Procedure for the Simulation of NOx Emissions Produced by Combustion Turbines

SIM-00001-Nox

Revision: 1

Status: Issued

July 2016

CONFIDENTIAL

Proprietary Class 2

CHANGE HISTORY

SimGenics-Simulation LLC
Suite B210, 25½ Road,Grand Junction, Colorado, 81505,United States of America
Tel: +1 (970) 639 2498, Fax:+1 (817) 736 0427
e-mail:mgregg@simgenics.com or flaubscher@simgenics.com

Configuration Control	
Created by:	Franli Laubscher
Original creation Date:	01 July 2016

Revision History

Revision	Date	Author	Description of Changes
A	12-July-2016	Franli Laubscher	Draft issue
B	12-July-2016	Francois Laubscher	Review of Document
1	20-July-2016	Franli Laubscher	Updates

Authorization

Action	Designate / Signature	Position	Date
Prepared	__________________		
Reviewer	__________________		
Approver	__________________	Management Representative	

Original Signed Copy at Configuration Management Centre (CMC)

Contents

Figures

Tables

Abbreviations and Acronyms

Abbreviations	Description
NOx	Nitrogen Oxides
SCR	Selective Catalytic Reduction
Ω	Omega
ppm	Parts per Million

1. PURPOSE

This document describes the formation and simulation of NOx emissions produced by combustion turbines, using the SimuPact Flame Model.

2. SCOPE

NOx emissions are formed by the combustion of nitrogen and oxygen in furnace flames, and rapidly increase when the flames reach temperatures above 1800°F [4]. These emissions are a big cause for concern as they can combine with other pollutants in the atmosphere to form O_3, or ground level ozone, as well as nitric acid, which is harmful to the environment.

In order to limit the amount of emission gases discharged, aqueous ammonia is diluted which, through a process called selective catalytic reduction (SCR), will break down the NO molecules to form by-products such as water and nitrogen gases.

3. REFERENCES

Reference	Title	Rev/Date
[1] NOx and SOx	Chemistry of NOx Formation	03/14/2016
[2] NRGLB	Long Beach Open Cycle Combustion Turbine Simulation	SAT Version
[3] NOx and SOx	NOx – Sources and Control Methods	NOx.ppt
[4] Alentecinc	The formation of NOx	
[5] Wikipedia	Ammonium Hydroxide	07/11/2016
[6] Wikipedia	Selective Catalytic Reduction	07/01/2016
[7] Wikipedia	Parts-per Notation	06/05/2016

4. DEFINITIONS, TERMS, ACRONYMS AND ABBREVIATIONS

Thermal NOx (nitrogen oxides) — A generic term used to describe mostly NO (nitric oxide) and NO_2 (nitrogen dioxide) that form when nitrogen and oxygen combusts at high temperatures. NO contributes to approximately 90-95% of the NOx emissions released into the atmosphere.

Fogger — A device that injects water vapor into the inlet air mixture, thus lowering the flame temperature and decreasing the NOx emissions.

Aqueous Ammonia	A solution of ammonia in water [5]. The fraction of ammonia needed for the reaction will depend on the efficiency of the catalyst bed.
Selective Catalytic Reduction (SCR)	A method of converting the NO emissions, with the support of a catalyst bed and the aqueous ammonia, into diatomic nitrogen (N_2) and water (H_2O) [6].
Ammonia Slip	The unreacted ammonia that passes through the SCR bed [2].
Omega (Ω)	The ratio of the mass flow of the water injection flowing into the flame over the mass flow of the gas supply entering the flame [2].
Parts per million (ppm)	This is a unit that describes the concentration of small values with dimensionless quantities, e.g. mole fractions or mass fractions [7].

5. PROCEDURE

The following procedures can be followed after the entire flow diagram is setup with all the nodes and boundary conditions defined. To ensure the operations explained below will be successful, the first step is to confirm that the diagram is able to solve and produce answers in steady state.

5.1 ADDING THE AMMONIA

Depending on the quality of the inlet air, the injected water, and the supplied gas, the top five gases present in the mixture after combustion will be varying amounts of CO_2, H_2O, N_2, NO, and O_2. Although ammonia (NH_3) is not a product of the combustion process, this compound will have to be added to the list of mixed gases in order for one to be able to change the mass fraction on another node in the future. To add NH_3 to the list, complete the following steps:

- Under *Charts/Lookup Tables,* expand the *Flow Solver* and choose *Materials and Fluids.* For the purpose of this example, *Mixed Fluids* were used. Expand *Des Moines* and double click on *Mixed Gases*.
- A window, as shown in **Figure 5-1** on the next page, will appear. Under the *Pure Gas Mixture* tab, double click on *Matheson – NH_3,* and it will shift over to the window on the right. Click OK. Ammonia has now been added to all the nodes used in your diagram.

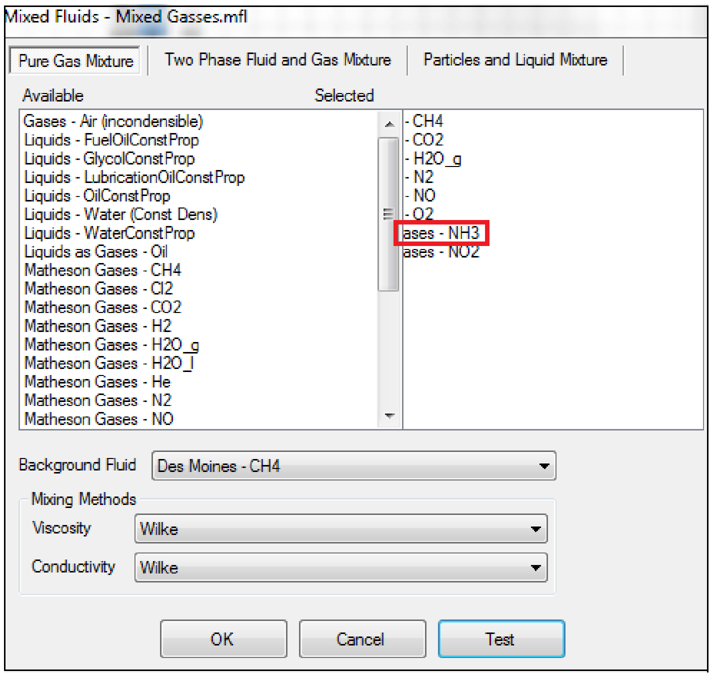

Figure 5-1: Mixed Fluids Window

5.2 CHANGING THE REACTION RATES FOR CORRECT NOX EMISSIONS IN THE FLAME

In the flame model, there are four major chemical reactions that occur when combustion takes place. The quantity of the products formed after each reaction will vary according to the amounts that enter the combustion chamber. In order for the correct amount of NOx emissions to be released, as specified by the plant data, the *reaction rate value* together with the *reaction rate factor* need to be adjusted accordingly.

5.2.1 ADJUSTING THE REACTION RATE VALUE

- Click on the flame model and scroll down to *Reaction Chain* (as circled in red). Below the reaction chain data, there will be four balanced chemical reactions. Scroll down until the reaction (also circled in red) as displayed in **Figure 5-2** on the next page, is found.

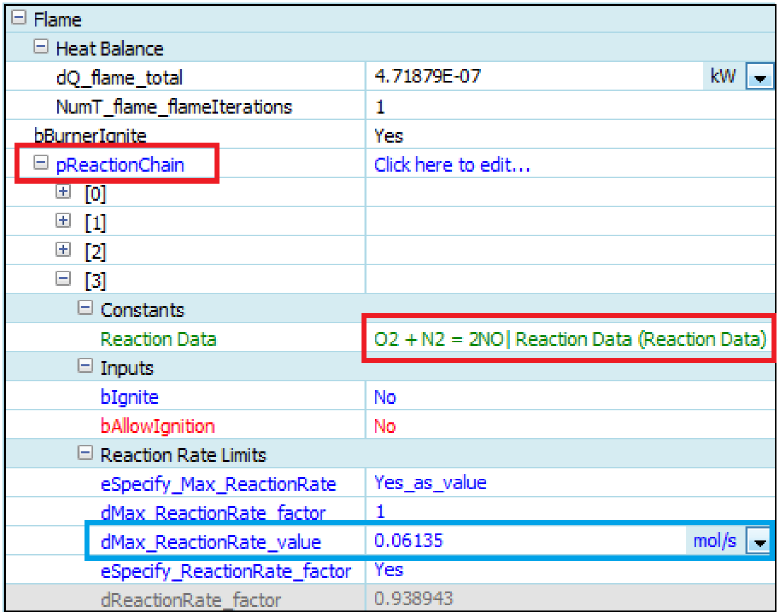

Figure 5-2: NOx Reaction in Flame Model

- As mentioned before, both the chemical reaction's *reaction rate value* and *reaction rate factor* will have to be adjusted. The reaction rate value will limit the amount of NO moles that are produced every second. As seen in **Figure 5-2** above, the value (circled in blue) was changed until there were no more than 13.5 ppm NOx emissions present at the turbine exit.

5.2.2 ADJUSTING THE REACTION RATE FACTOR

The flame's temperature will increase if either the fogger, or the water injection, or both are to be switched off. It will be very difficult to change the *reaction rate value* manually every time there is a fluctuation in the temperature. Therefore, the *reaction rate factor* will act as a multiplier to the *reaction rate value,* and will be changed by means of a function generator. Follow the steps below to pair the correct *reaction rate factor* with the flame.

- Insert a function generator, from the *DCS Library*.
- Using a *Data Transfer Link*, connect the temperature of the flame to the input of the function generator.
- The function generator will interpolate linearly between the number of X-values and Y-values that are inserted. On the function generator listed below, **Figure 5-3**, there are six points inserted. The X-values correspond to the flame temperature (°C is used in this example), and the Y-values correspond to the *reaction rate factor*. The X-values should be inserted according to what the flame temperature is, if the fogger and water injection are switched on or shut off respectively. The corresponding Y-values should be changed in order to reflect what the plant data indicates the NOx emissions should be at these various temperatures.

