JuliaHub

Prototyping Composable Simplification Passes for Equation-Oriented Models Using ModelingToolkit.jl

Yingbo Ma, Keno Fischer, Chris Rackauckas, and Viral Shah

Core Point: ModelingToolkit makes it easy to design new and better simplification algorithms interactively

ModelingToolkit

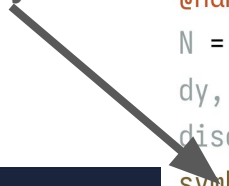
```
function EMF(; name, k)
    @named p = Pin()
    @named n = Pin()
    @named flange = Flange()
    @named support = Support()
    @parameters k = k
    @variables v(t)=0.0 i(t)=0.0 phi(t)=0.0 w(t)=0.0
    eqs = [v ~ p.v - n.v
           0 ~ p.i + n.i
           i ~ p.i
           phi ~ flange.phi - support.phi
           D(phi) ~ w
           k * w ~ v
           flange.tau ~ -k * i]
    ODESystem(eqs, t, [v, i, phi, w], [k]; name = name, systems = [p, n, flan
end
```

```
1  @named ground = Ground()
2  @named source = Voltage()
3  @named ref = Blocks.Step(height = 1, start_time = 0)
4  @named pi_controller = Blocks.LimPI(k = 1.1, T = 0.035, u_max = 0.6, Ta = 0.035)
5  @named feedback = Blocks.Feedback()
6  @named R1 = Resistor(R = R)
7  @named L1 = Inductor(L = L)
8  @named emf = EMF(k = k)
9  @named fixed = Fixed()
10 @named load = Torque(use_support = false)
11 @named load_step = Blocks.Step(height = tau_L_step, start_time = 3)
12 @named inertia = Inertia(J = J)
13 @named friction = Damper(d = f)
14 @named speed_sensor = SpeedSensor()
15
16 connections = [connect(fixed.flange, emf.support, friction.flange_b)
17                 connect(emf.flange, friction.flange_a, inertia.flange_a)
18                 connect(inertia.flange_b, load.flange)
19                 connect(inertia.flange_b, speed_sensor.flange)
20                 connect(load_step.output, load.tau)
21                 connect(ref.output, feedback.input1)
22                 connect(speed_sensor.w, feedback.input2)
23                 connect(feedback.output, pi_controller.err_input)
24                 connect(pi_controller.ctr_output, source.V)
25                 connect(source.p, R1.p)
26                 connect(R1.n, L1.p)
27                 connect(L1.n, emf.p)
28                 connect(emf.n, source.n, ground.g)]
29
30 @named model = ODESystem(connections, t,
```

Composability of MTK

XSystem ↦ YSystem

PDESystem ↦ ODESystem



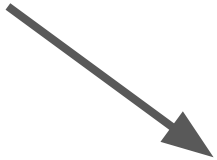
```
eq = [Dt(u(x,y,t)) ~ 1. + v(x,y,t)*u(x,y,t)^2 - 4.4*u(x,y,t) + α*∇²(u(x,y,t)) +
      brusselator_f(x, y, t),
      Dt(v(x,y,t)) ~ 3.4*u(x,y,t) - v(x,y,t)*u(x,y,t)^2 + α*∇²(v(x,y,t))]
domains = [x ∈ Interval(x_min, x_max),
           y ∈ Interval(y_min, y_max),
           t ∈ Interval(t_min, t_max)]
# Periodic BCs
bcs = [u(x,y,0) ~ u0(x,y,0),
      u(0,y,t) ~ u(1,y,t),
      u(x,0,t) ~ u(x,1,t),
      v(x,y,0) ~ v0(x,y,0),
      v(0,y,t) ~ v(1,y,t),
      v(x,0,t) ~ v(x,1,t)]
@named pdesys = PDESystem(eq,bcs,domains,[x,y,t],[u(x,y,t),v(x,y,t)])
N = 32
dy, dx = (y_max-y_min)/N, (x_max-x_min)/N
discretization = MOLFiniteDifference([x⇒dx, y⇒dy], t)
symbolic_discretize(sys, discretization)
```

Composability of MTK

XSystem ↦ YSystem

MomentClosure.jl

ReactionNetwork ↦ ODESystem



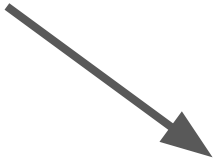
```
using Catalyst, MomentClosure
rn = @reaction_network begin
  # including system-size parameter Ω
  (c₁/Ω^2), 2X + Y → 3X
  (c₂), X → Y
  (c₃*Ω, c₄), 0 ↔ X
end c₁ c₂ c₃ c₄ Ω
generate_raw_moment_eqs(rn, 2, combinatoric_ratelaw=false)
```

