Table of Contents

Parameter optimisation

```
lorenz! (generic function with 1 method)
```

```julia
begin
    using DifferentialEquations, Flux, DiffEqFlux, Plots

    function lorenz!(du, u, p, t)
      x, y, z = u
      α, β, ρ = p
      du[1] = dx = α*y - α*x
      du[2] = dy = x*(ρ-z) -y
      du[3] = dz = x*y - β*z
    end
end
```

```
u0 =  Float64[1.0,  1.0,  1.0]
```

```julia
u0 = [1.0, 1.0, 1.0]
```

```
0.0:0.1:10.0
```

```julia
begin
    tspan = (0.0, 10.0)
    tsteps = 0.0:0.1:10.0
end
```

```
p =  Float64[10.0,  2.66667,  28.0]
```

```julia
p = [10, 8/3, 28]
```

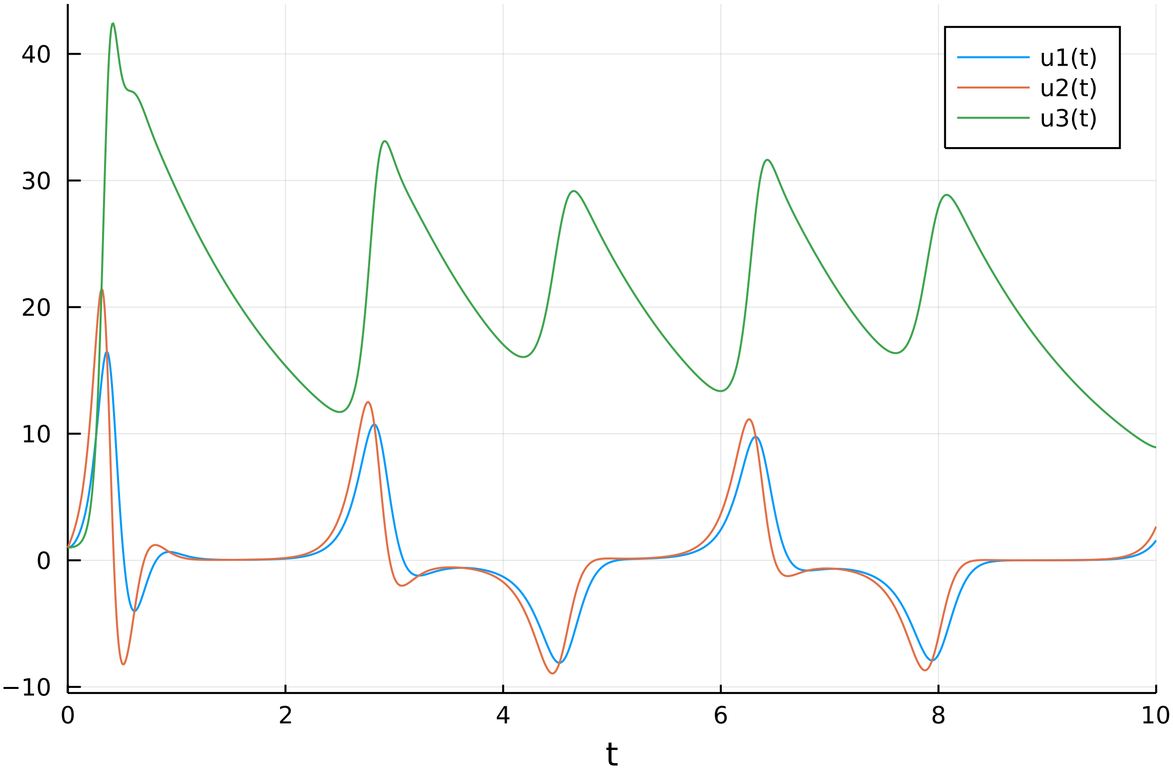
	timestamp	value1	value2	value3
1	0.0	1.0	1.0	1.0
2	0.0372378	1.148	1.83331	1.03124
3	0.0618418	1.3902	2.48107	1.08143
4	0.103775	2.07305	3.95208	1.27958

	timestamp	value1	value2	value3
5	0.146359	3.20537	6.18654	1.79939
6	0.19905	5.49456	10.4709	3.587
7	0.251097	9.06276	16.4154	8.46477
8	0.306398	13.8872	21.3667	20.3403
9	0.363248	16.46	15.5197	36.7049

