12th Homework - Solution

- In this homework, we shall model chemical reactions using reaction rate equations only.
- We shall program the reaction rate equations directly in the equation window and learn to use ***Modelica's*** matrix notation.

- Hydrogen-Bromine Reaction
- Oxy-hydrogen Gas Reaction

Hydrogen-Bromine Reaction

- We wish to simulate the ***hydrogen-bromine reaction*** described during the lectures. We concentrate on mass flows only, i.e., we only model the reaction rate equations.
- We wish to plot the molar masses of the five species as functions of time.
- We shall program the reaction rate equations in the ***equation window*** of ***Dymola***, making use of a ***matrix-vector notation***, i.e., the chemical reaction network is described by the corresponding ***N-matrix***.

- Although the reactions are occurring under ***isothermic conditions***, we still wish to take the ***Arrhenius' law*** into account, and program the reaction rate constants as functions of temperature:

$$ak_1 = 1.39 \cdot 10^8 \cdot \sqrt{T} \cdot \left(\frac{189243.0}{R \cdot T}\right)^{1.97}$$

$$k_1 = ak_1 \cdot \exp\left(\frac{-189243.0}{R \cdot T}\right)$$

$$k_2 = \frac{k_1}{K(T)}$$

$$k_3 = 10^{11.43} \cdot \exp\left(\frac{-82400.0}{R \cdot T}\right)$$

$$k_5 = 10^{11.97} \cdot \exp\left(\frac{-149800.0}{R \cdot T}\right)$$

$$k_4 = 0.1 \cdot k_5$$

- where ***R*** is the gas constant (***R = 8.314 J K^{-1} mole^{-1}***).

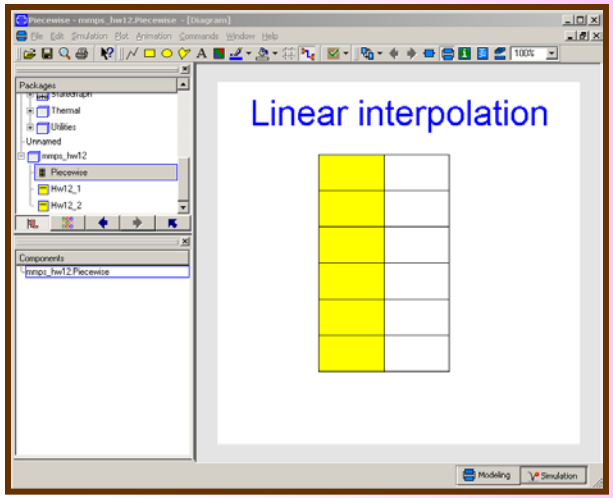
- Reaction k_2 contains a temperature dependence $K(T)$ that was experimentally found:

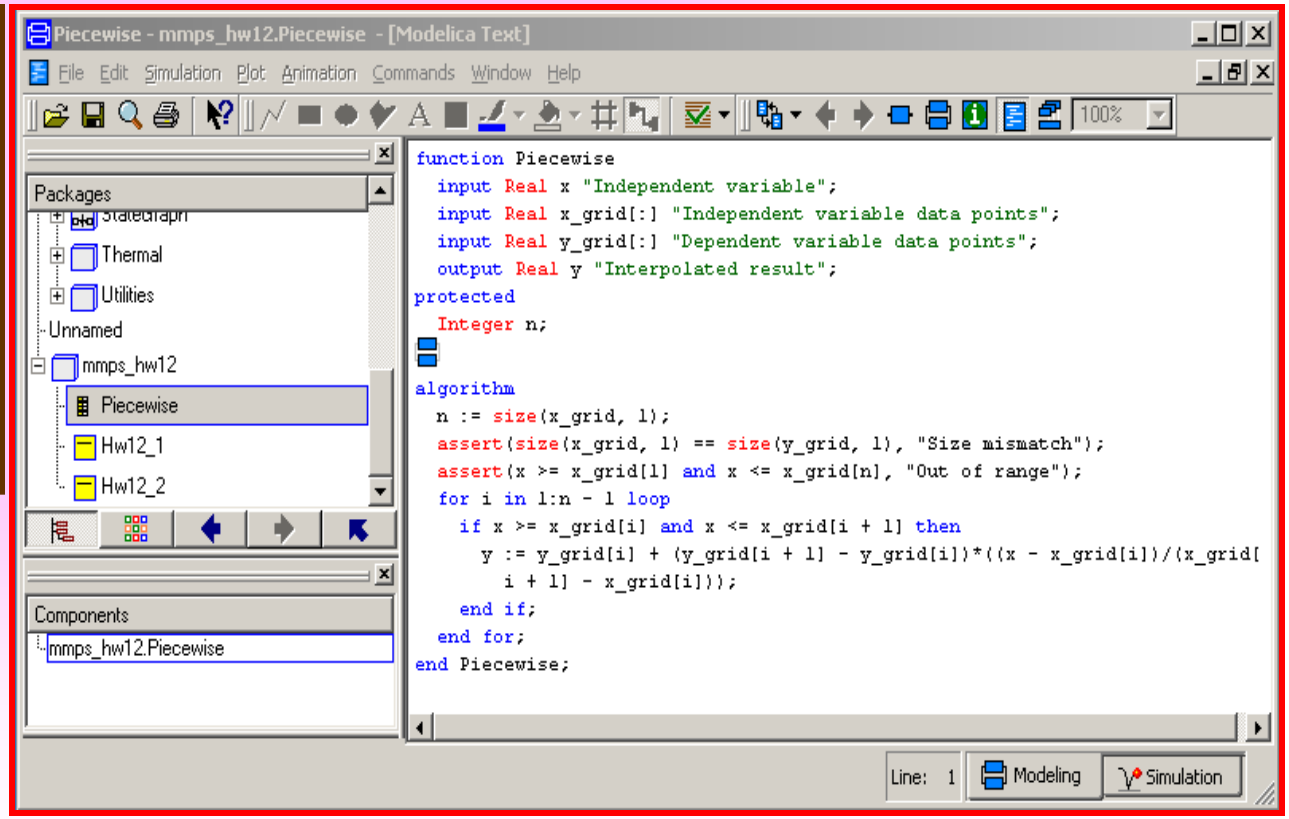
Abs. Temperature T [K]	Equilibrium Const. K [mole m^{-3}]
300.0	7.7446×10^{-29}
400.0	1.9543×10^{-20}
500.0	2.2182×10^{-15}
600.0	5.2844×10^{-12}
700.0	1.3867×10^{-9}
800.0	9.0782×10^{-8}
900.0	2.3768×10^{-6}
1000.0	3.2509×10^{-5}
1100.0	2.7861×10^{-4}
1200.0	1.6788×10^{-3}
1300.0	7.6913×10^{-3}
1400.0	2.8510×10^{-2}
1500.0	8.8716×10^{-2}
1600.0	2.4044×10^{-1}
1700.0	5.8344×10^{-1}
1800.0	1.7947
1900.0	2.6061
2000.0	4.9431

- Program $K(T)$ using a table-lookup function.

