Composing models

Illustrative Example

Here is the complete example of composing, visualizing and solving the SuperFast model and the Fast-JX model, with explanation to follow:

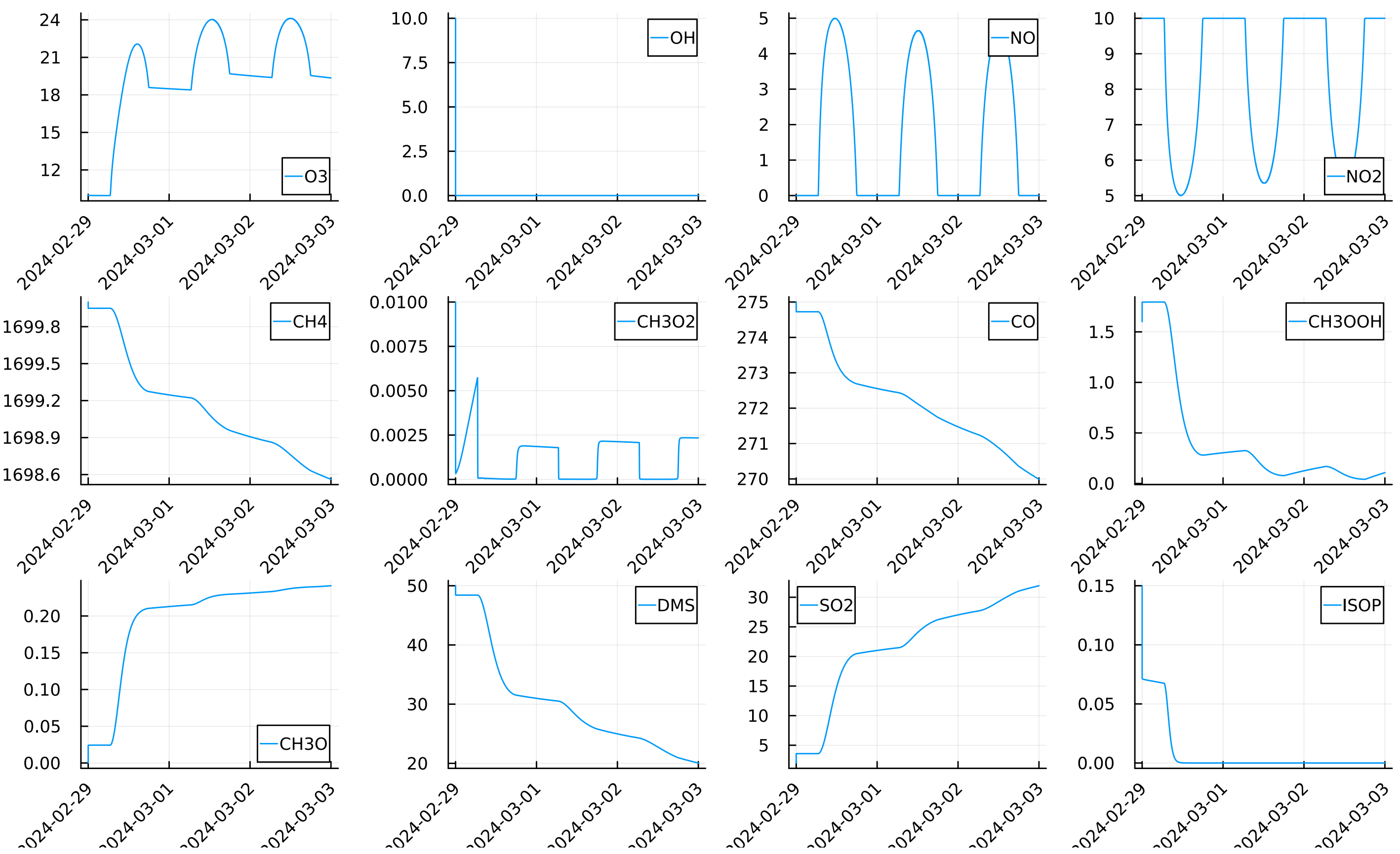
```
using EarthSciMLBase, GasChem, ModelingToolkit, OrdinaryDiffEq, Dates, Unitful, Diff


@parameters t
composed_ode = SuperFast(t) + FastJX(t) # Compose two models simply use the "+" oper

start = Dates.datetime2unix(Dates.DateTime(2024, 2, 29))
tspan = (start, start+3600*24*3)
sys = structural_simplify(get_mtk(composed_ode)) # Define the coupled system
sol = solve(ODEProblem(sys, [], tspan, []),AutoTsit5(Rosenbrock23()), saveat=10.0) #
```

In the composed system, the variable name for O_3 is not `O3` but `superfast₊O3(t)`. So we need some preparation of the result before visualizing.

```
vars = states(sys)  # Get the variables in the composed system
var_dict = Dict(string(var) => var for var in vars)
pols = ["O3", "OH", "NO", "NO2", "CH4", "CH3O2", "CO","CH3OOH", "CH3O", "DMS", "SO2"
var_names_p = ["superfast₊$(v)(t)" for v in pols]

x_t = unix2datetime.(sol[t]) # Convert from unixtime to date time for visualizing
```

Then, we could plot the results as:

```
using Plots
pp = []
for (i, v) in enumerate(var_names_p)
    name = pols[i]
    push!(pp, Plots.plot(x_t,sol[var_dict[v]],label = "$name", size = (1000, 600), x
end
Plots.plot(pp..., layout=(3, 4))
```



