Exploiting OpenMP in the Initial Section of Modelica Models

- Work in Progress -

Javier Bonilla[a], Luis J. Yebra[a] and Sebastián Dormido[b]

[a] CIEMAT-PSA, Centro de Investigaciones Energéticas, MedioAmbientales y Tecnológicas
Plataforma Solar de Almería, Almería, Spain, {javier.bonilla,luis.yebra}@psa.es

[b] UNED, Universidad Nacional de Educación a Distancia, Madrid, Spain, sdormido@dia.uned.es

5th of September, 2011

Index

Introduction

Modern equation-based object-oriented (EOO) modelling languages are continuously increasing their expressiveness to model complex systems. However, the required computational effort to simulate is also increasing.

Commonly, EOO models are compiled as single-threaded executables not taking advantage of the newest multi-core processors.

This work describes a straightforward parallelization method in the initial section of the resulting C source code obtained from a Modelica model.

OpenMP - Introduction

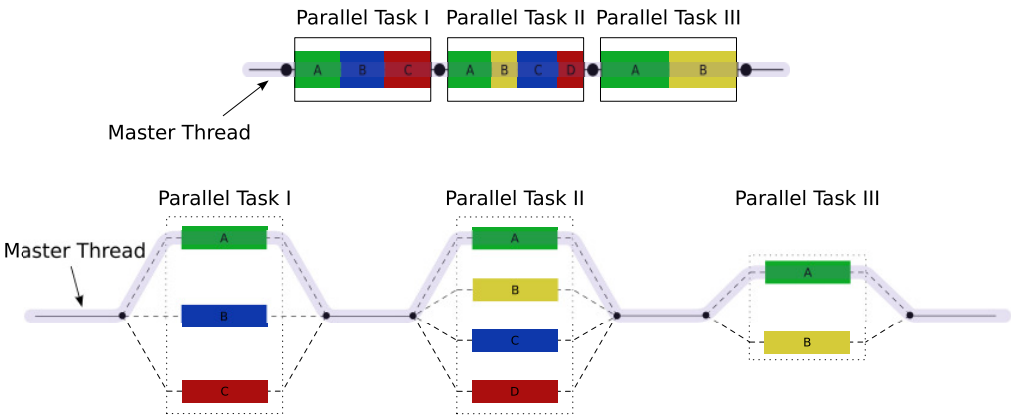
OpenMP (Open Multi-Processing) is an application programming interface (API) for multi-platform (Unix and Microsoft Windows platforms) shared-memory parallel programming in C/C++ and Fortran.

Version	Year	Languages
1.0	1997	Fortran
1.0	1998	C/C++
2.0	2000	Fortran
2.0	2002	C/C++
2.5	2005	Fortran,C/C++
3.0	2008	Fortran,C/C++
3.1	2011	Fortran,C/C++

OpenMP is maintained by the OpenMP Architecture Review Board (ARB).

OpenMP - Features

Key concept - Multithreading



Core elements

- Parallel control directives. Control the flow execution (i.e. *parallel*).
- Work Sharing. Distribute the code among cores (i.e. *parallel for dir.*).
- Synchronization. *critical*, *atomic* and *barrier* directives.
- Run-time functions. Set and get run-time information.

La Plataforma Solar de Almería (PSA)

CIEMAT-PSA, a Spanish government research and test center, is the biggest Europen research center devoted to solar concentrating technologies.