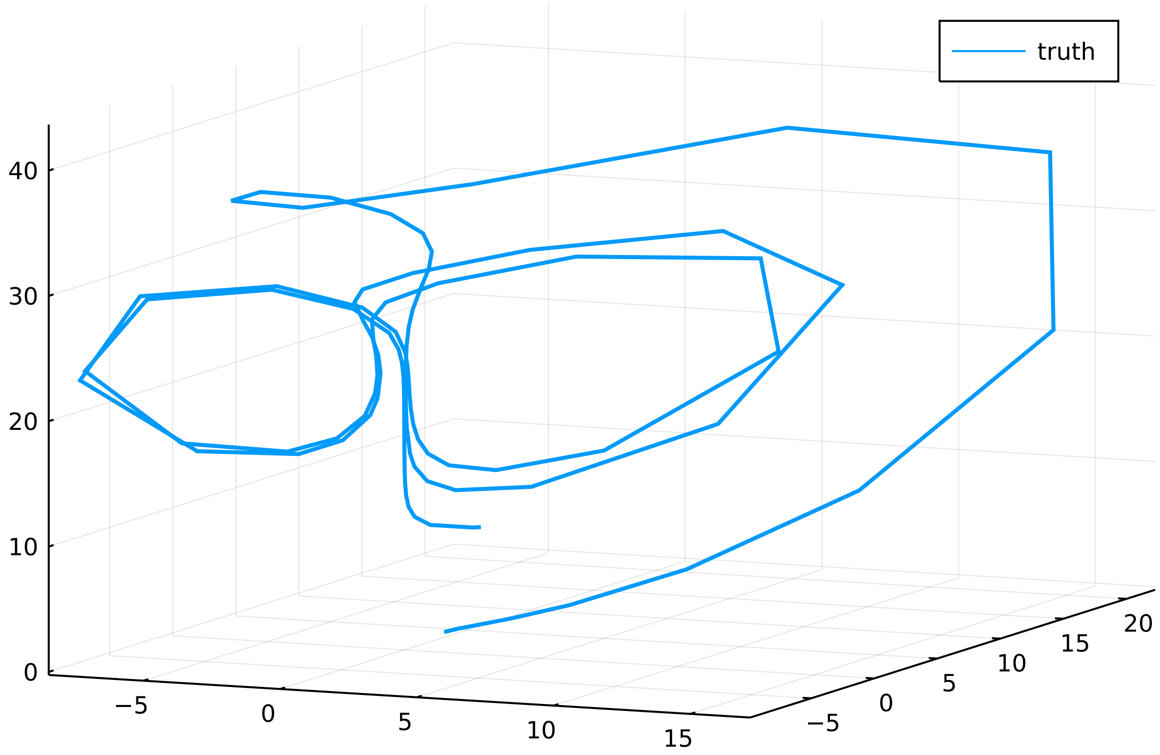
```
begin
    prob = ODEProblem(lorenz!, u0, tspan, p)
    sol = solve(prob, Tsit5())
end
```

```
Float64[1.0,  1.03124,  1.08143,  1.27958,  1.79939,  3.587,  8.46477,  20.3403,  36.7049,
```

```
sol[3,:]
```



```
begin
    plot(sol)
end
```



```
plot(sol[1,:], sol[2,:], sol[3,:], linewidth = 2, label = "truth", legend = true)
```

