NonLinearSystemNeuralNetworkFMU.jl

August 18, 2022

Contents

Part I

Home

Chapter 1

NonLinearSystemNeuralNetworkFMU.jl

Generate Neural Networks to replace non-linear systems inside OpenModelica 2.0 FMUs.

1.1 Table of Contents

1.2 Overview

The package generates an FMU from a modelica file in 3 steps (+ 1 user step):

1. Find non-linear equation systems to replace.
   - Simulate and profile Modelica model with OpenModelica using OMJulia.jl.
   - Find slowest equations below given threshold.
   - Find depending variables specifying input and output for every non-linear equation system.
   - Find min-max ranges for input variables by analyzing the simulation results.
2. Generate training data.
   - Generate 2.0 Model Exchange FMU with OpenModelica.
   - Add C interface to evaluate single non-linear equation system without evaluating anything else.
   - Re-compile FMU.
   - Initialize FMU using FMI.jl.
   - Generate training data for each equation system by calling new interface.
3. Train neural network.

   - Step performed by user.

4. Integrate neural network into FMU

   - Replace equations with neural network in generated C code.
   - Re-compile FMU.

1.3 Installation

Clone this repository to your machine and use the package manager Pkg to develop this package.

```
(@v1.7) pkg> dev /path/to/NonLinearSystemNeuralNetworkFMU
julia> using NonLinearSystemNeuralNetworkFMU
```

Part II

Profiling

Chapter 2

Profiling Modelica Models

2.1 Functions

NonLinearSystemNeuralNetworkFMU.profiling – Function.

```
profiling(modelName, pathToMo, pathToOmc, workingDir; threshold = 0.03)
```

Find equations of Modelica model that are slower then threashold.

Arguments

- `modelName::String`: Name of the Modelica model.
- `pathToMo::String`: Path to the *.mo file containing the model.
- `pathToOm::Stringc`: Path to omc used for simulating the model.

Keywords

- `workingDir::String = pwd()`: Working directory for omc. Defaults to the current directory.
- `threshold = 0.01`: Slowest equations that need more then `threshold` of total simulation time.

Returns

- `profilingInfo::Vector{ProfilingInfo}`: Profiling information with non-linear equation systems slower than `threshold`.

source

NonLinearSystemNeuralNetworkFMU.minMaxValuesReSim – Function.

```
minMaxValuesReSim(vars::Array{String}, modelName::String, pathToMo::String, pathToOmc::String;
↪  workingDir::String = pwd())
```

(Re-)simulate Modelica model and find miminum and maximum value each variable has during simulation.

Arguments

- `vars::Array{String}`: Array of variables to get min-max values for.
- `modelName::String`: Name of Modelica model to simulate.
- `pathToMo::String`: Path to .mo file.

- `pathToOm::Stringc`: Path to OpenModelica Compiler omc.

Keywords

- `workingDir::String = pwd()`: Working directory for omc. Defaults to the current directory.

Returns

- `min::Array{Float64}`: Minimum values for each variable listed in `vars`, minus some small epsilon.
- `max::Array{Float64}`: Maximum values for each variable listed in `vars`, plus some small epsilon.

source

2.2 Examples

Find Slowest Non-linear Equation Systems

We have a Modelica model `SimpleLoop`, see test/simpleLoop.mo with some non-linear equation system

$$r^2 = x^2 + y^2$$
$$rs = x + y$$

We want to see how much simulation time is spend solving this equation. So let's start `profiling`:

```
julia> using NonLinearSystemNeuralNetworkFMU


julia> modelName = "simpleLoop";


julia> pathToMo = joinpath("test","simpleLoop.mo");



julia> profilingInfo = profiling(modelName, pathToMo, omc; threshold=0)
ERROR: UndefVarError: omc not defined
```

We can see that non-linear equation system 14 is using variables `s` and `r` as input and has iteration variable `y`. `x` will be computed in the inner equation.

```
julia> profilingInfo[1].usingVars
ERROR: UndefVarError: profilingInfo not defined

julia> profilingInfo[1].iterationVariables
ERROR: UndefVarError: profilingInfo not defined
```

So we can see, that equations 14 is the slowest non-linear equation system. It is called 2512 times and needs around 15% of the total simulation time, in this case that is around 592 μs.

If we want to get the minimal and maximal values for the used variables `s` and `r` can get we can use `minMaxValuesReSim`. This will re-simulate the Modelica model and read the simulation results to find the smallest and largest values for each given variable.

```
julia> (min, max)  = minMaxValuesReSim(profilingInfo[1].usingVars, modelName, pathToMo, omc)
ERROR: UndefVarError: profilingInfo not defined
```