Add Emission model

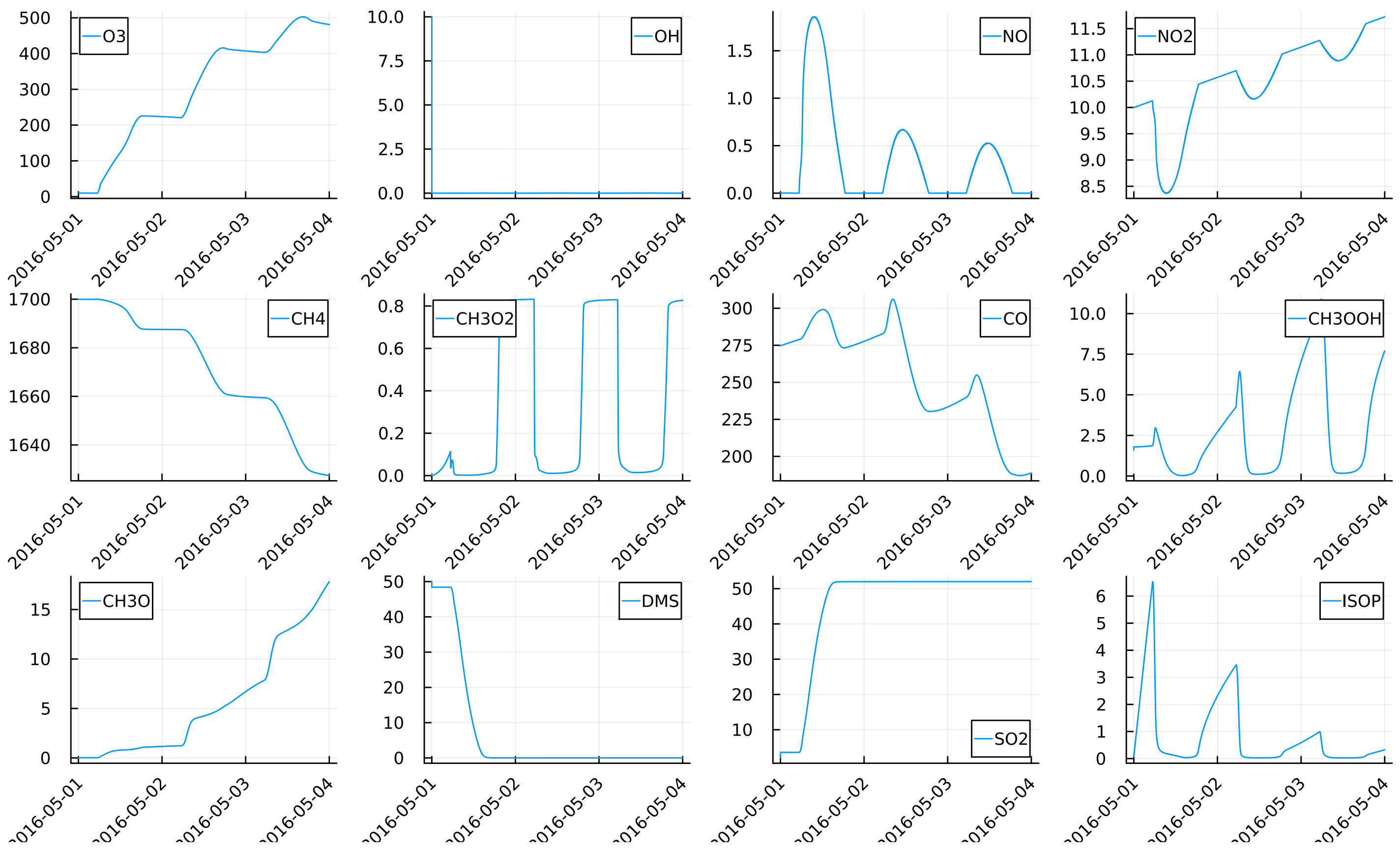
GasChem.jl incorporates an emissions model that utilizes data from the US National Emissions Inventory for the year 2016. This model is activated as an extension when the `EarthSciData` package is used. Here's a simple example:

```
using GasChem, EarthSciData # This will trigger the emission extension
using Dates, ModelingToolkit, OrdinaryDiffEq, DifferentialEquations, EarthSciMLBase

ModelingToolkit.check_units(eqs...) = nothing
@parameters t
composed_ode = SuperFast(t)+FastJX(t)+Emission(t) # Compose SuperFast, FastJX and th
sys = structural_simplify(get_mtk(composed_ode))

start = Dates.datetime2unix(Dates.DateTime(2016, 5, 1))
tspan = (start, start+3*24*3600)
sol = solve(ODEProblem(sys, [], tspan, []),AutoTsit5(Rosenbrock23()), saveat=10.0)
```

We could visualize the results with the following codes:

```
vars = states(sys)  # Get the variables names
var_dict = Dict(string(var) => var for var in vars)

using Plots
x_t = unix2datetime.(sol[t])
```

```
pols = ["O3", "OH", "NO", "NO2", "CH4", "CH3O2", "CO","CH3OOH", "CH3O", "DMS", "SO2"
var_names_p = ["superfast₊$(v)(t)" for v in pols]
pp = []
for (i, v) in enumerate(var_names_p)
    name = pols[i]
    p = Plots.plot(x_t,sol[var_dict[v]],label = "$name", size = (1000, 600), xrotati
    push!(pp, p)
end
Plots.plot(pp..., layout=(3, 4))
```



« Parameters

API »

Powered by Documenter.jl and the Julia Programming Language.