General	
Identifier	CD-Function Generator-1982
Solving	☑
Description	
Connectable Outputs	
Output	0.938943
Slope	0.00167188
Configuration	
Low Limit	0.4
High Limit	7
Limit to Range	Yes
Filter Time	5
Execution Number	0
Inputs	
Number of Points	6
X values	
[0]	500
[1]	700
[2]	952
[3]	984
[4]	1120
[5]	1151
Y values	
[0]	0.443
[1]	0.621
[2]	0.9386
[3]	0.9921
[4]	5.7562
[5]	6.2663
Connectable Inputs	
Input	952.205
Disable	0

Figure 5-3: Function Generator for the Reaction Rate Factor

- Set the high and low limits so that the Y-values fall in between these limitations (as depicted in blue in **Figure 5-3** above).
- Use a *Data Transfer Link* to connect output of the function generator to the *reaction rate factor* of the flame.
- To see how the quantity of the products change as the *reaction rate value* and the *reaction rate factor* change, the mass fractions at the turbine exit can be examined. **Figure 5-4** below, indicates that the mass fraction of NO can be converted to ppm if multiplied by 1 x 10^6 in order to equal 13.5 ppm.

Mass Fraction	
Mass fractions	
CH4	0
CO2	0.0542063
H2O_g	0.14427
N2	0.680848
NO	1.35008E-05
O2	0.120662
NH3	0
NO2	0

Figure 5-4: NO Mass Fraction at Turbine Exit

5.3 FUNCTION GENERATOR OF THE FOGGER

The fogger's influence on the temperature of the flame is relatively small, but it is still an important component that must be included in the design. In order to maximize the fogger's influence follow the steps listed below:

- Insert a function generator, from the *DCS Library*.
- Using a *Data Transfer Link*, connect the opening of the valve to the input of the function generator as shown in **Figure 5-5**.
- In this instance, there are 3 points inserted on the function generator. The X-values will correspond to the opening of the valve and the Y-values will correspond to the amount of heat lost (AU) in the node. The air inlet entering the node is 23.5°C, which the fogger has to reduce to 17.5°C if the valve is fully open. Insert different Y-values until this desired temperature is reached at the node. See **Figure 5-6**.

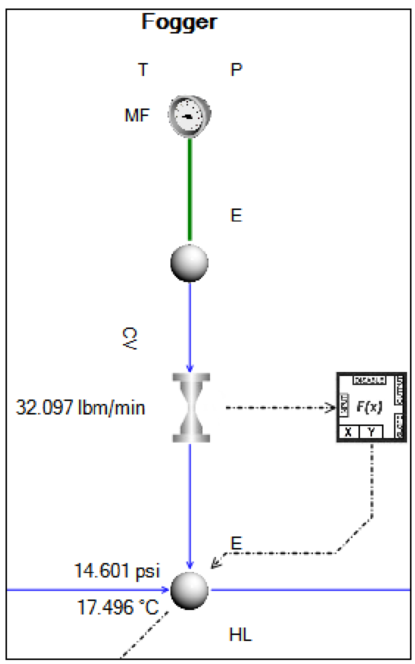

Figure 5-5: Fogger Illustration

General	
Identifier	01-NOX-Function Generator - 33
Solving	☑
Description	
Connectable Outputs	
Output	116
Slope	132
Configuration	
Low Limit	0
High Limit	150
Limit to Range	No
Filter Time	5
Execution Number	0
Inputs	
Number of Points	3
X values	
[0]	0
[1]	0.5
[2]	1
Y values	
[0]	0
[1]	50
[2]	116
Connectable Inputs	
Input	1
Disable	0

Figure 5-6: Function Generator of the Fogger

- Remember to set the high and low limits for the Y-values to ensure the function generator will operate optimally. The limits are shown in blue in **Figure 5-6** above.
- The ambient temperature on the node is also added here as 40°F to ensure that there will be a heat loss occurring to the atmosphere.
- Using a *Data Transfer Link,* connect the output of the function generator to the heat leak (AU) of the node as shown in **Figure 5-5** on the previous page.

5.4 THE CHEMICAL REACTION USING AMMONIA TO REDUCE NITROGEN OXIDES

The chemical reaction, as presented in **Figure 5-7**, involving the nitrogen oxides and diluted aqueous ammonia will be illustrated in a separate script (see Appendix for the completed code) where the final mass fractions will be calculated and transferred to a node representative of the SCR bed.

$$4NO + 4NH_3 + O_2 \xrightarrow{\mathrm{TiO_2\ or\ V_2O_5\ supported\ catalyst}} 4N_2 + 6H_2O$$

Figure 5-7: The Balanced Chemical Reaction to Reduce NOx Emissions [3]

The following instructions will be a guide to calculating the correct NOx emissions and Ammonia Slip.

- Insert a Script under the *Scripting* tab in *Components*.
- Create input properties for each of the mass fractions coming from the turbine exit node and the ammonia injection node respectively. Use the *Data Transfer Link* to connect the various mass fractions to these properties. See **Figure 5-8** below.

- Create two more input properties, also shown in **Figure 5-8**, to account for the mass flow rate at the exit of the turbine and the mass flow rate of the diluted ammonia. Use the *Data Transfer Tool* again to connect these properties. REMEMBER: To ensure all calculations are accurate, check that the units of the mass flow rates are in **kg/s** before connecting to the input properties.

⊟ Inputs (Mass Fractions)	
Mass_Fraction_CO2_Turbine	0.0542063
Mass_Fraction_H2Og_Turbine	0.14427
Mass_Fraction_N2_Turbine	0.680848
Mass_Fraction_NO_Turbine	1.35008E-05
Mass_Fraction_O2_Turbine	0.120662
Mass_Fraction_NH3_Turbine	0
Mass_Fraction_CO2_Ammonia	0.0510036
Mass_Fraction_H2Og_Ammonia	0.183604
Mass_Fraction_N2_Ammonia	0.640621
Mass_Fraction_NO_Ammonia	1.27031E-05
Mass_Fraction_O2_Ammonia	0.113533
Mass_Fraction_NH3_Ammonia	0.011226
⊟ Inputs [kg/s]	
MassFlow_Turbine	246.813
MassFlow_Ammonia	0.269239

Figure 5-8: Mass Fractions from Turbine Exit and Ammonia Injection

- Compute the mole rates [**mol/s**] for each individual gas present at the exit of the turbine and for the gases present in the diluted ammonia. To compute the mole rate, the mass fraction is multiplied by the mass flow rate, then divided by the molar mass of each gas. In order for the units to cancel out, it must be multiplied by a 1000. An example of the calculated mole rate, as it is written in the script, is expressed in **Figure 5-9** below.

```
//Mole Rate at Turbine Exit                    n = [mol/s]
n_CO2_T = (Mass_Fraction_CO2_T*MassFlow_Turb*1000.0)/(44.01);

//Mole Rate at Ammonia Injection
n_CO2_A = (Mass_Fraction_CO2_A*MassFlow_Amm*1000.0)/(44.01);
```

Figure 5-9: Mole Rate for Turbine Exit and Ammonia Injection

- It will now be necessary to compute the mole rate [**mol/s**] of the reaction. Since the ammonia is diluted, it will act as the limiting reactant. For every mole of ammonia used, products will form in relation to what their coefficient, in the balanced chemical equation, is. Refer back to **Figure 5-7** as a guide to the formation of products and dissolution of reactants. **Figure 5-10** shows the reaction for every gas present in the chemical equation.

```
//Mole Rate of Reaction                                n = [mol/s]
//4NO + 4NH3 + O2 ---> 4N2 + 6H2O
n_CO2_R = 0.0;
n_H2Og_R = (6.0/4.0)*n_NH3_Turbine + (6.0/4.0)*(n_NH3_Ammonia);
n_N2_R = n_NH3_Turbine + (n_NH3_Ammonia);
n_NO_R = n_NH3_Turbine + (n_NH3_Ammonia);
n_O2_R = (1.0/4.0)*(n_NH3_Turbine) + (1.0/4.0)*(n_NH3_Ammonia);
n_NH3_R = n_NH3_Turbine + (n_NH3_Ammonia);
```

Figure 5-10: Mole Rate of the Reaction

- It is important to notice that the ammonia will not react at 100% efficiency. There will be ammonia exiting the SCR bed, which is defined as the ammonia slip. Therefore, add the efficiency as an input property, and multiply it to the mole rate of the reaction as seen in **Figure 5-11**. The value for the efficiency can now be changed until the desired quantity of NOx emissions are produced.

```
//Mole Rate of Reaction                              n = [mol/s]
//4NO + 4NH3 + O2 ---> 4N2 + 6H2O
Ammonia_Fraction = Ammonia_Efficiency/100.0;
n_CO2_R = 0.0;
n_H2Og_R = (6.0/4.0)*n_NH3_Turbine + (6.0/4.0)*(n_NH3_Ammonia*Ammonia_Fraction);
n_N2_R = n_NH3_Turbine + (n_NH3_Ammonia*Ammonia_Fraction);
n_NO_R = n_NH3_Turbine + (n_NH3_Ammonia*Ammonia_Fraction);
n_O2_R = (1.0/4.0)*(n_NH3_Turbine) + (1.0/4.0)*(n_NH3_Ammonia*Ammonia_Fraction);
n_NH3_R = n_NH3_Turbine + (n_NH3_Ammonia*Ammonia_Fraction);
```

Figure 5-11: Added Efficiency to the Reaction Mole Rate

- The total mole rate [**mol/s**] can now be computed by adding the mole rate for each gas originating from the turbine exit and the mole rate for each gas originating from the ammonia injection. The mole rate of the reaction will be added or subtracted, depending whether products have been formed, or reactants have been used. The equation for each gas is demonstrated in **Figure 5-12** below.

```
//Total Mole Rate from Reaction                        [mol/s]
n_CO2_Total = n_CO2_Turbine + n_CO2_Ammonia + n_CO2_R;
n_H2Og_Total = n_H2Og_Turbine + n_H2Og_Ammonia + n_H2Og_R;
n_N2_Total = n_N2_Turbine + n_N2_Ammonia + n_N2_R;
n_NO_Total = n_NO_Turbine + n_NO_Ammonia - n_NO_R;
n_O2_Total = n_O2_Turbine + n_O2_Ammonia - n_O2_R;
n_NH3_Total = n_NH3_Turbine + n_NH3_Ammonia - n_NH3_R;
```

