dhcaeLTSThermoParcelSolver

Solver and Test Cases

Martin Becker

martin_becker@dhcae-tools.de

06/13/2012

Abstract

This is a short description of the dhcaeLTSThermoParcelSolver, which can be used to simulate the cooling effect of evaporating droplets of water in a quenching device, and a desciption of a series of test cases for the solver.
The focus of the content lies on the Local Time Stepping (LTS) approach as an acceleration technique for problems which have a steady state solution. Next to the description of the idea behind local time steps some advice is given on the usage of the numerous parameters controlling the LTS method.
While the reduction of computational time can be outstanding with a LTS based solver, especially the visualization of the Lagrangian particles is a problem. To overcome this drawback the combination of the dhcaeLTSThermoParcelSolver with a PISO/PIMPLE based solver is presented.

Contents

Trademarks

OpenFOAM is a registered trademark of SGI Corp.

Chapter 1

Introduction

1.1 Motivation

Let us take a toxic waste incineration plant. The waste is burned at very high temperatures and the resulting flue gas is an awful mixture of toxic gases. Sometimes a thermal utilization is even omitted due to the high effort required. However it is necessary to do a gas scrubbing before emitting the flue gas to the environment. The facilities for cleaning the flue gas need a much lower gas temperature to do their job. Therefor an important procedure is to cool down the flue gas to the desired target temperature. This is done by a quenching device, wherein droplets of water are injected into the hot gas and the evaporative cooling effect is used. The quench and some of the final results are presented in figure 1.1. The simulation of such a quenching device was the motivation for the modification of the "LTSReactingParcelFoam" solver to become the "dhcaeLTSThermoParcelSolver" solver.
Due to the very special objective of simulating the quenching device there are some simplifications possible. For example it can be granted that the water droplets do not hit the wall. Therefor a special treatment of wall films is not necessary. Furthermore it is known, that the water droplets evaporate before reaching their boiling temperature. So the standard liquid evaporation of OpenFOAM can be used.
To do extensive studies on the solver and investigate its behaviour it was necessary to simplify the original quench. As shown in figure 1.2 the resulting geometry can be meshed with hexahedral elements and gives the possibility to make use of symmetry planes or even wedge elements. The simplified meshes and cases are provided to the community.

1.2 About the solver

dhcaeLTSThermoParcelSolver is a solver for steady, compressible, laminar or turbulent non-reacting flow with multiphase Lagrangian particles. It uses the Local Time Stepping (LTS) approach to speed up the convergence.
The focus of this solver is to model the strong coupled thermal interaction of evaporating water droplets in a quenching device for cooling of hot gas. In addition to the momentum and thermal coupling there is a mass transfer between the dispersed and the continuous phase.

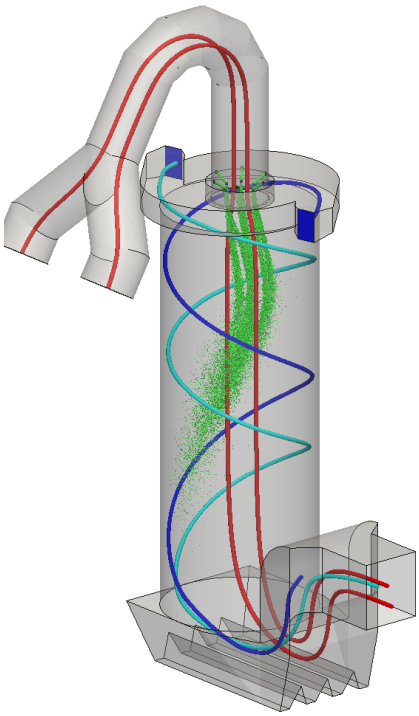
1.2.1 The origins

The dhcaeLTSThermoParcelSolver is derived from LTSReactingParcelFoam. In the anouncement of the solver[1] it was described as a prototype solver that requires further testing and/or development. This was done in a project dealing with the quenching device described above.

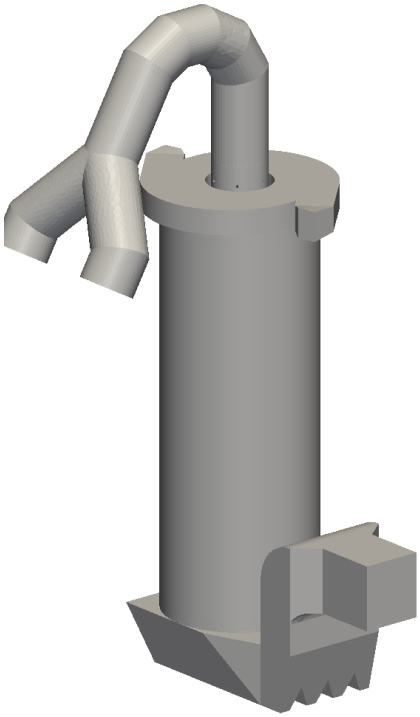
1.2.2 Removed features

In comparison to the original LTSReactingParcelFoam several features are removed:

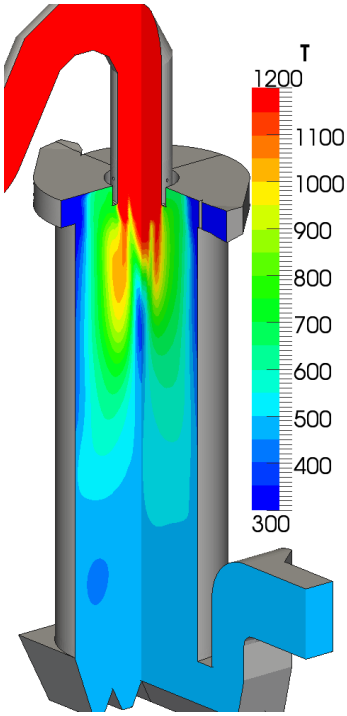
[1] http://www.openfoam.org/version2.0.0/lagrangian.php



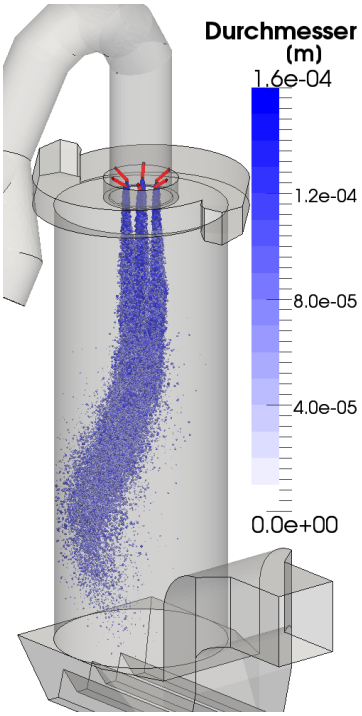
(a) Concept of the quench: The way of the hot gas is indicated by the red lines. A secondary stream of air (blue) causes a twist to keep droplets (green) and hot gas in the center of the tube and away from the quench's wall.



