Authors

Author	Matrikelnummer
Paul Römer	7377945
Philip Julius Pupkes	7360318
Alice Coors	7392745
Muhammad Fakhar	7432447


```
load(; N=100, η=3.0, cfl=0.5) =  DataFrame(CSV.File("results/
results_$(N)_$(η)_$(cfl).csv"));
```

```
data = load(; cfl=0.5);
```

Simulation state at different time steps

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

```
function closest_timestep(t; data=data)
    i = findmin(eachrow(data)) do row
        abs(row.time - t)
    end[2]
    data.time_step[i]
end
```

```
snapshot(time_step; data=data) = filter(data) do row
    row.time_step == time_step
end;
```

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

```
function plot_density(time_step; data=data, kwargs...)
    s = snapshot(time_step; data);
    scatter(s.position, s.density; title="t≈$(round(s.time[1], digits=1))",
    xlabel="position", ylabel="density", label=false, ms=2, kwargs...)
end
```

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

```
function plot_velocity(time_step; data=data, kwargs...)
    s = snapshot(time_step; data);
    a, b = linear_fit(s.position, s.velocity)
    scatter(s.position, s.velocity; title="t≈$(round(s.time[1], digits=1))",
    xlabel="position", ylabel="velocity", label=false, ms=2, kwargs...)
    plot!(x -> a + b * x; lc=:black, ls=:dash, label="linear fit")
end
```

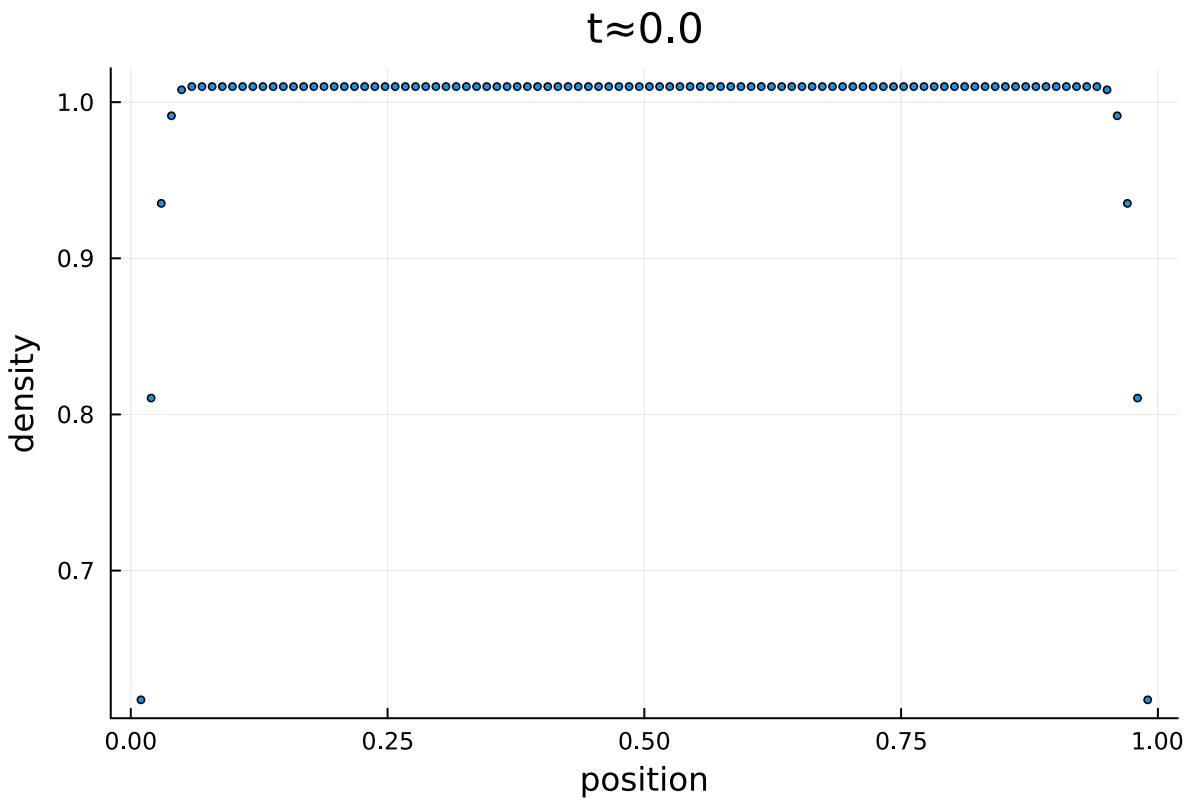
```
t_max = 1.0
```

```
t_max = round(maximum(data.time), digits=1)
```

