

# Comparing Results of Transient Simulation

E.B. Rudnyi, 2005, <http://Evgenii.Rudnyi.Ru/>, <http://www.imtek.uni-freiburg.de/simulation/mor4ansys/>

## Setting up Directory and Loading Functions

Write the directory name with the files and run

```
SetDirectory["."]
```

It is assumed that the functions are in the current directory. They have been enclosed in the archive for your convenience.

```
<< Post4MOR.m
```

The latest version of functions and the documentation can be found at <http://Evgenii.Rudnyi.Ru/soft/Post4MOR.tar.gz>. If you have *Mathematica* starting 5.1 and Internet access, the latest version can be loaded by executing

```
ToExpression[Import["http://Evgenii.Rudnyi.Ru/soft/Post4MOR/Post4MOR.m","Text"]]
```

## Loading Files

If you have different files names, please make changes accordingly.

Loading the reduced system

```
sys = ReadSystem["mor"]  
DynamicSystem[{30,1,4}, ...]
```

Loading ANSYS results for transient simulation

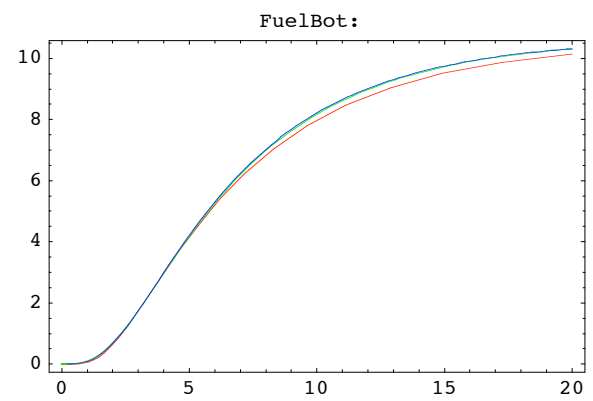
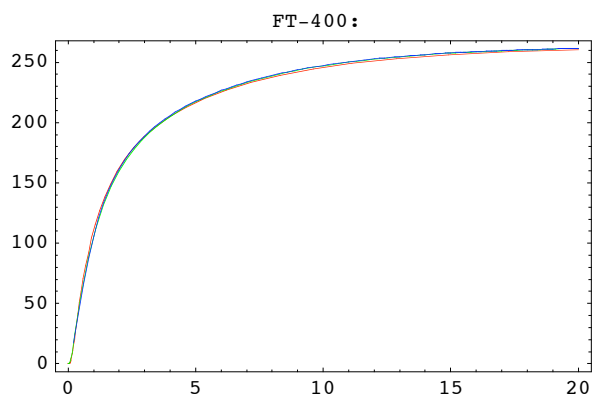
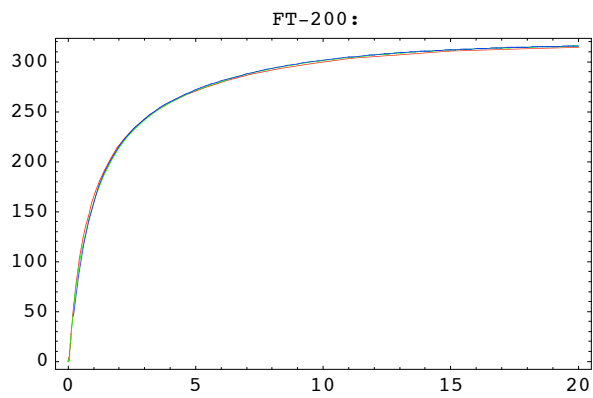
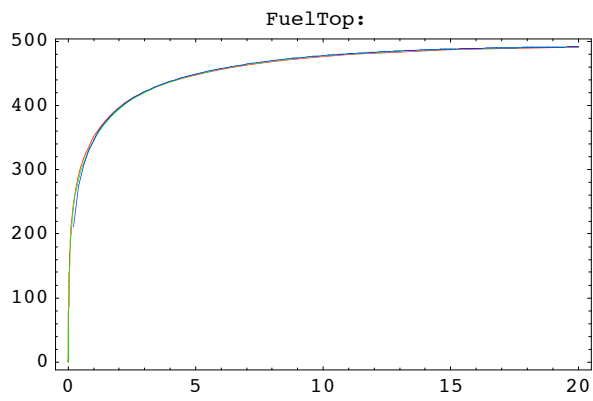
```
fixstep = ReadResult["fixstep.txt"]  
adaptive = ReadResult["adaptive.txt"]  
logtime = ReadResult["logtime.txt"]  
  
- SimulationResult -  
  
- SimulationResult -  
  
- SimulationResult -
```

## Comparing Different Integration Strategies in ANSYS

First, let us compare different simulation strategies in ANSYS to integrate in time between each other.