- The initial molar masses of **Br_2** and **H_2** are both equal to ***0.0075***. The total reaction volume is ***$V = 0.001\ m^3$***. The temperature is ***$T = 800\ K$***.
- Simulate the system during ***5000 seconds***. You need to reduce the tolerance value for the ***DASSL integration algorithm*** to ***10^{-10}***.
- Plot on one graph the molar masses of **Br_2**, **H_2**, and **HBr** during the first ***0.1 seconds***.
- Plot on a second graph the molar mass of **$H^\bullet$** during the first ***0.2 seconds***.
- Plot on a third graph the molar mass of **$Br^\bullet$** during the first ***0.3 seconds***.

- We shall need a ***table look-up function***. We can program it manually.





Piecewise - mmps_hw12.Piecewise - [Modelica Text]

```
function Piecewise
  input Real x "Independent variable";
  input Real x_grid[:] "Independent variable data points";
  input Real y_grid[:] "Dependent variable data points";
  output Real y "Interpolated result";
protected
  Integer n;

algorithm
  n := size(x_grid, 1);
  assert(size(x_grid, 1) == size(y_grid, 1), "Size mismatch");
  assert(x >= x_grid[1] and x <= x_grid[n], "Out of range");
  for i in 1:n - 1 loop
    if x >= x_grid[i] and x <= x_grid[i + 1] then
      y := y_grid[i] + (y_grid[i + 1] - y_grid[i])*((x - x_grid[i])/(x_grid[
        i + 1] - x_grid[i]));
    end if;
  end for;
end Piecewise;
```

We use a function.

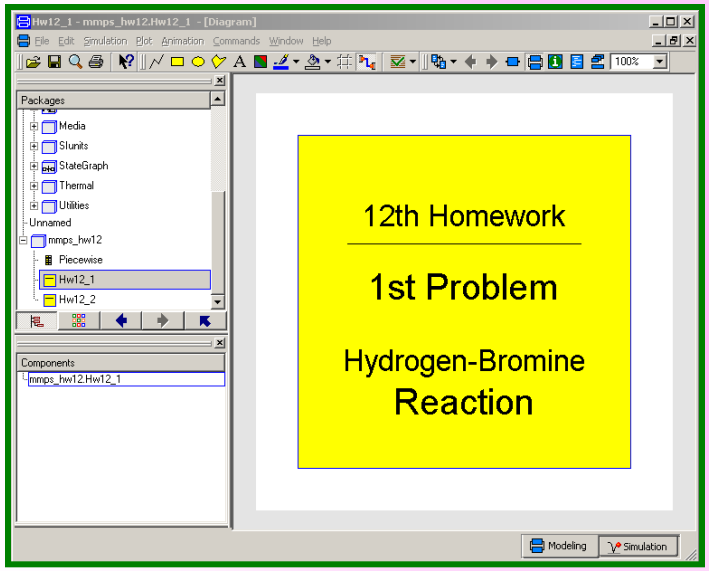
The colon here means that the array will adjust its dimension upon invocation.

Protected variables are hidden from the outside. They are strictly local.

The algorithm instruction allows users to encode sequential code. While equations are declarative, algorithms are procedural.

Assert is a clean way to ensure proper usage of the function. If the assert clause is not met, the simulation dies with the indicated error message.

- We can now create the model for the ***hydrogen-bromine reaction***.



The problem is modeled entirely in the equation window. The code is a bit long. The declarations and equations shall be explained separately.

Hw12_1 - mmps_hw12.Hw12_1 - [Modelica Text]

```
model Hw12_1
  package Math = Modelica.Math;
  parameter Modelica.SIunits.Temperature T=800.0 "Reaction temperature";
  parameter Modelica.SIunits.MolarHeatCapacity R = Modelica.Constants.R
    "Gas constant";
  parameter Modelica.SIunits.Volume V=0.001 "Reaction volume";
  parameter Integer nbr_subs=5 "Number of reactants";
  parameter Integer nbr_reac=5 "Number of reactions";

  //Reaction rate constants
  parameter Modelica.SIunits.Temperature Temp_vals[18]=300:100:2000;
  parameter Real K_vals[18](unit="mol/m3") = {7.7446e-29,1.8543e-20,2.2182e-15,
      5.2844e-12,1.3867e-9,9.0782e-8,2.3768e-6,3.2509e-5,2.7861e-4,1.6788e-3,
      7.6913e-3,2.851e-2,8.8716e-2,2.4044e-1,5.8344e-1,1.7947,2.6061,4.9431};
  Real RT(unit="m3/mol.K");
  Real K(unit="mol/m3");
  Real akl(unit="1/s");
  Real k[nbr_reac];  // Vector components have different units!

  // Reaction rate equations
  parameter Integer N[nbr_subs, nbr_reac]={{-1,1,0,0,-1},{2,-2,-1,1,1},{0,0,
      -1,1,0},{0,0,1,-1,-1},{0,0,1,-1,1}};
  Modelica.SIunits.AmountOfSubstance n_subs[nbr_subs](start={0.0075,0.0,0.0075,0.0,0.0});
  Real nu_subs[nbr_subs](unit="mol/s");
  Real nu_reac[nbr_reac](unit="mol/s");
  Modelica.SIunits.AmountOfSubstance nBr2;
  Modelica.SIunits.AmountOfSubstance nBr;
  Modelica.SIunits.AmountOfSubstance nH2;
  Modelica.SIunits.AmountOfSubstance nH;
  Modelica.SIunits.AmountOfSubstance nHBr;

equation
```

This is an abbreviation. Whenever I write "Math," I mean "Modelica.Math."

For convenience, I use the Matlab notation here.

Array sizes can be parameter expressions.

The Modelica notation is used here to declare this two-dimensional array. The Matlab notation would have worked equally well.

Initial conditions for the state vector.

Hw12_1 - mmps_hw12.Hw12_1 - [Modelica Text]

```
equation

  // Reaction rate constants
  // Empirical equations - numerical constants have units!
  K = Piecewise(x=T, x_grid=Temp_vals, y_grid=K_vals);
  RT = 1.0/(R*T);
  ak1 = 1.39e8*sqrt(T)*(189243.0*RT)^1.97;
  k[1] = ak1*Math.exp(-189243.0*RT);
  k[2] = k[1]/K;
  k[3] = 10^11.43*Math.exp(-82400.0*RT);
  k[5] = 10^11.97*Math.exp(-149800.0*RT);
  k[4] = 0.1*k[5];

  // Extract substances from substance vector
  nBr2 = n_subs[1];
  nBr = n_subs[2];
  nH2 = n_subs[3];
  nH = n_subs[4];
  nHBr = n_subs[5];

  // Reaction rate equations
  nu_reac[1] = k[1]*nBr2;
  nu_reac[2] = k[2]*nBr^2/V;
  nu_reac[3] = k[3]*nH2*nBr/V;
  nu_reac[4] = k[4]*nHBr*nH/V;
  nu_reac[5] = k[5]*nH*nBr2/V;
  nu_subs = N*nu_reac;

  // Integration
  nu_subs = der(n_subs);
end Hw12_1;
```

Call of interpolation function.