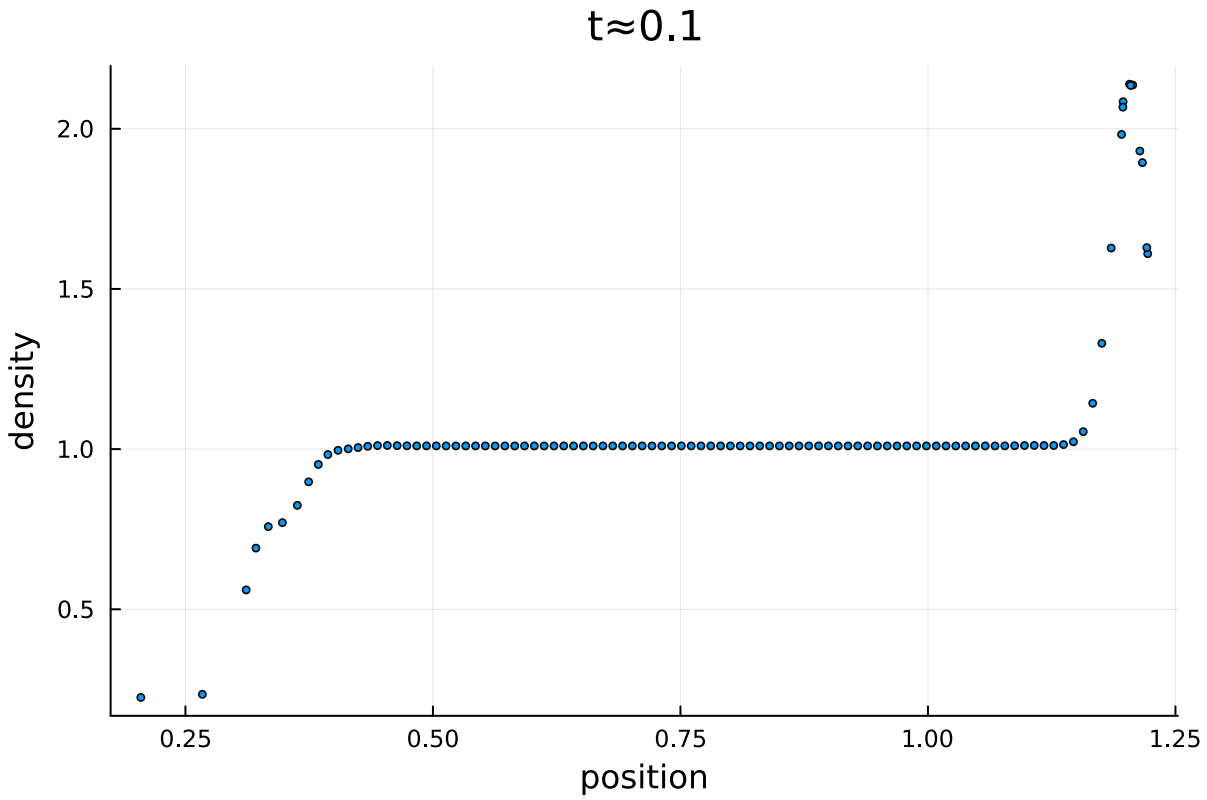
```
time_steps =  [0, 33, 107, 195, 295, 386, 465, 542, 615, 674, 726]
```

```
time_steps = [closest_timestep(t) for t in 0.0:0.1:t_max]
```