Composability of MTK

XSystem ↦ YSystem

MomentClosure.jl

SDESystem ↦ ODESystem



```
using ModelingToolkit, MomentClosure
@variables t, x₁(t), x₂(t)
@parameters ϵ, ω_n, ω_g, A
drift_eqs = [Differential(t)(x₁) ~ x₂;
             Differential(t)(x₂) ~ ϵ*(1-x₁^2)*x₂ -
                                   ω_n^2*x₁ + A*cos(ω_g*t)]
diff_eqs = [0; A]
@named vdp_model = SDESystem(drift_eqs, diff_eqs, t,
                             [x₁, x₂], [ϵ, ω_n, ω_g, A])
generate_raw_moment_eqs(vdp_model, 2)
```

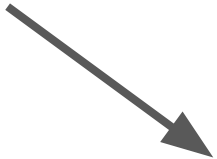
Composability of MTK

XSystem ↦ YSystem

Proper Orthogonal Decomposition and Discrete Empirical Interpolation Method (POD-DEIM)

ModelOrderReduction.jl

ODESystem ↦ ODESystem



```
pod_dim = deim_dim = 5
deim_sys = deim(simp_sys, snapshot_simpsys,
                pod_dim; deim_dim = deim_dim)
```

Composability of MTK

XSystem and others ↦ YSystem

OM.jl

Modelica ↦ ODESystem

```
model SimpleMechanicalSystem
  parameter Real J1 = 1 , J2 = 1;
  constant Real C = 1;
  Real u;
  Real Phi_1, Phi_2;
  Real omega_1 , omega_2;
  Real tau_1, tau_2, tau_3;
initial equation
  omega_1 = 0;
  omega_2 = 0;
  Phi_1 = 0;
  Phi_2 = 0;
equation
  u = time;
  // The Algebraic variables
  tau_1 = u;
  tau_2 = C * (Phi_2 - Phi_1);
  tau_3 = -tau_2;
  der(Phi_1) = omega_1;
  der(Phi_2) = omega_2;
  der(omega_1) = (tau_1 + tau_2) / J1;
  der(omega_2) = tau_3 / J2;
end SimpleMechanicalSystem;
```

Clearly, many (young) contributors have easily added their own transformation passes in record time.

Why is MTK so hackable for writing transformations?

I will show the interactivity by describing the structural simplification passes!

```
julia>
```

```
julia> ts = TearingState(expand_connections(rc_model)); find_solvables!(ts); ts.structure
SystemStructure with 22 equations and 23 variables
  #  ∂ₜ  eq            ∂ₜ  v
  1      [3, 4, 5]     2↓ [8]
  2      [6, 7]        1↑ [5]
  3      [7, 8]           [1, 18]
  4      [4, 8]           [1, 4]
  5      [2, 9, 10]       [1, 16]
  6      [11, 12]         [2, 19]
  7      [12, 13]         [2, 3, 17]
  8      [1, 13]          [3, 4]
  9      [14]             [5, 20, 21]
 10      [15, 16, 17]     [5, 18]
 11      [18, 19]         [6, 22]
 12      [19, 20]         [6, 7, 19]
 13      [16, 21]         [7, 8]
 14      [22]             [9, 15]
 15      [14, 21]         [10, 20]
 16      [5, 17]          [10, 13]
 17      [7, 19]          [10, 16]
 18      [3, 10]          [11, 22]
 19      [6, 12]          [11, 12, 17]
 20      [9, 15]          [12]
 21      [9, 22]          [13, 15]
 22      [11, 18, 23]     [14, 21]
 23      ·                [22]
```

```
julia> dummy_derivative(ts)
Matched SystemStructure with 22 equations and 23 variables
  #  ∂ₜ  eq            ∂ₜ  v
  1      [3, 4, 5]     2↓ [8]
  2      [6, 7]        1↑ ∫ [5]
  3      [7, 8]           [1, 18]
  4      [4, 8]           [1, 4]
  5      [2, 9, 10]       [1, 16]
  6      [11, 12]         [2, 19]
  7      [12, 13]         [2, 3, 17]
  8      [1, 13]          [3, 4]
  9      [14]             [5, 20, 21]
 10      [15, 16, 17]     [5, 18]
 11      [18, 19]         [6, 22]
 12      [19, 20]         [6, 7, 19]
 13      [16, 21]         [7, 8]
 14      [22]             [9, 15]
 15      [14, 21]         [10, 20]
 16      [5, 17]          [10, 13]
 17      [7, 19]          [10, 16]
 18      [3, 10]          [11, 22]
 19      [6, 12]          [11, 12, 17]
 20      [9, 15]          [12]
 21      [9, 22]          [13, 15]
 22      [11, 18, 23]     [14, 21]
 23      ·                [22]
```