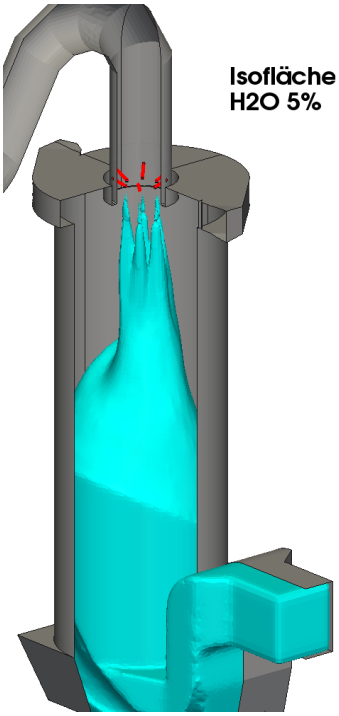
(b) General view of the quench. The cylinder has a height of some 19m and a diameter of 7m.



(c) Temperature [K]

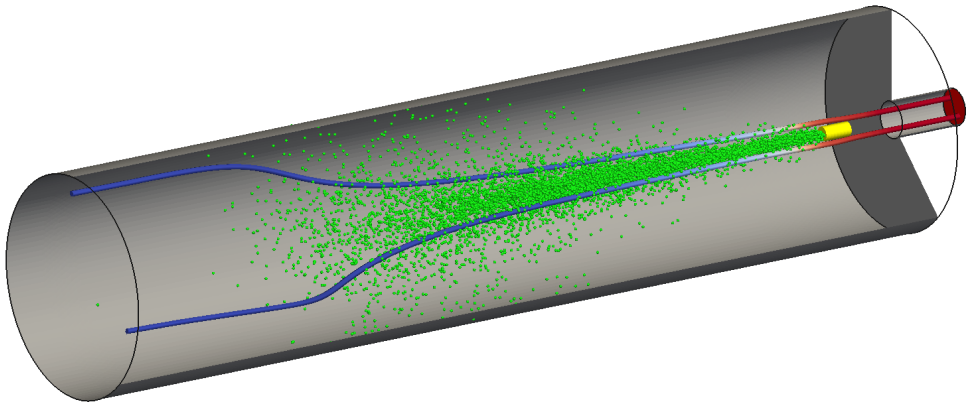


(d) Particle cloud with the diameter of the water droplets.

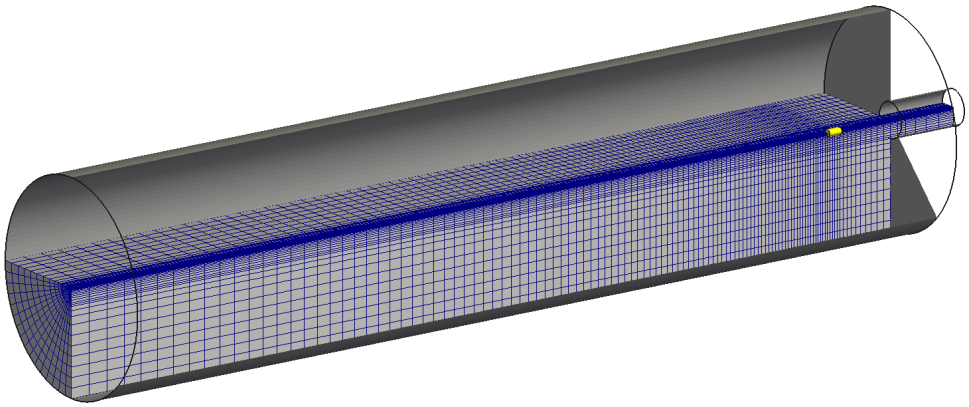


(e) Iso surface 5% H2O fraction.

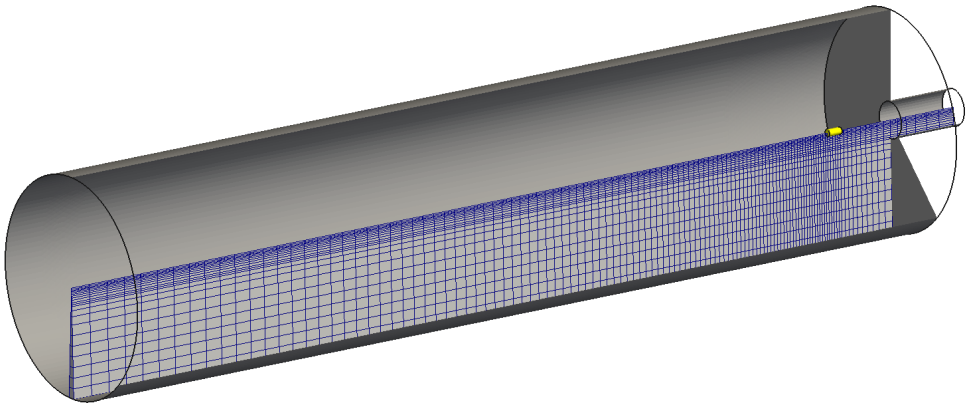
Figure 1.1: A quenching device for cooling of hot gas.



(a) A single levitating nozzle (yellow) is placed on the center line of a tube.



(b) 90° segment with symmetryPlane.



(c) 4° segment with wedge mesh.

Figure 1.2: Simplifications of the quenching device. Symmetry features are used to reduce the mesh size.

- porous zones
- chemical reactions
- explicit source terms
- radiation

The main reason to do this was to focus the solver on the main goal, namely the simulation of the quenching device. The number of text files to be provided for the solver was reduced, the necessary features got clearly visible.
However it should be easy to add one or all of these removed functions again, just have a look at the current LTSReactingParcelFoam and its implementation.