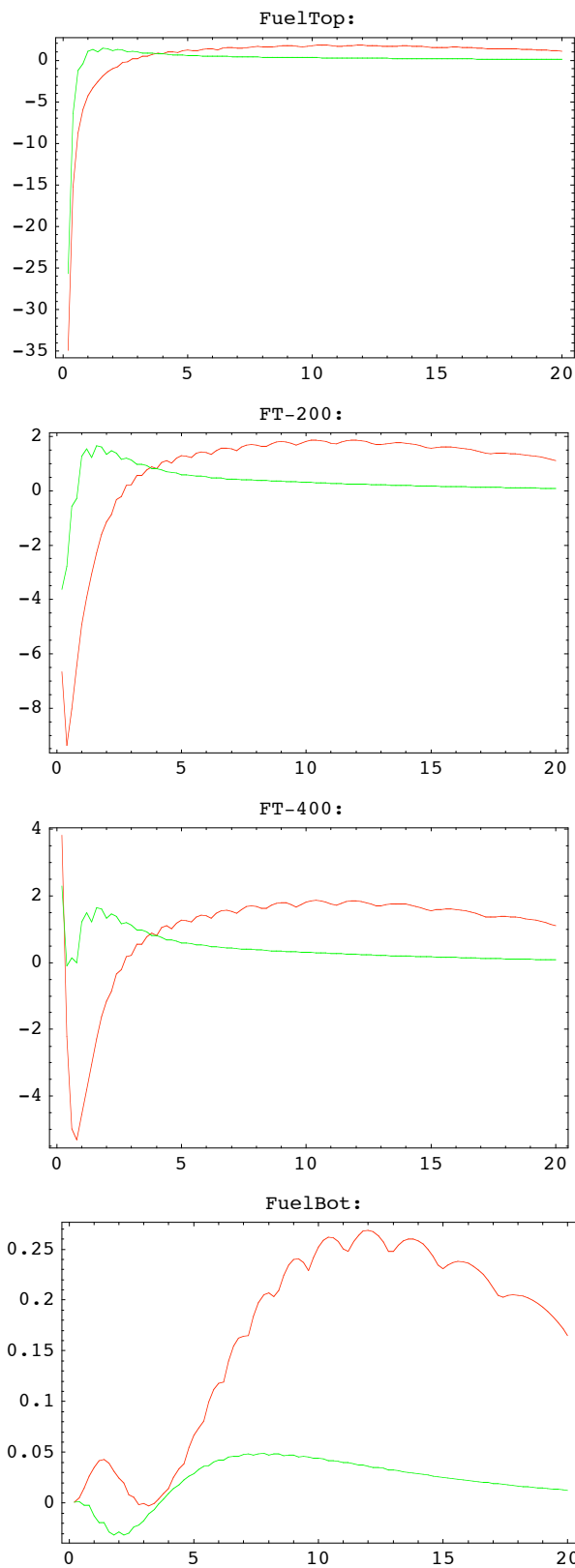
Plot temperature vs. time. logtime is red, adaptive is green, fixstep is blue.

```
PlotResult[{logtime, adaptive, fixstep},  
PlotStyle -> {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1]}];
```