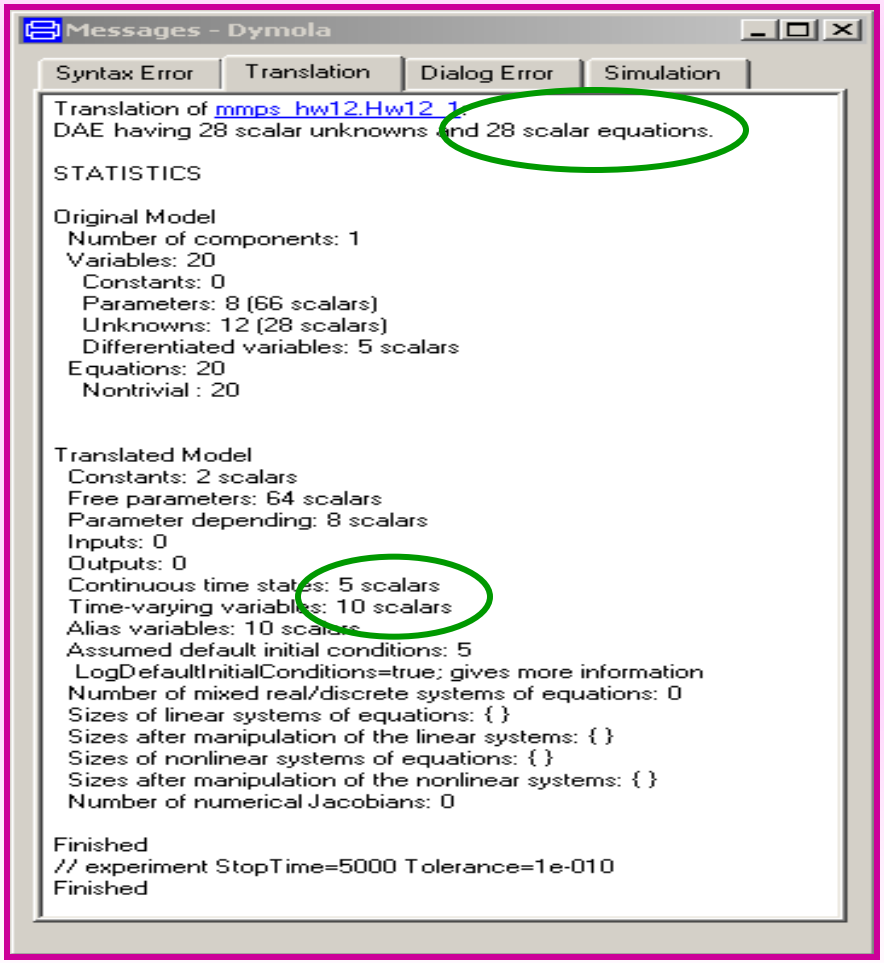
***Sqrt** is an intrinsic Modelica function, whereas **exp** is a function that is contained in the Modelica.Math library.*

Comment.

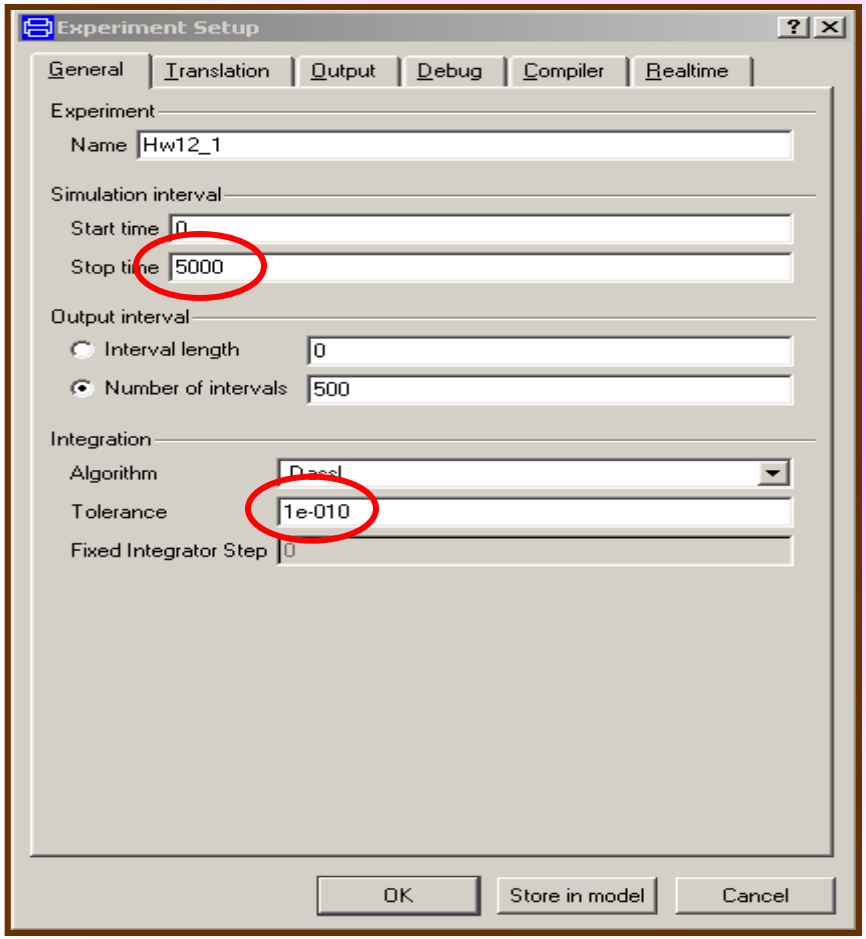
Matrix multiplication.