Dummy derivative

$$
\begin{aligned}
x_1 + x_2 + u_1(t) &= 0 \\
x_1 + x_2 + x_3 + u_2(t) &= 0
\end{aligned}
$$

$$
\begin{aligned}
x_1' + \dot{x}_2 + \dot{u}_1(t) &= 0 \\
x_1' + \dot{x}_2 + x_3' + \dot{u}_2(t) &= 0 \\
x_1 + x_3' + x_4 + u_3(t) &= 0
\end{aligned}
$$

$$
\begin{aligned}
x_1'' + \ddot{x}_2 + \ddot{u}_1(t) &= 0 \\
x_1'' + \ddot{x}_2 + x_3'' + \ddot{u}_2(t) &= 0 \\
x_1' + x_3'' + x_4' + \dot{u}_3(t) &= 0 \\
2x_1'' + \ddot{x}_2 + x_3'' + x_4' + u_4(t) &= 0
\end{aligned}
$$

Do all the variables with a prime have the same interpretation?

$$\dot{x} = x'$$

$$\ddot{x} = \dot{x'} = f(x)$$

Symbols could be misleading

$$x_1 + x_2 + u_1(t) = 0$$
$$x_1 + x_2 + x_3 + u_2(t) = 0$$

$$\dot{x}_1 + \dot{x}_2 + \dot{u}_1(t) = 0$$
$$\dot{x}_1 + \dot{x}_2 + \dot{x}_3 + \dot{u}_2(t) = 0$$
$$x_1 + \dot{x}_3 + x_4 + u_3(t) = 0$$

Algebraic variable

$$\ddot{x}_1 + \ddot{x}_2 + \ddot{u}_1(t) = 0$$
$$\ddot{x}_1 + \ddot{x}_2 + \ddot{x}_3 + \ddot{u}_2(t) = 0$$
$$\dot{x}_1 + \ddot{x}_3 + \dot{x}_4 + \dot{u}_3(t) = 0$$
$$2\ddot{x}_1 + \ddot{x}_2 + \ddot{x}_3 + \dot{x}_4 + u_4(t) = 0$$

Differential variable

$$\dot{x} = x'$$
$$\ddot{x} = \dot{x}' = f(x)$$

Dummy derivative: structural encoding

Tearing tells us which variable is solved by which equation

$$\text{Int} \mapsto \text{Union\{Int, Nothing\}}$$

Actual differential variables cannot be algebraically solved by any equation

$$\text{Int} \mapsto \text{Union\{Int, Nothing, SelectedState\}}$$

Hence, dummy derivatives are variables that appear to be differentiated, but its lower derivative is not a SelectedState.

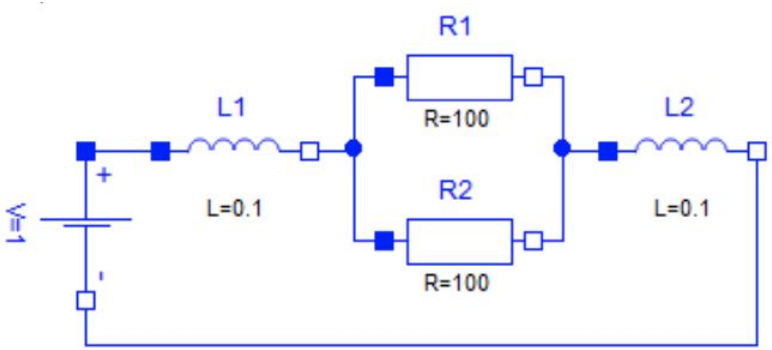
$$x_1 + x_2 + u_1(t) = 0$$
$$x_1 + x_2 + x_3 + u_2(t) = 0$$

$$x_1' + \dot{x}_2 + \dot{u}_1(t) = 0$$
$$x_1' + \dot{x}_2 + x_3' + \dot{u}_2(t) = 0$$
$$x_1 + x_3' + x_4 + u_3(t) = 0$$

$$x_1'' + \ddot{x}_2 + \ddot{u}_1(t) = 0$$
$$x_1'' + \ddot{x}_2 + x_3'' + \ddot{u}_2(t) = 0$$
$$x_1' + x_3'' + x_4' + \dot{u}_3(t) = 0$$
$$2x_1'' + \ddot{x}_2 + x_3'' + x_4' + u_4(t) = 0$$