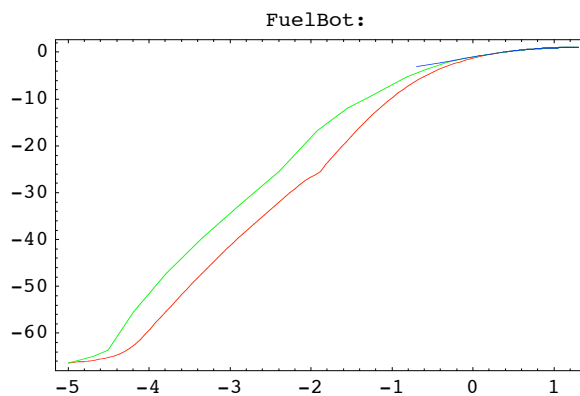
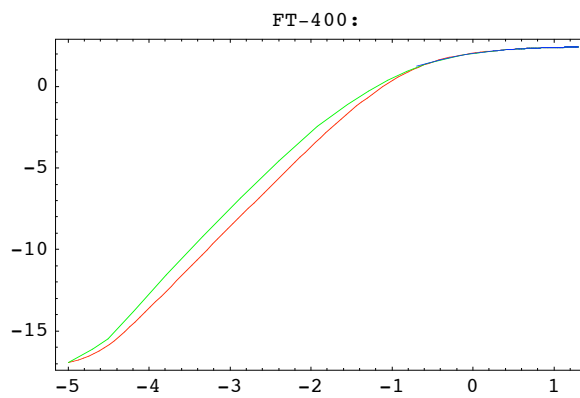
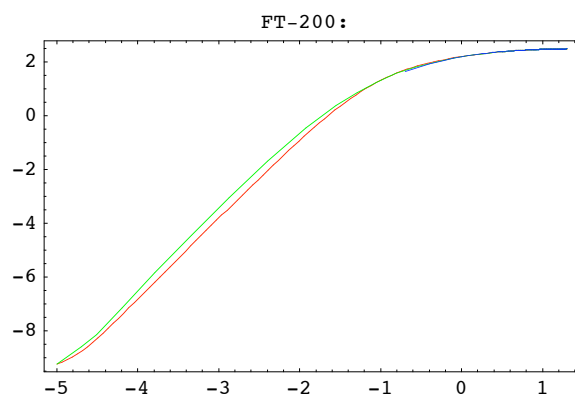
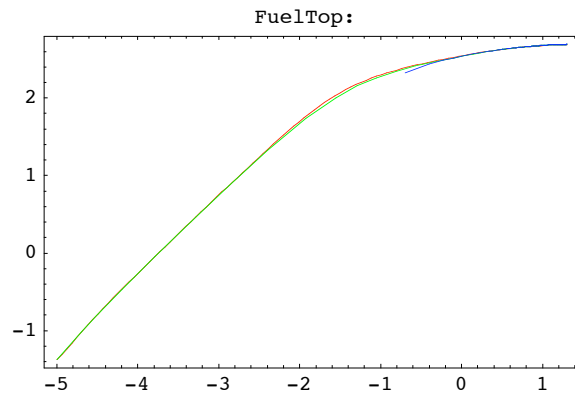
Plot difference temperature vs. time in respect to fixstep: logtime-fixstep in red, adaptive-fixstep in green.

```
PlotResult[Difference[fixstep, {logtime, adaptive}],  
PlotStyle -> {RGBColor[1, 0, 0], RGBColor[0, 1, 0]}];
```



Plot  $\text{Log}_{10}[\text{temperature}]$  vs.  $\text{Log}_{10}[\text{time}]$ . It is common to make such a plot in electro-thermal simulation. logtime is red, adaptive is green, fixstep is blue. It was necessary to take Abs as because of numerical errors there were negative values.

```
PlotResult[{logtime, adaptive, fixstep},  
  PlotStyle -> {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1]},  
  FunctionX -> Log10, FunctionY -> (Log10[Abs[#]] &)];
```



As one can see, there is considerable difference in results for different strategies. This means that the numerical integration in time introduces numerical errors and it is important to remember about this while comparing results of the

reduced model with those from ANSYS. The question what a simulation strategy for integration in time is the best is left as an exercise.

Below we make comparison of the reduced system with ANSYS results for logtime only.

## Transient Simulation of the Reduced System

In order to integrate the reduced system in time, there are two choices. First is to use the same integration method and the same integration points as in ANSYS.

```
res1 = AnsysTransientSolution[XSeries[logtime], sys]
- SimulationResult -
```

Second is to employ NDSolve (an integrator available in *Mathematica*). NDSolve chooses integration points adaptively by itself.

```
res2 = TransientSolution[XSeries[logtime], sys, Verbose -> True]
NDSolve has made 1111 steps for 2.271340 Second
- SimulationResult -
```

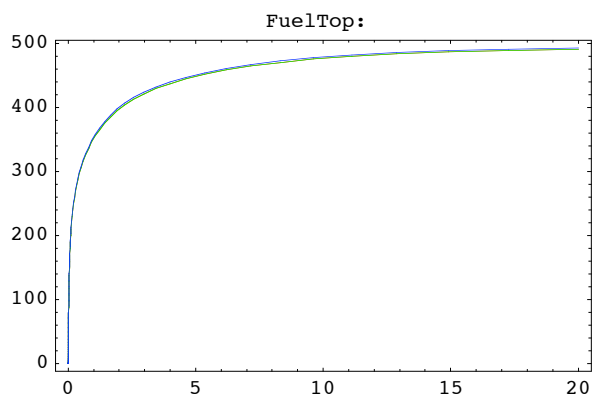
We can expect that the solution by NDSolve is more accurate but it takes more time. The simulation time can be recorded by means of function `AbsoluteTiming`.