Figure 5-12: Total Mole Rate

- The total mole rate [**mol/s**] for every gas, now needs to be converted back to mass flow rate [**kg/s**] in order to be displayed as a mass fraction at the SCR bed. This is achieved by multiplying each mole rate with the gases' individual molar mass, and then dividing by a 1000 so that the units will be correct. The conversion from mole rate to mass flow rate is shown in **Figure 5-13**.

```
MassFlow_CO2 = (n_CO2_Total*44.01)/1000.0;
MassFlow_H2Og = (n_H2Og_Total*18.01528)/1000.0;
MassFlow_N2 = (n_N2_Total*28.0134)/1000.0;
MassFlow_NO = (n_NO_Total*30)/1000.0;
MassFlow_O2 = (n_O2_Total*31.998)/1000.0;
MassFlow_NH3 = (n_NH3_Total*17.03052)/1000.0;
```

Figure 5-13: Conversion from Mole Rate to Mass Flow Rate

- Compute the total mass flow rate [**kg/s**] by adding the mass flow rate at the exit of the turbine and the mass flow rate of the diluted ammonia together.
- The final mass fractions can now be calculated by taking the individual mass flow rate of every gas and dividing it by the total mass flow rate. Examine **Figure 5-14** on the next page to see the

calculations. REMEMBER: To prevent the denominator from being undefined when the total mass flow rate is zero, add a very small number (e.g. 1 x 10^{-6}) to the total mass flow rate before dividing.

```
Total_MassFlow = MassFlow_Turbine + MassFlow_Ammonia;
mf_CO2_out = MassFlow_CO2/(Total_MassFlow +0.0000001);  //mf = Mass_Fraction
mf_N2_out = MassFlow_N2/(Total_MassFlow +0.0000001);
mf_NO_out = MassFlow_NO/(Total_MassFlow +0.0000001);
mf_O2_out = MassFlow_O2/(Total_MassFlow +0.0000001);
mf_NH3_out = MassFlow_NH3/(Total_MassFlow +0.0000001);
mf_H2Og_out = 1 - mf_CO2_out- mf_N2_out- mf_NO_out- mf_O2_out- mf_NH3_out;
```

Figure 5-14: Final Mass Fractions

- Before the mass fractions can be linked to the SCR bed, check that they add up to one (1). See **Figure 5-15** below.

```
Total_Mass_Frac = mf_CO2_out + mf_N2_out + mf_NO_out + mf_O2_out + mf_NH3_out + mf_H2Og_out;
```

Figure 5-15: Total Mass Fraction

- Finally, if the total mass fraction adds up to one, create output properties in the script in order to display the individual mass fractions for each gas to the outside world.
- Link the individual mass fractions for each gas to the boundary condition of the SCR node using the *Data Transfer Link*. The quantity of the NOx emissions can now vary if the efficiency at which the ammonia reacts is changed.
- Calculate the NOx emissions and ammonia slip, in ppm, by simply multiplying the NO and NH_3 mass fractions at the SCR bed by 1 x 10^6 respectively, as done in **Figure 5-16**.

```
//Calculations of parts per million    [ppm]
NOx_Amm = mf_NO_out*(1000000.0);
Amm_Slip = mf_NH3_out*(1000000.0);
```

Figure 5-16: NOx and Ammonia Slip ppm Calculations

5.5 CALCULATING OMEGA (Ω)

Omega is calculated as the ratio of the mass flow rate of the water injection flowing into the flame over the mass flow rate of the gas supply entering the flame. The following steps will require more data to be added to the existing script so that only a small calculation is needed to determine Omega.

- Create two new input properties in the script and connect them, using the *Data Transfer Link*, to the mass flow rate of the water injection and to the mass flow rate of the gas supply. REMEMBER: Since Omega is a ratio, the units need to cancel out. Thus be sure to check that both properties are in the same unit. An example of the input properties are shown in **Figure 5-17**.

MassFlow_H2O_Injection	9.07983
MassFlow_Gas_Supply	5.17637

Figure 5-17: Water Injection and Gas Supply Mass Flow Rates

- Calculate Omega by dividing the mass flow rate of the water injection by the mass flow rate of the gas supply as in **Figure 5-18**. Remember to add the small value to the denominator in case the mass flow rate of the gas supply should equal zero.

```
//Calculation of Omega
//Omega = Water Injection Mass Flow / Gas Supply Mass Flow
Omega = MassFlow_H2O / (MassFlow_Gas_Supply +0.0000001);
```

Figure 5-18: Calculation of Omega

- Lastly, create an output property in the script in order to make Omega visible to the outside world.

6. VERIFICATION AND VALIDATION

To ensure that this new method for determining the NOx emissions, as described in the previous sections, is accurate enough to use on simulators in the future, it has been compared to the NRG Long Beach simulator. The Long Beach simulator makes use of empirical methods to determine the NOx emissions, as opposed to this new method that uses the gases' mass fractions and flow rates. An analysis of the emissions were done at startup, shutdown, and while the simulators were running at steady state.

6.1 TEST ANALYSIS AT STEADY STATE

At steady state (65 MW), the Long Beach simulator will produce the lowest quantity of NOx emissions. At the turbine exit, the emissions should equal to 13.5 ppm. When ammonia has been added to the system, the NOx emissions will reduce to 2.3 ppm. The following table, **Table 6-1**, compares the influence of the foggers, and the influence of the ammonia injection, on the presence of the NOx emissions. The water injection will always be switched on.

Table 6-1: Steady State NOx Emissions Analysis

	NRG Long Beach	New Method
Fogger ON Ammonia ON	2.3 ppm	2.299 ppm
Fogger ON Ammonia OFF	13.5 ppm	13.5 ppm
Fogger OFF Ammonia OFF	14.7 ppm	14.35 ppm
Fogger OFF Ammonia ON	2.4 ppm	2.91 ppm

	NRG Long Beach	New Method
Omega	1.76	1.75
Ammonia Slip with Fogger ON	$\boxed{0.1 \text{ ppm}}$	5.874 ppm

It can be observed from **Table 6-1,** that the new method for determining the NOx emissions, is very close to the values produced on the Long Beach simulator. Thus, it is possible that the new method can be considered for determining NOx emissions on other simulators. The ammonia slip is the only property that shows a significant difference between the two methods. The power plant data [2] indicates that the ammonia slip needs to be around 5 ppm, when the NOx emissions are 2 ppm. **It was noticed that the slip never changed on the Long Beach simulator, which proves that this value is incorrect**. Thus, it can be concluded that the ammonia slip determined from the new method would provide the more accurate number.

6.2 TEST ANALYSIS AT STARTUP

When starting up a simulator, it is expected that the NOx emissions will have an increase in concentration, and then gradually reduce as steady state is reached. A study of the startup was done on both simulators in order to compare the accuracy of the new method to the values of the Long Beach simulator. **Figure 6-1** and **Figure 6-2** below indicates the differences and similarities in the two methods.