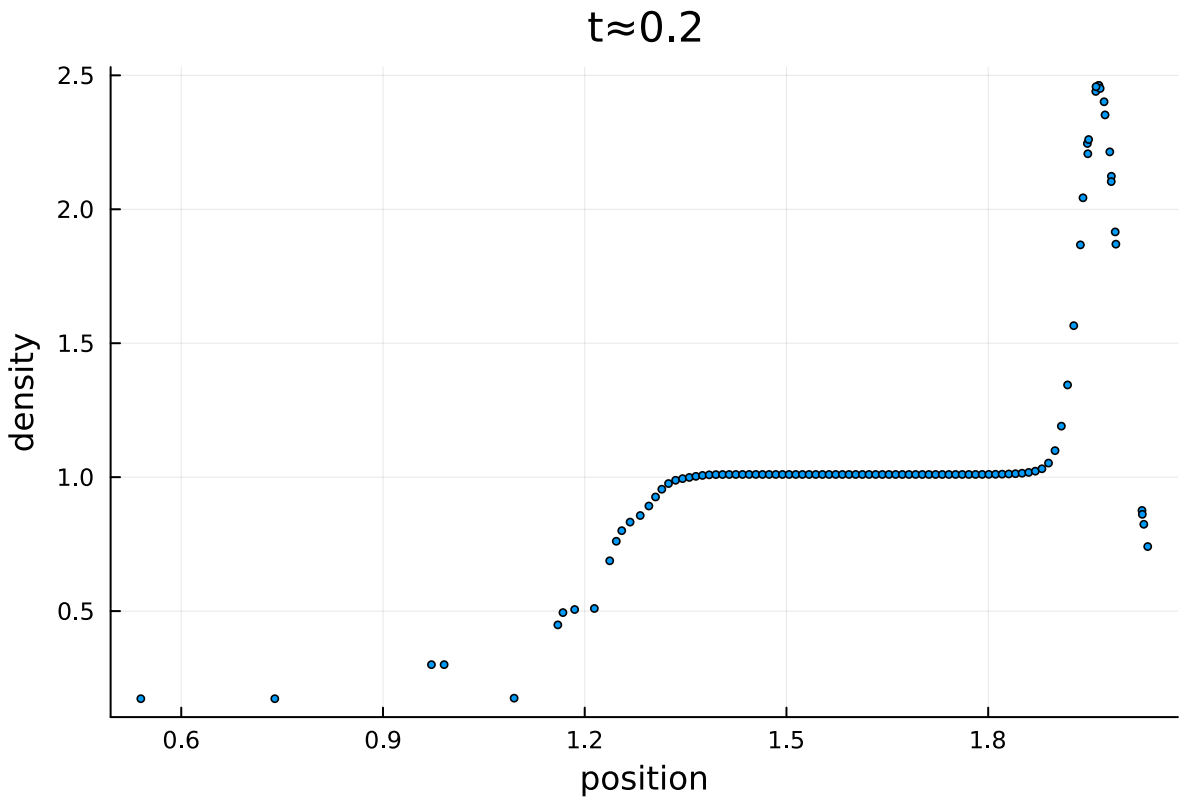
Density



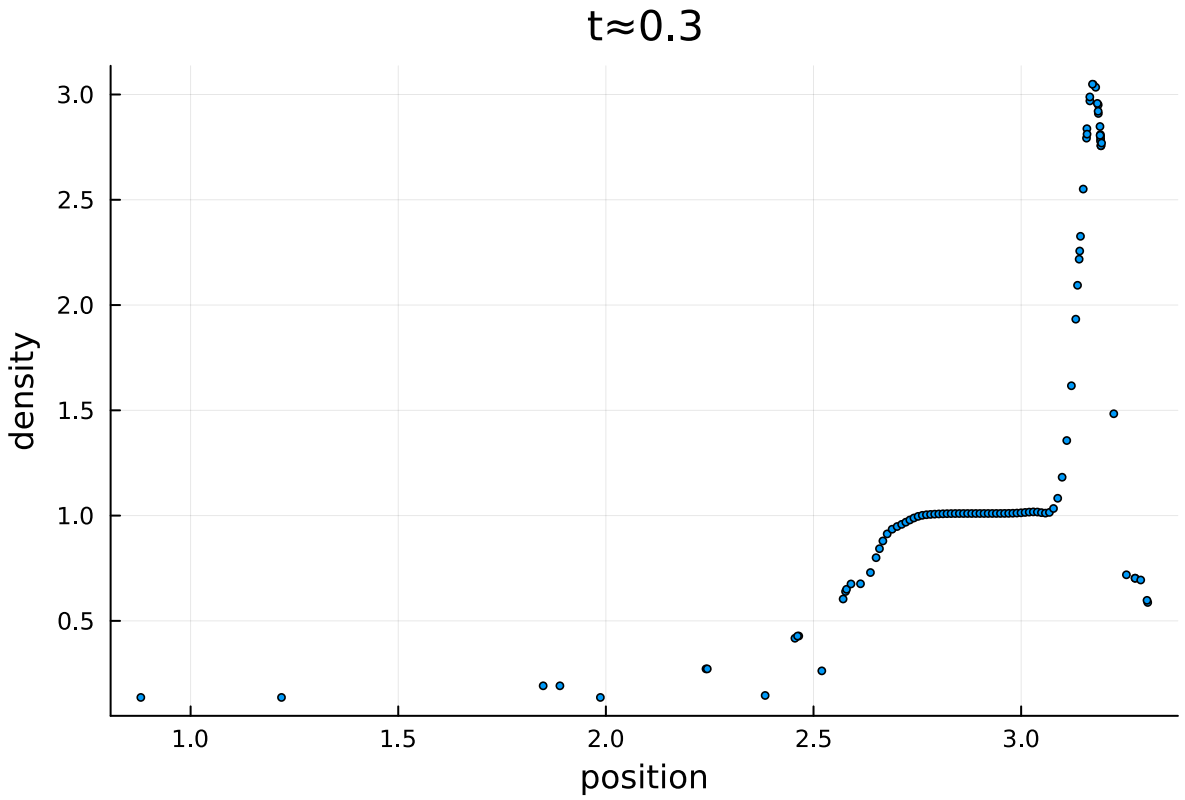

```
plot_density(time_steps[1])
```




```
1 plot_density(time_steps[2])
```