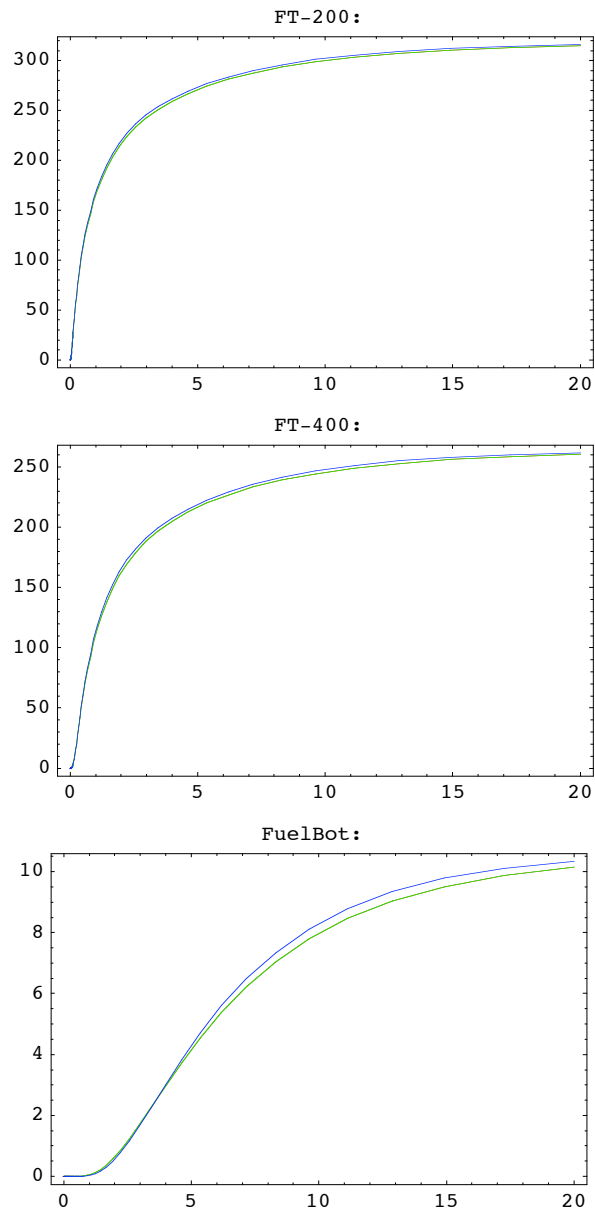
## Comparing Results of Transient Simulation of Reduced and ANSYS models

As was mentioned above, we will use logtime simulation results from ANSYS.

Plot temperature vs. time. ANSYS model is red, reduced model is green for res1 and blue for res2.

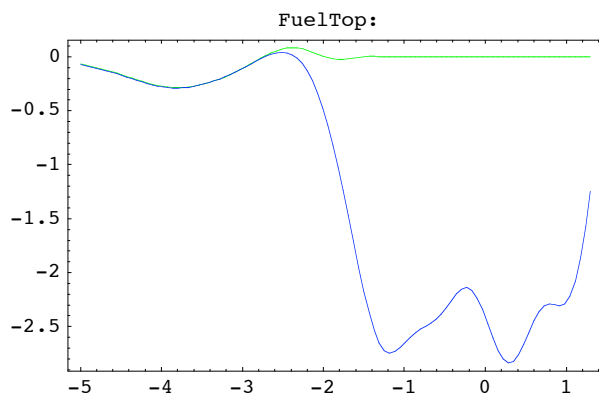
```
PlotResult[{logtime, res1, res2},
  PlotStyle -> {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1]};
```