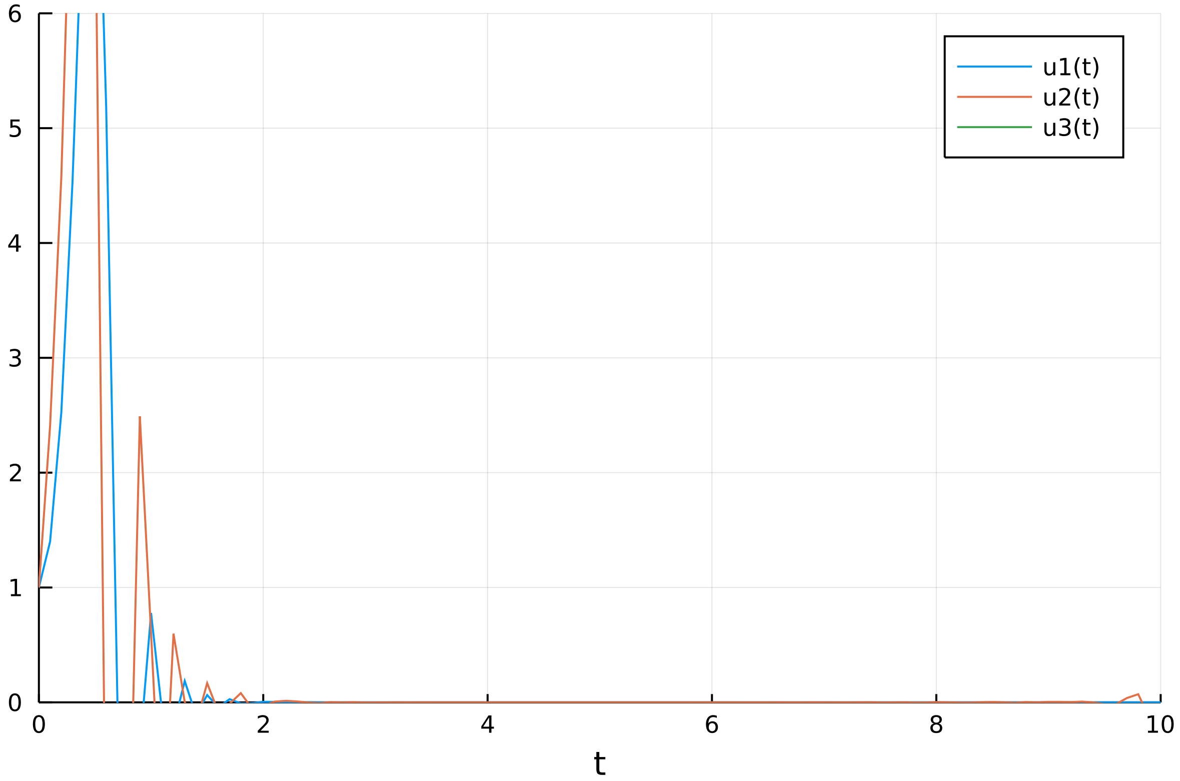
```
Enter cell code...
```

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

```
begin
    params = Flux.params(p)

function predict_rd() # Our 1-layer "neural network"
  solve(prob,Tsit5(),p=p,saveat=0.1)[1,:] # override with new parameters
end

loss_rd() = sum(abs2,x-1 for x in predict_rd()) # loss function

end
```

InterruptException:

```
begin
    data = Iterators.repeated((), 100)
    opt = ADAM(0.1)

    Flux.train!(loss_rd, params, data, opt)
end
```

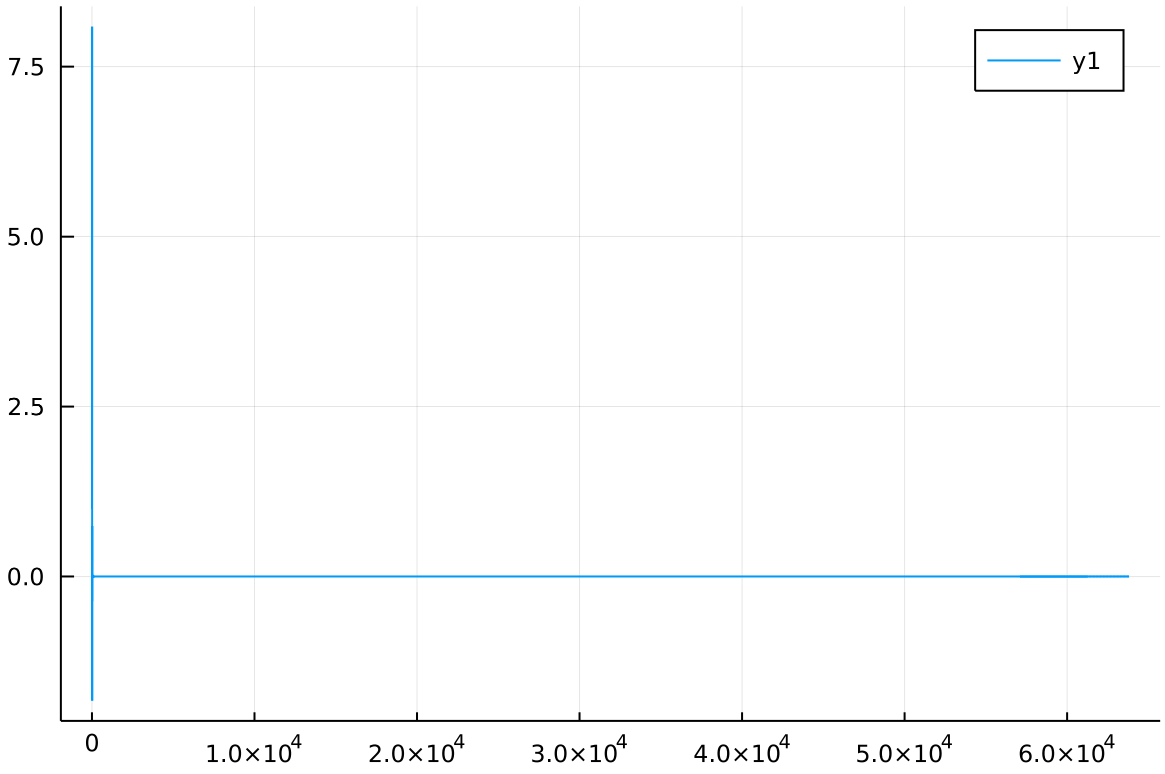


```
plot(solve(remake(prob,p=p),Tsit5(),saveat=0.1),ylim=(0,6))
```