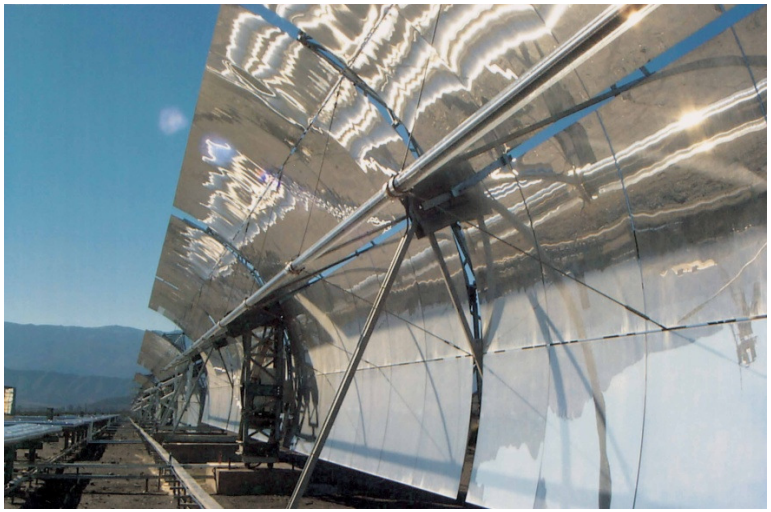
The DISS test facility

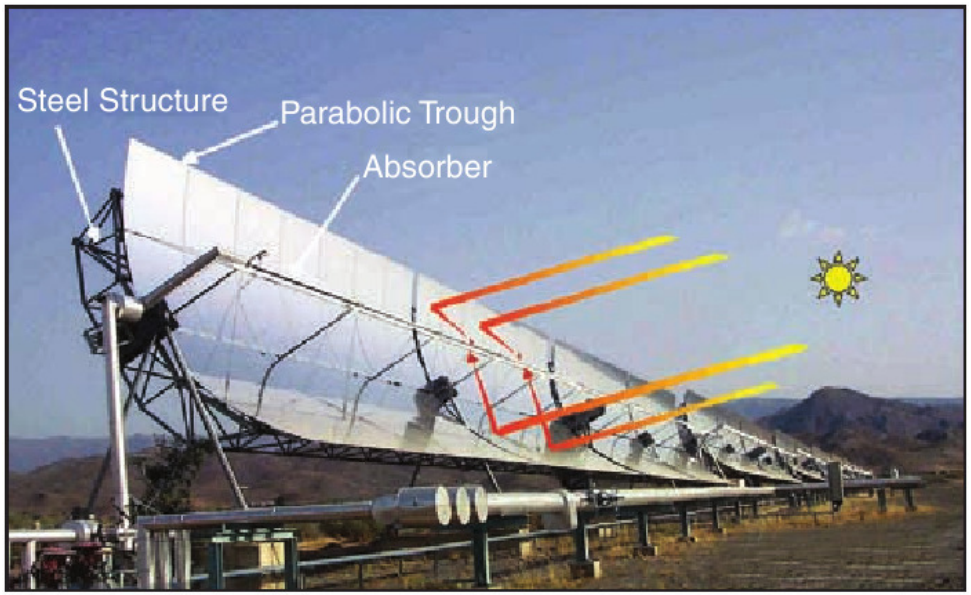
The DISS test facility is the first facility built in the world where two-phase-flow steam-water processes in parabolic-trough collectors can be studied under real solar conditions.

Parabolic-Trough Collectors (PTCs)