1.2.3 New features

There are several new features added to the basic solver, too:

- dhcaeLTSThermoParcelSolver uses hsPsiThermo classes instead of hRhoThermo classes. Therefor you can use the same material definition as with reactingParcelFoam.
- The parameters to manage the Local Time Stepping (LTS) procedure are moved into the system/-controlDict file. Further more the values for maxCourant, rDeltaTSmoothingCoeff, maxDeltaT and alphaTemp are interpolated from a user defined table. The idea behind is to use stable but slow parameters in the early stage of convergence, and change these values in the direction of faster convergence later on.
- A bit of additional timing is added, so that one can easily check how much computational time the particle tracking part needs, and how much the solving of the continuous phase takes.

1.3 The Local Time Stepping (LTS) approach

For the PISO algorithm to be stable it is necessary to have the maximum Courant number stay below 1 or even lower (see CFL condition on Wikipedia). The Courant number in its general form is defined as

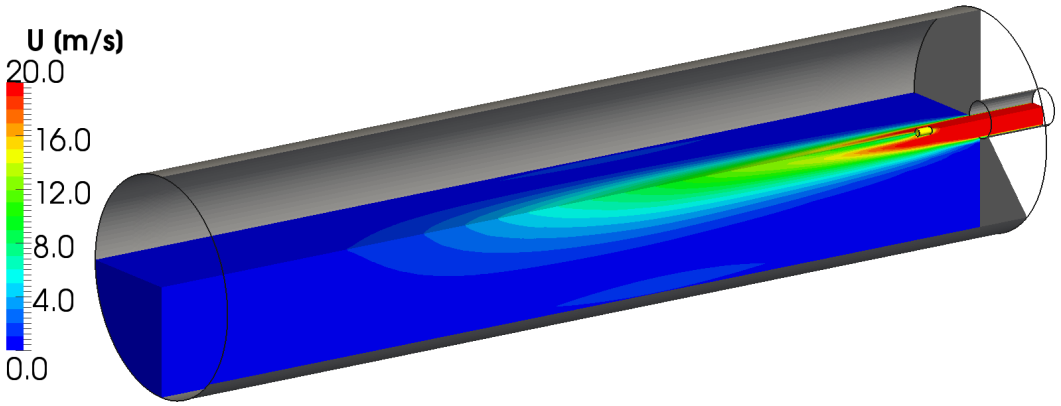
$$Co = \frac{|u| \cdot \Delta t}{\Delta x}$$

with velocity u, time step Δt and length interval Δx.
With the definition of a maxCo value and enabling the adjustTimeStep option in the system/controlDict file it is possible in many OpenFOAM solvers to use a global time step that fulfills the CFL condition. This global time step is adjusted continuously at the beginning of an iteration:

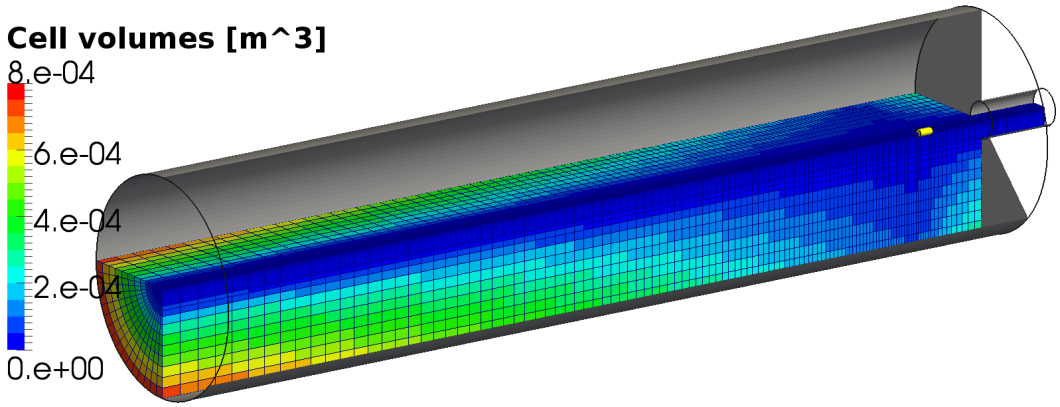
$$\Delta t^{(n+1)} = min(\frac{maxCo \cdot \Delta x_i}{|u^{(n)}|})$$

The global time step is used throughout the entire computational domain for the actual iteration. The big problem of this approach is in short: the smallest cell with the highest velocity determines the speed of the simulations progress! Most of the cells could be driven with a much larger time step, but they are not allowed to do so.
At this point the Local Time Stepping approach comes into play. The idea is to use an individual time step of the maximum possible value for each cell. So instead of using a global maximum time step the LTS technique uses a global maximum Courant number. The individual time step for cell with index "i" is:

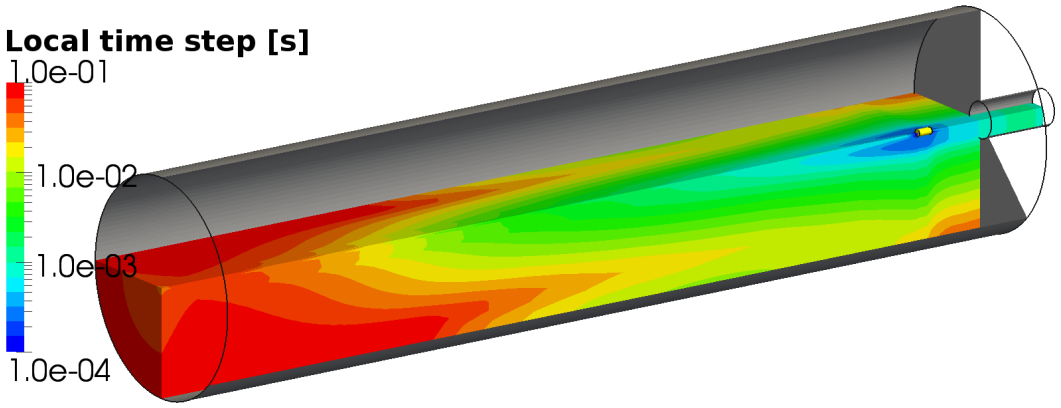
$$\Delta t_i^{(n+1)} = \frac{maxCo \cdot \Delta x}{|u_i^{(n)}|}$$



(a) Velocity U [m/s]



(b) Cell volumes [m³]



(c) Local time steps [s]

Figure 1.3: The cell velocity is combined with the maximum global Courant number and the local cell size. The result after an additional smoothing step is a field for the local time steps.