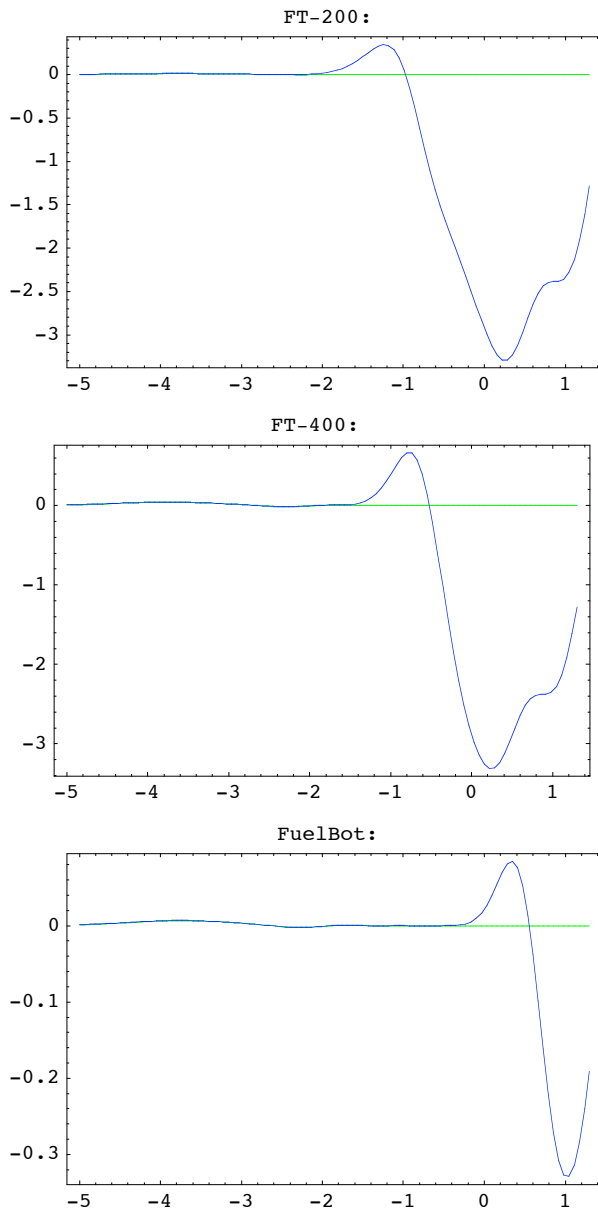




Plot difference temperature vs.  $\text{Log}_{10}[\text{time}]$  in respect to ANSYS.

```
PlotResult[Difference[logtime, {res1, res2}],  
PlotStyle -> {RGBColor[0, 1, 0], RGBColor[0, 0, 1]}, FunctionX -> Log10];
```