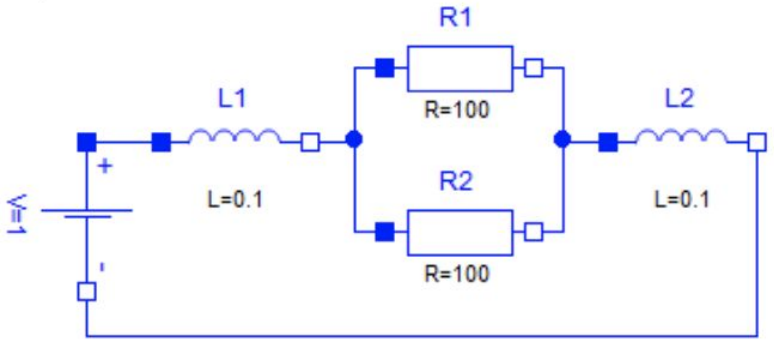
```
julia> dummy_derivative(ts)
Matched SystemStructure with 9 equations and 11 variables
  #  ∂ₜ      eq             ∂ₜ       v
  1  5↑      [10, 11]       7↑8↓     [3, 7]
  2  7↑      [8, 10, 11]    9↓       [4, 9]
  3  9↑      [1, 9, 10]     4↓       [4, 6, 8]
  4          [2, 3, 5, 7]   3↑10↓     [5, 7, 9]
  5  6↑1↓    [4, 6]         6↓       [4, 6, 8]
  6  5↓      [3, 5]         5↑11↓    ∫ [5, 7]
  7  8↑2↓    [1, 4, 6]      1↓       [4, 8, 9]
  8  7↓      [3, 5, 7]      1↑       [2]
  9  3↓      [2, 4, 7]      2↑       [3]
 10          ·              4↑       [1, 2, 3]
 11          ·              6↑       ∫ [1, 2]

julia> ts.fullvars[[6, 11]]
2-element Vector{Any}:
 Differential(t)(x2(t))
 x2(t)
```

Alias elimination




```
julia> model_ex = expand_connections(model); equations(model_ex)
34-element Vector{Equation}:
 R1₊v(t) ~ R1₊p₊v(t) - R1₊n₊v(t)
 0 ~ R1₊n₊i(t) + R1₊p₊i(t)
 R1₊i(t) ~ R1₊p₊i(t)
 R1₊v(t) ~ R1₊R*R1₊i(t)
 R2₊v(t) ~ R2₊p₊v(t) - R2₊n₊v(t)
 0 ~ R2₊n₊i(t) + R2₊p₊i(t)
 R2₊i(t) ~ R2₊p₊i(t)
 R2₊v(t) ~ R2₊R*R2₊i(t)
 L1₊v(t) ~ L1₊p₊v(t) - L1₊n₊v(t)
 0 ~ L1₊n₊i(t) + L1₊p₊i(t)
 L1₊i(t) ~ L1₊p₊i(t)
 Differential(t)(L1₊i(t)) ~ L1₊v(t) / L1₊L
 L2₊v(t) ~ L2₊p₊v(t) - L2₊n₊v(t)
 0 ~ L2₊n₊i(t) + L2₊p₊i(t)
 L2₊i(t) ~ L2₊p₊i(t)
 Differential(t)(L2₊i(t)) ~ L2₊v(t) / L2₊L
 source₊v(t) ~ source₊p₊v(t) - source₊n₊v(t)
 0 ~ source₊n₊i(t) + source₊p₊i(t)
 source₊i(t) ~ source₊p₊i(t)
 source₊v(t) ~ source₊V₊u(t)
 constant₊output₊u(t) ~ constant₊k
 ground₊g₊v(t) ~ 0
 constant₊output₊u(t) ~ source₊V₊u(t)
 source₊p₊v(t) ~ L1₊p₊v(t)
 0 ~ L1₊p₊i(t) + source₊p₊i(t)
 R1₊p₊v(t) ~ R2₊p₊v(t)
 R1₊p₊v(t) ~ L1₊n₊v(t)
 0 ~ L1₊n₊i(t) + R1₊p₊i(t) + R2₊p₊i(t)
 L2₊p₊v(t) ~ R2₊n₊v(t)
 L2₊p₊v(t) ~ R1₊n₊v(t)
 0 ~ L2₊p₊i(t) + R1₊n₊i(t) + R2₊n₊i(t)
 source₊n₊v(t) ~ L2₊n₊v(t)
 source₊n₊v(t) ~ ground₊g₊v(t)
 0 ~ L2₊n₊i(t) + ground₊g₊i(t) + source₊n₊i(t)
```

