**Instructions for Automated Modular Modeling and Control**

This file contains instructions for symbolically deriving the interconnected state-space model, given a connection between objects (modules). This can be either the open-loop dynamic model or the closed-loop dynamic depending on whether the objects have control built into them or not.

To produce an input file for running the code, follow the below steps

1. Construct objects of the appropriate class. Each constructor takes as input the index name (which is appended to the end of the variables from that class), and the rotating reference frame angular position and angular velocity (if you just want a stationary reference frame, just choose phi=0 and dphidt = 0). So that, the KCL and KVL equations between modules hold, the reference frame for each object should be the same. The one exception for this is the PE\_Flywheel class, where the reference frame for the flywheel side of the AC/DC/AC can be different since the power electronics side, not the flywheel side, is connected to the rest of the grid.

Here is an example of constructing objects:

syms phi dphidt real

syms phi2 dphi2dt real

SM1IndexName = {'1'};

SM1 = SmGcEc(SM1IndexName,phi,dphidt);

TL1IndexName = {'2'};

TL1 = TransmissionLine(TL1IndexName,phi,dphidt);

IM1IndexName = {'3'};

IM1 = InductionMachine(IM1IndexName,phi,dphidt);

PE\_Flywheel1IndexName = {'4'};

PE\_Flywheel1 = PE\_Flywheel(PE\_Flywheel1IndexName,[phi;phi2],[dphidt;dphi2dt]);

Load1IndexName = {'6'};

Load1 = Load(Load1IndexName,phi,dphidt);

2. Combine all objects into one cell array. For example,

Modules = {SM1, TL1, IM1, PE\_Flywheel1, Load1};

3. Enter the G matrix describing the connections between modules. The dimensions of the G matrix are the number of junctions (each phase counts as a separate junction) by the total number of ports of all objects.

G = [1 0 1 0 0 0 0 0 0 0 0 0

0 1 0 1 0 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 1 0 1 0

0 0 0 0 0 1 0 1 0 1 0 1];

Since for larger systems, it can be tedious to write the entire G matrix, a function “ProduceGMatrix” was created for forming the G matrix. Here, the user enters all the objects connected to each bus. (Here it is assumed that all phases of the object are connected to the bus. If you want an unbalanced system where different phases of the object are connected differently, it is necessary to form the G by hand.) For an object with two ports per phase, such as a transmission line, either ‘L’ or ‘R’ to distinguish whether it is the left port or the right port of the transmission line that is connected to the bus.

Bus1 = { {D1}, {Load1}, {TL1, 'L'}, {TL2, 'L'} };

Bus2 = { {W1}, {PE\_Flywheel1}, {Load2}, {TL1, 'R'} };

Bus3 = { {H1}, {TL2,'R'} };

Buses = {Bus1, Bus2, Bus3};

G = ProduceGMatrix(Modules,Buses);

4. Construct the PowerSystem object, whose constructor takes as input the cell array Modules and the connection matrix G.

PS = PowerSystem(G,Modules);

5. Write the state-space equations for the interconnected system using the PrintStateSpace method of the PowerSystem class. The second argument is the text file where the equations will be written.

PS.PrintStateSpace(PS,'Equations/TwoBusEx.txt')

Here is an example of running the automated modular code for forming the dynamic equations for a system with a synchronous machine (with governor and exciter control) connected through a transmission line to an induction machine, load, and power electronics/flywheel.

% -- Load

% SM -- TL -- IM

% -- PE -- Flywheel

clear classes

syms phi dphidt real

syms phi2 dphi2dt real

SM1IndexName = {'1'};

SM1 = SmGcEc(SM1IndexName,phi,dphidt);

TL1IndexName = {'2'};

TL1 = TransmissionLine(TL1IndexName,phi,dphidt);

IM1IndexName = {'3'};

IM1 = InductionMachine(IM1IndexName,phi,dphidt);

PE\_Flywheel1IndexName = {'4'};

PE\_Flywheel1 = PE\_Flywheel(PE\_Flywheel1IndexName,[phi;phi2],[dphidt;dphi2dt]);

Load1IndexName = {'6'};

Load1 = Load(Load1IndexName,phi,dphidt);

Modules = {SM1, TL1, IM1, PE\_Flywheel1, Load1};

G = [1 0 1 0 0 0 0 0 0 0 0 0

0 1 0 1 0 0 0 0 0 0 0 0

0 0 0 0 1 0 1 0 1 0 1 0

0 0 0 0 0 1 0 1 0 1 0 1];

PS = PowerSystem(G,Modules);

PS.PrintStateSpace(PS,'Equations/TwoBusEx.txt')

Here is a description of the classes I already implemented for my research.