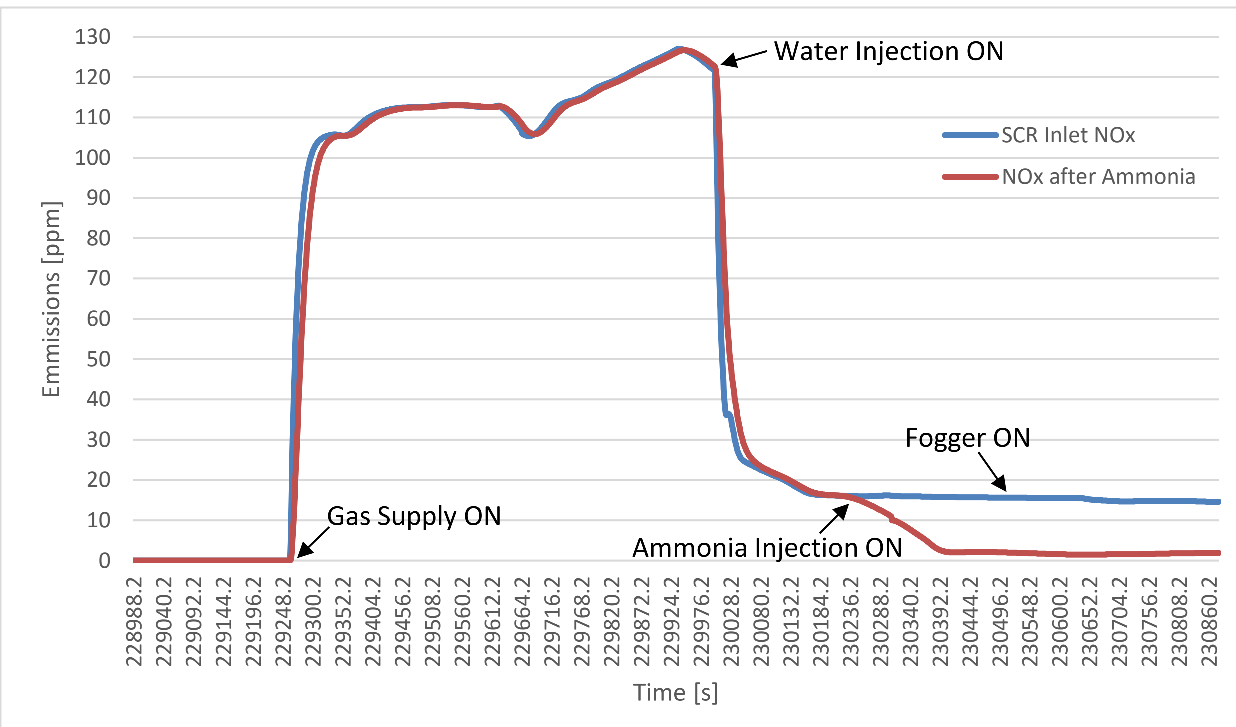

Figure 6-1: NOx Emissions of the Long Beach Simulator at Startup

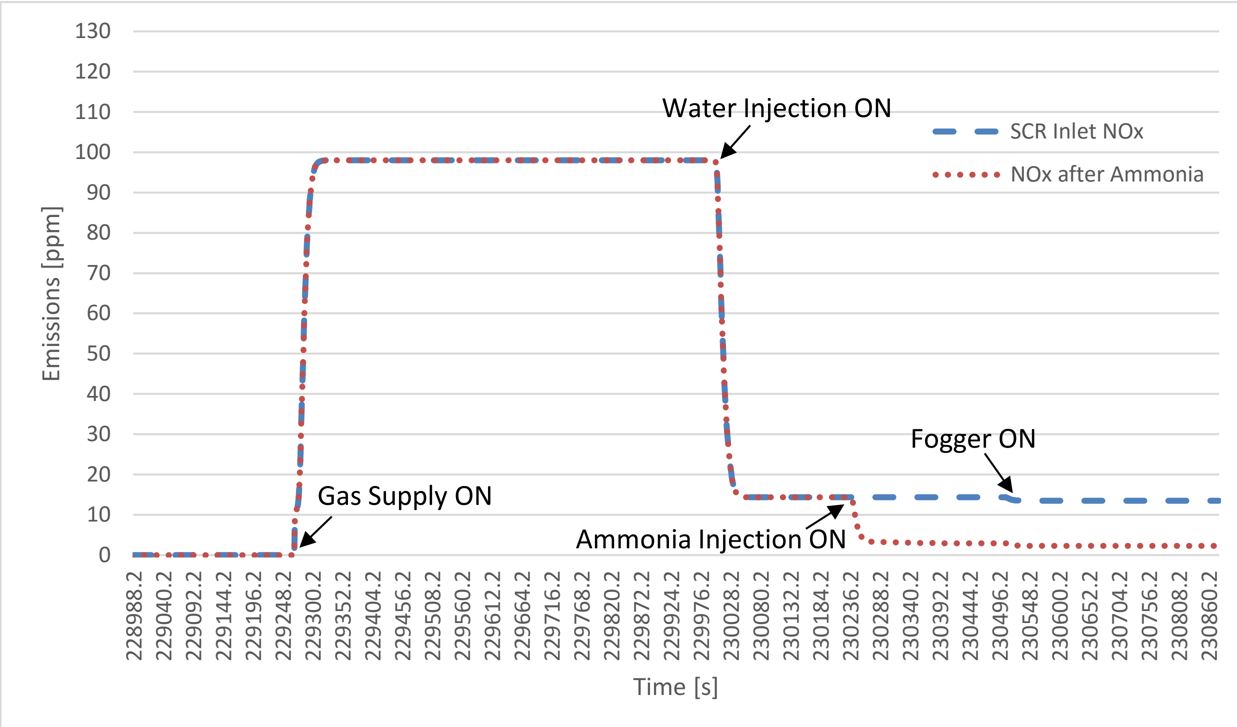

Figure 6-2: NOx Emissions of the Test Simulator at Startup

NOTE:

Sensor Dynamics was not included in the Test Simulator.

From the figures above, it can be observed that the NOx emissions of the test simulator do not reach the same maximum concentration than that of the NOx emissions at the Long Beach simulator. At the Long Beach simulator, the emissions are much higher because more gas is initially injected than is needed, in order to get the shaft speed up to 3600 rpm. The test simulator was tested at constant shaft speed and constant gas supply. Had the test simulator been tested at varying shaft speeds and varying gas supply, the results would have been similar to the values obtained at Long Beach. This is true because the final concentrations of the NOx emissions, as the simulator approached steady state, proved to be exactly the same.

As mentioned before, Omega fluctuates when the gas supply - and water injection mass flow rates' fluctuate. From **Figure 6-3**, on the next page, it can be seen that the Long Beach simulator's Omega value, increased and decreased significantly before it reached a stable value. This oscillation in the ratio occurred because the simulator attempted to increase the shaft speed up to 3600 rpm as fast as possible, which caused water injection and gas supply to be uncontrolled. There was no fluctuation in the test simulator's Omega, since the gas supply and water injection reached a constant value once the valves were opened. Apart from the oscillation, both simulators reached the same ratio as soon as the shaft speed reached a constant state.

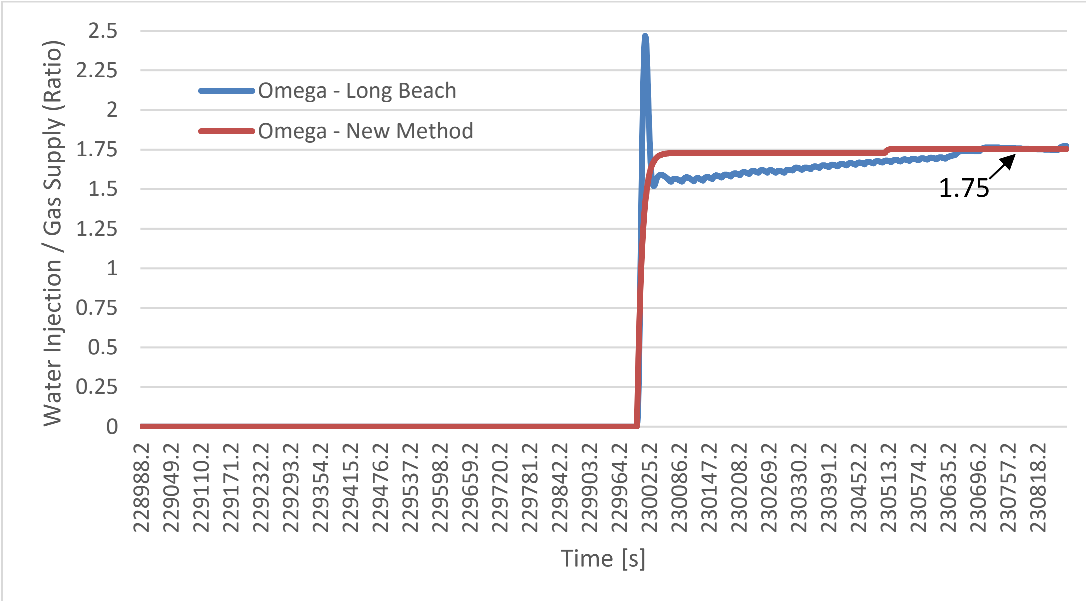

Figure 6-3: Omega Value for Long Beach - and the Test Simulator at Startup (Sensor Dynamics not Included)

It make sense for the ammonia slip to be present only when there is ammonia being injected into the system. From **Figure 6-4** below, it can be observed that the ammonia slip at Long Beach stays at a constant rate of 0.105 ppm throughout the entire startup, even when there is no ammonia present. This is incorrect. As mentioned before, the ammonia slip needs to be around 5 ppm when the simulator is at steady state, which close to the value obtained by the test simulator.

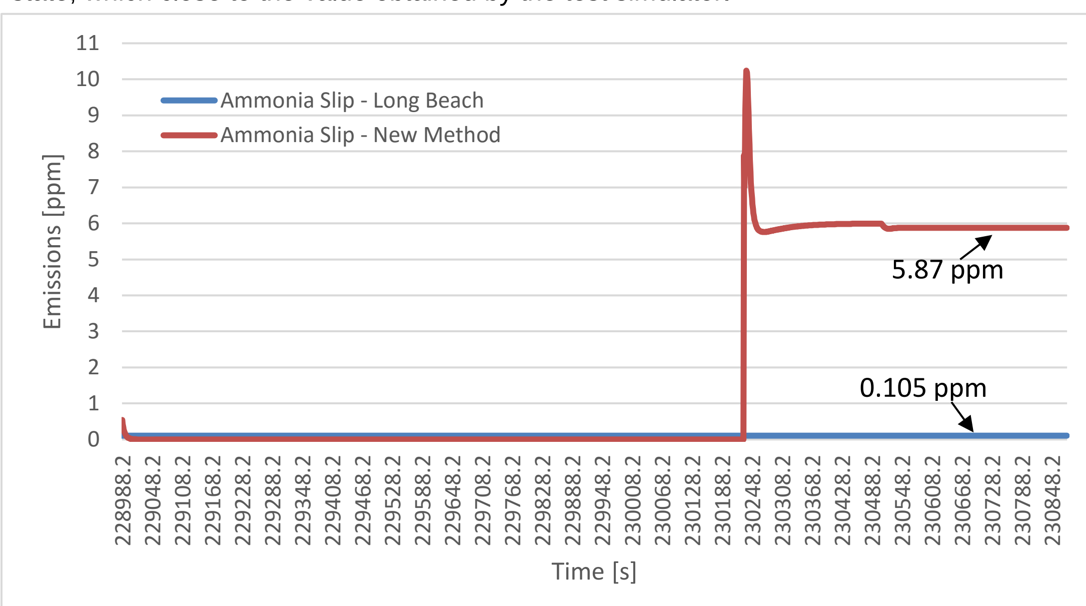

Figure 6-4: Ammonia Slip for Long Beach - and the Test Simulator at Startup

6.3 TEST ANALYSIS AT SHUTDOWN

The NOx emissions released by the simulators, ranging from steady state to shutdown, can be examined in this section. The emissions will again increase in concentration as the ammonia injection, foggers, and water injection are switched off, but will ultimately decrease to zero once there is no more gas supplied to the combustion chamber. Look at **Figure 6-5** below, and **Figure 6-6** on the next page to compare the results between the Long Beach simulator and the test simulator.

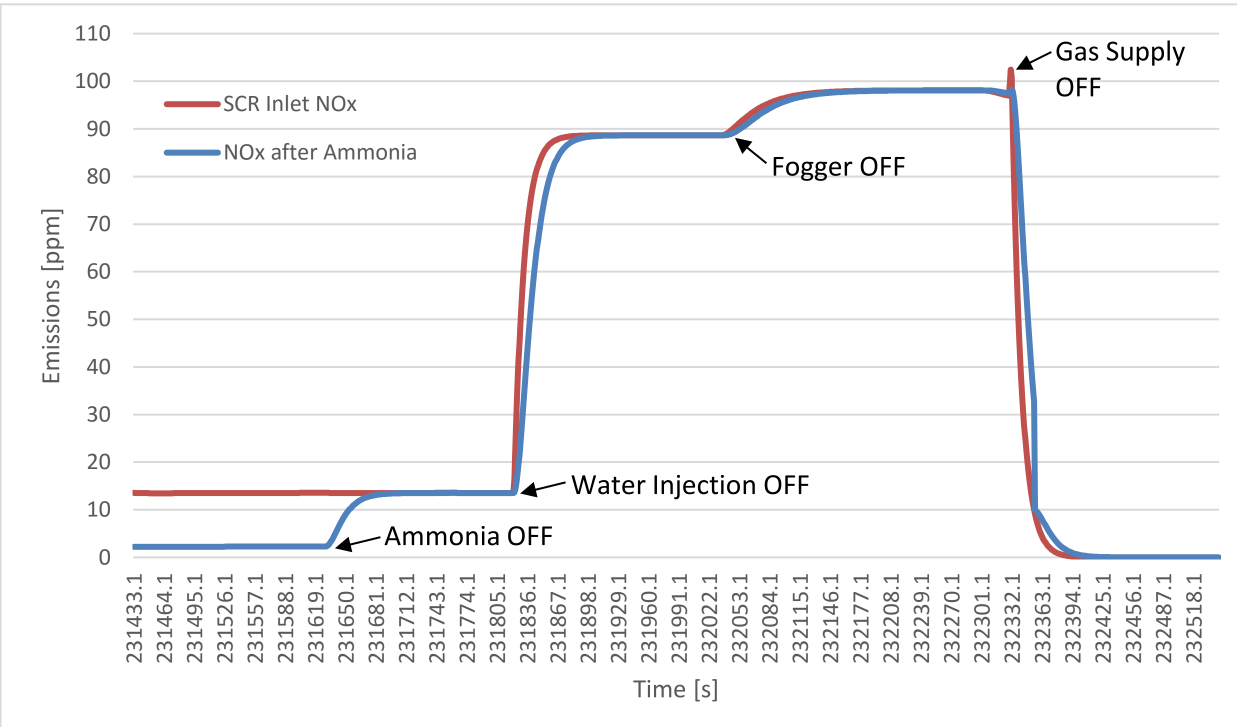

Figure 6-5: NOx Emissions of the Long Beach Simulator at Shutdown

From these figures, it can be observed that the NOx emissions on both simulators are very similar to each other when the ammonia injection, water injection, and foggers are switched off at the indicated time intervals. The concentration of the NOx emissions never exceed 98 ppm on either simulator, and after the gas supply is switched off, both simulators decrease to 0 ppm. Since both simulators yielded the same results, it proves that the test simulator (containing the new method) is functioning correctly.