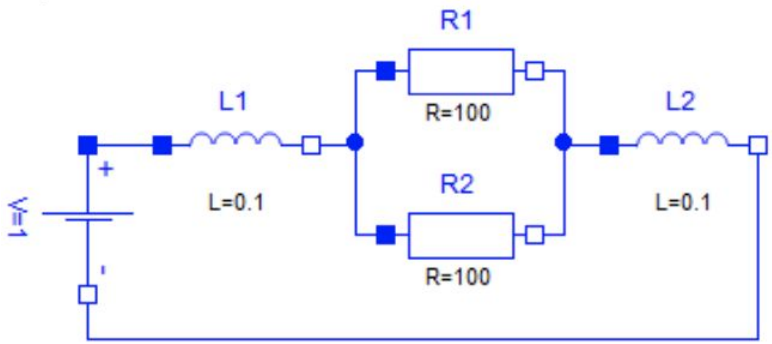
Alias elimination



Dummy derivative and tearing

```
julia> full_equations(dummy_derivative(model_ex))
4-element Vector{Equation}:
 Differential(t)(L1₊i(t)) ~ (constant₊k - L2₊p₊v(t) - R2₊R*R2₊p₊i(t)) / L1₊L
 Differential(t)(L2₊i(t)) ~ L2₊p₊v(t) / L2₊L
 0 ~ L2₊i(t) - L1₊i(t)
 0 ~ R1₊R*(R2₊p₊i(t) - L2₊i(t)) + R2₊R*R2₊p₊i(t)
```

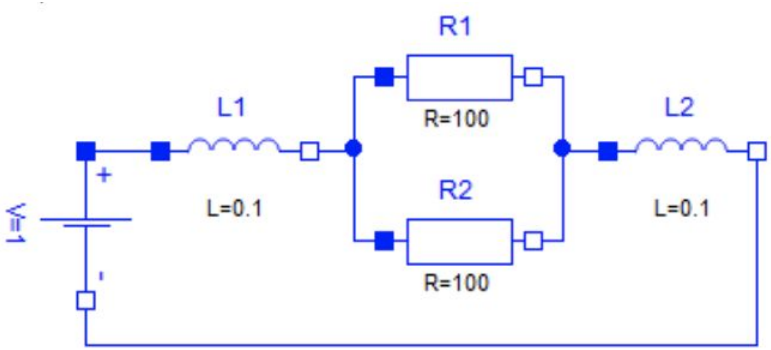
Alias elimination



Not index 1

```
julia> full_equations(dummy_derivative(model_ex))
4-element Vector{Equation}:
 Differential(t)(L1₊i(t)) ~ (constant₊k - L2₊p₊v(t) - R2₊R*R2₊p₊i(t)) / L1₊L
 Differential(t)(L2₊i(t)) ~ L2₊p₊v(t) / L2₊L
 0 ~ L2₊i(t) - L1₊i(t)
 0 ~ R1₊R*(R2₊p₊i(t) - L2₊i(t)) + R2₊R*R2₊p₊i(t)
```