Informations in the flow field are transported much faster through the domain. The LTS approach can be used if a steady state solution exists and if you are not interested in the intermediate steps. Before reaching the steady state the solution is most likely invalid and possibly even physically nonsense. Finally let's have a look an the way the local time step is calculated by the solver in figure 1.3.

Chapter 2

Preparations

2.1 Compiling the dhcaeLTSThermoParcelSolver

I assume, that you were able to download the sources of dhcaeLTSThermoParcelSolver already. A good place to unpack the solver is your OpenFOAM's user directory. Quite often this is the directory "~/OpenFOAM/username-2.1.x". So the dhcaeLTSThermoParcelSolver solver should be placed at "~/OpenFOAM/username-2.1.x/dhcaeLTSThermoParcelSolver".
Open a shell and navigate to this location. Activate your OpenFOAM environment and call "wmake". That's it. You can check the success of the compilation by typing "which dhcaeLTSThermoParcelSolver", which should return the path to the newly created binary executable.

2.2 Usage of rotational symmetry (wedge elements)

There are two series of cases provided with the dhcaeLTSThermoParcelSolver. One of them uses a very small wedge based mesh. However in the standard OpenFOAM 2.1.x release the treatment of the situation, where a particle hits the symmetry face of a wedge element, is to give a fatal error and shutdown the simulation.
Indeed, there are good reasons to do so, have a look at this thread:
http://www.cfd-online.com/Forums/openfoam-solving/62970-lagrangian-track-not-support-empty-patch-interaction.html
This situation of a "hitWedgePatch" arises when using dhcaeLTSThermoParcelSolver on the example geometries used in this text. Since we were able to get satisfying results with dhcaeLTSThermoParcelSolver on a wedge mesh, you might want to change the OpenFOAM sources a bit. An alternative handling of the hitWedgePatch is already prepared and you can perform the changes in the source code quickly.

- Open the file "OpenFOAM-2.1.x/src/lagrangian/basic/particle/particleTemplates.C" with your favorite text editor.
- Search for "Foam::particle::hitWedgePatch" in the file.
- Comment out the lines with the "FatalErrorIn" call, so that the following lines of code, which do a simple reflection of the particle, get active.
- Open a shell and activate your OpenFOAM-2.1.x environment. Navigate to the location "OpenFOAM-2.1.x/src/lagrangian/basic", type in a "wclean" and a "wmake libso".
- Recompile the dhcaeLTSThermoParcelSolver as described in section 2.1.
- If you want to run reactingParcelFoam (original PISO based solver of the OpenFOAM-2.1.x assembly), for example for veryfication purpose or to create fancy animations with particles flying aroung, then you must recompile this solver, too: open a shell, navigate to "OpenFOAM-2.1.x/applications/solvers/lagrangian/reactingParcelFoam/" and type "wclean" and "wmake".

The resulting particleTemplates.C code snippet should look like this:

```
template<class TrackData>
void Foam::particle::hitWedgePatch
(
    const wedgePolyPatch& wpp,
    TrackData&
)
{
    /*
    FatalErrorIn
    (
        "void Foam::particle::hitWedgePatch"
        "("
            "const wedgePolyPatch& wpp, "
            "TrackData&"
        ")"
    )   << "Hitting a wedge patch should not be possible."
        << abort(FatalError);
    */
    vector nf = normal();
    nf /= mag(nf);
    transformProperties(I - 2.0*nf*nf);
}
```

Chapter 3

General workflow

3.1 Initialization with rhoSimpleFoam

It's a good idea to initialize the dhcaeLTSThermoParcelSolver with the standard rhoSimpleFoam solver first. Especially the U file is quite useful to prevent the dhcaeLTSThermoParcelSolver to struggle with the particle movement too much in the early stages of convergence. It's not necessary to have fully converged rhoSimpleFoam simulation; in the cases provided here 1000 iterations are assumed to be sufficient.

3.2 Simulation with dhcaeLTSThermoParcelSolver

As soon as the rhoSimpleFoam solver has done the initialization, the results are mapped to the 0 directory of the dhcaeLTSThermoParcelSolver.

3.2.1 Parameters for the LTS management

There are several parameters the manage the LTS procedure. In contrast to the original LTSReactingParcelFoam the definition of the relevant settings are placed in the system/controlDict. Moreover the values do not need to be constant for the complete simulation, but they can be interpolated from a table. Here is a snippet of the system/controlDict:

```
_maxCourantTable constant 0.5; // this entry is "deactivated" with the underscore
maxCourantTable table // replaces the maxCo entry in fvSolutions PIMPLE dictionary
(
    (0     0.1) // start with a stable value
    (500   0.2) // increase it until iteration 500
    (20000 0.2) // stay at 0.2 until the end
);
rDeltaTSmoothingCoeffTable constant 0.5;
_rDeltaTSmoothingCoeffTable table // this entry is "deactivated" with the underscore
(
    (0 0.2)
    (500 0.5)
    (20000 0.5)
);
maxDeltaTTable constant 1000000;
_maxDeltaTTable table // this entry is "deactivated" with the underscore
(
    (0 1.0)
    (500 100000.0)
    (20000 100000.0)
);
_alphaTempTable constant 0.1;
alphaTempTable table // replaces the alphaTemp entry fvSolutions PIMPLE dictionary
(
    (0     0.2) // start with a stable value
    (500   0.5) // increase it until iteration 500
    (20000 0.5) // stay at 0.5 until the end
);
```

maxCourantTable: The value for the maximum Courant number is interpolated from this table. A typical usage would be to start with a small maxCo value to have a stable beginning of the simulation. The maxCo might then be increased after a couple of iterations. It might be useful to reduce the maxCo again when reaching the final stages of the simulation. Instead of using the table you can define a constant value here, too. Keep in mind, that this maxCo is responsible for the continuous phase firsthand. The maxCo for the dispersed phase it managed in the constant/reactingCloud1Properties file.

rDeltaTSmoothingCoeffTable: To avoid abrupt transitions from small local time steps to large local time steps in neighbored cells the field with the reciprocal value of the time step is smoothed by this parameter. This is especially useful when using tetrahedral cells or more general a mesh with large local differences in cell sizes. A value of 1.0 does no smoothing, so each cell has its own time step fitting the maxCo number. A value of zero will use a unique time step for the complete domain.

maxDeltaTTable: You can limit the maximum time step allowed. If you have dead zones in your domain the time step can rise ad infinity, so a limitation does no harm. However this value should be really large (let's say: 100000) to take the best out of the LTS approach.

alphaTempTable: This parameter limits the rise of the temperature in a cell. A value of 1.0 sets the new value completely, a value of 0.5 blends between 50% of the previous value and 50% of the new value. While this seems to be quite useful in simulations with chemical reactions, it's not an important parameter for the quenching device investigated here.