NOTE:

Sensor dynamics was not included in the Test Simulator graphs on the next page.

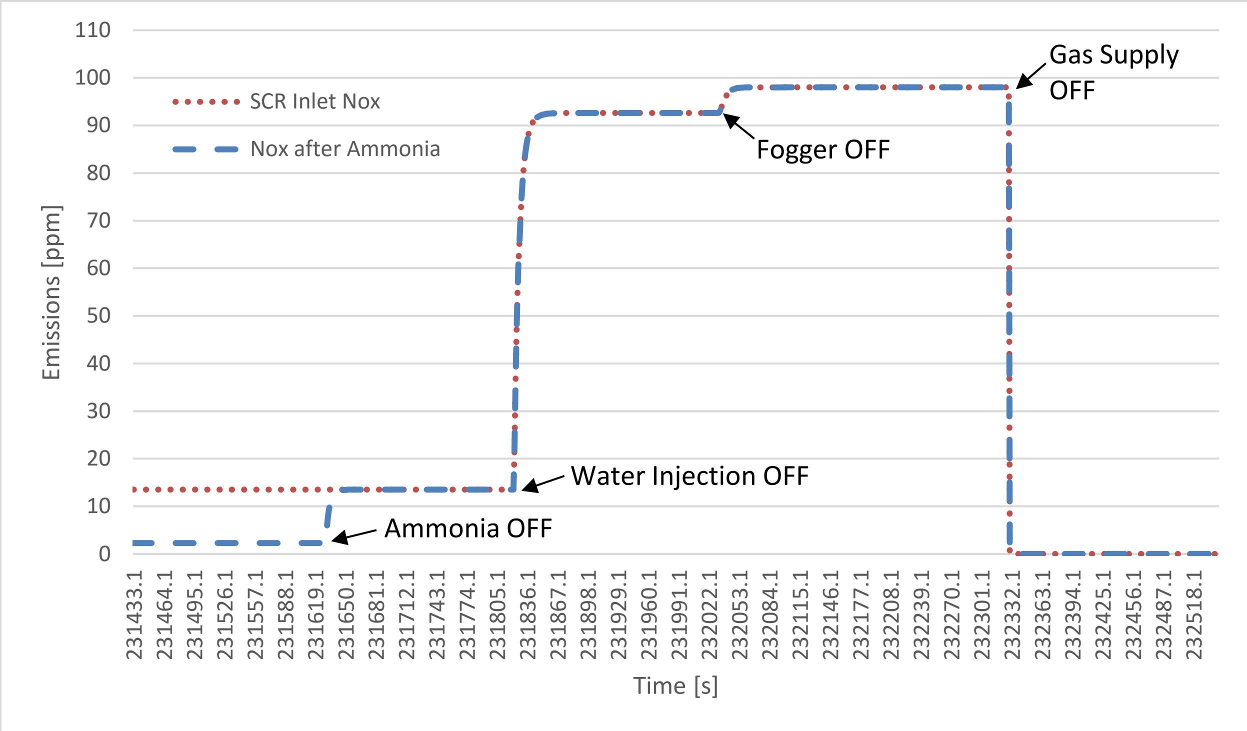

Figure 6-6: NOx Emissions of the Test Simulator at Shutdown

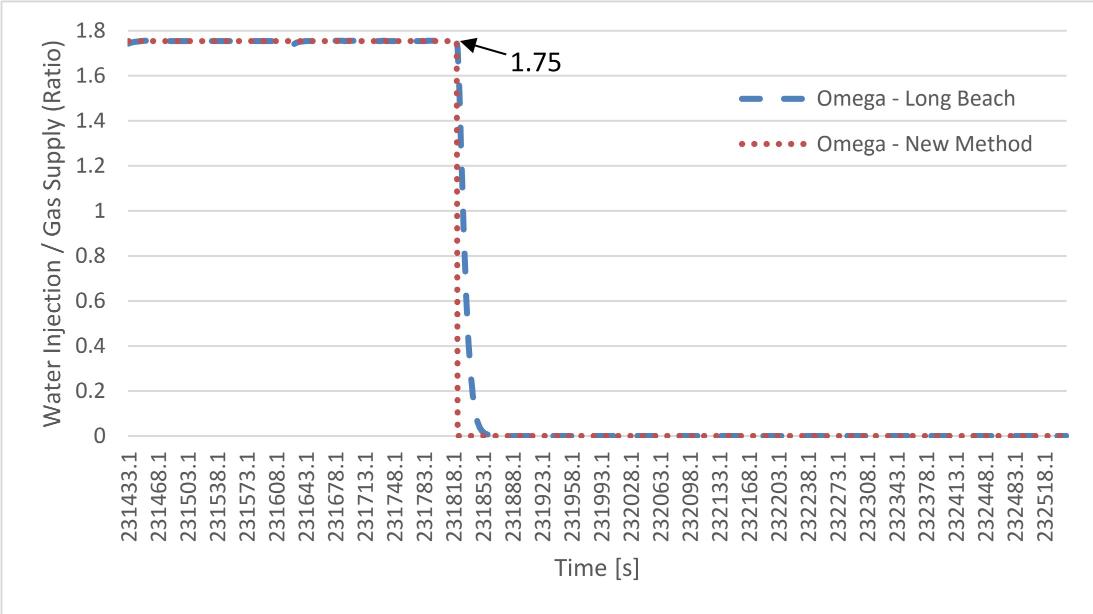

Figure 6-7: Omega Value for Long Beach - and the Test Simulator at Shutdown

The Omega value on both simulators were also examined at shutdown. It can be seen from **Figure 6-7** on the previous page, that Omega was immediately decreased to zero, on both simulators, when the water injection was turned off. From the figure, is seems that the Long Beach simulator takes a little longer to reach zero; this is only because there might be time constants inserted into the simulator preventing the calculation from happening too fast and making the simulator unstable.

The ammonia slip for each simulator was also calculated starting at steady state and ending when both simulators were completely shut down. As observed in **Figure 6-8** below, the ammonia slip at the Long Beach simulator was kept at a constant rate of 0.105 ppm, even after the ammonia injection had been shut off. This is proven to be incorrect by the test simulator, which demonstrates that the ammonia slip decreases from 5.87 ppm to 0 ppm at the moment the ammonia injection is turned off.

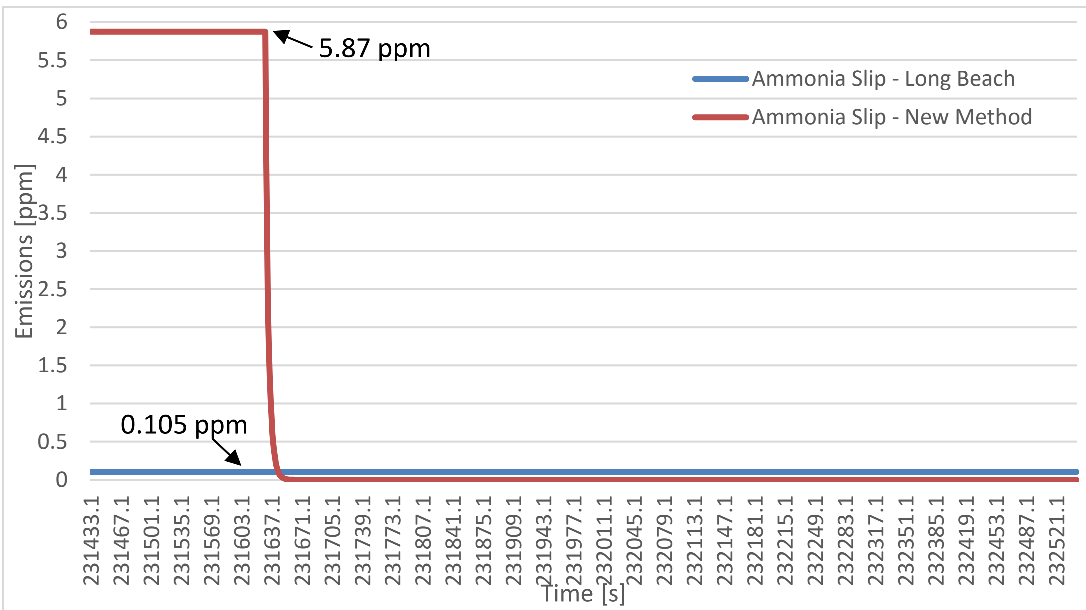

Figure 6-8: Ammonia Slip for Long Beach – and the Test Simulator at Shutdown

NOTE:

Sensor Dynamics was not included in the Test Simulator.

