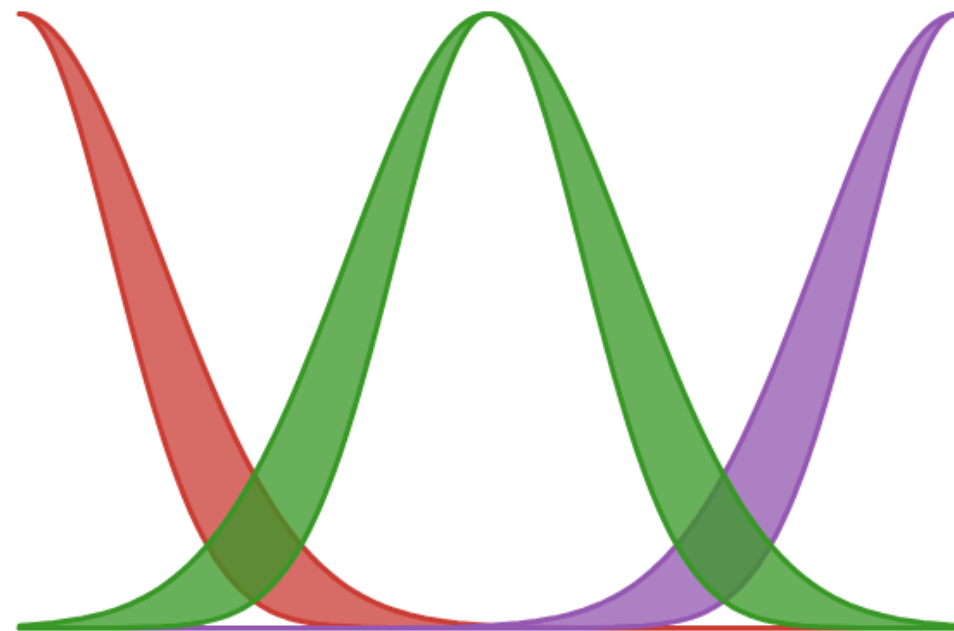


FuzzyLogic.jl

Luca Ferranti, University of Vaasa

FUZZ-IEEE 2023



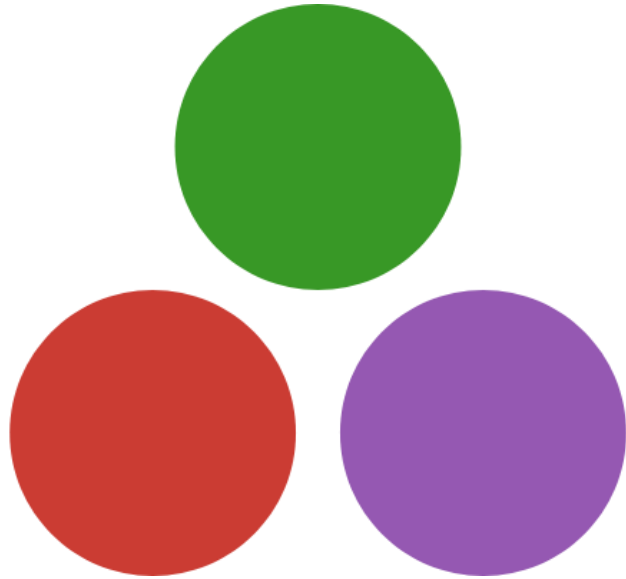


What is FuzzyLogic.jl ?