More settings must be done in the constant/reactingCloud1Properties file. Despite the correct definition of the mass to be injected you can adjust some specific LTS parameters here:

```
transient       no;  // no -> activate LTS; yes -> deactivate LTS
calcFrequency   2;   // do the particle tracking every 2nd iteration
maxTrackTime    6.0; // track the particle for 6s at most
maxCo           0.05; // max Courant number for particle tracking
```

transient: Setting to "no" activates the LTS approach, setting to "yes" uses the PISO algorithm.

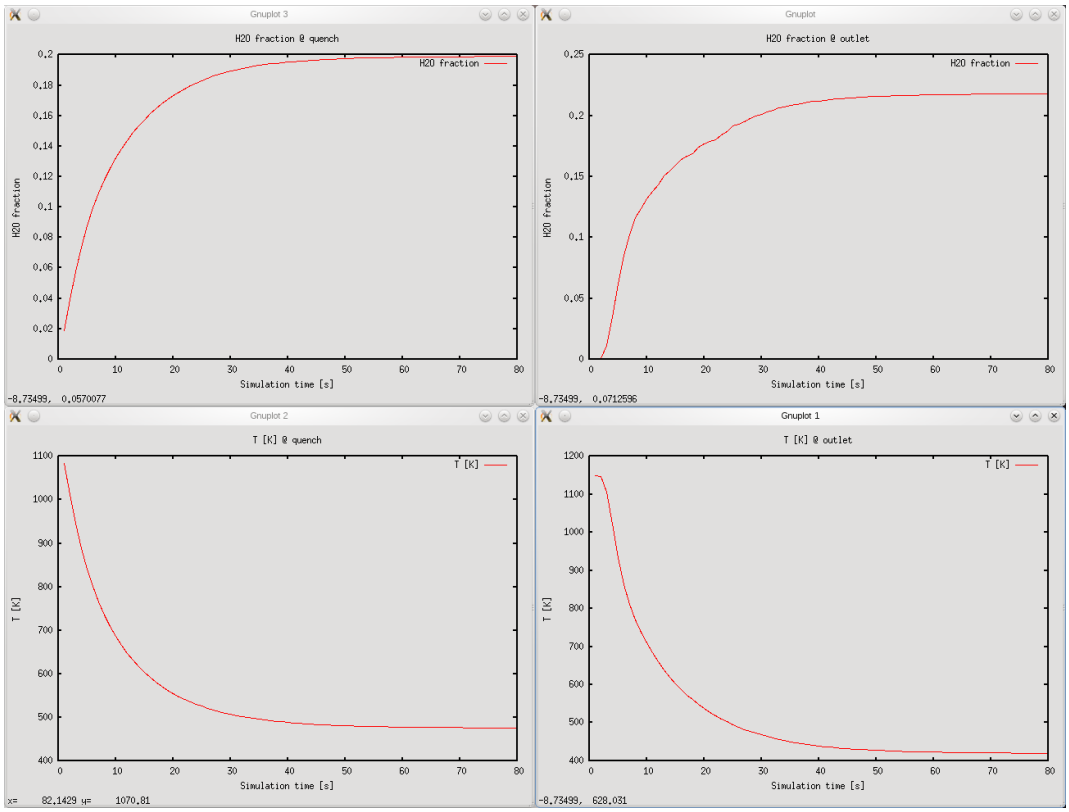
calcFrequency: Defines the frequency of the particle tracking step. This is a way to save lots of computational time. A value of 1 is very accurate, a value of 2 is fine for the cases presented here. Further increment of the calcFrequency provokes warnings in the thermodynamic libraries and oscillation in the flow field, so be careful with this setting.

maxTrackTime: Defines the longest period of simulation time that a particle is tracked. When having dead zones, or when a particle is sticked to the wall, the tracking algorithm might never end. So limiting the maximum amount of tracking time available for each particle is a good idea.

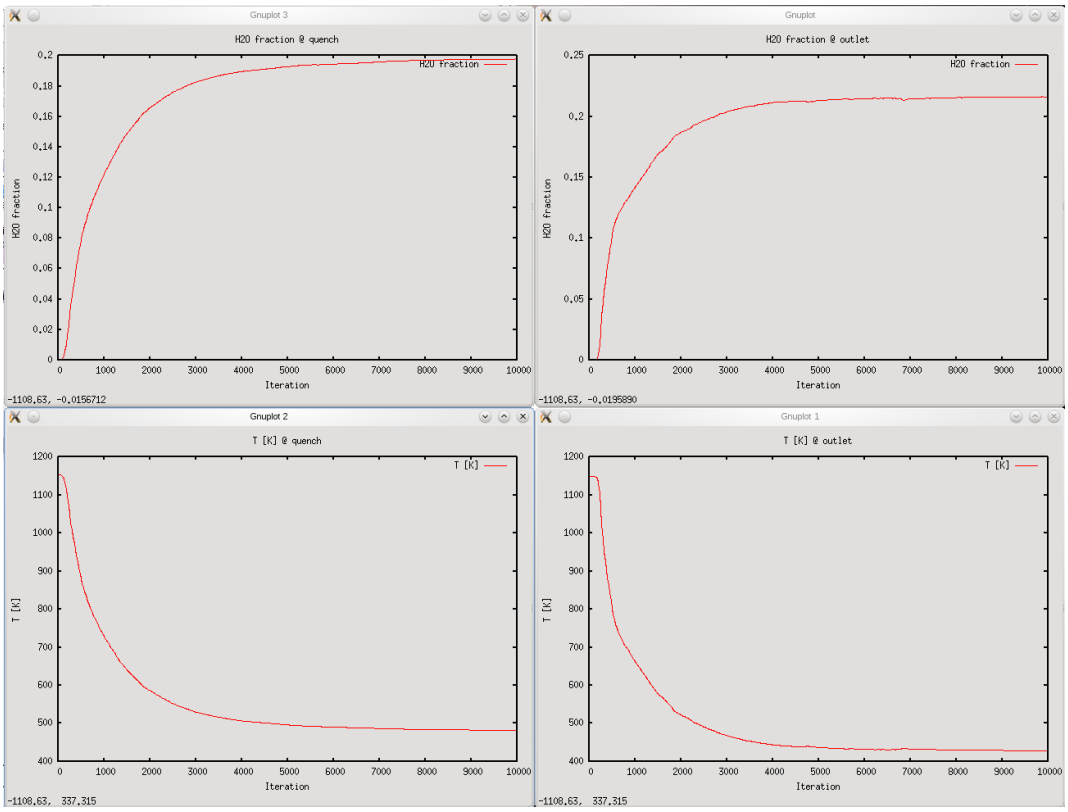
maxCo: This value determines the accuracy of the particle tracking. A value of 0.2 means that each cell is traversed by a particle in some 5 steps. You can save computational time by increasing this value. For the very coarse mesh in the case provided here a very small value of 0.05 must be chosen to fit the results from the PISO based reactingParcelFoam. When using a finer mesh this value can be increased.

