```julia
begin
    # numerical libraries
    using Expokit, PROPACK, Arpack, SparseArrays
    # output and plotting
    using ProgressLogging, JLD
    # modelling and statistics
    using Catalyst, JumpProcesses, StatsBase, DifferentialEquations
    using Interpolations
    # importing local fsp package
    using Revise
    local_mod = include("../src/DiscStochSim.jl")
    using .local_mod.DiscStochSim
end
```

```
Replacing docs for `Main.var"workspace#2".DiscStochSim.FindLowestValuesPercent :
: Union{Tuple{T}, Tuple{Vector{T}, Number}} where T` in module `Main.var"workspa
ce#2".DiscStochSim`
```

rn =

$$\varnothing \underset{\mathrm{d}_1 + \frac{s\gamma}{1+s}}{\overset{\left(\frac{\mathrm{K}_1^3 \beta_1}{\mathrm{K}_1^3 + V^3} + \alpha_1 \right) \eta}{\rightleftharpoons}} \mathrm{U}$$

$$\varnothing \underset{\mathrm{d}_2}{\overset{\left(\frac{\mathrm{K}_2^3 \beta_2}{\mathrm{K}_2^3 + U^3} + \alpha_2 \right) \eta}{\rightleftharpoons}} \mathrm{V}$$

```julia
rn = @reaction_network begin
    (η*(α₁ + (β₁*K₁^3/(K₁^3 + V^3))), d₁+s*γ/(1+s)), 0 <--> U
    (η*(α₂ + (β₂*K₂^3/(K₂^3 + U^3))), d₂), 0 <--> V
end
```

$$\left[\left(\frac{\mathrm{K}_1^3 \beta_1}{\mathrm{K}_1^3 + \left(V\left(t \right) \right)^3} + \alpha_1 \right) \eta, \ \left(\mathrm{d}_1 + \frac{s\gamma}{1 + s} \right) U\left(t \right), \ \left(\frac{\mathrm{K}_2^3 \beta_2}{\mathrm{K}_2^3 + \left(U\left(t \right) \right)^3} + \alpha_2 \right) \eta, \ \mathrm{d}_2 V\left(t \right) \right]$$

```julia
begin
    model = DiscreteStochasticSystem(rn);
    jumpratelaw.(Catalyst.get_rxs(rn));
end
```

```
def_params = 0.0:0.25:30.0
```