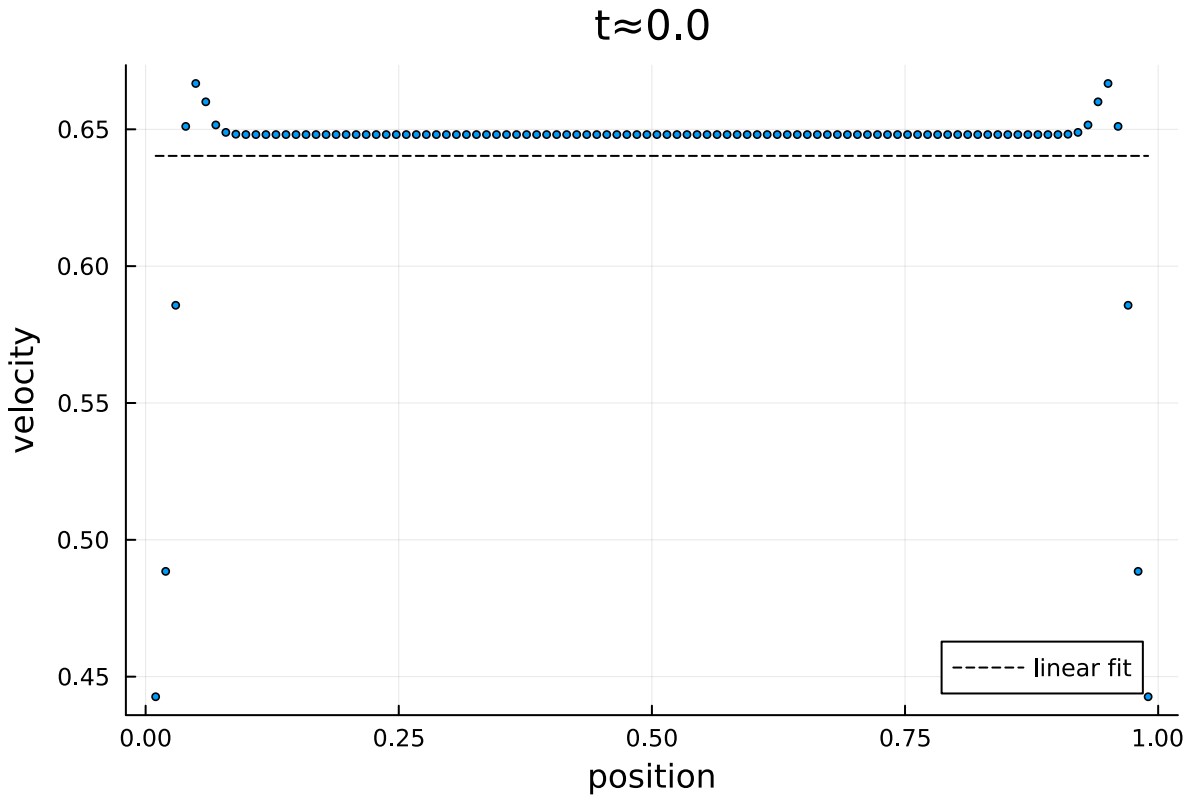



```
1 plot_density(time_steps[3])
```

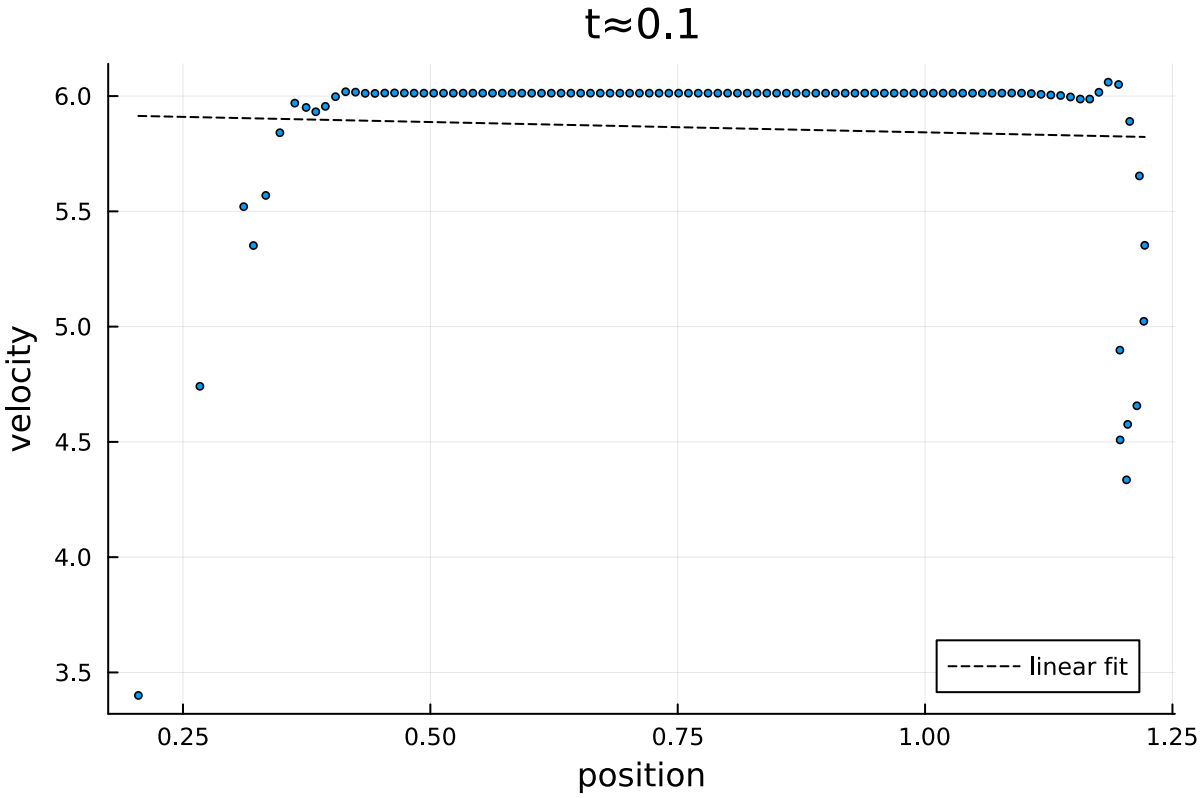



```
1 plot_density(time_steps[4])
```