7. APPENDIX

The completed script, containing the code for solving the chemical equation using mass fractions and mass flow rates, as explained in **Section 5.4** above, is shown here to serve as an example for solving other chemical reactions.

```
//script using directives
//css_ref IPS.Core.dll;
//css_ref IPS.PluginInterface.dll;
using System;
using IPS.Properties;
using IPS.Scripting;

//script must be derived from IComponentScript
public class Script: IPS.Scripting.IComponentScript
{
    //Inputs
    IPS.Properties.Double _Mass_Fraction_CO2_Turbine;           //Mass Fractions
    IPS.Properties.Double _Mass_Fraction_H2Og_Turbine;          //T = At turbine exit
    IPS.Properties.Double _Mass_Fraction_N2_Turbine;
    IPS.Properties.Double _Mass_Fraction_NO_Turbine;
    IPS.Properties.Double _Mass_Fraction_O2_Turbine;
    IPS.Properties.Double _Mass_Fraction_NH3_Turbine;

    IPS.Properties.Double _Mass_Fraction_CO2_Ammonia;           //A = Ammonia Injection
    IPS.Properties.Double _Mass_Fraction_H2Og_Ammonia;
    IPS.Properties.Double _Mass_Fraction_N2_Ammonia;
    IPS.Properties.Double _Mass_Fraction_NO_Ammonia;
    IPS.Properties.Double _Mass_Fraction_O2_Ammonia;
    IPS.Properties.Double _Mass_Fraction_NH3_Ammonia;

    IPS.Properties.Double _MassFlow_Turbine;     //[kg/s]
    IPS.Properties.Double _MassFlow_Ammonia;
    IPS.Properties.Double _MassFlow_H2O_Injection;
    IPS.Properties.Double _MassFlow_Gas_Supply;
    IPS.Properties.Double _Ammonia_Efficiency;   //[%]

    //Variables
    IPS.Properties.Double _n_CO2_T;              //[mol/s]
    IPS.Properties.Double _n_H2Og_T;
    IPS.Properties.Double _n_N2_T;
    IPS.Properties.Double _n_NO_T;
    IPS.Properties.Double _n_O2_T;
    IPS.Properties.Double _n_NH3_T;

    IPS.Properties.Double _n_CO2_A;              //[mol/s]
    IPS.Properties.Double _n_H2Og_A;
    IPS.Properties.Double _n_N2_A;
    IPS.Properties.Double _n_NO_A;
    IPS.Properties.Double _n_O2_A;
    IPS.Properties.Double _n_NH3_A;

    IPS.Properties.Double _n_CO2_R;              //[mol/s]
    IPS.Properties.Double _n_H2Og_R;
    IPS.Properties.Double _n_N2_R;
    IPS.Properties.Double _n_NO_R;
    IPS.Properties.Double _n_O2_R;
    IPS.Properties.Double _n_NH3_R;

    IPS.Properties.Double _n_CO2_Total;          //[mol/s]
    IPS.Properties.Double _n_H2Og_Total;
    IPS.Properties.Double _n_N2_Total;
    IPS.Properties.Double _n_NO_Total;
    IPS.Properties.Double _n_O2_Total;
    IPS.Properties.Double _n_NH3_Total;
```

```

    IPS.Properties.Double _MassFlow_CO2;        //[kg/s]
    IPS.Properties.Double _MassFlow_H2Og;
    IPS.Properties.Double _MassFlow_N2;
    IPS.Properties.Double _MassFlow_NO;
    IPS.Properties.Double _MassFlow_O2;
    IPS.Properties.Double _MassFlow_NH3;

    IPS.Properties.Double _Ammonia_Fraction;

    //Outputs
    IPS.Properties.Double _Mass_Fraction_CO2_Out;           //Mass Fractions
    IPS.Properties.Double _Mass_Fraction_H2Og_Out;
    IPS.Properties.Double _Mass_Fraction_N2_Out;
    IPS.Properties.Double _Mass_Fraction_NO_Out;
    IPS.Properties.Double _Mass_Fraction_O2_Out;
    IPS.Properties.Double _Mass_Fraction_NH3_Out;

    IPS.Properties.Double _Total_MassFlow;
    IPS.Properties.Double _Total_Mass_Fraction;
    IPS.Properties.Double _NOx_at_SCR_Inlet;
    IPS.Properties.Double _NOx_after_Ammonia;
    IPS.Properties.Double _Ammonia_Slip;
    IPS.Properties.Double _Omega;

    //do pre simulation initialisation here
    public override void Initialise()
    {
  }

    //do post simulation cleanup here
    public override void Cleanup()
    {
  }

  //script main execution function - called every cycle
    public override void Execute(double Time)
    {
        //Local Variables
        double mf_CO2_T, mf_H2Og_T, mf_N2_T, mf_NO_T, mf_O2_T, mf_NH3_T;//Inputs
        double mf_CO2_A, mf_H2Og_A, mf_N2_A, mf_NO_A, mf_O2_A, mf_NH3_A;//Inputs
        double MF_Turb, MF_Amm, Amm_Eff, MF_H2O, MF_GS;//Inputs
        double n_CO2_T, n_H2Og_T, n_N2_T, n_NO_T, n_O2_T, n_NH3_T;//Variables
        double n_CO2_A, n_H2Og_A, n_N2_A, n_NO_A, n_O2_A, n_NH3_A;//Variables
        double n_CO2_R, n_H2Og_R, n_N2_R, n_NO_R, n_O2_R, n_NH3_R;//Variables
        double n_CO2_Total, n_H2Og_Total, n_N2_Total, n_NO_Total, n_O2_Total, n_NH3_Total;//Variables
        double MF_CO2, MF_H2Og, MF_N2, MF_NO, MF_O2, MF_NH3, Amm_Frac; //Variables
        double mf_CO2_out, mf_H2Og_out, mf_N2_out, mf_NO_out, mf_O2_out, mf_NH3_out; //Outputs
        double Total_MF, Total_Mass_Frac, NOx_SCR, NOx_Amm, Amm_Slip, Omega; //Outputs

        //Inputs
        mf_CO2_T= _Mass_Fraction_CO2_Turbine.Value;
        mf_H2Og_T = _Mass_Fraction_H2Og_Turbine.Value;
        mf_N2_T = _Mass_Fraction_N2_Turbine.Value;
        mf_NO_T = _Mass_Fraction_NO_Turbine.Value;
        mf_O2_T = _Mass_Fraction_O2_Turbine.Value;
        mf_NH3_T = _Mass_Fraction_NH3_Turbine.Value;

        mf_CO2_A= _Mass_Fraction_CO2_Ammonia.Value;
```

```
        mf_H2Og_A = _Mass_Fraction_H2Og_Ammonia.Value;
        mf_N2_A = _Mass_Fraction_N2_Ammonia.Value;
        mf_NO_A = _Mass_Fraction_NO_Ammonia.Value;
        mf_O2_A = _Mass_Fraction_O2_Ammonia.Value;
        mf_NH3_A = _Mass_Fraction_NH3_Ammonia.Value;

        MF_Turb = _MassFlow_Turbine.Value;
        MF_Amm = _MassFlow_Ammonia.Value;
        Amm_Eff = _Ammonia_Efficiency.Value;
        MF_H2O = _MassFlow_H2O_Injection.Value;
        MF_GS = _MassFlow_Gas_Supply.Value;

        //Calculates the mass fractions of the remaining products after ammonia had been added

        //Mole Rate at Turbine Exit                 n = [mol/s]
        n_CO2_T = (mf_CO2_T*MF_Turb*1000.0)/(44.01);
        n_H2Og_T = (mf_H2Og_T*MF_Turb*1000.0)/(18.01528);
        n_N2_T = (mf_N2_T*MF_Turb*1000.0)/(28.0134);
        n_NO_T = (mf_NO_T*MF_Turb*1000.0)/(30.0);
        n_O2_T = (mf_O2_T*MF_Turb*1000.0)/(31.998);
        n_NH3_T = (mf_NH3_T*MF_Turb*1000.0)/(17.03052);

        //Mole Rate at Ammonia Injection
        n_CO2_A = (mf_CO2_A*MF_Amm*1000.0)/(44.01);
        n_H2Og_A = (mf_H2Og_A*MF_Amm*1000.0)/(18.01528);
        n_N2_A = (mf_N2_A*MF_Amm*1000.0)/(28.0134);
        n_NO_A = (mf_NO_A*MF_Amm*1000.0)/(30.0);
        n_O2_A = (mf_O2_A*MF_Amm*1000.0)/(31.998);
        n_NH3_A = (mf_NH3_A*MF_Amm*1000.0)/(17.03052);

        //Mole Rate of Reaction                                     n = [mol/s]
        //4NO + 4NH3 + O2 ---> 4N2 + 6H2O
        Amm_Frac = Amm_Eff/100.0;
        n_CO2_R = 0.0;
        n_H2Og_R = (6.0/4.0)*n_NH3_T + (6.0/4.0)*(n_NH3_A*Amm_Frac);    //Pos
        n_N2_R = n_NH3_T + (n_NH3_A*Amm_Frac);                           //Pos
        n_NO_R = n_NH3_T + (n_NH3_A*Amm_Frac);                           //Neg
        n_O2_R = (1.0/4.0)*(n_NH3_T) + (1.0/4.0)*(n_NH3_A*Amm_Frac);    //Neg
        n_NH3_R = n_NH3_T + (n_NH3_A*Amm_Frac);                          //Neg

        //Total Mole Rate from Reaction
        n_CO2_Total = n_CO2_T + n_CO2_A + n_CO2_R;                    //[mol/s]
        n_H2Og_Total = n_H2Og_T + n_H2Og_A + n_H2Og_R;
        n_N2_Total = n_N2_T + n_N2_A + n_N2_R;
        n_NO_Total = n_NO_T + n_NO_A - n_NO_R;
        n_O2_Total = n_O2_T + n_O2_A - n_O2_R;
        n_NH3_Total = n_NH3_T + n_NH3_A - n_NH3_R;

        MF_CO2 = (n_CO2_Total*44.01)/1000.0;                     //MF = Mass Flow [kg/s]
        MF_H2Og = (n_H2Og_Total*18.01528)/1000.0;
        MF_N2 = (n_N2_Total*28.0134)/1000.0;
        MF_NO = (n_NO_Total*30)/1000.0;
        MF_O2 = (n_O2_Total*31.998)/1000.0;
        MF_NH3 = (n_NH3_Total*17.03052)/1000.0;

        Total_MF = MF_Turb + MF_Amm;
        mf_CO2_out = MF_CO2/(Total_MF +0.0000001);  //mf = Mass_Fraction
        mf_N2_out = MF_N2/(Total_MF +0.0000001);
        mf_NO_out = MF_NO/(Total_MF +0.0000001);
```

```
        mf_O2_out = MF_O2/(Total_MF +0.0000001);
        mf_NH3_out = MF_NH3/(Total_MF +0.0000001);
        mf_H2Og_out = 1 - mf_CO2_out- mf_N2_out- mf_NO_out- mf_O2_out- mf_NH3_out;

        Total_Mass_Frac = mf_CO2_out + mf_N2_out + mf_NO_out + mf_O2_out + mf_NH3_out + mf_H2Og_out;

        //Calculations of parts per million
        NOx_SCR = mf_NO_T*(1000000.0);              //[ppm]
        NOx_Amm = mf_NO_out*(1000000.0);
        Amm_Slip = mf_NH3_out*(1000000.0);

        //Calculations of parts per million   [ppm]
        NOx_Amm = mf_NO_out*(1000000.0);
        Amm_Slip = mf_NH3_out*(1000000.0);

        //Calculation of Omega
        //Omega = Water Injection Mass Flow / Gas Supply Mass Flow
        Omega = MF_H2O /  (MF_GS+0.0000001);

        //Global Variables
        _MassFlow_CO2.Value = MF_CO2;
        _MassFlow_H2Og.Value = MF_H2Og;
        _MassFlow_N2.Value = MF_N2;
        _MassFlow_NO.Value = MF_NO;
        _MassFlow_O2.Value = MF_O2;
        _MassFlow_NH3.Value = MF_NH3;

        _n_CO2_T.Value = n_CO2_T;
        _n_H2Og_T.Value = n_H2Og_T;
        _n_N2_T.Value = n_N2_T;
        _n_NO_T.Value = n_NO_T;
        _n_O2_T.Value = n_O2_T;
        _n_NH3_T.Value = n_NH3_T;

        _n_CO2_A.Value = n_CO2_A;
        _n_H2Og_A.Value = n_H2Og_A;
        _n_N2_A.Value = n_N2_A;
        _n_NO_A.Value = n_NO_A;
        _n_O2_A.Value = n_O2_A;
        _n_NH3_A.Value = n_NH3_A;

        _n_CO2_R.Value = n_CO2_R;
        _n_H2Og_R.Value = n_H2Og_R;
        _n_N2_R.Value = n_N2_R;
        _n_NO_R.Value = n_NO_R;
        _n_O2_R.Value = n_O2_R;
        _n_NH3_R.Value = n_NH3_R;

        _n_CO2_Total.Value = n_CO2_Total;
        _n_H2Og_Total.Value = n_H2Og_Total;
        _n_N2_Total.Value = n_N2_Total;
        _n_NO_Total.Value = n_NO_Total;
        _n_O2_Total.Value = n_O2_Total;
        _n_NH3_Total.Value = n_NH3_Total;

        _Ammonia_Fraction.Value = Amm_Frac;

        //Outputs
        _Mass_Fraction_CO2_Out.Value = mf_CO2_out;
```

```
            _Mass_Fraction_H2Og_Out.Value = mf_H2Og_out;
            _Mass_Fraction_N2_Out.Value = mf_N2_out;
            _Mass_Fraction_NO_Out.Value = mf_NO_out;