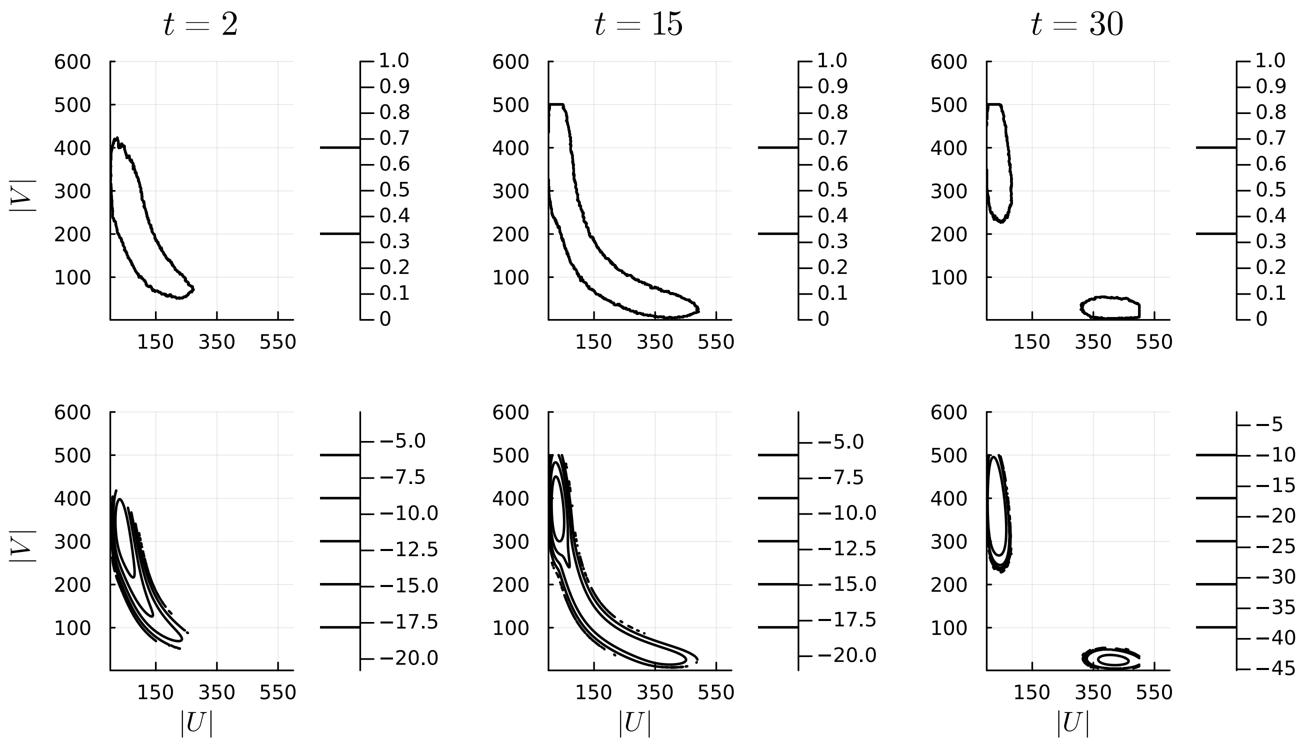
```
def_params = begin
    # reaction rates
    scale=100
    rates = [1.0, 0.2*scale,
             4.0*scale,
             1.0*scale,
             1.0,
             0.1,
             1.0,
             0.2*scale,
             4.0*scale,
             1.0*scale,
             1.0]
    # boundary
    bounds = (0, 500) #(lower limit, upper limit)
    boundary_condition(x) = RectLatticeBoundaryCondition(x, bounds);
    # time interval and initial values
    δt = 0.25
    T = 0:δt:30
end
```

```
init_fsp_vars = begin
    global U₀ = CartesianIndex(85, 5)
    global 𝒮₀ = Set([U₀])
    global 𝒮₀ = expand!(𝒮₀, model, rates, 0.0, boundary_condition, 2);
    # initial probability vector (only for active states)
    global p₀ = zeros(𝒮₀ |> length)
    global p₀[FindElement(U₀, 𝒮₀)] = 1
end;
```

fsp_sim =

```
fsp_sim = begin
    # copy initial values
    pₜ = copy(p₀)
    𝒮ₜ = copy(𝒮₀)
    # time stepping loop
    iter = 1
    pₜ = p₀
    # variables to store simulation observables
    size_𝒮ₜ = Int.(zeros(length(T))) # system sizes
    ϵₜ = zeros(length(T)) #local truncation err
    sol = Array{SparseMatrixCSC,1}(undef, length(T))
    @progress for (iter, t) ∈ enumerate(T)
        # expand state space
        global 𝒮ₜ, pₜ = expand!(𝒮ₜ, pₜ, model, rates , t, boundary_condition, 50)
        global size_𝒮ₜ[iter] = length(𝒮ₜ)
        A = MasterEquation(𝒮ₜ, model, rates, boundary_condition, t)
        # solve system and normalize (using expokit)
        global pₜ = expmv(δt, A, pₜ)
        # add add states in sparse arrays
        I = [a[1] for a in 𝒮ₜ]
        J = [a[2] for a in 𝒮ₜ]
        global sol[iter] = sparse(I, J, pₜ)
        # purge state space
        𝒮ₜ, pₜ = purge!(𝒮ₜ, pₜ, 1e-8)
        ϵₜ[iter] = 1.0 - sum(pₜ)
        pₜ ./= sum(pₜ)
    end
end
```

100%




```
Plot{Plots.GRBackend() n=6}
```