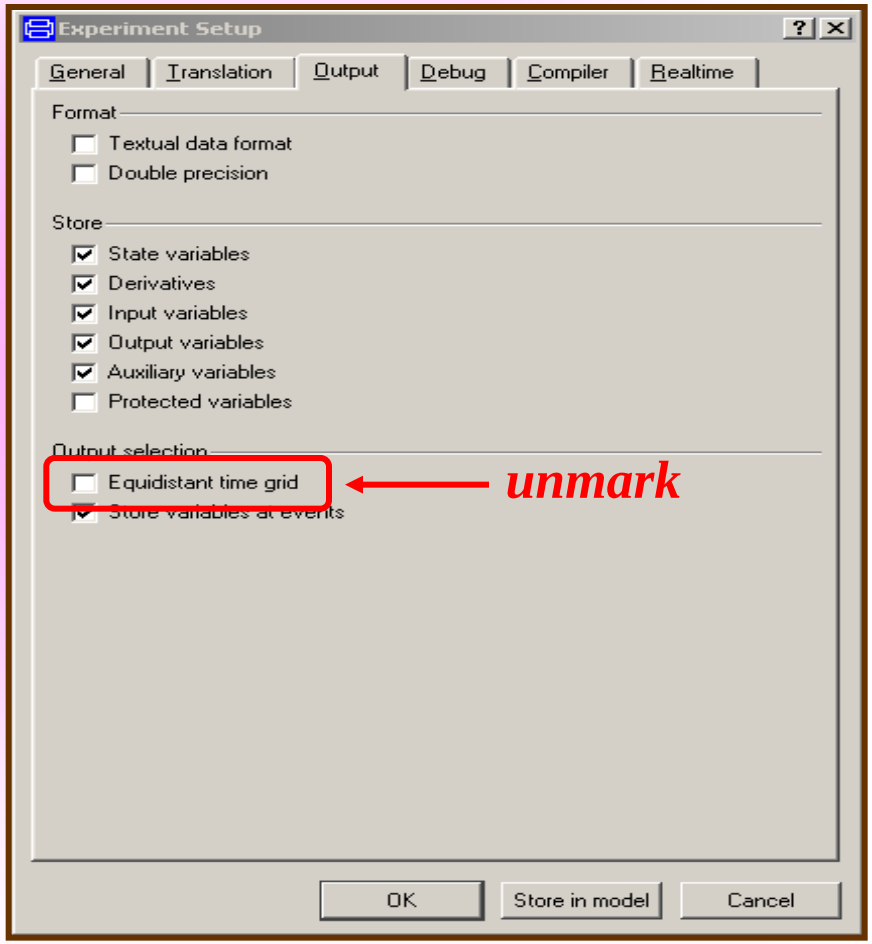
Matrix integration.

- We are now ready to compile the model.



- We require some simulation control.





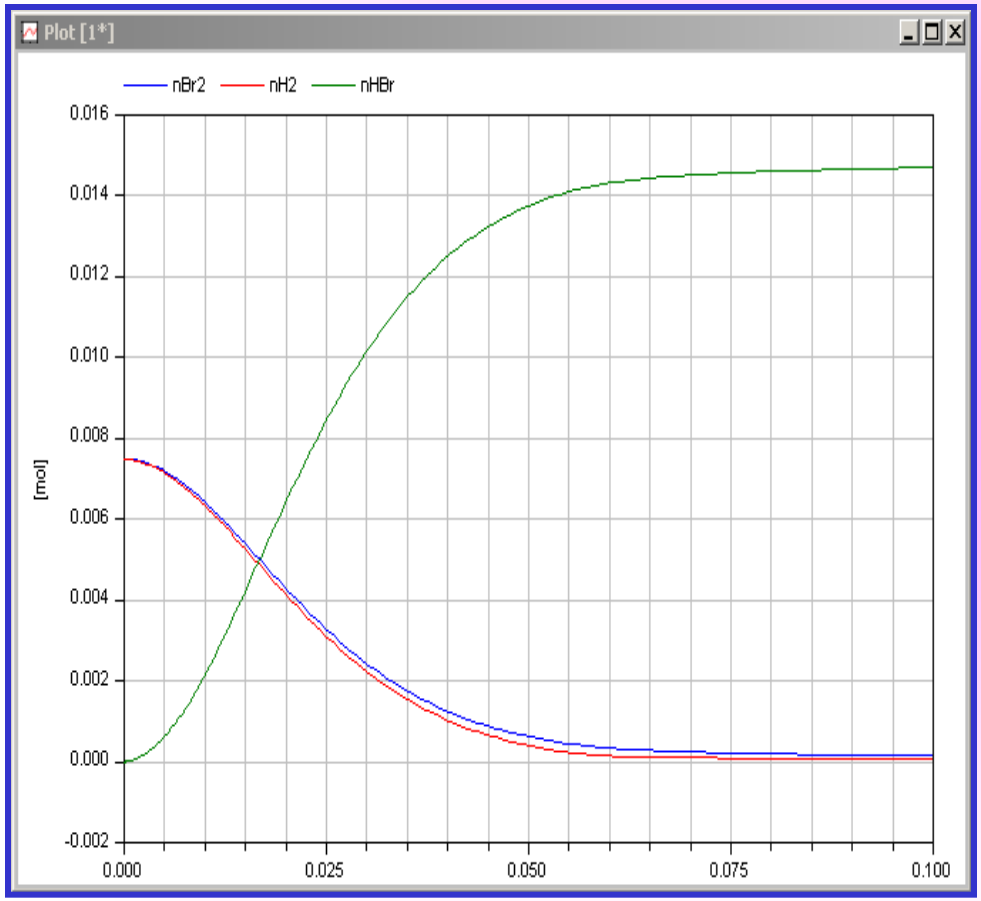
- We can now simulate the model.