            _Mass_Fraction_O2_Out.Value = mf_O2_out;
            _Mass_Fraction_NH3_Out.Value = mf_NH3_out;

            _Total_MassFlow.Value = Total_MF;
            _Total_Mass_Fraction.Value = Total_Mass_Frac;
            _NOx_at_SCR_Inlet.Value = NOx_SCR;
            _NOx_after_Ammonia.Value = NOx_Amm;
            _Ammonia_Slip.Value = Amm_Slip;
            _Omega.Value = Omega;

        }

    //any processing you want to do before steady state
      public override void ExecuteBeforeSteadyState()
      {
      Execute(0.0);
      }
     //any processing you want to do while solving steady state
      public override void ExecuteSteadyState()
      {
      Execute(0.0);
      }

    //any processing you want to do after steady state
      public override void ExecuteAfterSteadyState()
      {
      Execute(0.0);
      }

      //constructer initialises parameters
      public Script()
      {
          //Inputs
          _Mass_Fraction_CO2_Turbine = new IPS.Properties.Double();
          _Mass_Fraction_CO2_Turbine.Value = 0.0;
          _Mass_Fraction_H2Og_Turbine = new IPS.Properties.Double();
          _Mass_Fraction_H2Og_Turbine.Value = 0.0;
          _Mass_Fraction_N2_Turbine = new IPS.Properties.Double();
          _Mass_Fraction_N2_Turbine.Value = 0.0;
          _Mass_Fraction_NO_Turbine = new IPS.Properties.Double();
          _Mass_Fraction_NO_Turbine.Value = 0.0;
          _Mass_Fraction_O2_Turbine = new IPS.Properties.Double();
          _Mass_Fraction_O2_Turbine.Value = 0.0;
          _Mass_Fraction_NH3_Turbine = new IPS.Properties.Double();
          _Mass_Fraction_NH3_Turbine.Value = 0.0;

          _Mass_Fraction_CO2_Ammonia = new IPS.Properties.Double();
          _Mass_Fraction_CO2_Ammonia.Value = 0.0;
          _Mass_Fraction_H2Og_Ammonia = new IPS.Properties.Double();
          _Mass_Fraction_H2Og_Ammonia.Value = 0.0;
          _Mass_Fraction_N2_Ammonia = new IPS.Properties.Double();
          _Mass_Fraction_N2_Ammonia.Value = 0.0;
          _Mass_Fraction_NO_Ammonia = new IPS.Properties.Double();
          _Mass_Fraction_NO_Ammonia.Value = 0.0;
          _Mass_Fraction_O2_Ammonia = new IPS.Properties.Double();
          _Mass_Fraction_O2_Ammonia.Value = 0.0;
```

```
            _Mass_Fraction_NH3_Ammonia = new IPS.Properties.Double();
            _Mass_Fraction_NH3_Ammonia.Value = 0.0;

            _MassFlow_Turbine = new IPS.Properties.Double();
            _MassFlow_Turbine.Value = 0.0;
            _MassFlow_Ammonia = new IPS.Properties.Double();
            _MassFlow_Ammonia.Value = 0.0;
            _Ammonia_Efficiency = new IPS.Properties.Double();
            _Ammonia_Efficiency.Value = 0.0;
            _MassFlow_H2O_Injection = new IPS.Properties.Double();
            _MassFlow_H2O_Injection.Value = 0.0;
            _MassFlow_Gas_Supply = new IPS.Properties.Double();
            _MassFlow_Gas_Supply.Value = 0.0;

            //Variables
            _n_CO2_T = new IPS.Properties.Double();
            _n_CO2_T.Value = 0.0;
            _n_H2Og_T = new IPS.Properties.Double();
            _n_H2Og_T.Value = 0.0;
            _n_N2_T = new IPS.Properties.Double();
            _n_N2_T.Value = 0.0;
            _n_NO_T = new IPS.Properties.Double();
            _n_NO_T.Value = 0.0;
            _n_O2_T = new IPS.Properties.Double();
            _n_O2_T.Value = 0.0;
            _n_NH3_T = new IPS.Properties.Double();
            _n_NH3_T.Value = 0.0;

            _n_CO2_A = new IPS.Properties.Double();
            _n_CO2_A.Value = 0.0;
            _n_H2Og_A = new IPS.Properties.Double();
            _n_H2Og_A.Value = 0.0;
            _n_N2_A = new IPS.Properties.Double();
            _n_N2_A.Value = 0.0;
            _n_NO_A = new IPS.Properties.Double();
            _n_NO_A.Value = 0.0;
            _n_O2_A = new IPS.Properties.Double();
            _n_O2_A.Value = 0.0;
            _n_NH3_A = new IPS.Properties.Double();
            _n_NH3_A.Value = 0.0;

            _n_CO2_R = new IPS.Properties.Double();
            _n_CO2_R.Value = 0.0;
            _n_H2Og_R = new IPS.Properties.Double();
            _n_H2Og_R.Value = 0.0;
            _n_N2_R = new IPS.Properties.Double();
            _n_N2_R.Value = 0.0;
            _n_NO_R = new IPS.Properties.Double();
            _n_NO_R.Value = 0.0;
            _n_O2_R = new IPS.Properties.Double();
            _n_O2_R.Value = 0.0;
            _n_NH3_R = new IPS.Properties.Double();
            _n_NH3_R.Value = 0.0;

            _n_CO2_Total = new IPS.Properties.Double();
            _n_CO2_Total.Value = 0.0;
            _n_H2Og_Total = new IPS.Properties.Double();
            _n_H2Og_Total.Value = 0.0;
            _n_N2_Total = new IPS.Properties.Double();
```

```
            _n_N2_Total.Value = 0.0;
            _n_NO_Total = new IPS.Properties.Double();
            _n_NO_Total.Value = 0.0;
            _n_O2_Total = new IPS.Properties.Double();
            _n_O2_Total.Value = 0.0;
            _n_NH3_Total = new IPS.Properties.Double();
            _n_NH3_Total.Value = 0.0;

            _MassFlow_CO2 = new IPS.Properties.Double();
            _MassFlow_CO2.Value = 0;
            _MassFlow_H2Og = new IPS.Properties.Double();
            _MassFlow_H2Og.Value = 0;
            _MassFlow_N2 = new IPS.Properties.Double();
            _MassFlow_N2.Value = 0;
            _MassFlow_NO = new IPS.Properties.Double();
            _MassFlow_NO.Value = 0;
            _MassFlow_O2 = new IPS.Properties.Double();
            _MassFlow_O2.Value = 0;
            _MassFlow_NH3 = new IPS.Properties.Double();
            _MassFlow_NH3.Value = 0;

            _Ammonia_Fraction = new IPS.Properties.Double();
            _Ammonia_Fraction.Value = 0;

            //Outputs
            _Mass_Fraction_CO2_Out = new IPS.Properties.Double();
            _Mass_Fraction_CO2_Out.Value = 0.0;
            _Mass_Fraction_H2Og_Out = new IPS.Properties.Double();
            _Mass_Fraction_H2Og_Out.Value = 0.0;
            _Mass_Fraction_N2_Out = new IPS.Properties.Double();
            _Mass_Fraction_N2_Out.Value = 0.0;
            _Mass_Fraction_NO_Out = new IPS.Properties.Double();
            _Mass_Fraction_NO_Out.Value = 0.0;
            _Mass_Fraction_O2_Out = new IPS.Properties.Double();
            _Mass_Fraction_O2_Out.Value = 0.0;
            _Mass_Fraction_NH3_Out = new IPS.Properties.Double();
            _Mass_Fraction_NH3_Out.Value = 0.0;

            _Total_MassFlow = new IPS.Properties.Double();
            _Total_MassFlow.Value = 0.0;
            _Total_Mass_Fraction = new IPS.Properties.Double();
            _Total_Mass_Fraction.Value = 0.0;
            _NOx_at_SCR_Inlet = new IPS.Properties.Double();
            _NOx_at_SCR_Inlet.Value = 0.0;
            _NOx_after_Ammonia = new IPS.Properties.Double();
            _NOx_after_Ammonia.Value = 0.0;
            _Ammonia_Slip = new IPS.Properties.Double();
            _Ammonia_Slip.Value = 0.0;
            _Omega = new IPS.Properties.Double();
            _Omega.Value = 0.0;

        }

        //property declarations to make
        //parameters visible to outside world

        //Inputs
        //-----
        [PropertyUsage(UseProperty.DYNAMIC)]
```

```
        [GridCategory(new String[]{"Inputs [%]"})]
        [GridOrder(100)]
        public IPS.Properties.Double Ammonia_Efficiency
        {
            get
            {
                return _Ammonia_Efficiency;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(101)]
        public IPS.Properties.Double Mass_Fraction_CO2_Turbine
        {
            get
            {
                return _Mass_Fraction_CO2_Turbine;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(102)]
        public IPS.Properties.Double Mass_Fraction_H2Og_Turbine
        {
            get
            {
                return _Mass_Fraction_H2Og_Turbine;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(103)]
        public IPS.Properties.Double Mass_Fraction_N2_Turbine
        {
            get
            {
                return _Mass_Fraction_N2_Turbine;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(104)]
        public IPS.Properties.Double Mass_Fraction_NO_Turbine
        {
            get
            {
                return _Mass_Fraction_NO_Turbine;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(105)]
        public IPS.Properties.Double Mass_Fraction_O2_Turbine
        {
            get
            {
                return _Mass_Fraction_O2_Turbine;
            }
        }
```

```
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(106)]
        public IPS.Properties.Double Mass_Fraction_NH3_Turbine
        {
            get
            {
                return _Mass_Fraction_NH3_Turbine;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(110)]
        public IPS.Properties.Double Mass_Fraction_CO2_Ammonia
        {
            get
            {
                return _Mass_Fraction_CO2_Ammonia;
            }
        }

        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(111)]
        public IPS.Properties.Double Mass_Fraction_H2Og_Ammonia
        {
            get
            {
                return _Mass_Fraction_H2Og_Ammonia;
            }
        }

        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(112)]
        public IPS.Properties.Double Mass_Fraction_N2_Ammonia
        {
            get
            {
                return _Mass_Fraction_N2_Ammonia;
            }
        }

        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(113)]
        public IPS.Properties.Double Mass_Fraction_NO_Ammonia
        {
            get
            {
                return _Mass_Fraction_NO_Ammonia;
            }
        }

        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(114)]
        public IPS.Properties.Double Mass_Fraction_O2_Ammonia
        {
```

```
            get
            {
                return _Mass_Fraction_O2_Ammonia;