ClassName: SmGcEc

Description: Synchronous machine with governor control and exciter control

The state variables for this object are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the stator current |
|  | quadrature component of the stator current |
|  | rotor current |
|  | machine angular speed |
|  | machine angular position |
|  | load (mechanical) torque |
|  | valve position |
|  | integral of difference between the speed and the speed set point |
|  | rotor voltage |

The port inputs determined by this object’s connection to the rest of the grid are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the stator voltage |
|  | quadrature component of the stator voltage |

The parameters are:

|  |  |
| --- | --- |
| Symbol | Description |
| LR | Self-inductance of the rotor winding |
| LS | Self-inductance of the stator windings |
| LSS | Mutual inductance between the stator windings |
| M | Mutual inductance between the stator and rotor windings when parallel |
| RR | Resistance of the rotor winding |
| RS | Resistance of the stator windings |
| J | Inertia of the rotor |
| B | Damping coefficient of the rotor |
|  | Time-constant for the turbine |
|  | Time-constant for the governor |

The controller gains are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | Gain for the mechanical torque |
|  | Gain for the valve position |
|  | Proportional gain for governor controller |
|  | Integral gain for governor controller |
|  | Gain for exciter controller |

The controller set points are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | Reference speed for governor controller |
|  | Terminal reference voltage for exciter controller |

ClassName: SynchronousMachine

Description: Synchronous machine without governor control or exciter control

The state variables for this object are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the stator current |
|  | quadrature component of the stator current |
|  | rotor current |
|  | machine angular speed |
|  | machine angular position |

The port inputs determined by this object’s connection to the rest of the grid are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the stator voltage |
|  | quadrature component of the stator voltage |

The parameters:

|  |  |
| --- | --- |
| Symbol | Description |
| LR | Self-inductance of the rotor winding |
| LS | Self-inductance of the stator windings |
| LSS | Mutual inductance between the stator windings |
| M | Mutual inductance between the stator and rotor windings when parallel |
| RR | Resistance of the rotor winding |
| RS | Resistance of the stator windings |
| J | Inertia of the rotor |
| B | Damping coefficient of the rotor |
| VR | Voltage applied to the rotor winding |
|  | Load (mechanical) torque |

ClassName: InductionMachine

Description: Induction machine with no control

The state variables for this object are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the stator current |
|  | quadrature component of the stator current |
|  | direct component of the rotor current |
|  | quadrature component of the rotor current |
|  | machine angular speed |
|  | machine angular position |

The port inputs determined by this object’s connection to the rest of the grid are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the stator voltage |
|  | quadrature component of the stator voltage |

The parameters for this object:

|  |  |
| --- | --- |
| Symbol | Description |
| LR | Self-inductance of the rotor winding |
| LRR | Mutual inductance between the rotor windings |
| LS | Self-inductance of the stator windings |
| LSS | Mutual inductance between the stator windings |
| M | Mutual inductance between the stator and rotor windings when parallel |
| RR | Resistance of the rotor windings |
| RS | Resistance of the stator windings |
| J | Inertia of the rotor |
| B | Damping coefficient of the rotor |
|  | Load (mechanical) torque |

Technically, the mechanical torque is an “exogenous input” because it can vary as a function of time unlike a “parameter.” However, for the purposes of symbolically deriving the interconnected state space model, there is no need to distinguish between “exogenous inputs” and “parameters”

ClassName: TransmissionLine

Description: Transmission line using the pi model with shunt capacitances

The state variables for this object are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the transmission line left shunt capacitor charge |
|  | quadrature component of the transmission line left shunt capacitor charge |
|  | direct component of the transmission line inductor current |
|  | quadrature component of the transmission line inductor current |
|  | direct component of the transmission line right shunt capacitor charge |
|  | quadrature component of the transmission line right shunt capacitor charge |

The port inputs determined by this object’s connection to the rest of the grid are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the current entering left side of transmission line |
|  | quadrature component of the current entering left side of transmission line |
|  | direct component of the current entering right side of transmission line |
|  | quadrature component of the current entering right side of transmission line |

The parameters are:

|  |  |
| --- | --- |
| Symbol | Description |
| RTL | Resistance of the transmission line |
| CTL | Shunt capacitance of the transmission line |
| LTL | Self-inductance of the transmission line |

ClassName: Load

Description: Load with a constant resistance and constant inductance

The state variables for this object are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the load inductor current |
|  | quadrature component of the load inductor current |

The port inputs determined by this object’s connection to the rest of the grid are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the voltage applied to load |
|  | quadrature component of the voltage applied to load |

Parameters:

|  |  |
| --- | --- |
| Symbol | Description |
| RL | Resistance of the load |
| LL | Self-inductance of the load |

ClassName: PE\_Flywheel

Description: The Flywheel energy storage system consists of a synchronous machine controlled through switches in the power electronics (AC/DC/AC converter), which interface between the machine and the rest of the system. The power electronic current set points are chosen so that, in closed-loop, the aggregate of the power electronics and all the wind generators on the bus behaves as a constant current source.