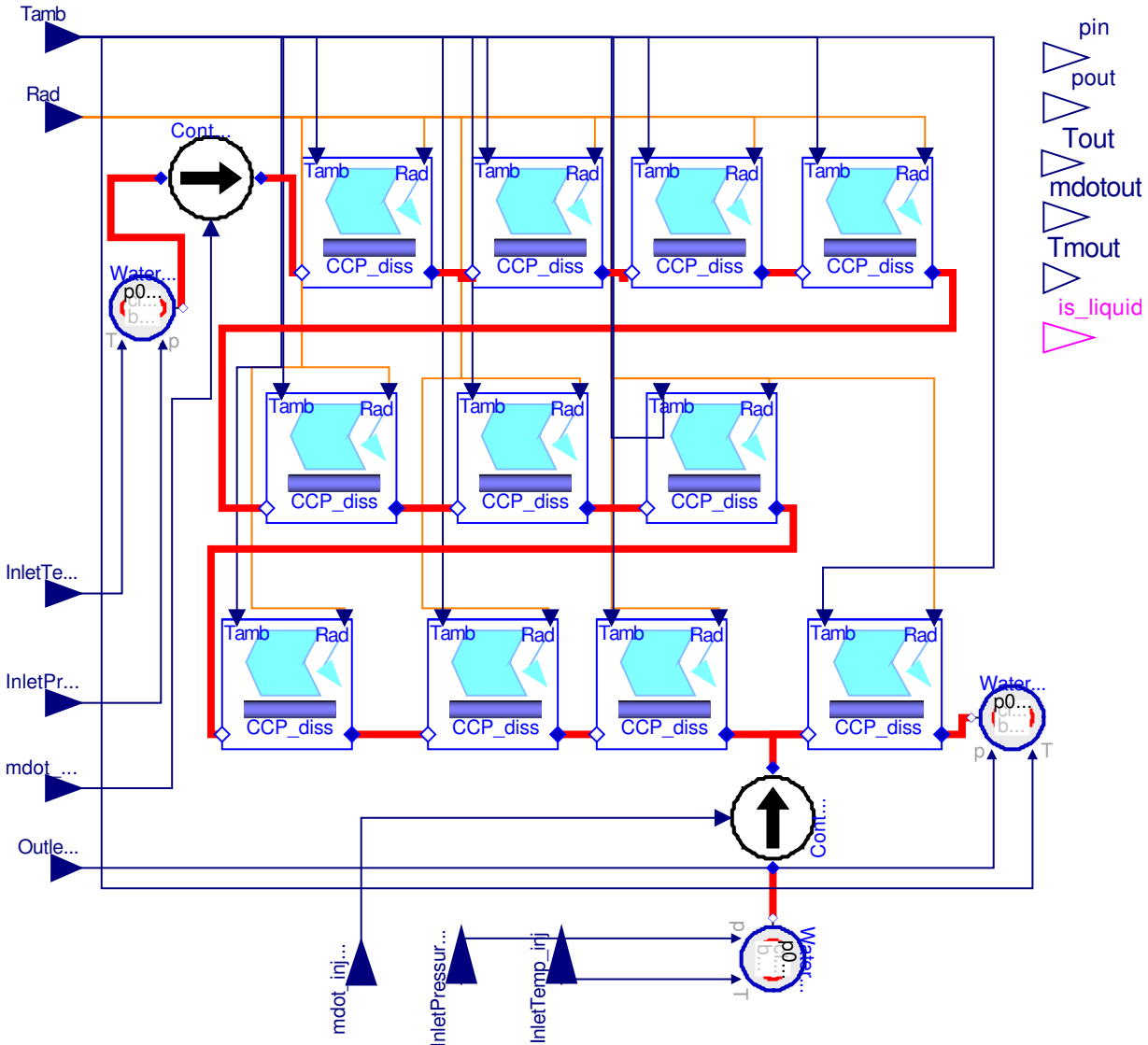
Main elements

- Parabolic-trough receiver.
- Absorber tube.