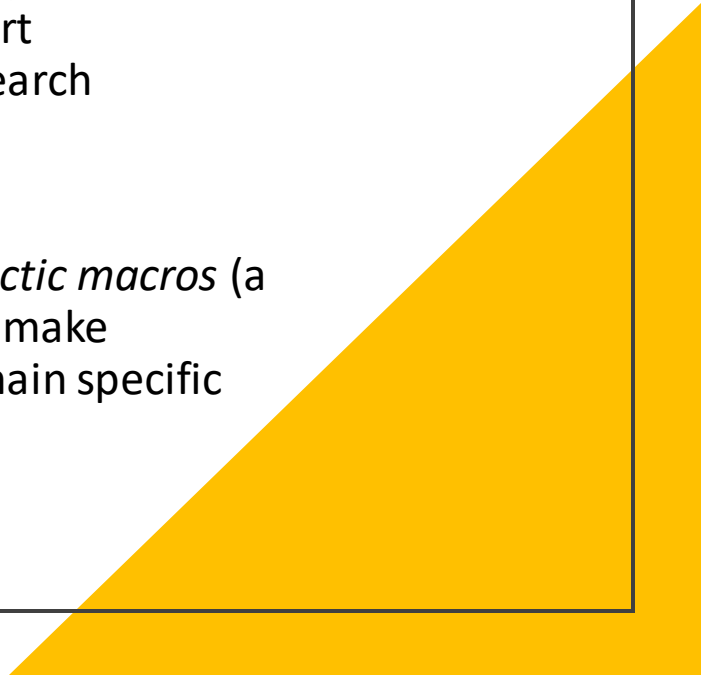
Library for fuzzy inference written fully in Julia

- **Feature rich:** Supports type-1 and interval type-2 membership functions, has several T-norms, defuzzifiers, visualization functionalities, etc.
- **Compatible:** read fuzzy models from matlab .fis files, Fuzzy Markup Language and Fuzzy Control Language
- **Portable:** Generate stand-alone Julia code and soon(ish) C-code
- **Extensible:** Easy to add your own features

Why Julia?



- Solves the **2 languages problem**
 - Runs like C, reads like Python
- Rich ecosystem, especially for scientific computing
- Composable, different libraries can be combined and "just work"
- Great package manager and environments support for reproducible research
- Open-source!
- Expressive
 - Thanks to *syntactic macros* (a la Lisp), easy to make embedded domain specific languages



Domain specific language

- @mamfis is a **macro**: it takes code parsed as Abstract syntax tree, does transformations and executes the new code.
- Macros allow to arbitrarily change the semantics of Julia.

```
tipper = @mamfis function tipper(service, food)::tip
    service := begin
        domain = 0:10
        poor = GaussianMF(0.0, 1.5)
        good = GaussianMF(5.0, 1.5)
        excellent = GaussianMF(10.0, 1.5)
    end

    food := begin
        domain = 0:10
        rancid = TrapezoidalMF(-2, 0, 1, 3)
        delicious = TrapezoidalMF(7, 9, 10, 12)
    end

    tip := begin
        domain = 0:30
        cheap = TriangularMF(0, 5, 10)
        average = TriangularMF(10, 15, 20)
        generous = TriangularMF(20, 25, 30)
    end

    service == poor || food == rancid --> tip == cheap
    service == good --> tip == average
    service == excellent || food == delicious --> tip == generous

    and = ProdAnd
end
```