The state variables are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | Initial condition of direct component of inductor current |
|  | Initial condition of quadrature component of inductor current |
|  | Initial condtion of dc-bus capacitor charge |
|  | direct component of the stator current |
|  | quadrature component of the stator current |
|  | rotor current |
|  | machine angular speed |
|  | machine angular position |
|  | integral of difference between the speed and the speed set point |
|  | the flywheel reference speed |
|  | the desired dc-bus capacitor charge |
|  | the desired direct component of the stator current |

The port inputs determined by this object’s connection to the rest of the grid are:

|  |  |
| --- | --- |
| Symbol | Description |
|  | direct component of the voltage applied to power electronics |
|  | quadrature component of the voltage applied to power electronics |

The parameters are:

|  |  |
| --- | --- |
| Symbol | Description |
| C | Capacitance of dc-bus capacitor in power electronics |
| L | Inductance of power electronics inductor in power electronics |
| R | Resistance in series with inductance of power electronics |
| RC | Resistance of resistor in parallel with capacitor in power electronics |
| LR | Self-inductance of the rotor winding in the machine |
| LS | Self-inductance of the stator windings in the machine |
| LSS | Mutual inductance between the stator windings in the machine |
| M | Mutual inductance between the stator and rotor windings when parallel |
| RR | Resistance of the rotor winding in the machine |
| RS | Resistance of the stator windings in the machine |
| J | Inertia of the machine |
|  | Load (mechanical) torque applied to the machine |
| B | Damping coefficient of the machine |
| VR | Voltage applied to the rotor winding of the machine |

The controller gains are

|  |  |
| --- | --- |
| Symbol | Description |
| *KP* | Proportional gain in the speed controller |
| *KI* | Integral gain in the speed controller |
| *KS* | Gain for the flywheel speed set point dynamics |
| *ρ* | Percent of maximum reference speed that is desired to obtain |

The controller set points:

|  |  |
| --- | --- |
| Symbol | Description |
|  | Set point for the direct component of current entering the power electronics |
|  | Set point for the quadrature component of current entering the power electronics |

It should be noted that the flywheel speed is not an independent set point, because as described in my thesis, it is made a state variable with dynamics that depend on the power electronic current set points.

The class definition for the load (which is probably the simplest object) is shown below.

classdef Load < Module

methods

function this = Load(IndexName,RefFrameAngle,RefFrameSpeed)

ParameterNames={'RL','LL'};

StateVariableNames = {'iLd','iLq'};

PortInputNames = {'vLd','vLq'};

PortStateNames = {'iLd','iLq'};

ControllableInputNames = {};

this.RefFrameAngle = RefFrameAngle;

this.RefFrameSpeed = RefFrameSpeed;

this.Parameters = sym(sym(strcat(ParameterNames,IndexName)),'real');

this.StateVariables = sym(sym(strcat(StateVariableNames,IndexName)),'real');

this.ControllableInputs = sym(sym(strcat(ControllableInputNames,IndexName)),'real');

this.PortInputs = sym(sym(strcat(PortInputNames,IndexName)),'real');

this.PortStates = sym(sym(strcat(PortStateNames,IndexName)),'real');

this.PortVoltages = this.PortInputs;

this.PortCurrents = this.PortStates;

this.StateVariableDerivatives = sym(sym(strcat('d',StateVariableNames,IndexName,'dt')),'real');

this.PortStateDerivatives = sym(sym(strcat('d',PortStateNames,IndexName,'dt')),'real');

this.PortStates\_Time = sym(sym(strcat(PortStateNames,IndexName,'\_t(t)')));

this.PortStateTypes = {'Current','Current'};

this.StateSpace = Load.LoadDynamics(this.Parameters,this.StateVariables,this.ControllableInputs,...

this.PortInputs,this.RefFrameAngle,this.RefFrameSpeed);

end

end

methods(Static)

function StateSpace = LoadDynamics(Parameters,StateVariables,ControllableInputs,PortInputs,phi,dphidt)

%Inputs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

%StateVariables: iLd,iLq

%InputVariables: vLd,vLq

%Parameters: RL, LL

%phi: angle of rotating reference frame

%dphidt: speed of rotating reference frame

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

%Outputs\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

%StateSpace = [ diLddt ; diLqdt];

%\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

iLd = StateVariables(1);

iLq = StateVariables(2);

vLd = PortInputs(1);

vLq = PortInputs(2);

RL = Parameters(1);

LL = Parameters(2);

%Transmission Line Dynamics

diLddt = dphidt\*iLq + (vLd - RL\*iLd)/LL;

diLqdt = (vLq - RL\*iLq)/LL - dphidt\*iLd;

StateSpace = [ diLddt ; diLqdt];

end

end

end