3.2.2 Convergence monitoring

Although dhcaeLTSThermoParcelSolver tries to reach a steady state solution, the initial residuals are not that useful to determine wether convergence is reached. Due to the momentum exchange and the non determinism of the stochasticDispersion model there is always some level of change in the flow fields and the initial residuals stay at a rather high level. But of course it would be nice to have a concrete criteria to rate the progress of the simulation. One way to do such a rating is to use probes in the flow field, another way, which is applied here, is the usage of the fieldFunctionsObjects library by definition in the system/controlDict:



(a) Convergence visualization for reactingParcelFoam...



(b) ... and for dhcaeLTSThermoParcelSolver.

Figure 3.1: The plots created when calling "gnuplot gnuplot.Me" in the case directory. These charts are a convenient way to rate the convergence level of the simulation.

```
functions
{
    outlet_T_H2O
    {
        type
        faceSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   timeStep; // for reactingParcelFoam: outputTime;
        outputInterval  25;       // for reactingParcelFoam: 1;
        log             true;
        valueOutput     false;
        source          patch;
        sourceName      outlet;
        operation       weightedAverage;
        weightField     phi;
        fields ( T H2O );
    }
    quench_average
    {
        type            cellSource;
        functionObjectLibs ("libfieldFunctionObjects.so");
        enabled         true;
        outputControl   timeStep; // for reactingParcelFoam: outputTime;
        outputInterval  25;       // for reactingParcelFoam: 1;
        log             true;
        valueOutput     false;
        source          all;
        sourceName      region0;
        operation       volAverage;
        fields ( T H2O );
    }
}
```

So the average temperature and the H2O fraction is evaluated at the outlet patch and for the whole domain. The aggregated values are written to the HDD and can be visualized with small gnuplot scripts named "gnuplot.Me" that are placed in the case directories:

```
set term X11 3
set title 'H2O fraction @ quench'
set ylabel 'H2O fraction'
set xlabel 'Iteration'
plot "quench_average/0/cellSource.dat" using 1:4 \
    title 'H2O fraction' with lines
pause 10
reread
```

Just open another shell and navigate to your case directory. After the first lines of the output files are written you can call the script with "gnuplot gnuplot.Me". The resulting plots are updated automatically. See figure 3.1 for an example.

3.2.3 Definition massFlowRate

The massFlowRate in constant/reactingCloud1Properties takes care of the mass conservation: if you run your simulation on a 4 degree wedge mesh you must reduce the massFlowRate to fit for the smaller injection area. So when running your simulation on a 90 degree symmetryPlane mesh you must give in the 22.5 time of the massFlowRate, and when running with the full 360 degree quench, 90 times the value of the 4 degree wedge mesh case.

3.2.4 Pros and cons of LTSReactingParcelFoam

The big advantage of LTSReactingParcelFoam is the significantly reduced computational time. Indeed, this benefit can be outstanding!
The big drawback however is the missing of useful parcel information. During each iteration the droplets evaporate completely, so there is no chance to store the position, diameter, temperature etc. There are some indirect possibilities to visualize the particles: if you uncomment the "//#include "write.H" line in the dhcaeLTSThermoParcelSolver.C the source terms are written out with the other result fields, and you can have a look at them in the postprocessor. If these source terms are missing somewhere in the flow field, then no droplet is there... you can visualize the amount of H2O, id est the evaporated fraction of the droplets. But thats not satisfactory either. Therefor, if you need

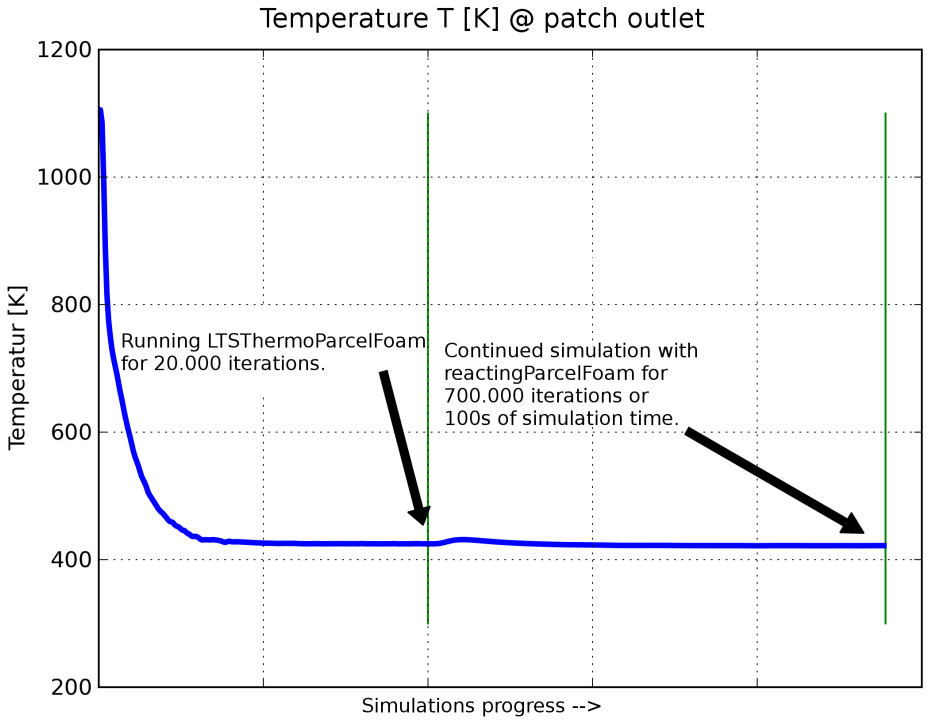


Figure 3.2: Concept of the continued simulation. After 20.000 iterations with dhcaeLTSThermoParcelSolver another 700.000 iterations with reactingParcelFoam were appended.

to make a cute visualization of the particles cloud then you should do a continued simulation with reactingParcelFoam...