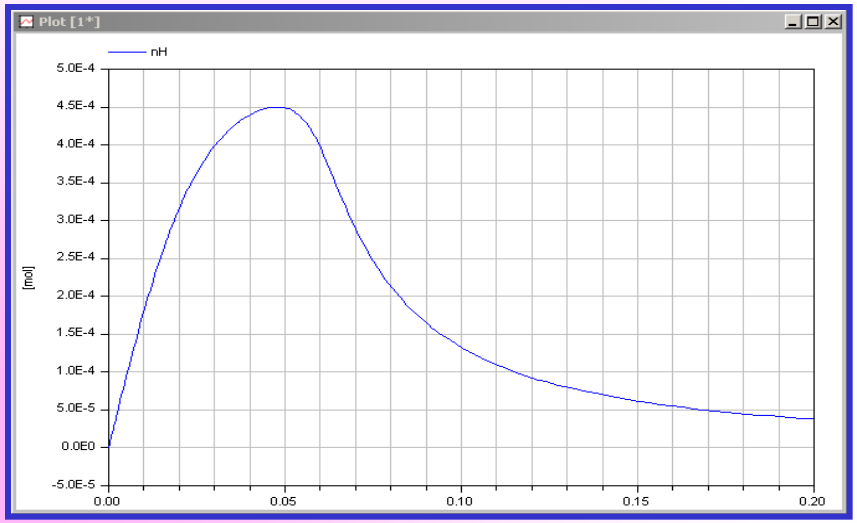
Messages - Dymola

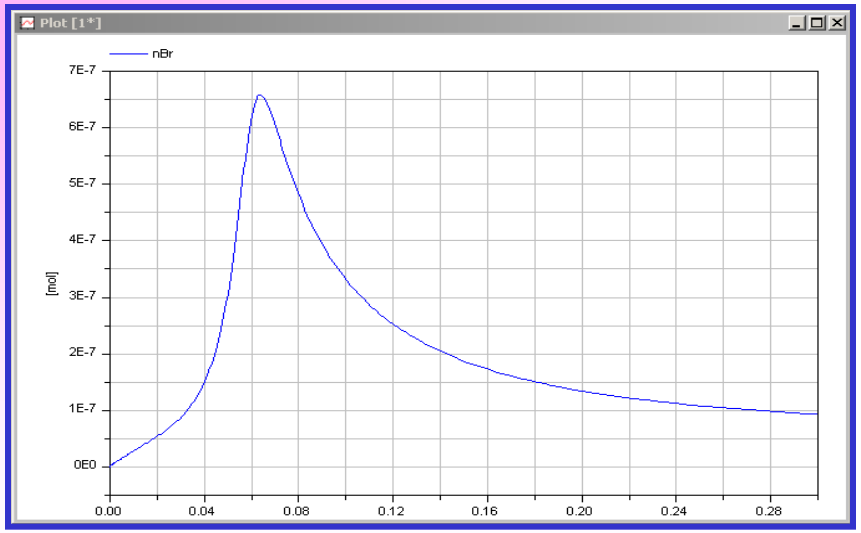
Syntax Error | Translation | Dialog Error | Simulation

```
Log-file of program ./dymosim
(generated: Wed Dec 17 14:07:47 2008)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "Hw12_1.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Integration terminated successfully at T = 5000
   CPU-time for integration      : 0.02 seconds
   CPU-time for one GRID interval: 0.0443 milli-seconds
   Number of result points       : 452
   Number of GRID   points       : 452
   Number of (successful) steps  : 451
   Number of F-evaluations       : 1186
   Number of Jacobian-evaluations: 51
   Number of (model) time events : 0
   Number of (U) time events     : 0
   Number of state    events     : 0
   Number of step     events     : 0
   Minimum integration stepsize  : 5.27e-009
   Maximum integration stepsize  : 401
   Maximum integration order     : 5
Calling terminal section
... "dsfinal.txt" creating (final states)
```

- Simulation results:







Oxy-hydrogen Gas Reaction

- When oxygen and hydrogen gases are mixed in similar proportions, a spark can bring the mixture to explosion. The process can be described by the following set of step reactions:

$$H_2 + O_2 \xrightarrow{k_0} H^\bullet + HO_2^\bullet$$
$$H_2 + OH^\bullet \xrightarrow{k_1} H^\bullet + H_2O$$
$$O_2 + H^\bullet \xrightarrow{k_2} OH^\bullet + O^\bullet$$
$$H_2 + O^\bullet \xrightarrow{k_3} H^\bullet + OH^\bullet$$
$$OH^\bullet + W \xrightarrow{a_1}$$
$$H^\bullet + W \xrightarrow{a_2}$$
$$O^\bullet + W \xrightarrow{a_3}$$

W stands for the wall. At the wall, the unstable atoms $H^\bullet$ and $O^\bullet$, as well as the unstable radical $OH^\bullet$ can be absorbed. The absorption rates are proportional to the molar masses of the absorbed species:

$$\nu_{OH^\bullet} = -a_1 \cdot n_{OH^\bullet}$$

- The reaction rate constants at the given temperature are as follows:

$k_0 = 60.0$
$k_1 = 2.3 \cdot 10^{11}$
$k_2 = 4.02 \cdot 10^{9}$
$k_3 = 2.82 \cdot 10^{12}$
$a_1 = 920.0$
$a_2 = 80.0$
$a_3 = 920.0$

Model the system in the ***Dymola equation window*** using a matrix-vector notation.

Simulate the system during 0.1 seconds. The initial conditions are $n_{H_2} = 10^{-7}$, and $n_{O_2} = 0.5 \cdot 10^{-7}$. The reaction volume is $V = 1.0\ m^3$.

You need to reduce the tolerance value of the ***DASSL integration algorithm*** to 10^{-17}.

- Plot the molar masses of H_2, O_2, and H_2O on one plot. Plot the molar masses of the other four species on separate plots.

- Let us find the ***N-matrix***:

$$H_2 + O_2 \xrightarrow{k_0} H^\bullet + HO_2^\bullet$$
$$H_2 + OH^\bullet \xrightarrow{k_1} H^\bullet + H_2O$$
$$O_2 + H^\bullet \xrightarrow{k_2} OH^\bullet + O^\bullet$$
$$H_2 + O^\bullet \xrightarrow{k_3} H^\bullet + OH^\bullet$$
$$OH^\bullet + W \xrightarrow{a_1}$$
$$H^\bullet + W \xrightarrow{a_2}$$
$$O^\bullet + W \xrightarrow{a_3}$$

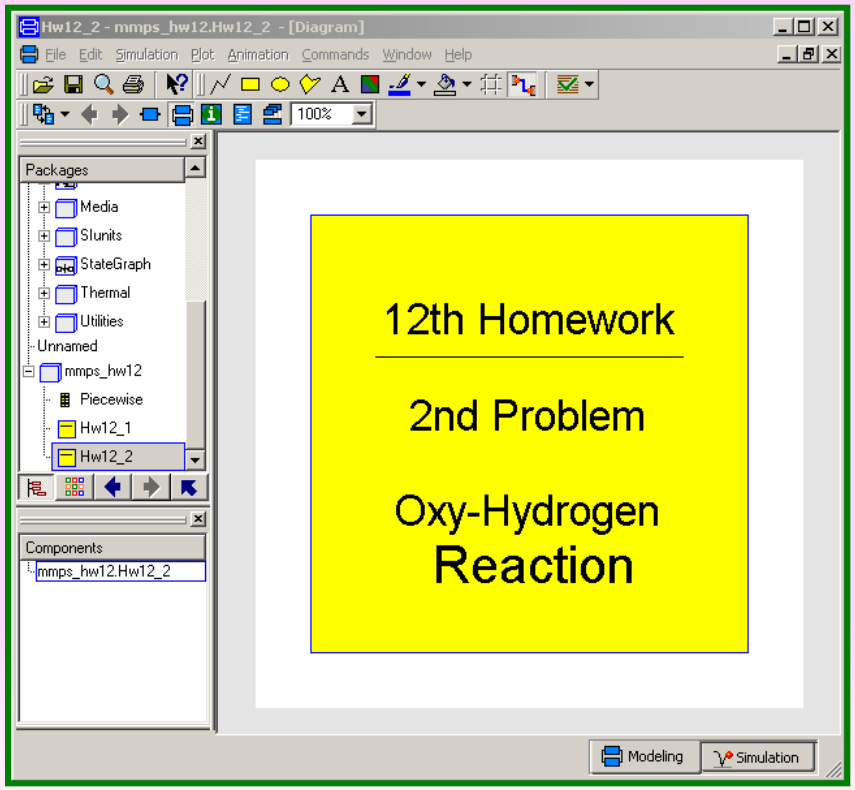
$$N = \begin{array}{c} \begin{array}{ccccccc} k_0 & k_1 & k_2 & k_3 & a_1 & a_2 & a_3 \end{array} \\ \begin{bmatrix} -1 & -1 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & -1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 1 & -1 & 0 & 0 & -1 \\ 0 & -1 & 1 & 1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \begin{array}{c} H_2 \\ O_2 \\ H_2O \\ H^\bullet \\ O^\bullet \\ OH^\bullet \\ HO_2^\bullet \end{array}$$