	timestamp	value1	value2	value3
54	2.0775	0.000896601	0.0341199	411.713
55	2.09501	0.00398003	0.0146233	424.665
56	2.11208	0.00346929	-0.0140014	437.695
57	2.12754	0.000618749	-0.0278579	449.826

	timestamp	value1	value2	value3
59	2.16052	-0.00315801	0.00308916	476.804
60	2.17659	-0.00129291	0.0208206	490.618
61	2.1922	0.00129083	0.0201954	504.36
62	2.2075	0.00253962	0.00417206	518.192
63	2.22342	0.00161113	-0.0140126	533.002
64	2.23784	-0.000356968	-0.0186565	546.78
65	2.25336	-0.00188756	-0.00800282	562.003
66	2.26851	-0.00164618	0.00810934	577.267
67	2.28207	-0.000248799	0.0156317	591.285
68	2.29716	0.00137793	0.0103742	607.283

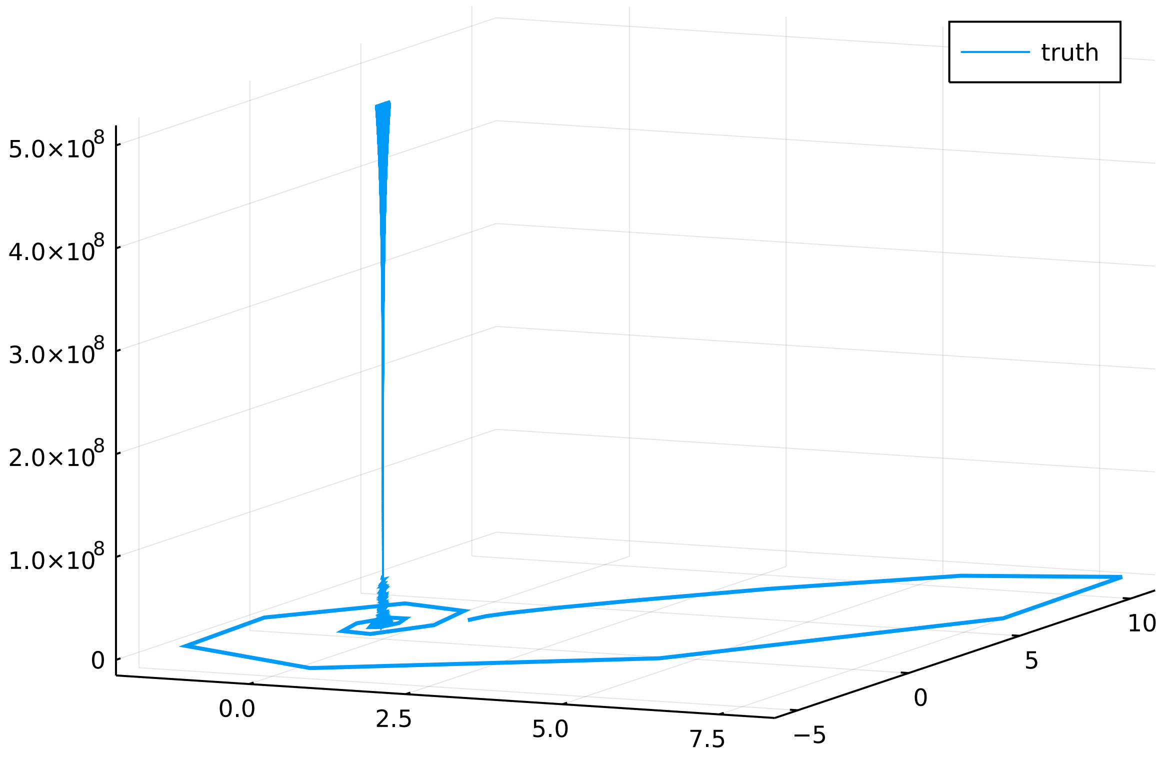
```
new_sol = solve(prob,Tsit5())
```



```
plot(new_sol[1,:])
```

```
Float64[7.48035,  -1.76945,  15.0526]
```

```
p
```



```
plot(new_sol[1,:], new_sol[2,:], new_sol[3,:], linewidth = 2, label = "truth", legend
= true)
```