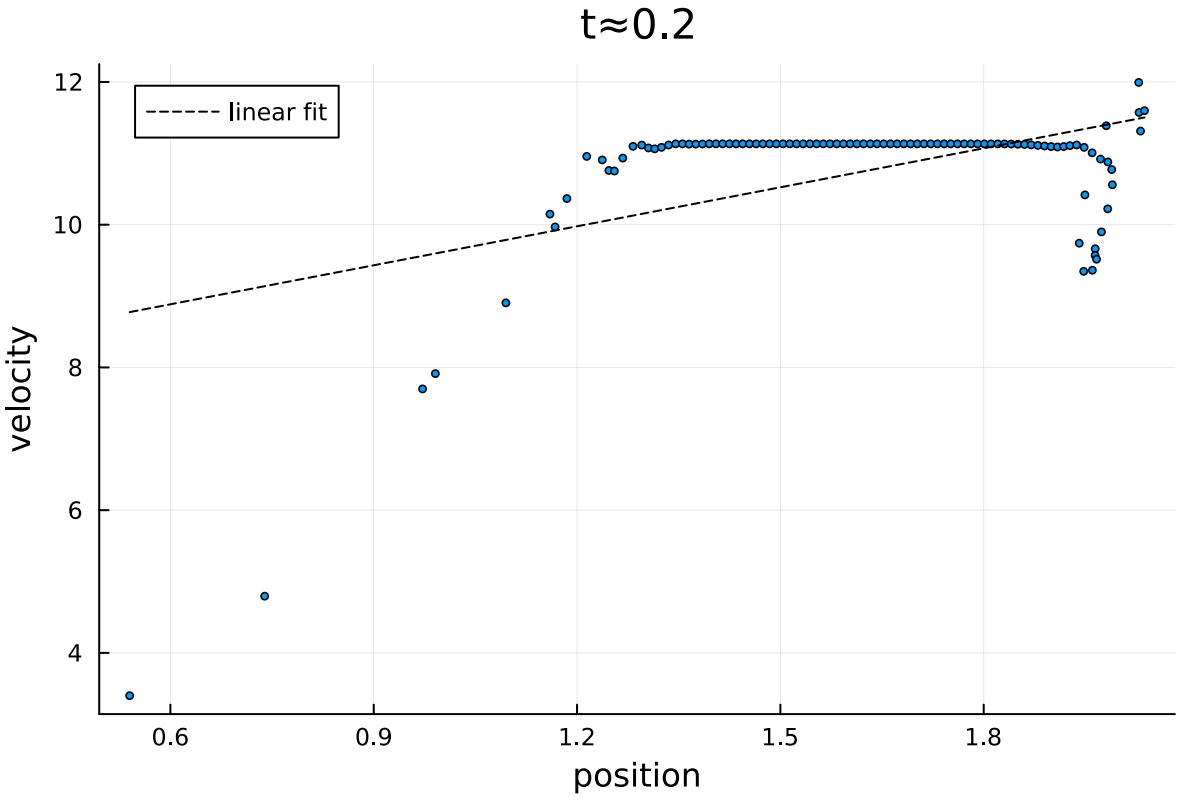
Velocity



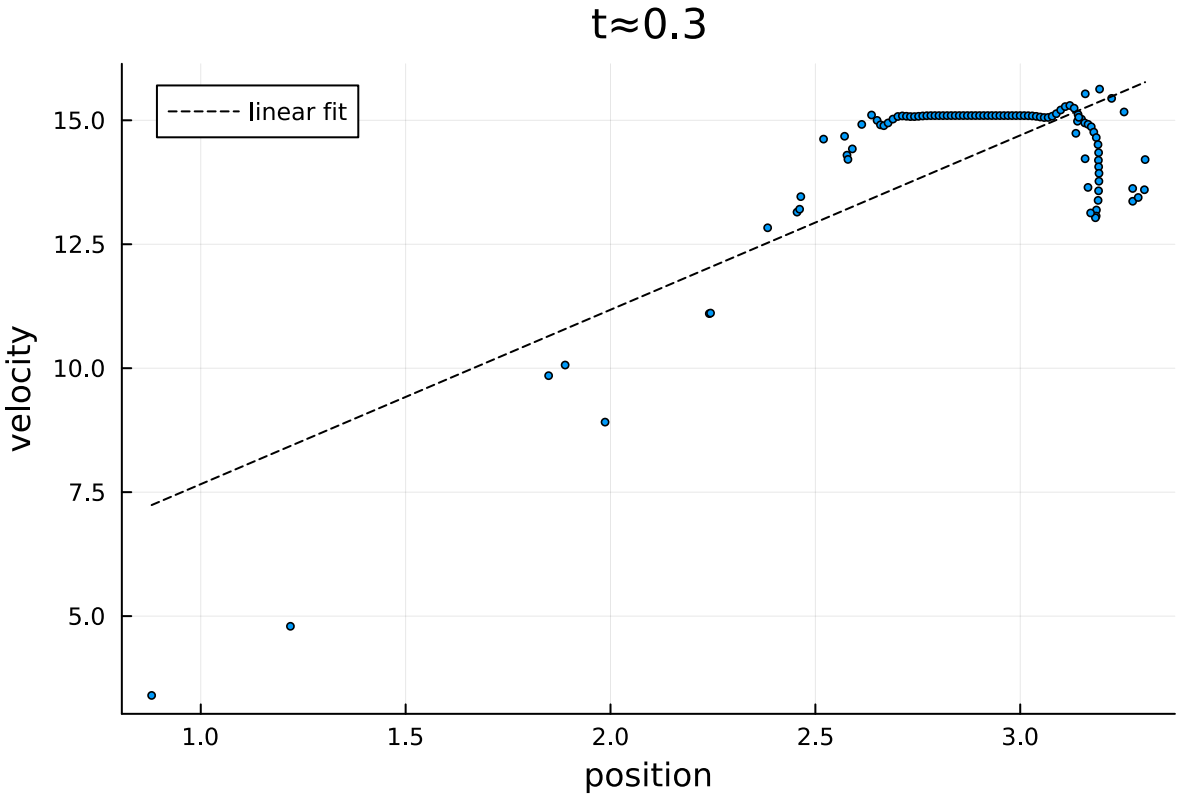

```
1 plot_velocity(time_steps[1])
```