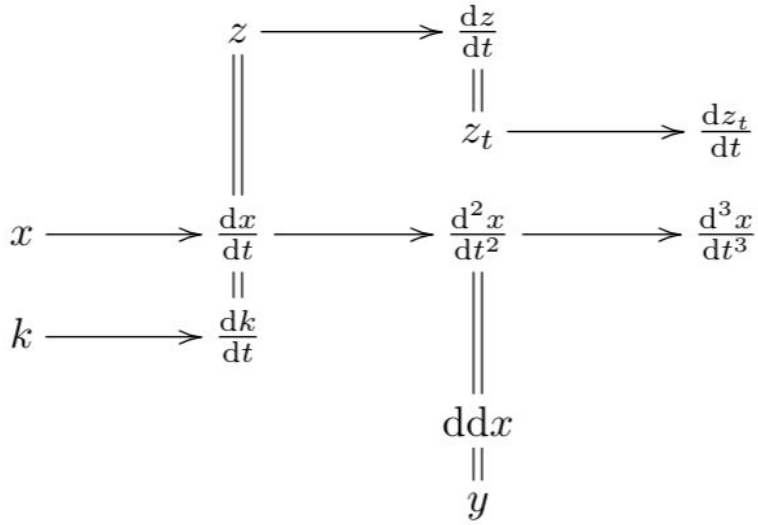
Alias elimination: singularity removal




```
julia> full_equations(dummy_derivative(alias_elimination(model_ex)))
3-element Vector{Equation}:
 Differential(t)(L1₊i(t)) ~ -((-(constant₊k - L1₊v(t) - R1₊R*(L1₊i(t) - R2₊i(t)))) / L2₊L)
 0 ~ R2₊R*R2₊i(t) - R1₊R*(L1₊i(t) - R2₊i(t))
 0 ~ (-(constant₊k - L1₊v(t) - R1₊R*(L1₊i(t) - R2₊i(t)))) / L2₊L + L1₊v(t) / L1₊L
```

Stronger alias elimination

```
eqs = [D(D(x)) ~ ddx
       ddx ~ y
       D(x) ~ z
       D(z) ~ z_t
       D(z_t) ~ sin(t)
       D(x) ~ D(k)
       D(D(D(x))) ~ sin(t)]
```

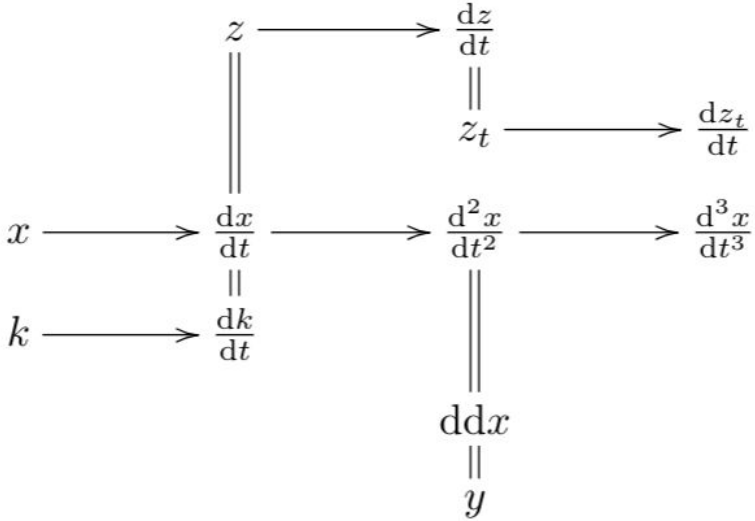


Stronger alias elimination

$$z := \frac{\mathrm{d}k}{\mathrm{d}t} := \frac{\mathrm{d}x}{\mathrm{d}t}$$

$$\frac{\mathrm{d}z}{\mathrm{d}t} := z_t := \mathrm{d}\mathrm{d}x := y := \frac{\mathrm{d}^2x}{\mathrm{d}t^2}$$

$$\frac{\mathrm{d}z_t}{\mathrm{d}t} := \frac{\mathrm{d}^3x}{\mathrm{d}t^3}.$$



Stronger alias elimination

$$z := \frac{\mathrm{d}k}{\mathrm{d}t} := \frac{\mathrm{d}x}{\mathrm{d}t}$$
$$\frac{\mathrm{d}z}{\mathrm{d}t} := z_t := \mathrm{d}\mathrm{d}x := y := \frac{\mathrm{d}^2x}{\mathrm{d}t^2}$$
$$\frac{\mathrm{d}z_t}{\mathrm{d}t} := \frac{\mathrm{d}^3x}{\mathrm{d}t^3}.$$

```
julia> aliases
7-element Vector{Any}:
  Differential(t)(k(t)) => Differential(t)(x(t))
                   z(t) => Differential(t)(x(t))
                 ddx(t) => Differential(t)(Differential(t)(x(t)))
                   y(t) => Differential(t)(Differential(t)(x(t)))
  Differential(t)(z(t)) => Differential(t)(Differential(t)(x(t)))
                 zₜ(t) => Differential(t)(Differential(t)(x(t)))
 Differential(t)(zₜ(t)) => Differential(t)(Differential(t)(Differential(t)(x(t))))
```

plot_tearing_state