 3819 steps, improv/step: 0.143 (last = 0.1265), fit
ness=8061.887455267
12.55 secs, 4103 evals, 3986 steps, improv/step: 0.143 (last = 0.1377), fit
ness=4443.356625241

Optimization stopped after 4001 steps and 12.6008939743042 seconds
Termination reason: Max number of steps (4000) reached
Steps per second = 317.5171545890996
Function evals per second = 326.8022100969538
Improvements/step = 0.143
Total function evaluations = 4118


Best candidate found: [1.34452, 1.60542, 3.41174, 1.23799]

Fitness: 4443.356625241
```

```
opt = Opt(:GN_ORIG_DIRECT_L, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj.cost_function2)
```

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

```
6.356342 seconds (25.43 M allocations: 2.094 GiB, 9.67% gc time)
(23525.885834893048, [8.2716, 7.42112, 7.40283, 3.7037], :XTOL_REACHED)
```

```
opt = Opt(:GN_CRS2_LM, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 20000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
39.138491 seconds (157.81 M allocations: 12.996 GiB, 9.84% gc time)
(1.2705814379989609e-14, [1.5, 1.0, 3.0, 1.0], :XTOL_REACHED)
```

```
opt = Opt(:GN_ISRES, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 50000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
148.948490 seconds (607.55 M allocations: 50.034 GiB, 9.85% gc time)
(13788.53018501453, [1.99399, 3.86693, 2.42806, 0.751493], :MAXEVAL_REACHED
)
```

```
opt = Opt(:GN_ESCH, 4)
lower_bounds!(opt,[0.0,0.0,0.0,0.0])
upper_bounds!(opt,[10.0,10.0,10.0,10.0])
min_objective!(opt, obj.cost_function2)
xtol_rel!(opt,1e-12)
maxeval!(opt, 20000)
@time (minf,minx,ret) = NLopt.optimize(opt,glo_init)
```

```
58.300421 seconds (243.02 M allocations: 20.014 GiB, 10.00% gc time)
(11810.271274396095, [0.611902, 0.509101, 8.70263, 4.73246], :MAXEVAL_REACH
ED)
```

```
opt = Opt(:LN_BOBYQA, 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,loc_init)
```

```
1.510405 seconds (6.49 M allocations: 547.189 MiB, 10.25% gc time)
(1.2705773526988974e-14, [1.5, 1.0, 3.0, 1.0], :SUCCESS)
```

```
opt = Opt(:LN_NELDERMEAD, 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-9)
maxeval!(opt, 10000)
@time (minf,minx,ret) = NLopt.optimize(opt,loc_init)
```

```
1.328376 seconds (5.78 M allocations: 487.756 MiB, 10.02% gc time)
(1.7646805842304758e-14, [1.5, 1.0, 3.0, 1.0], :XTOL_REACHED)
```

```
opt = Opt(:LD_SLSQP, 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,loc_init)
```

```
2.872635 seconds (8.37 M allocations: 703.304 MiB, 6.91% gc time)
(21569.713608934053, [3.25971, 2.72384, 0.85817, 0.404497], :XTOL_REACHED)
```

```
obj_short =
    build_loss_objective(prob_short,Tsit5(),L2Loss(t_short,data_short),tstops=t_short)
lower = [0.0,0.0,0.0,0.0]
upper = [5.0,5.0,5.0,5.0]
splits = ([0.0,1.0,3.0],[0.0,1.0,3.0],[0.0,1.0,3.0],[0.0,1.0,3.0])
root, x0 = analyze(obj_short,splits,lower,upper)
```

```
minimum(root)
```

```
Box1.12840013604021850@[1.49017, 0.994299, 2.99232, 1.0]
```

```
obj = build_loss_objective(prob,Vern9(),L2Loss(t,data),tstops=t,reltol=1e-9,abstol=1e-9)
lower = [0.0,0.0,0.0,0.0]
upper = [10.0,10.0,10.0,10.0]
splits = ([0.0,3.0,6.0],[0.0,3.0,6.0],[0.0,3.0,6.0],[0.0,3.0,6.0])
root, x0 = analyze(obj,splits,lower,upper)
```

```
minimum(root)
```

```
Box23222.23744589307@[2.84639, 3.0, 5.78411, 3.0]
```

Parameter estimation on the longer sample proves to be extremely challenging for some of the global optimizers. A few give the accurate values, BlacBoxOptim also performs quite well while others seem to struggle with accuracy a lot.

3 Conclusion

In general we observe 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 `LD_SLSQP` `GN_ESCH` `GN_ISRES` `GN_ORIG_DIRECT_L` performing among the worse, BBO also gives a bit high fitness in comparison. QuadDIRECT gives accurate results in the case of the shorter problem but doesn't perform very well in the longer problem case.