- Let us find the ***reaction rates***:

$$H_2 + O_2 \xrightarrow{k_0} H^\bullet + HO_2^\bullet$$
$$H_2 + OH^\bullet \xrightarrow{k_1} H^\bullet + H_2O$$
$$O_2 + H^\bullet \xrightarrow{k_2} OH^\bullet + O^\bullet$$
$$H_2 + O^\bullet \xrightarrow{k_3} H^\bullet + OH^\bullet$$
$$OH^\bullet + W \xrightarrow{a_1}$$
$$H^\bullet + W \xrightarrow{a_2}$$
$$O^\bullet + W \xrightarrow{a_3}$$

$$\nu_1 = k_0 \cdot n_{H_2} \cdot n_{O_2} / V$$
$$\nu_2 = k_1 \cdot n_{H_2} \cdot n_{OH^\bullet} / V$$
$$\nu_3 = k_2 \cdot n_{O_2} \cdot n_{H^\bullet} / V$$
$$\nu_4 = k_3 \cdot n_{H_2} \cdot n_{O^\bullet} / V$$
$$\nu_5 = a_1 \cdot n_{OH^\bullet}$$
$$\nu_6 = a_2 \cdot n_{H^\bullet}$$
$$\nu_7 = a_3 \cdot n_{O^\bullet}$$

- We are now ready to program:



Hw12_2 - mmps_hw12.Hw12_2 - [Modelica Text]

File Edit Simulation Plot Animation Commands Window Help

Packages: Media, SIunits, StateGraph, Thermal, Utilities, Unnamed, mmps_hw12, Piecewise, Hw12_1, Hw12_2

Components: mmps_hw12.Hw12_2

```
model Hw12_2
  parameter Modelica.SIunits.Volume V=1.0 "Reaction volume";
  parameter Integer nbr_subs=7 "Number of reactants";
  parameter Integer nbr_reac=7 "Number of reactions";

  //Reaction rate constants
  parameter Real k0(unit="m3/mol.s") = 60;
  parameter Real k1(unit="m3/mol.s") = 2.3e11;
  parameter Real k2(unit="m3/mol.s") = 4.02e9;
  parameter Real k3(unit="m3/mol.s") = 2.82e12;
  parameter Real a1(unit="1/s") = 920;
  parameter Real a2(unit="1/s") = 80;
  parameter Real a3(unit="1/s") = 920;

  // Reaction rate equations
  parameter Integer N[nbr_subs, nbr_reac]={{-1,-1,0,-1,0,0,0},{-1,0,-1,0,0,0,
      0},{0,1,0,0,0,0,0},{1,1,-1,1,0,-1,0},{0,0,1,-1,0,0,-1},{0,-1,1,1,-1,0,
     0},{1,0,0,0,0,0,0}};
  Modelica.SIunits.AmountOfSubstance n_subs[nbr_subs](start={1.0e-7,0.5e-7,0,0,0,0,0});
  Real nu_subs[nbr_subs](unit="mol/s");
  Real nu_reac[nbr_reac](unit="mol/s");
  Modelica.SIunits.AmountOfSubstance nH2;
  Modelica.SIunits.AmountOfSubstance nO2;
  Modelica.SIunits.AmountOfSubstance nH2O;
  Modelica.SIunits.AmountOfSubstance nH;
  Modelica.SIunits.AmountOfSubstance nO;
  Modelica.SIunits.AmountOfSubstance nOH;
  Modelica.SIunits.AmountOfSubstance nHO2;

equation
```

Line: 1 Modeling Simulation

Hw12_2 - mmps_hw12.Hw12_2 - [Modelica Text]

File Edit Simulation Plot Animation Commands Window Help

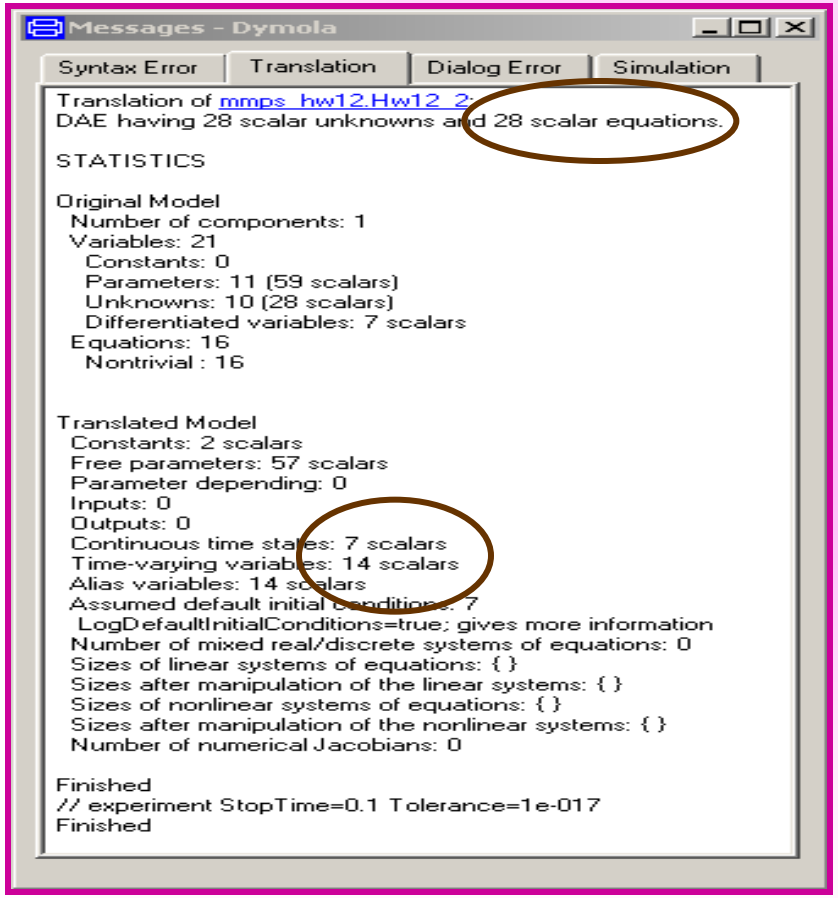
Packages: StateGraph, Thermal, Utilities, Unnamed, mmps_hw12, Piecewise, Hw12_1, Hw12_2