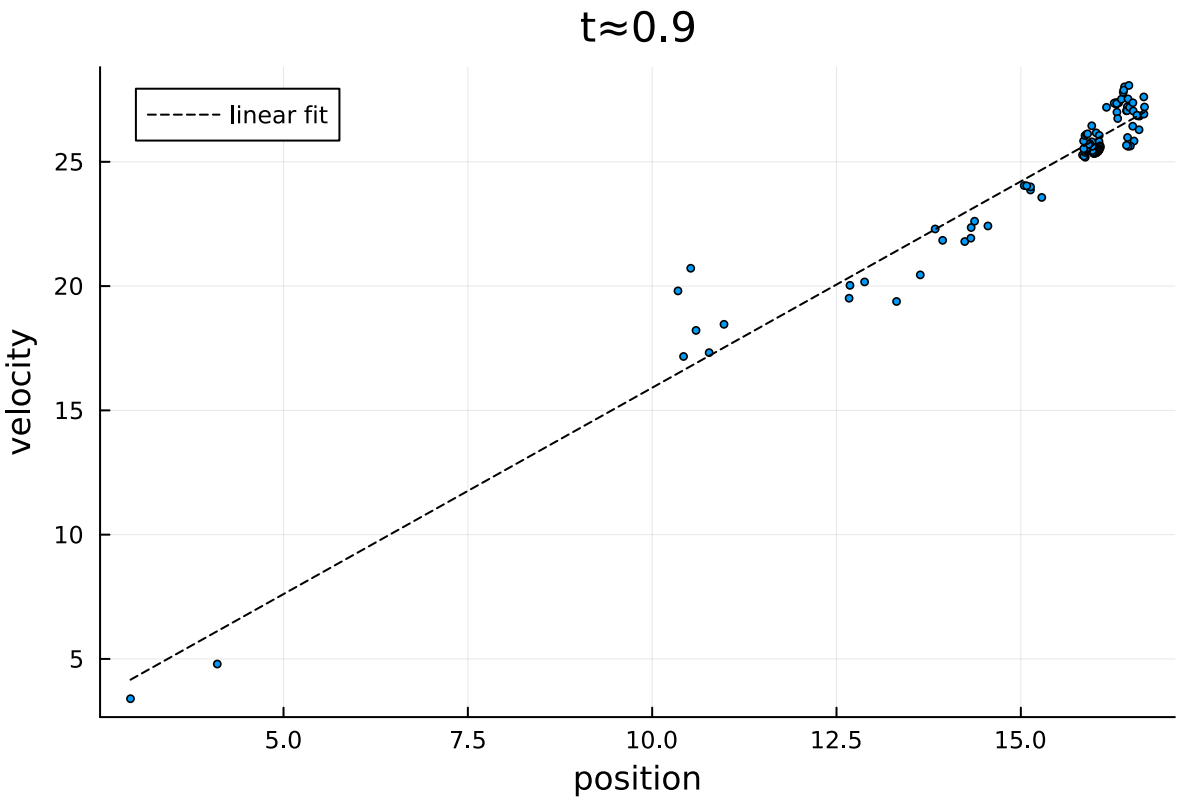

```
1 plot_velocity(time_steps[2])
```




```
1 plot_velocity(time_steps[3])
```




```
1 plot_velocity(time_steps[4])
```