3.3 Continued simulation with reactingParcelFoam

After the dhcaeLTSThermoParcelSolver reached steady state you can continue the simulation a bit further with reactingParcelFoam from OpenFOAM's standard solver assembly. It's easy to do, since you can reuse the thermophysicalProperties, the boundary conditions and all the results immediately. Customizations must be done especially in the constant/reactingCloud1Properties file. The way how to define the particle injection differs from the LTS steady state proceedings. Beside of that several dummy files for radiation, combustion etc must be provided in the constant/ directory.
There are a couple of good reasons to combine reactingParcelFoam with dhcaeLTSThermoParcelSolver:

- Since a steady state should have been reached with the LTS solver, the continuation with reactingParcelFoam should not lead to significant changes in the results.
- The lack of good particle visualization possibilities in dhcaeLTSThermoParcelSolver can be overcome with the particle tracking information provided by reactingParcelFoam.
- In cases where no steady state exists, the dhcaeLTSThermoParcelSolver can be used to initialize the flow field in a very efficient manner. From this starting point the thermoParcelFoam solver can provide time discretized results with far lower computational effort.

The combination of dhcaeLTSThermoParcelSolver and reactingParcelFoam is visalized in figure 3.2. The simulation was run with dhcaeLTSThermoParcelSolver for 20.000 iterations. Then the final results were mapped to reactingParcelFoam and the simulation was continued for another 700.000 iterations or 100s of simulation time. There is a small disturbance visible immediately after switching the solver:

since we don't have the droplets in place after stopping dhcaeLTSThermoParcelSolver (they evaporate completely in each iteration), reactingParcelFoam must start with a droplet free domain.
It should be obvious that you can stop the continued simulation after a few seconds of simulation time. While the complete simulation with reactingParcelFoam needs some 100s to be converged, the continued simulation does not change any more after 15s. So there is a saving when using dhcaeLTS-ThermoParcelSolver first!

3.4 Alternative simulation with reactingParcelFoam

Of course it makes sense to run reactingParcelFoam as an alternative to dhcaeLTSThermoParcelSolver. This is an efficient way to verify the correctness of the LTS convergence acceleration approach. And it can make you happy to see how much time you can save with the LTS procedure in contrast to a PISO or PIMPLE based solver.

Chapter 4

Test cases

Two series of test cases are provided. One serie makes use of wedge elements to reduce computational time and memory usage. To run this serie you must modify the OpenFOAM's source code as described in section 2.2. The other serie uses a 90 degree segement of the simplified quench with standard symmetryPlanes. These simulations do not need a modification of the source code, just a compilation of the provided dhcaeLTSThermoParcelSolver.

4.1 Specifics of the provided meshes

To keep computational time low the mesh resolution for both series is chosen very coarse! To improve the accuracy of the results you need to refine the meshes by editing the blockMeshDict files. However a fine resolution of the mesh will take the simulations to run for several days. For this quick introductionary cases we stay with the lower results quality and with the short computational times. The parameter configuration for the LTS approach is optimized for the coarse meshes to a certain extent. When using finer resolution you need to to further LTS parameter tuning to get optimum performance.

4.2 Wedge series

Four cases are provided which are partially dependend on each other.

4.2.1 case_rhoSimpleFoam

First of all you should run the initialization case with name "case_rhoSimpleFoam". You can do this by starting the "Allrun" script in the case_rhoSimpleFoam directory or by calling the "Allrun" script of the above level.

4.2.2 case_dhcaeLTSThermoParcelSolver

Next you can run the dhcaeLTSThermoParcelSolver case. This simulation does not take that much time, so it's a good point to start with. You can run the case by calling the "Allrun" script in the case_dhcaeLTSThermoParcelSolver directory.
After starting the simulation you can open another shell and navigate to the cases directory again. If you have gnuplot installed you can visualize the simulations progress by typing "gnuplot gnuplot.Me". The simulation will run for 4.000 iterations. As you can see in the gnuplot charts a steady state is reached by then.
The results in directory 4000 are used to initialize the case_reactingParcelFoam_continued simulation, so don't delete it too early!

4.2.3 case_reactingParcelFoam_continued

If you have a fast machine or plenty of time you can run the case_reactingParcelFoam_continued. It will be initialized with the results from case_dhcaeLTSThermoParcelSolver, so you need to run this simulation first.
The output of the solver is simply written into the shell, it is not written to the HDD. The reason is simple: it's a very huge file (hundreds of MB) and there's not that much interesting information in it. After starting the simulation you can start another shell and use "gnuplot gnuplot.Me" to visualize the simulations progress. You will have to wait until the first two seconds are written out, otherwise the necessary data is not available for gnuplot yet and there will be an error message. Just wait a little while and try the gnuplot command again.

4.2.4 case_reactingParcelFoam

If you have even more time available then you can run the complete case with the PISO based reactingParcelFoam. Just run the "Allrun" script in the case directory. The initialization is done by the results from the rhoSimpleFoam case.
Again the output of the solver is only written to the shell. And again you can watch the simulations progress with "gnuplot gnuplot.Me" after the first few seconds of simulation time.

4.3 3D series

Running the cases with the 90° segment of the quench will provide you even more reliable results than using the wedge mesh. However in the provided cases the mesh is still not large enough to get advantages from using parallel computation. You may want to adjust the system/decomposeParDict files and the Allrun scripts to fit your needs when having more CPU's available and dealing with bigger mesh sizes.
As described in the section 4.2 "Wedge series" you can start the individual cases by running the Allrun scripts.

4.4 Some results

To give you an impression of the results that you can expect from the test cases the H2O fraction and the temperature distribution are shown in figure 4.1 in a comparison of the LTS based solver and the PISO based solver.