DISS Modelica model



Model inputs

- Ambient temperature
- Direct solar radiation

1st PTC inlet flow

- Mass flow
- Flow temperature
- Flow pressure

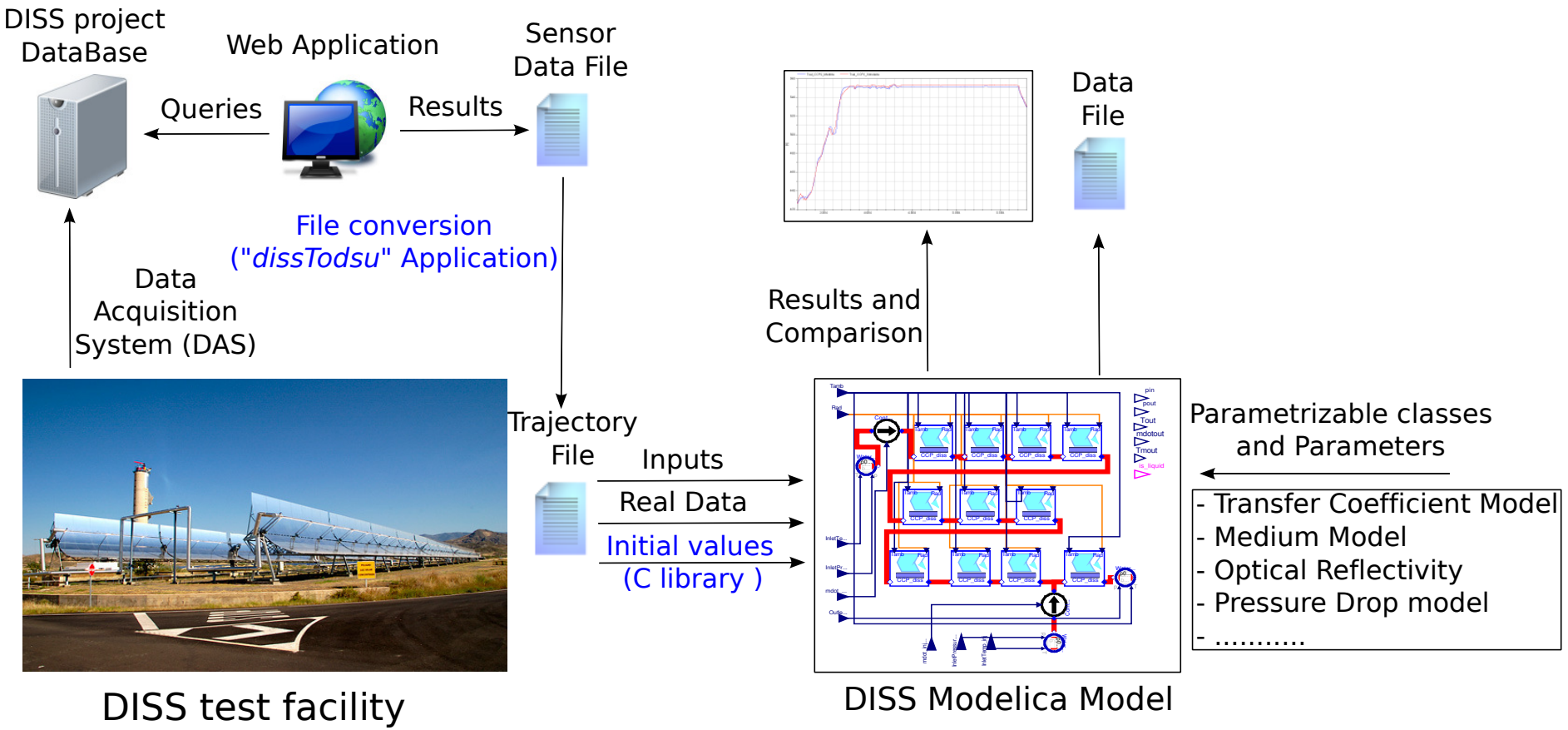
11th PTC injector

- Mass flow
- Flow temperature
- Flow pressure

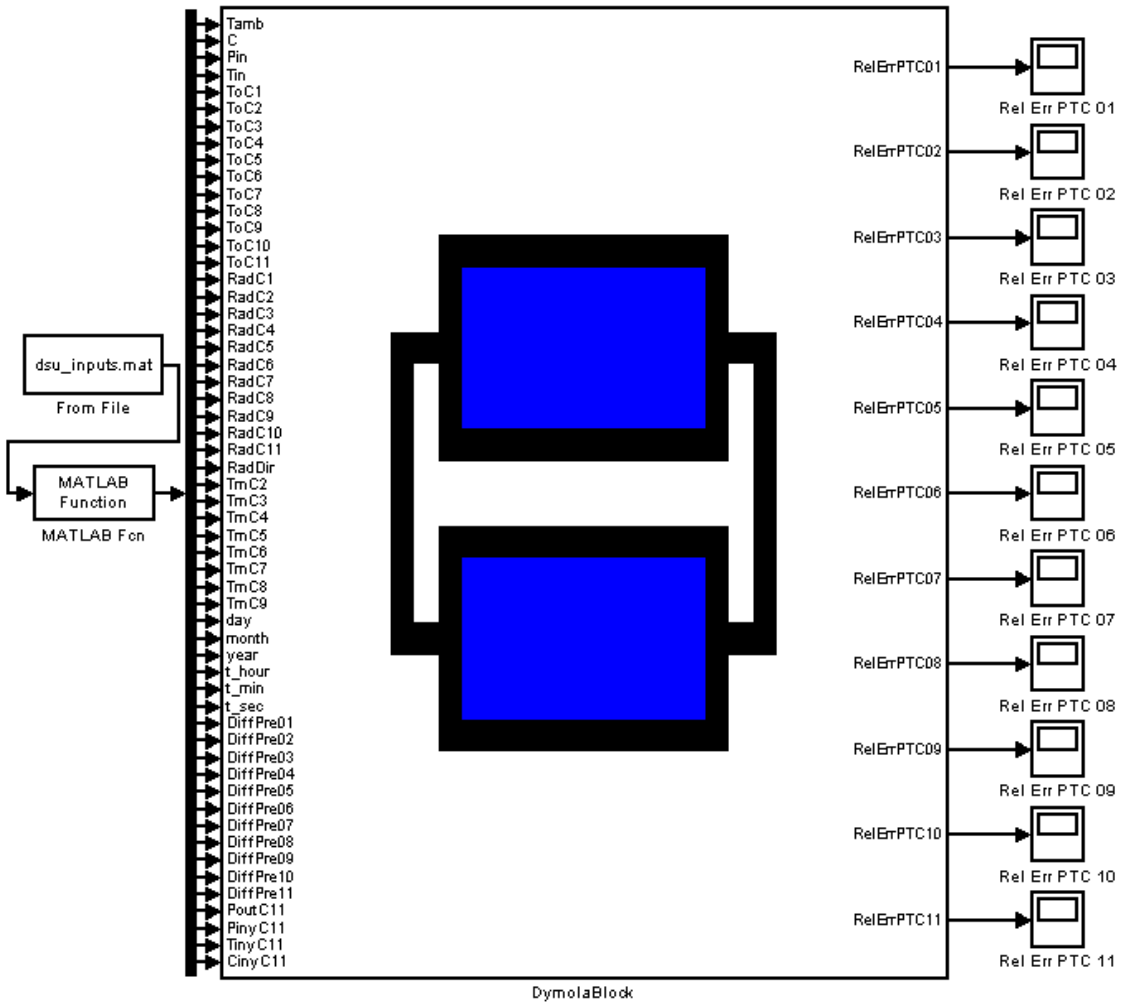
Initialization

- Outlet pressure

Simulator scheme



Model Calibration



Calibration process

- Exporting the DISS Modelica model to Matlab/Simulink.
- Using the Matlab Genetic Algorithm Toolbox.
- Multi-objective approach, minimizing the output temperature difference in each PTC.

Initial Section Parallelization (1/2)

Procedure

1. Reference the OpenMP header (*omp.h*) in the *dsmodel.c* file (Listing 1).
2. Set the maximum number of threads (*omp_set_num_threads*) (Listing 2).
3. Manually distribute the sentences among the threads (Listing 3).
4. Compile including the OpenMP library (*-fopenmp in GNU compilers*).

```
#include <omp.h>
```

Listing 1. OpenMP header inclusion.

```
constant int NCores = 4;
omp_set_num_threads(NCores);
```

Listing 2. Setting the maximum number of threads.

Initial Section Parallelization (2/2)

```
#pragma omp parallel
{
  #pragma omp sections
  {
    #pragma omp section
    {
      ........
      ........
     T[0]=getIniValue(iniTime,"Tem PTC1");
     T[1]=getIniValue(iniTime,"Tem PTC2");
      ........
      ........
    }
    #pragma omp section
    {
      ........
      ........
     p[0]=getIniValue(iniTime,"Pres PTC1");
     p[1]=getIniValue(iniTime,"Pres PTC2");
      ........
      ........
    }
    #pragma omp section
    {
      ........
      ........
     r[0]=getIniValue(iniTime,"Rad PTC1");
     r[1]=getIniValue(iniTime,"Rad PTC2");
      ........
      ........
    }
    #pragma omp section
    {
      ........
      ........
     m[0]=getIniValue(iniTime,"Mflow PTC1");
     m[1]=getIniValue(iniTime,"Mflow PTC2");
      ........
      ........
    }
  }
}
........
........
```

Listing 3. OpenMP parallel sections.