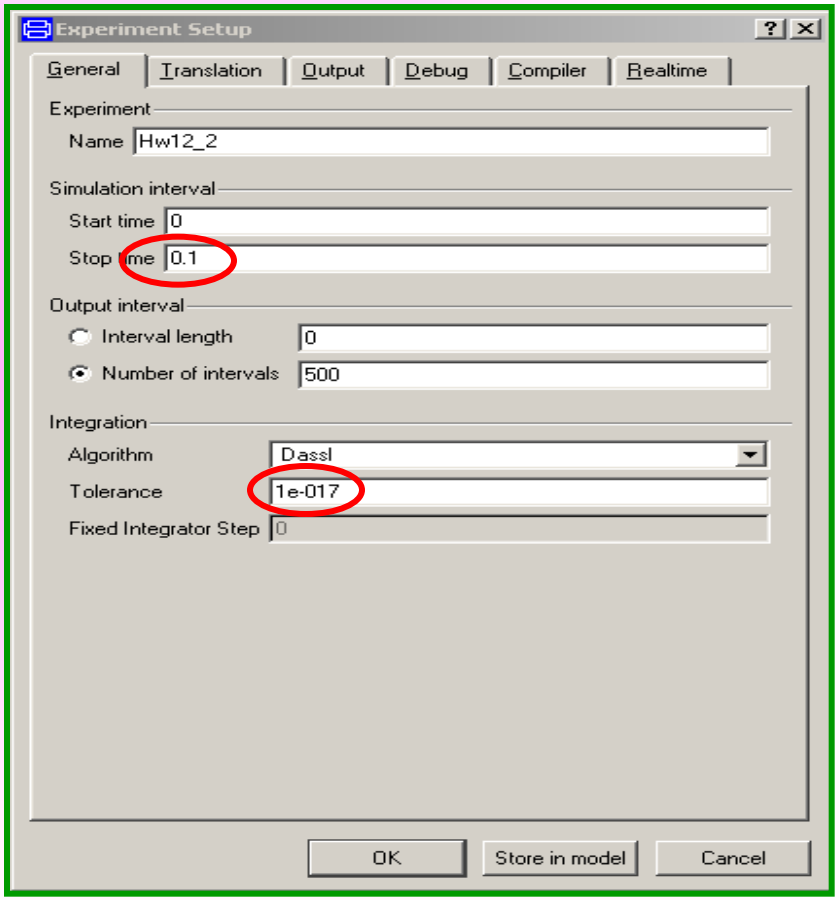
Components: mmps_hw12.Hw12_2

```
equation

  // Extract substances from substance vector
  nH2 = n_subs[1];
  nO2 = n_subs[2];
  nH2O = n_subs[3];
  nH = n_subs[4];
  nO = n_subs[5];
  nOH = n_subs[6];
  nHO2 = n_subs[7];

  // Reaction rate equations
  nu_reac[1] = k0*nH2*nO2/V;
  nu_reac[2] = k1*nH2*nOH/V;
  nu_reac[3] = k2*nO2*nH/V;
  nu_reac[4] = k3*nH2*nO/V;
  nu_reac[5] = a1*nOH;
  nu_reac[6] = a2*nH;
  nu_reac[7] = a3*nO;
  nu_subs = N*nu_reac;

  // Integration
  nu_subs = der(n_subs);
end Hw12_2;
```

Line: 1 Modeling Simulation

- Translation and simulation control:





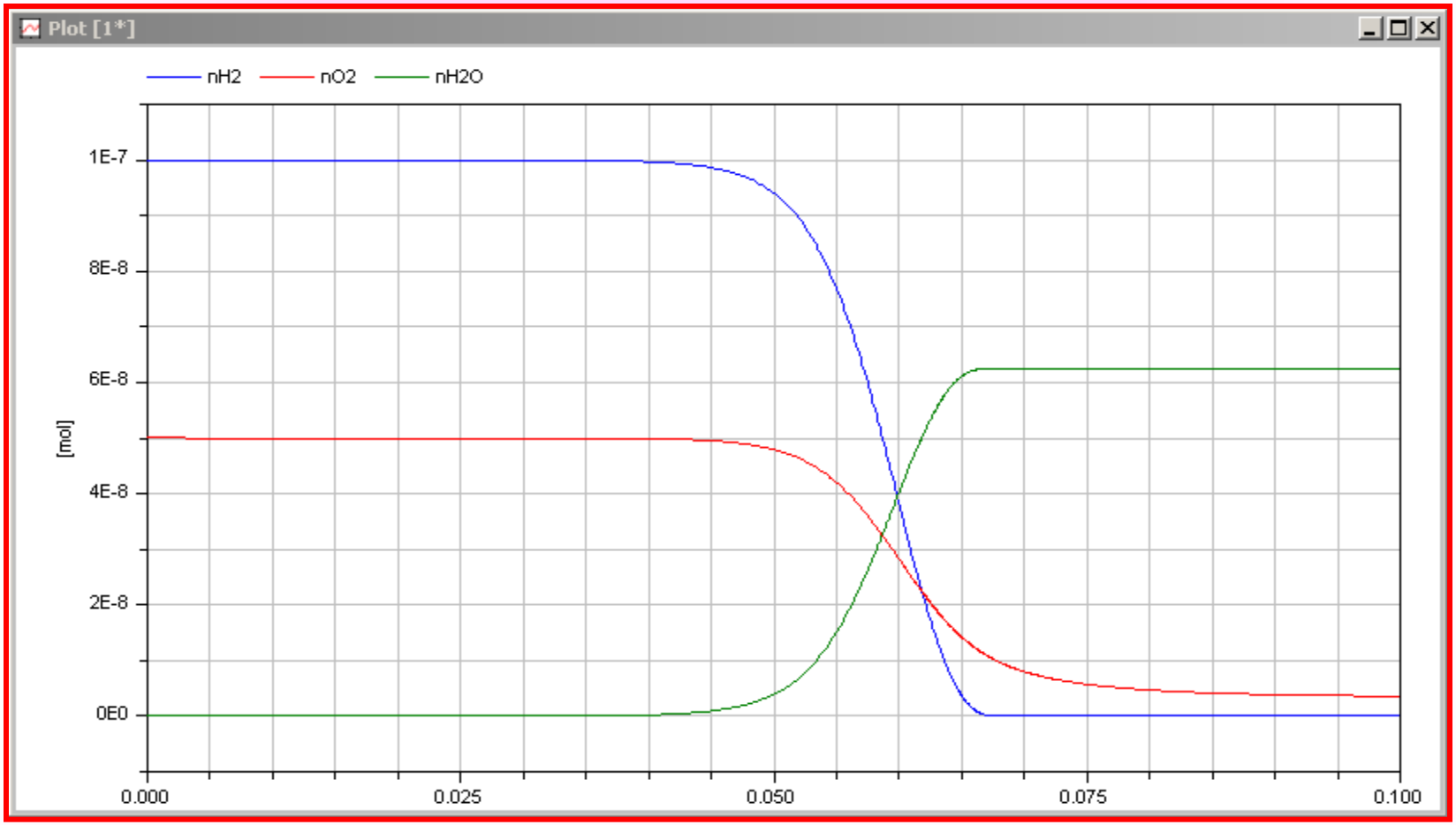
- Translation and simulation control:

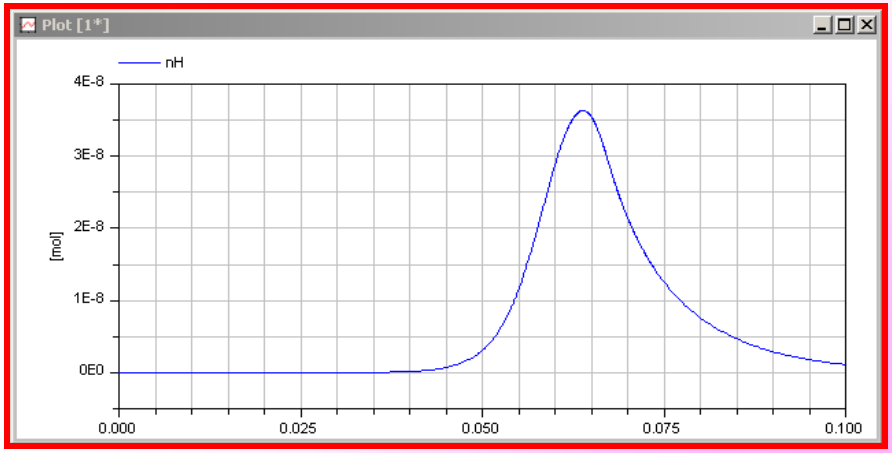
Messages - Dymola

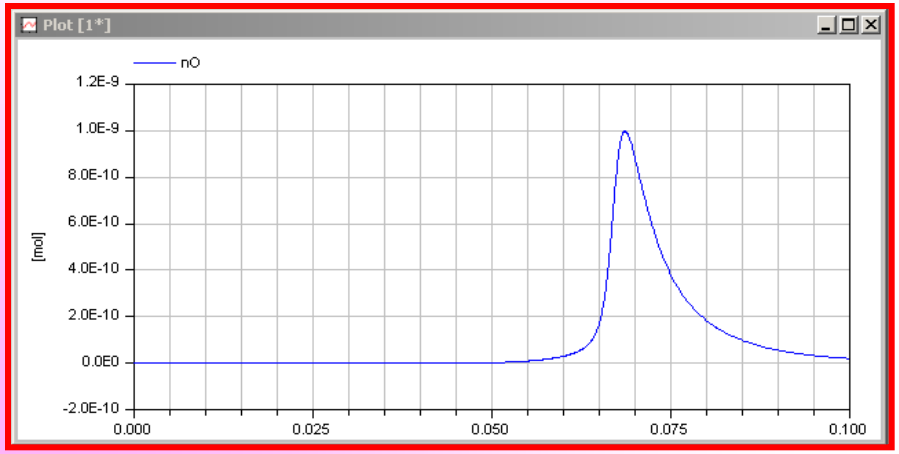
Syntax Error | Translation | Dialog Error | Simulation

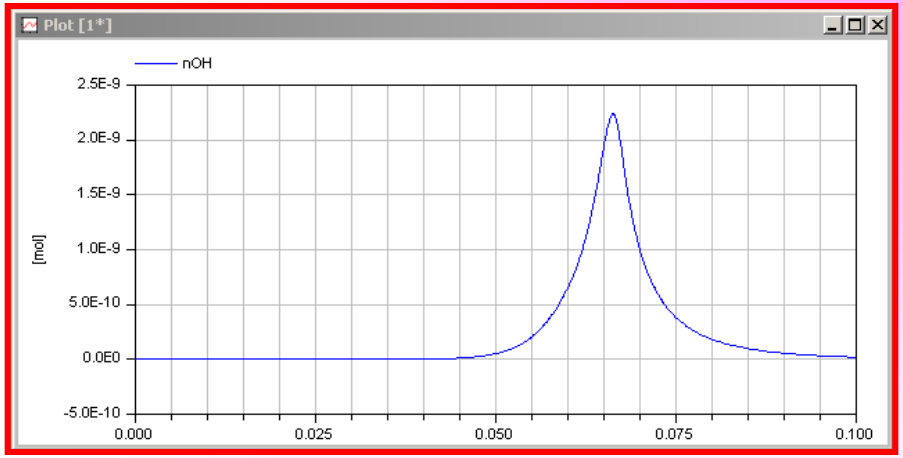
```
Log-file of program ./dymosim
(generated: Wed Dec 17 14:31:40 2008)

dymosim started
... "dsin.txt" loading (dymosim input file)
... "Hw12_2.mat" creating (simulation result file)

Integration started at T = 0 using integration method DASSL
(DAE multi-step solver (dassl/dasslrt of Petzold modified by Dynasim))
Integration terminated successfully at T = 0.1
   CPU-time for integration      : 0.04 seconds
   CPU-time for one GRID interval: 0.071 milli-seconds
   Number of result points       : 564
   Number of GRID   points       : 564
   Number of (successful) steps  : 563
   Number of F-evaluations       : 1277
   Number of Jacobian-evaluations: 18
   Number of (model) time events : 0
   Number of (U) time events     : 0
   Number of state    events     : 0
   Number of step     events     : 0
   Minimum integration stepsize  : 2.2e-005
   Maximum integration stepsize  : 0.000706
   Maximum integration order     : 5
Calling terminal section
... "dsfinal.txt" creating (final states)
```

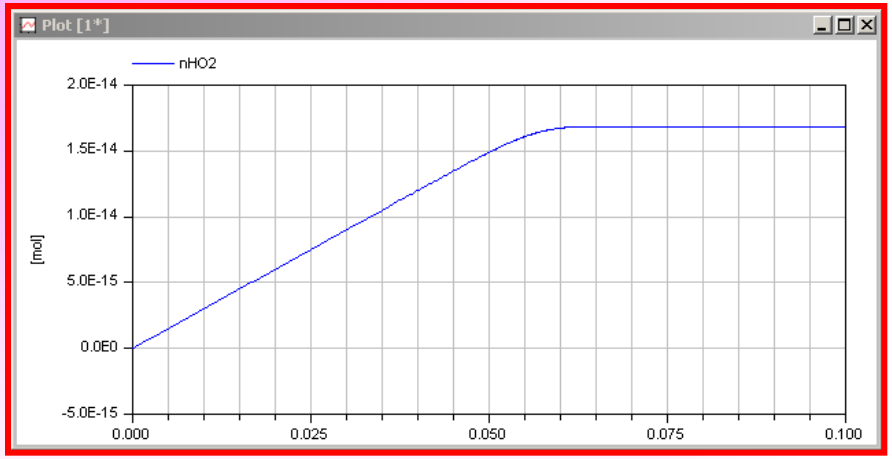
- Simulation results:











References

- Tiller, M.M. (2001), *Introduction to Physical Modeling with Modelica*, Kluwer Academic Publishers, Chapter 6.5: *Language Fundamentals*.