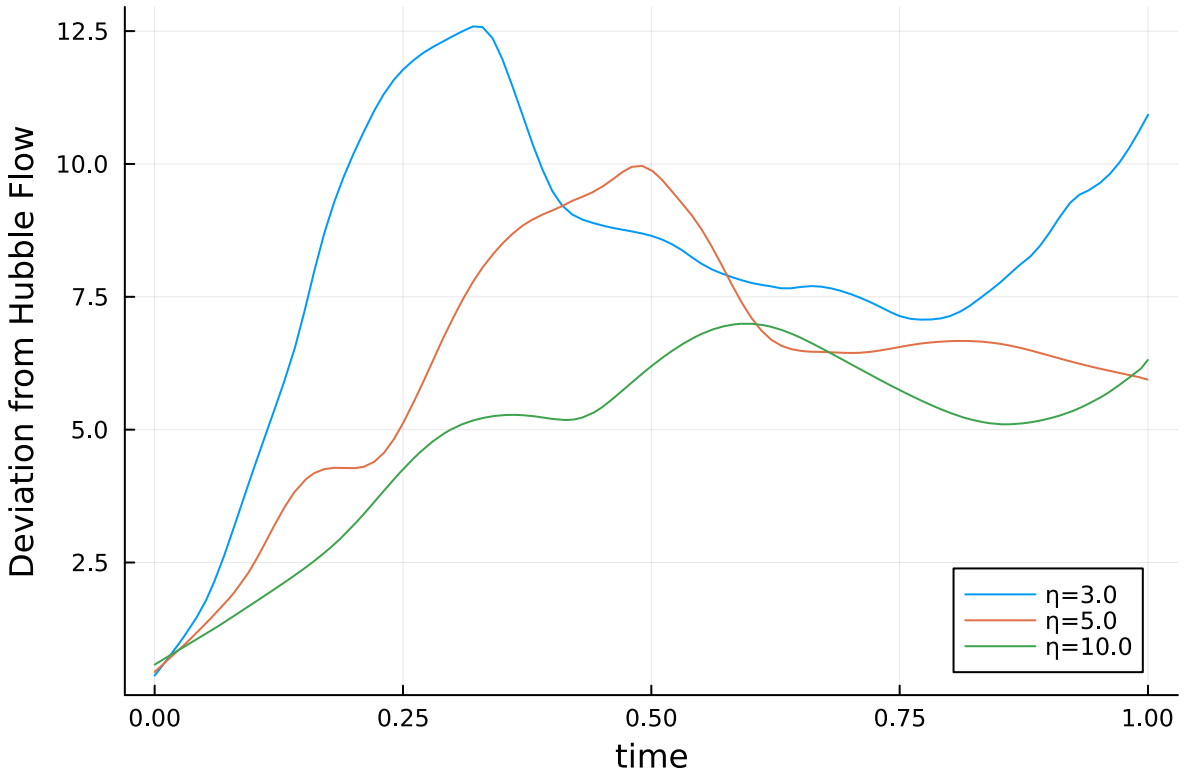



```
1 plot_velocity(time_steps[10])
```

Deviation from Hubble

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

```
function velocity_deviation_from_linear(time_step; data=data, kwargs...)
    s = snapshot(time_step; data);
    a, b = linear_fit(s.position, s.velocity)

    sqrt(sum(((a + b * s.position[i]) - s.velocity[i])^2 for i in
    1:length(s.position)))
end
```