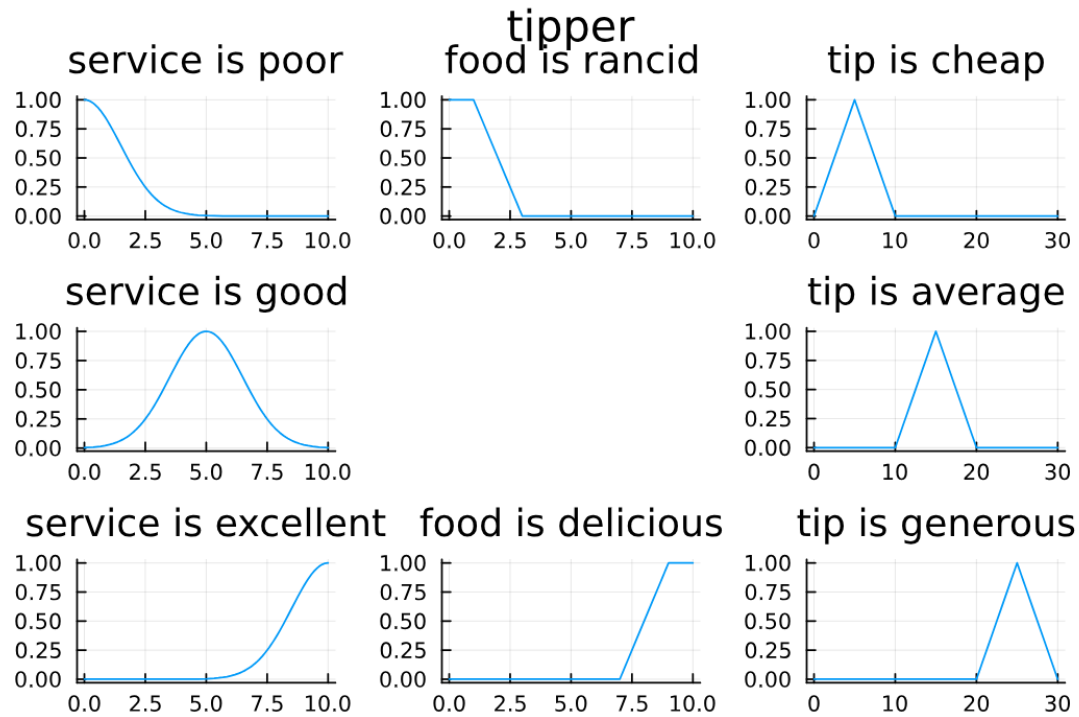
Doing inference

- The constructed Fuzzy system can be called like a function, so doing inference is just

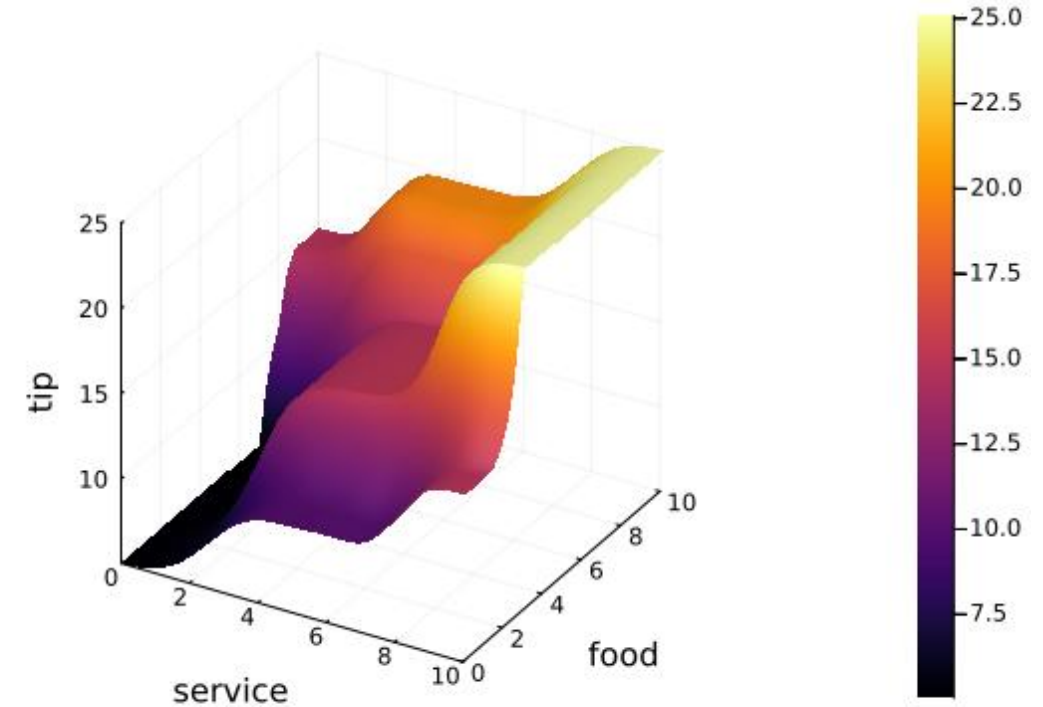
```
1-element Dictionaries.Dictionary{Symbol, Float64}  
:tip | 7.788531995027624
```

- tipper(service=2, food=3)