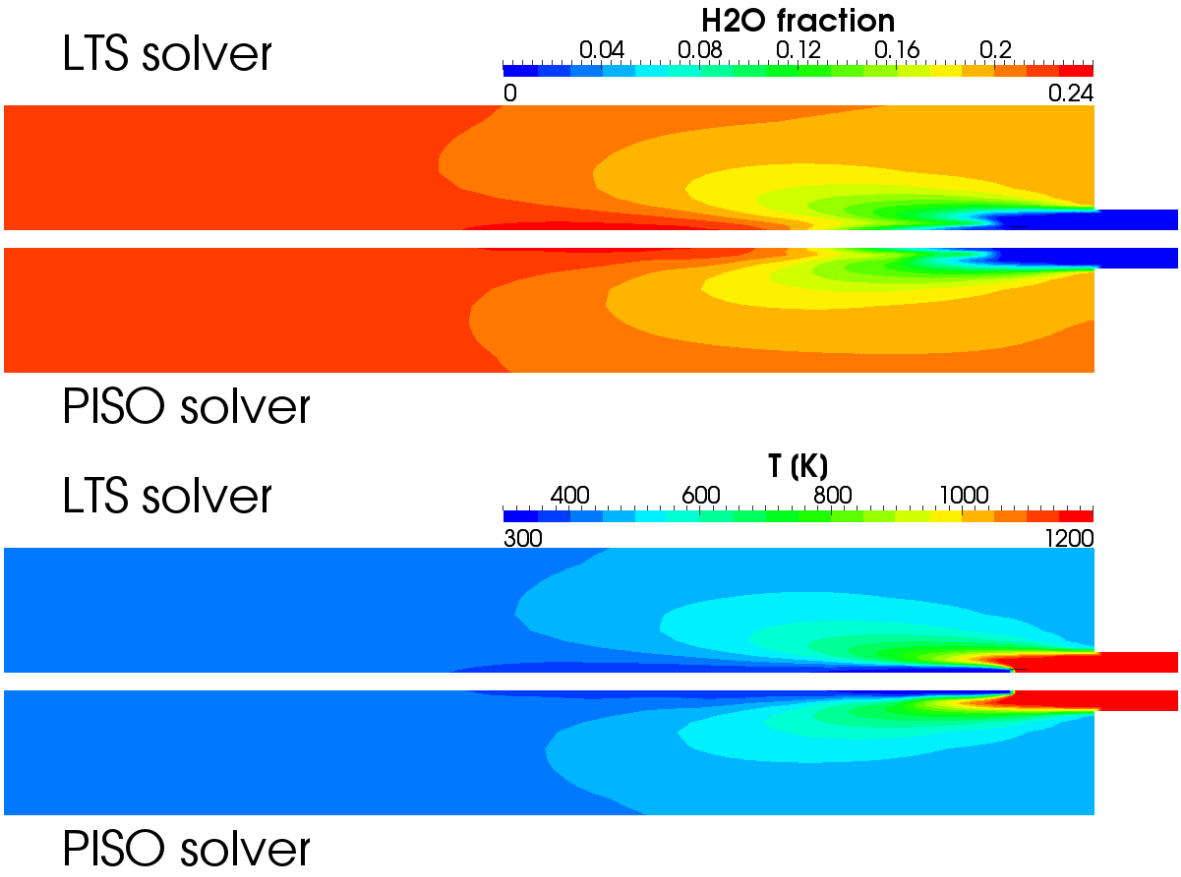


Figure 4.1: A comparison of results from dhcaeLTSThermoParcelSolver and reactingParcelFoam.

Chapter 5

Hints

5.1 reactingParcelFoam

Several of the files and there dictionaries need your full attention. From the wide variety of settings here is a selection of the most error-prone.

5.1.1 constant/reactingCloud1Properties

```
coneInjectionCoeffs
{
    SOI             0.0; // start of injection
    duration        100.0; // duration of injection in seconds
    positionAxis    (((0.0 0.0 0.0) (0 0 1)));
    massTotal       0.1; // 100s * 0.001kg/s = 0.1 kg
    parcelsPerInjector 5000000; // 50000 parcels * 100s
    parcelBasisType mass;
    flowRateProfile constant 1.0;
    Umag            constant 11.0;
    thetaInner      constant 0; // 0 -> full cone
    thetaOuter      constant 75;
    sizeDistribution
    {
        type          RosinRammler;
        RosinRammlerDistribution
        {
            minValue        3.7e-5;// 37 micrometer
            maxValue        1.65e-4;// 165 micrometer
            d               7.0e-05; // 70 micrometer
            n               0.5;
        }
    }
}
```

SOI: Start of injection. Does this value fit to your start time in system/controlDict? You may want to start injection immediately at the beginning of the simulation, or at any later point of time. If your simulation does not start at time 0.0 (as it is the case in the continued simulation with reactingParcelFoam), you must adjust this value to the desired start time.

duration: For how long shall the parcels be injected? This value in combination to the massTotal definition determines the injection rate! Both of these values must fit to the length of your simulation defined in the system/controlDict with startTime and endTime. Another effect is that you can't change the endTime on the fly during your simulation.

massTotal: How much mass shall be injected in the complete injection period? This value must fit the startTime and endTime in the system/controlDict, as well as the duration in the constan-

t/reactingCloud1Properties. It must fit the "area" of the coneInjector, too: when running the simulation on a wedge mesh, massTotal is smaller than in the case of the 90 degree segment.

parcelsPerInjector: How many parcels shall be injected during the complete injection period? To get a good resolution you must start a rather large number of parcels!

5.1.2 system/controlDict

```
adjustTimeStep   yes;
maxCo            0.2;
maxDeltaT        1e-03;
```

maxCo: The value maxCo for the maximum Courant number (see section 1.3) is crucial for the accuracy of the solution on the one hand, while being essential for the amount of computational time necessary on the other hand! In the provided cases you will get excellent compliance with the results from the dhcaeLTSThermoParcelSolver when setting the maxCo to 0.2 or lower. But it will take quite a long time to get the simulation finished. Setting maxCo to 0.5 or higher reduces computational time, but the results will be a bit different, too. If your specific geometries guarantees that the maxCo value is not reached in the cells which are actually being traversed by particles, you may be able to set higher values than in the provided cases.

5.1.3 Don't use PIMPLE instead of PISO

Although you can enable the PIMPLE algorithm by setting the nOuterCorrectors parameter to a value > 1, it is not recommended at all. The particle transport suffers awfully from the higher time step, and although you can speed up the simulation a bit, the results are bad.

DHCAE Tools UG (limited liability company)
Friedrich-Ebert-Str. 368
47800 Krefeld
Germany
Phone +49 2151 821493
Fax +49 2151 821494