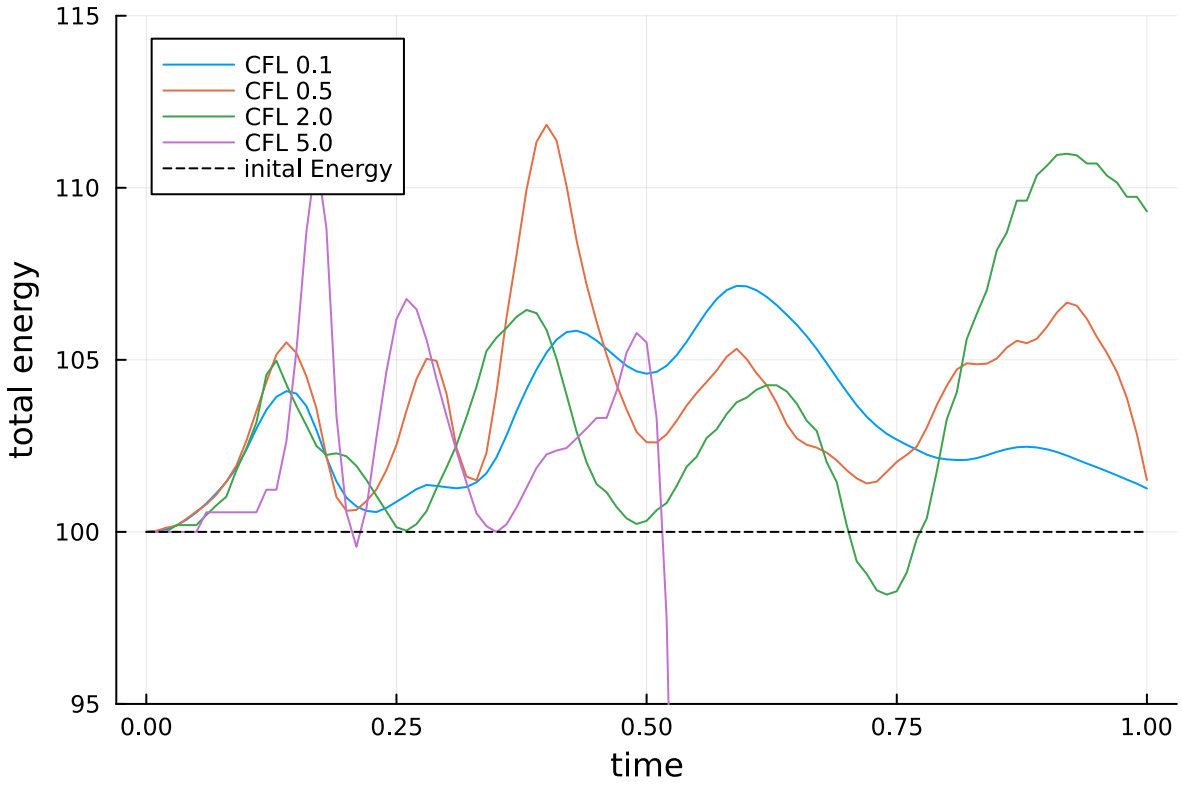

```
let
    plot(; xlabel="time", ylabel="Deviation from Hubble Flow")
    for η in [3.0, 5.0, 10.0]
        data = load(;η=η)
        time_steps = [closest_timestep(t; data) for t in 0.0:0.01:t_max]
        times = map(rows->rows.time[1], snapshot.(time_steps; data))

        Δ = [velocity_deviation_from_linear(ts; data=data) for ts in time_steps]
        plot!(times, Δ; label="η=$η")
    end
    plot!()
end
```

Increasing η, decreases deviations from a Hubble-like Flow

Conservation of Energy

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

```
1 total_energy(time_stamp; data=data) = sum(snapshot(time_stamp; data=data).energy)
```




```
1  let
2      u_0 = total_energy(closest_timestep(0.0))
3      plot(; xlabel="time", ylabel="total energy", ylims=(95, 115))
4      for cfl in [0.1, 0.5, 2.0, 5.0]
5          data = load(; cfl);
6          t_max = round(maximum(data.time), digits=1)
7          ts = 0.0:0.01:t_max
8
9          plot!(ts, [total_energy(closest_timestep(t; data); data) for t in ts];
           label="CFL $(cfl)")
10     end
11     plot!(x -> u_0; lc=:black, ls=:dash, label="inital Energy")
12 end
```

Reducing γ_{cfl} slightly reduces the amplitude of the energy fluctuations. When γ_{cfl} is set to 5, the simulation diverges. It is possible that the internal Energy is not conserved because $\frac{du}{dt}$ was not chosen symetrically.