Visualizations



• `plot(tipper)`



Inside the Fuzzy Inference System

```
Base.@kwdef struct MamdaniFuzzySystem{And <: AbstractAnd, Or <: AbstractOr,  
                                       Impl <: AbstractImplication,  
                                       Aggr <: AbstractAggregator,  
                                       Defuzz <: AbstractDefuzzifier,  
                                       R <: AbstractRule} <: AbstractFuzzySystem  
  
    name::Symbol  
    inputs::Dictionary{Symbol, Variable} = Dictionary{Symbol, Variable}()  
    outputs::Dictionary{Symbol, Variable} = Dictionary{Symbol, Variable}()  
    rules::Vector{R} = FuzzyRule[]  
    and::And = MinAnd()  
    or::Or = MaxOr()  
    implication::Impl = MinImplication()  
    aggregator::Aggr = MaxAggregator()  
    defuzzifier::Defuzz = CentroidDefuzzifier()  
end
```

Type Structure of fuzzy system

Note that algorithmic choices (what T-norm definition, what defuzzifier, etc.) are part of the type signature!

Hence they are known at compile time and the compiler can optimize, by already picking the right algorithm

The system is flexible, but no runtime overhead!

Rules representation

- Rules are represented as syntactic trees
- Hence arbitrary logic expressions are supported

```
FuzzyOr(  
  FuzzyRelation(:service, :poor),  
  FuzzyAnd(  
    FuzzyRelation(:service, :excellent),  
    FuzzyRelation(:food, :rancid)  
  )  
)
```

Extensible

- Flexible interface, easy to add custom membership functions, T-norms etc.

```
struct MySingleton { T <: Real } <: AbstractMembershipFunction
  c :: T
end

(mf :: MySingleton) (x) = x == mf.c ? 1 : 0
```

Code generation

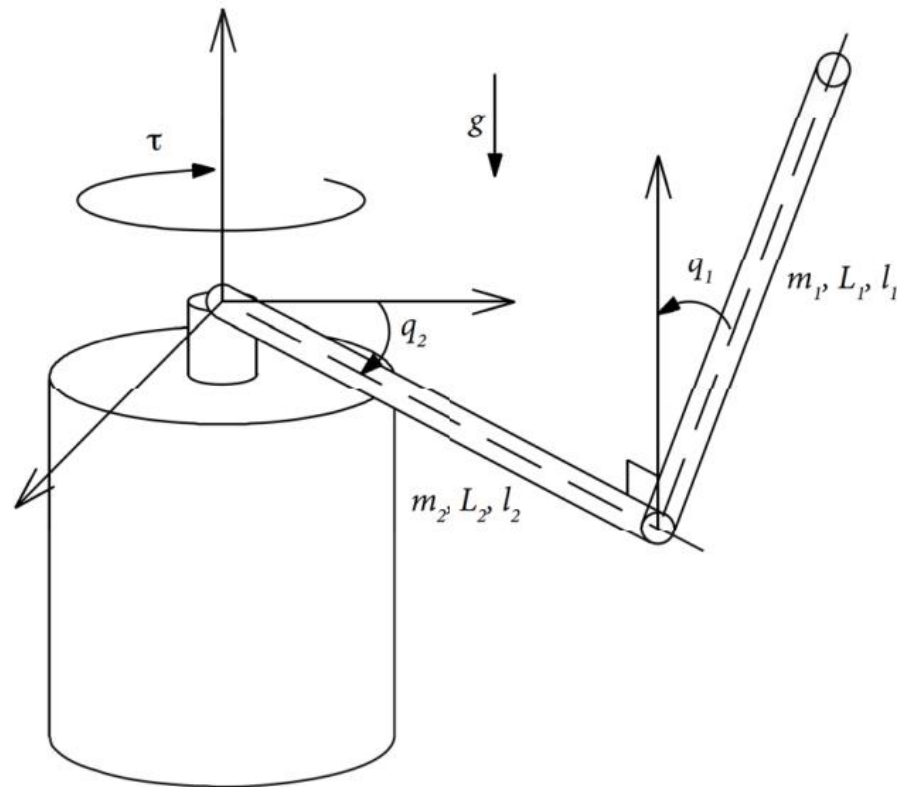
Generate stand-alone
optimized Julia code

```
function tipper(service, food)
    poor = exp(-((service - 0.0) ^ 2) / 4.5)
    good = exp(-((service - 5.0) ^ 2) / 4.5)
    excellent = exp(-((service - 10.0) ^ 2) / 4.5)
    rancid = max(min((food - -2) / 2, 1, (3 - food) / 2), 0)
    delicious = max(min((food - 7) / 2, 1, (12 - food) / 2), 0)
    ant1 = max(poor, rancid)
    ant2 = good
    ant3 = max(excellent, delicious)
    tip_agg = collect(LinRange{Float64}(0.0, 30.0, 101))
    @inbounds for (i, x) = enumerate(tip_agg)
        cheap = max(min((x - 0) / 5, (10 - x) / 5), 0)
        average = max(min((x - 10) / 5, (20 - x) / 5), 0)
        generous = max(min((x - 20) / 5, (30 - x) / 5), 0)
        tip_agg[i] = max(max(min(ant1, cheap), min(ant2, average)), min(ant3, g
enerous))
    end
    tip = ((2 * sum((mfi * xi for (mfi, xi) = zip(tip_agg, LinRange{Float64}(0.0, 3
0.0, 101)))) - first(tip_agg) * 0) - last(tip_agg) * 30) / ((2 * sum(tip_agg) - first
(tip_agg)) - last(tip_agg))
    return tip
end
```