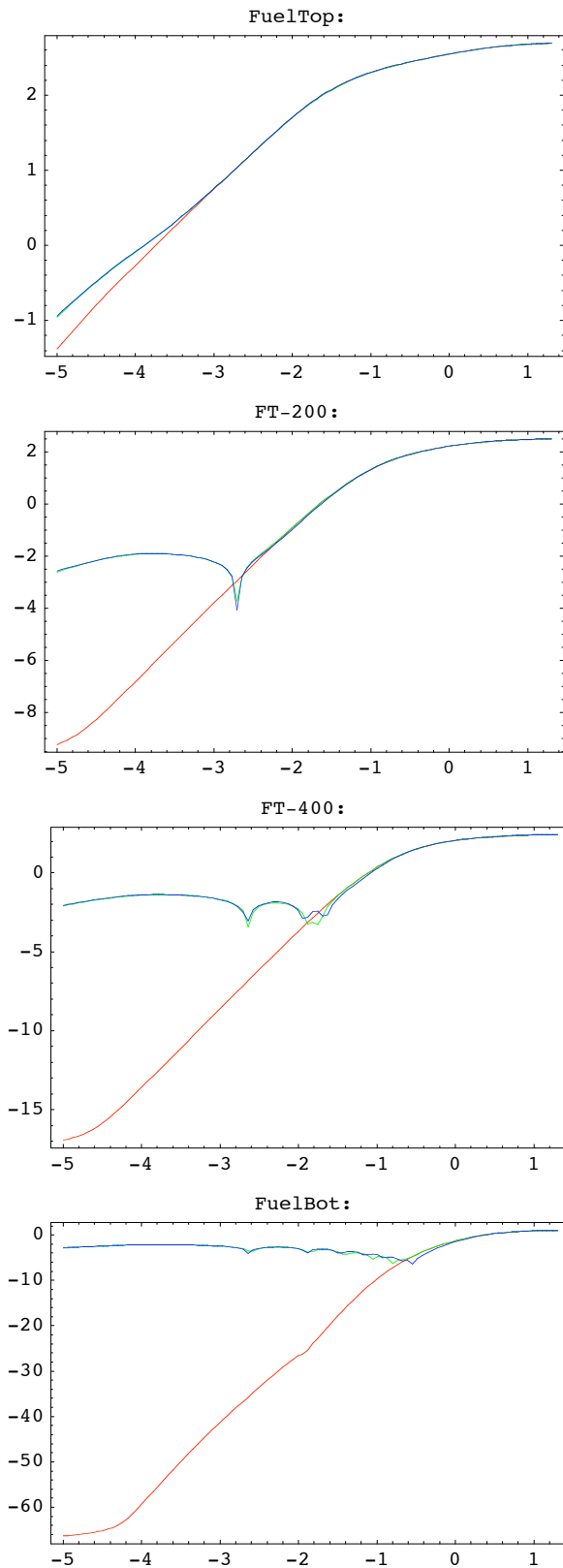




It is interesting to see that the difference between res1 and ANSYS results is smaller than for res2. However, it should be noted that res2 is more accurate than res1! The explanation is simple. When we use `AnsysSimulationResult`, it introduces about the same numerical errors during the time integration as ANSYS by itself. Note that the difference between res2 and ANSYS is comparable with that between `logtime` and `adaptive` for ANSYS.

Plot  $\text{Log}_{10}[\text{temperature}]$  vs.  $\text{Log}_{10}[\text{time}]$ . It is a common plot in electro-thermal simulation. ANSYS model is red, reduced model is green for res1 and blue for res2. It was necessary to take `Abs` as because of numerical errors there were negative values.

```
PlotResult[{logtime, res1, res2},
  PlotStyle → {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1]},
  FunctionX → Log10, FunctionY → (Log10[Abs[#]] &)];
```



The last plots reveals that the reduced model overestimate temperatures at the beginning. This was hidden above as the absolute error is actually quite small.

## Local Error Indicator

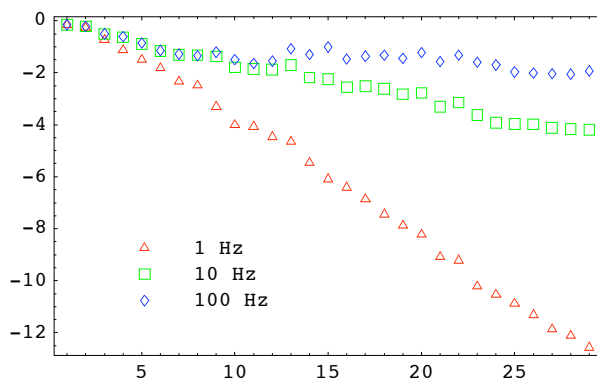
We can estimate local error and choose an optimal dimension of the reduced system as described in paper Error indicators for fully automatic extraction of heat-transfer macromodels for MEMS.

Compute an error indicator for 1, 10, and 100 Hz.

```
er1 = LocalErrorIndicator[FrequencyConvergence[1, sys]];
er10 = LocalErrorIndicator[FrequencyConvergence[10, sys]];
er100 = LocalErrorIndicator[FrequencyConvergence[100, sys]];
```

Make a plot Log10[Relative Error] vs. system dimension

```
MultipleListPlot[{er1, er10, er100},
  SymbolShape -> {PlotSymbol[Triangle, Filled -> False],
    PlotSymbol[Box, Filled -> False], PlotSymbol[Diamond, Filled -> False]},
  SymbolStyle -> {RGBColor[1, 0, 0], RGBColor[0, 1, 0], RGBColor[0, 0, 1]},
  PlotLegend -> {"1 Hz", "10 Hz", "100 Hz"}, LegendBorder -> {},
  LegendPosition -> {-0.7, -0.4}, LegendSize -> {0.8, 0.28},
  Evaluate[Imtek`Post4MOR`Private`defaultPlotOptions]];
```



What a frequency and what tolerance to choose in order to determine an optimal model dimension depends on an engineering problem.

## Using SLICOT for Sequential Model Reduction

A reduced model obtained by MOR for ANSYS can be reduced even further. For details, see paper cited in the previous section.

SLICOT (<http://www.slicot.de/>) is a library implementing several Grammian-based methods for model reduction. Mathlink interface to SLICOT is available at <http://evgenii.rudnyi.ru/soft/slicot.tar.gz>. In the documentation, you will find an example of the application of SLICOT to the microthruster model.