            }
        }

        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs (Mass Fractions)"})]
        [GridOrder(115)]
        public IPS.Properties.Double Mass_Fraction_NH3_Ammonia
        {
            get
            {
                return _Mass_Fraction_NH3_Ammonia;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs [kg/s]"})]
        [GridOrder(120)]
        public IPS.Properties.Double MassFlow_Turbine
        {
            get
            {
                return _MassFlow_Turbine;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs [kg/s]"})]
        [GridOrder(121)]
        public IPS.Properties.Double MassFlow_Ammonia
        {
            get
            {
                return _MassFlow_Ammonia;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs [kg/s]"})]
        [GridOrder(122)]
        public IPS.Properties.Double MassFlow_H2O_Injection
        {
            get
            {
                return _MassFlow_H2O_Injection;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Inputs [kg/s]"})]
        [GridOrder(123)]
        public IPS.Properties.Double MassFlow_Gas_Supply
        {
            get
            {
                return _MassFlow_Gas_Supply;
            }
        }
        //Variables
        //-----
        [PropertyUsage(UseProperty.DYNAMIC)]
```

```
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(200)]
        public IPS.Properties.Double Ammonia_Fraction
        {
            get
            {
                return _Ammonia_Fraction;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(201)]
        public IPS.Properties.Double n_CO2_T
        {
            get
            {
                return _n_CO2_T;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(202)]
        public IPS.Properties.Double n_H2Og_T
        {
            get
            {
                return _n_H2Og_T;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(203)]
        public IPS.Properties.Double n_N2_T
        {
            get
            {
                return _n_N2_T;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(204)]
        public IPS.Properties.Double n_NO_T
        {
            get
            {
                return _n_NO_T;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(205)]
        public IPS.Properties.Double n_O2_T
        {
            get
            {
                return _n_O2_T;
            }
        }
```

```
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(206)]
        public IPS.Properties.Double n_NH3_T
        {
            get
            {
                return _n_NH3_T;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(211)]
        public IPS.Properties.Double n_CO2_A
        {
            get
            {
                return _n_CO2_A;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(212)]
        public IPS.Properties.Double n_H2Og_A
        {
            get
            {
                return _n_H2Og_A;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(213)]
        public IPS.Properties.Double n_N2_A
        {
            get
            {
                return _n_N2_A;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(214)]
        public IPS.Properties.Double n_NO_A
        {
            get
            {
                return _n_NO_A;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(215)]
        public IPS.Properties.Double n_O2_A
        {
            get
            {
                return _n_O2_A;
            }
```

```
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(216)]
        public IPS.Properties.Double n_NH3_A
        {
            get
            {
                return _n_NH3_A;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(221)]
        public IPS.Properties.Double n_CO2_R
        {
            get
            {
                return _n_CO2_R;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(222)]
        public IPS.Properties.Double n_H2Og_R
        {
            get
            {
                return _n_H2Og_R;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(223)]
        public IPS.Properties.Double n_N2_R
        {
            get
            {
                return _n_N2_R;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(224)]
        public IPS.Properties.Double n_NO_R
        {
            get
            {
                return _n_NO_R;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(225)]
        public IPS.Properties.Double n_O2_R
        {
            get
            {
                return _n_O2_R;
```

```
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Variables"})]
    [GridOrder(226)]
    public IPS.Properties.Double n_NH3_R
    {
        get
        {
            return _n_NH3_R;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Variables"})]
    [GridOrder(231)]
    public IPS.Properties.Double n_CO2_Total
    {
        get
        {
            return _n_CO2_Total;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Variables"})]
    [GridOrder(232)]
    public IPS.Properties.Double n_H2Og_Total
    {
        get
        {
            return _n_H2Og_Total;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Variables"})]
    [GridOrder(233)]
    public IPS.Properties.Double n_N2_Total
    {
        get
        {
            return _n_N2_Total;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Variables"})]
    [GridOrder(234)]
    public IPS.Properties.Double n_NO_Total
    {
        get
        {
            return _n_NO_Total;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Variables"})]
    [GridOrder(235)]
    public IPS.Properties.Double n_O2_Total
    {
        get
        {
```

```
                return _n_O2_Total;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(236)]
        public IPS.Properties.Double n_NH3_Total
        {
            get
            {
                return _n_NH3_Total;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(241)]
        public IPS.Properties.Double MassFlow_CO2
        {
            get
            {
                return _MassFlow_CO2;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(242)]
        public IPS.Properties.Double MassFlow_H2Og
        {
            get
            {
                return _MassFlow_H2Og;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(243)]
        public IPS.Properties.Double MassFlow_N2
        {
            get
            {
                return _MassFlow_N2;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(244)]
        public IPS.Properties.Double MassFlow_NO
        {
            get
            {
                return _MassFlow_NO;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(245)]
        public IPS.Properties.Double MassFlow_O2
        {
            get
```

```
            {
                return _MassFlow_O2;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Variables"})]
        [GridOrder(246)]
        public IPS.Properties.Double MassFlow_NH3
        {
            get
            {
                return _MassFlow_NH3;
            }
        }
        //Outputs
        //-----
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
        [GridOrder(310)]
        public IPS.Properties.Double Mass_Fraction_CO2_Out
        {
            get
            {
                return _Mass_Fraction_CO2_Out;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
        [GridOrder(311)]
        public IPS.Properties.Double Mass_Fraction_H2Og_Out
        {
            get
            {
                return _Mass_Fraction_H2Og_Out;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
        [GridOrder(312)]
        public IPS.Properties.Double Mass_Fraction_N2_Out
        {
            get
            {
                return _Mass_Fraction_N2_Out;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
        [GridOrder(313)]
        public IPS.Properties.Double Mass_Fraction_NO_Out
        {
            get
            {
                return _Mass_Fraction_NO_Out;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
        [GridOrder(314)]
```

```
    public IPS.Properties.Double Mass_Fraction_O2_Out
    {
        get
        {
            return _Mass_Fraction_O2_Out;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
    [GridOrder(315)]
    public IPS.Properties.Double Mass_Fraction_NH3_Out
    {
        get
        {
            return _Mass_Fraction_NH3_Out;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Outputs (Mass Fractions)"})]
    [GridOrder(320)]
    public IPS.Properties.Double Total_Mass_Fraction
    {
        get
        {
            return _Total_Mass_Fraction;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Outputs [kg/s]"})]
    [GridOrder(330)]
    public IPS.Properties.Double Total_MassFlow
    {
        get
        {
            return _Total_MassFlow;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Outputs [ppm]"})]
    [GridOrder(340)]
    public IPS.Properties.Double NOx_at_SCR_Inlet
    {
        get
        {
            return _NOx_at_SCR_Inlet;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Outputs [ppm]"})]
    [GridOrder(350)]
    public IPS.Properties.Double NOx_after_Ammonia
    {
        get
        {
            return _NOx_after_Ammonia;
        }
    }
    [PropertyUsage(UseProperty.DYNAMIC)]
    [GridCategory(new String[]{"Outputs [ppm]"})]
```

```
        [GridOrder(360)]
        public IPS.Properties.Double Ammonia_Slip
        {
            get
            {
                return _Ammonia_Slip;
            }
        }
        [PropertyUsage(UseProperty.DYNAMIC)]
        [GridCategory(new String[]{"Outputs (Ratio)"})]
        [GridOrder(370)]
        public IPS.Properties.Double Omega
        {
            get
            {
                return _Omega;
            }
        }
    }
```