Cool application! Furuta pendulum control

$$0 = \theta_1 \ddot{q}_1 + \theta_2 \cos(q_1) \ddot{q}_2 - \theta_3 \sin(q_1) \cos(q_1) \dot{q}_2^2 - \theta_4 g \sin(q_1),$$

$$\tau = \theta_2 \cos(q_1) \ddot{q}_1 + (\theta_5 + \theta_3 \sin^2(q_1)) \ddot{q}_2 - \theta_2 \sin(q_1) \dot{q}_1^2 + \\ 2\theta_3 \sin(q_1) \cos(q_1) \dot{q}_1 \dot{q}_2,$$



Let's do it in Julia – define the parameters

```
begin
    @variables t q1(t) q2(t)
    @constants g = 9.81
    @parameters m1 = 67.9e-3 m2 = 0.2869 L1 = 0.14 L2 = 0.235 l1 = 0.07 l2 = 0.1175
    I1 = 5.5452e-5 I2 = 1.9e-3
    θ1 = I1 + m1 * L1
    θ2 = m1 * l1 * L2
    θ3 = m1 * l1^2
    θ4 = m1 * l1
    θ5 = I2 + m2 * l2^2

    d = Differential(t)

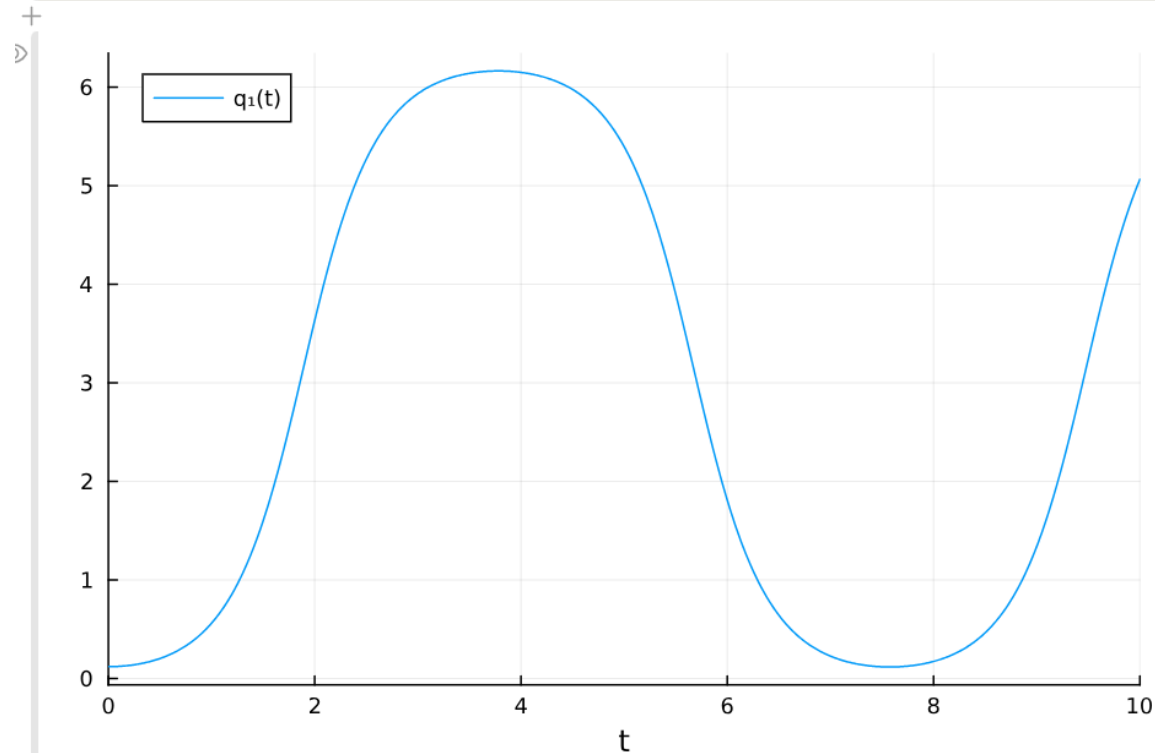
    M = [θ1      θ2*cos(q1);
         θ2*cos(q1) (θ5+θ3*sin(q1)^2)]

    rhs = [θ3 * sin(q1) * cos(q1) * d(q2) ^ 2 + θ4 * g * sin(q1);
           θ2 * sin(q1) * d(q1)^2 - 2θ3*sin(q1)*cos(q1)*d(q1)*d(q2)]

    |tspan = (0.0, 10.0)
    q0 = [d(q2) => 0, d(q1) => 0, q2 => 0, q1 => 0.12]
end
```