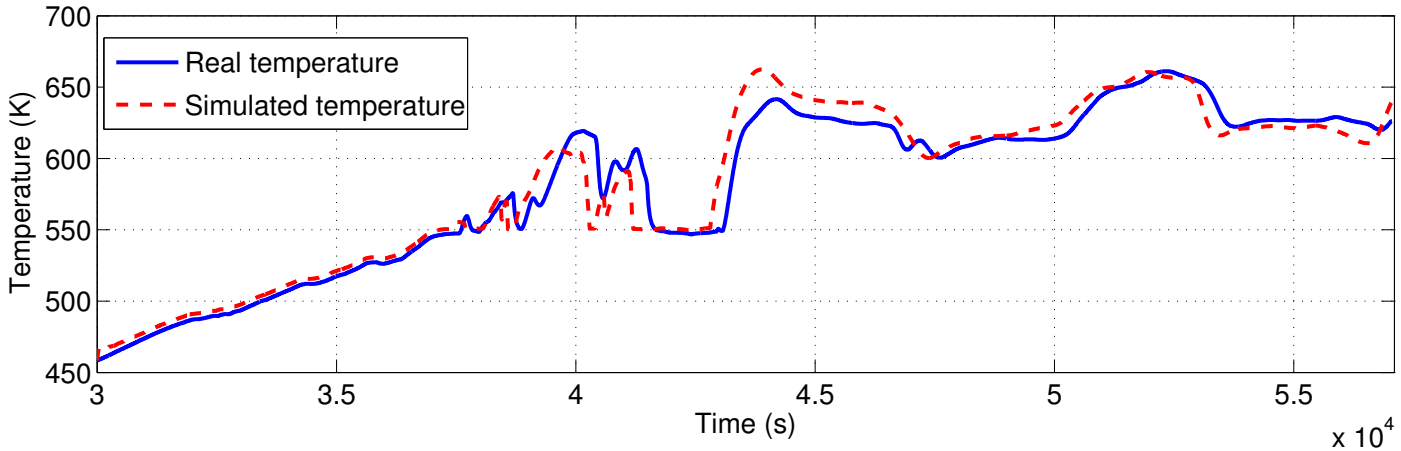
Simulation Statistic

The tests were performed using a Intel® Core™ i5 CPU M540, 2.53 GHz. Several tests were performed and mean execution times have been calculated.

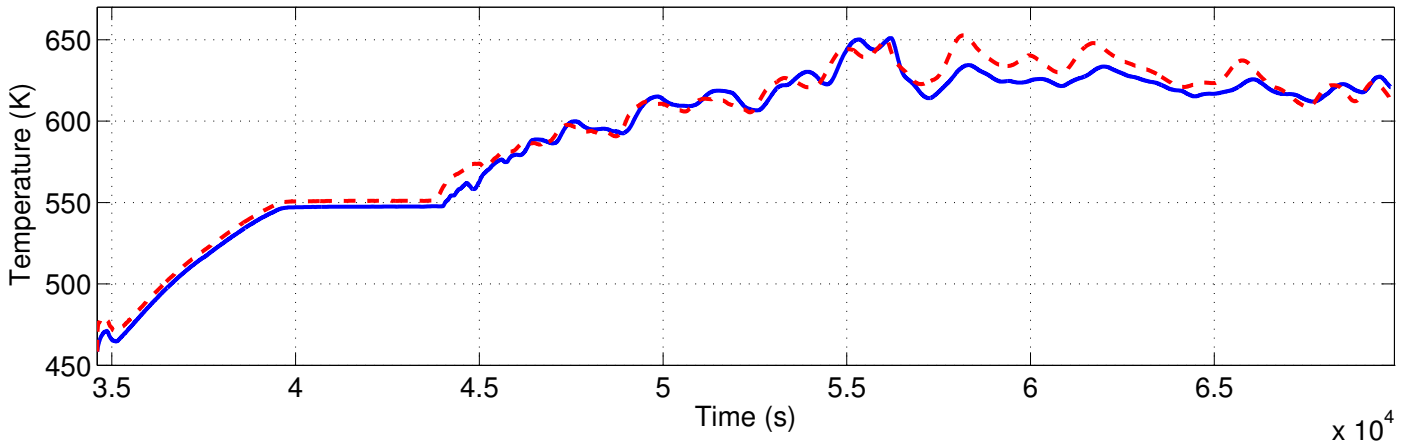
Kind of model	Original	Parallelized
Execution time (s)	170.727	151.415
Speedup	1	1.13
Initialization section exec. time (s)	31.231	11.127
Initialization section speedup	1	2.80
Calibration execution time	3 days 9 hours	3 days 4.5 hours

Simulation Result - output DISS field temperature

Calibration



Validation



Conclusions

- OpenMP can be easily used to parallelize the initial section in the resulting C source code obtained from Modelica models.
- A mean speedup of 1.13 was obtained in the whole simulation.
- A mean speedup of 2.80 was obtained in the initial section.
- The gain in speed was useful in simulation and calibration.

Future Work

- Study how to implement the use of OpenMP directly in the Modelica code.
- Study how to take advantage of OpenMP in the integration process.
- Take advantage of a 13-node cluster.
- Consider the parallelization not only in simulation but in the genetic algorithm calibration.