Let's do it in Julia -- define the ODE system

```
begin
    @named sys = ODESystem((d^2).([q1, q2]) .~ M\rhs)
    sys = structural_simplify(sys)
    prob = ODEProblem(sys, q0, tspan)
end
```



```
begin
    sol = solve(prob, Tsit5())
    plot(sol, idxs=(0, 2))
end
```

Let's define our Fuzzy controller

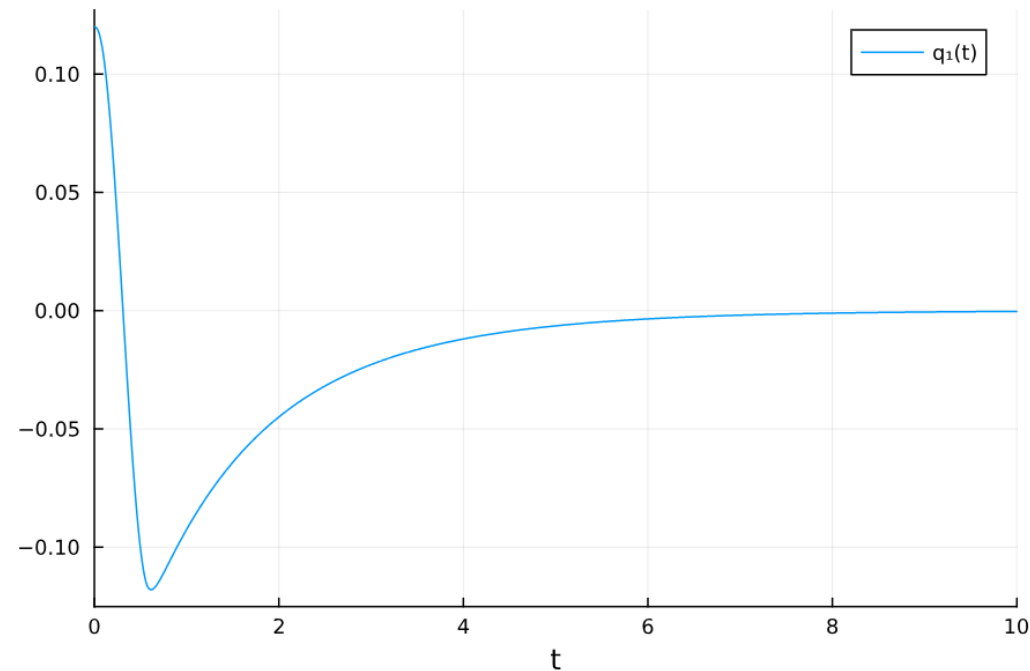
```
fis = @mamfis function controller(x[1:2])::τ
    for i in 1:2
        x[i] := begin
            domain = -2π:2π
            positive = SigmoidMF(-1.45, 0.0)
            negative = SigmoidMF(1.45, 0.0)
        end
    end

    τ := begin
        domain = -10:10
        negative = GaussianMF(5.0, 1/sqrt(2))
        zero = GaussianMF(0.0, 1/sqrt(2))
        positive = GaussianMF(-5.0, 1/sqrt(2))
    end

    x[1] == negative && x[2] == positive --> τ == negative
    x[1] == negative && x[2] == positive --> τ == zero
    x[1] == positive && x[2] == negative --> τ == zero
    x[1] == positive && x[2] == positive --> τ == positive
end
```

Controlled system -- solution

```
• begin
•    $\tau(x1, x2) = \text{fis}(\text{; } x1, x2)[:\tau]$ 
•   @register_symbolic  $\tau(x1, x2)$ 
•   @named sys_controlled = ODESystem(( $d^2$ ).([ $q_1, q_2$ ]) .~  $M \backslash (\text{rhs} + [0; \tau(q_1, d(q_1))])$ )
•   sys_controlled = structural_simplify(sys_controlled)
•   prob_controlled = ODEProblem(sys_controlled,  $q_0$ , tspan)
• end
```



```
• begin
•   sol_controlled = solve(prob_controlled, Tsit5())
•   plot(sol_controlled, idxs=(0, 2))
• end
```


Future work

- Automatically learning and tuning parameters
 - Integration with Enzyme for AD
- GUI / web interface
- Generate C – code
- ...
- ...
- ...

Future prospects -- teaching

The screenshot shows the homepage of the 'Fuzzy Systems' website. The header features the title 'Fuzzy Systems' in a large, white, serif font on a dark blue background. Below the header, there is a sidebar on the left with a search bar, a 'Choose your track:' dropdown menu, and a 'WELCOME' section with links to 'Class logistics', 'Software installation', and 'Cheatsheets'. The main content area is titled 'Highlights' and contains two sections. The first section, 'Learn fuzzy logic and applications to real-world problems', includes a paragraph about fuzzy logic and a graph of three overlapping bell curves in red, green, and purple. The second section, 'Revolutionary interactivity with Pluto.jl', includes a paragraph about the website's interactive features and a small plot with a blue line and a red dot.

Fuzzy Systems
an overview of fuzzy logic and applications
by Luca Ferranti

Search...

Choose your track:
Choose...

WELCOME

Class logistics
Software installation
Cheatsheets

1. BASIC CONCEPTS

1.1 Introduction to fuzzy thinking
1.2 Fuzzy Sets
1.3 Fuzzy logic operators
1.4 Introduction to fuzzy numbers

Homework 0: Intro to Julia
Homework 1: fuzzy sets and logic

2. FUZZY INFERENCE SYSTEMS

2.1 Introduction to Fuzzy

Highlights

Learn fuzzy logic and applications to real-world problems

While traditional logic only has true or false, fuzzy logic allows truth values to be in between. This can model vagueness and express concepts like *partially true*. **Fuzzy logic mimics human reasoning and offers a framework to model uncertainty, with applications to explainable AI and other engineering domains.**

Revolutionary interactivity with Pluto.jl

Thanks to Pluto.jl, the website is built using real code, and instead of a book, we have a series of interactive notebooks. **On the website,**

- I am building a **fully open and fully free** class on fuzzy systems
- Inspired by Computational thinking with Julia taught at MIT
 - <https://computationalthinking.mit.edu/>
- Interactive lecture material, youtube videos
- To be taught in Spring 2024

Get involved!

- Check out the repository:
<https://github.com/lucaferranti/FuzzyLogic.jl>
- Use the package and open an issue / send me an email if you encounter problems
- If you have feature suggestions, open an issue
- Come talk to me any time! I like talking with other humans!

Questions?

- Software repo:
<https://github.com/lucaferranti/FuzzyLogic.jl>
- Software docs: <https://www.lucaferranti.com/FuzzyLogic.jl/stable/>
- My website: <https://lucaferranti.com>

These slides:

