NTNU

Norwegian University of
Science and Technology

Computer Code for Thermal Analysis of Rocket Motors

Jørn Arnold Kvistad Riise

Master of Science in Engineering and ICT
Submission date: July 2008
Supervisor: Erling Næss, EPT

Norwegian University of Science and Technology
Department of Energy and Process Engineering

Problem Description

1. Det eksisterende numeriske beregningsverktøyet skal modifiseres for bedre å kunne håndtere transiente beregninger, samt mer realistiske beregningsmodeller for ablative materialer. I dette inngår å endre det numeriske beregningsskjemaet til implisitt skjema for tidsintegrasjonen, samt at kildeledd skal inkluderes i den løste energilikningen. Endringene skal presenteres, implementeres og diskuteres.
2. Anbefalte modeller for ablasjon og pyrolyse av isolasjonsmaterialer presentert i prosjektoppgaven skal implementeres. Det skal legges vekt på at modellene benytter informasjon som allerede er tilgjengelige for de materialene Nammo Raufoss benytter.
3. Beregninger skal gjenomføres med det modifiserte programmet, og resultatene skal sammenliknes med tilsvarende beregninger foretatt med de kommersielle verktøyene CMA3 og ASTHMA, samt resultater fra målinger tilgjengelige fra Nammo Raufoss. Resultatene skal kommenteres og eventuelle avvik/forskjeller skal diskuteres.
4. Det skal utarbeides forslag til videre arbeid.

Assignment given: 04. February 2008
Supervisor: Erling Næss, EPT

Preface

This thesis is submitted as a conclusion of a Master of Science degree in Engineering & ICT with specialisation in the field of energy and process engineering at the Norwegian University of Science and Technology (NTNU). The work was performed during the spring semester 2008.

The supervisors for this thesis have been Professor Erling Næss from the Norwegian University of Science and Technology (NTNU) and Erland Ørbekk from Nammo AS.

Jørn Arnold Kvistad Riise
Trondheim, 14.07.2008

Acknowledgements

First I would like to express a special thank to my primary supervisor for this thesis, Professor Erling Næss from the Norwegian University of Science and Technology (NTNU), for guiding and helping me through the work, encouraging me to follow my ideas, and giving me directions on how to best carry out the thesis.

I would also like to thank Erland Ørbekk, Nils Kubberud, Kristian Lium and John Myklebust for giving me an introduction to the field of rocket technology during my stay at Nammo Raufoss the summer of 2007. The work performed by John Myklebust to provide me with necessary background information is also highly appreciated.

Special thanks to Audun Bråthen, Solveig Slepvold and Mari Aass for reading the report and giving me feedback, which helped me correct a few errors and improve the overall report.

Last, but by no means least, thanks to all of my friends and family for their support during this work, and for engaging me in other activities, making me regain energy and motivation.

Abstract

Further development of a two-dimensional thermal analysis code (G2DHeat) to include internal decomposition and charring ablation of insulation materials is presented. An overview of the structural changes made to the program code by implementing an implicit solution routine, including source term is given, before testing and verification of accuracy is performed. A kinetic model for decomposition reactions, as well as routines for handling the generated gas from the decomposition reactions, changes concerning the material properties and erosion of surface material are implemented and explained. Comparisons of results are made with similar results obtained by commercial programs. Possible reasons affecting the results are pointed out, before additional comparisons with experimentally observed measurements are performed. Based on the simulated results it is concluded that a great deal of testing remains for proper validation of the program. How to include better boundary conditions for simulating charring ablation is suggested and recommended for further development of the program.

List of contents

List of figures

List of tables

Nomenclature

A_e east cell face [m^2]

A_w west cell face [m^2]

A_n north cell face [m^2]

A_s south cell face [m^2]

A_i frequency factor of component “i” [s^{-1}]

$B_c^{'}$ dimensionless ablation rate [-]

$B_g^{'}$ dimensionless pyrolysis gas rate [-]

c specific heat [$J.Kg^{-1}K^{-1}$]

C_p specific heat capacity of a cell volumes material [$J.Kg^{-1}K^{-1}$]

C_{pp} specific heat capacity of virgin material [$J.Kg^{-1}K^{-1}$]

C_{pr} specific heat capacity of residual mass [$J.Kg^{-1}K^{-1}$]

$C_{p,g}$ specific heat capacity of pyrolysis gas [$J.Kg^{-1}K^{-1}$]

$C_{v,s}$ specific heat capacity of solid material [$J.Kg^{-1}K^{-1}$]

C_H Stanton number for heat transfer [-]

C_M Stanton number for mass transfer [-]

E_i activation energy for component “i” [$J.kmol^{-1}$]

$\overline{h}$ convective heat transfer coefficient [$W.\ m^{-2}K^{-1}$]

h_* enthalpy at face *(=e,w,n or s) [$J.Kg^{-1}$]

h_g enthalpy of pyrolysis gas [$J.Kg^{-1}$]

$h_{f,*}^0$ enthalpy of formation *(=residue, virgin) [$J.Kg^{-1}$]

$h_{f,*}^{T_w}$ enthalpy of formation evaluated at wall temperature (T_w) [$J.Kg^{-1}$]

h_P enthalpy in cell volume P [$J.Kg^{-1}$]

h_r enthalpy of residual mass [$J.Kg^{-1}$]

h_0 enthalpy of virgin material [$J.Kg^{-1}$]

H_r recovery enthalpy [$J.Kg^{-1}$]

i denotes i-direction

j denotes j-direction

k thermal conductivity [$W.\ m^{-1}K^{-1}$]

k_p thermal conductivity for virgin material [$W.\ m^{-1}K^{-1}$]

k_r thermal conductivity for residual mass [$W.\ m^{-1}K^{-1}$]

$\dot{m}_*$ mass flow rate of gases at face *(=e,w,n or s) [$Kg.\ s^{-1}$]

$\dot{m}_g$ mass flow rate of pyrolysis gases [$Kg.\ s^{-1}$]

$\dot{m}_r$ mass flow rate of residual mass(char) which is eroded [$Kg.\ s^{-1}$]

$\dot{m}_{pyr}$ mass rate of gases produced by pyrolysis [Kg. s^{-1}]

P total pressure [J. m^{-2}]

$\dot{r}$ recession rate [$m.s^{-1}$]

$\overline{\dot{r}}_{cell}$ mean recession rate for the cell volume [$m.s^{-1}$]

R universal gas constant [$J.kmol^{-1}K^{-1}$]

$R_{I,*}$ thermal resistance in I-direction [$m^2K.W^{-1}$]

$R_{J,*}$ thermal resistance in J-direction [$m^2K.W^{-1}$]

S source term [W]

S_u cell source term [W]

S_p source term as a function of cell temperature [$W.K^{-1}$]

$\overline{S}$ source term [$W.m^{-3}$]

T temperature [K]

T_P cell temperature [K]

T_E east cell temperature [K]

T_W west cell temperature [K]

T_N north cell temperature [K]

T_S south cell temperature [K]

$\overline{u}$ internal energy [$J.Kg^{-1}$]

u gas velocity in i-direction [$m.s^{-1}$]

$\dot{Q}$ heat flux [W]

x fraction parameter [-]

Z_i^* diffusion driving force (See ASTHMA3 1972) [-]

ε emissivity [-]

v gas velocity in j-direction [$m.s^{-1}$]

α_i degree of weight loss component “i”

ΔV cell volume [m^3]

Δt time step [s]

Δh_{pyr} heat of pyrolysis [$J.Kg^{-1}$]

Δh_{abl} heat of ablation [$J.Kg^{-1}$]

λ_* fraction parameter for direction *(=i,j) [-]

ρ current density [$Kg.m^{-3}$]

ρ_0 initial density [$Kg.m^{-3}$]

ρ_r residual density (often the same as the char density) [$Kg.m^{-3}$]

ρ_i current density of component “i” [$Kg.m^{-3}$]

ρ_{0i} initial density of component “i” [$Kg.m^{-3}$]

$\rho_{r\,i}$ residual density of component “i” [$Kg.m^{-3}$]

$\dot{\rho}$ weight loss to volume [$Kg.m^{-3}s^{-1}$]

Γ	volume fraction
σ	Stefan-boltzmann constant (=5,67×10^{-8} [W/m^2K^4]).
0	denote values at the previous time step

Introduction

Heat transfer in rocket motors is being increasingly investigated to better understand the physical material changes during the operational time of the motor. Material durability is important since the motor structure requires strength and manoeuvrability to bring off a successful flight. The more demanding the customer specifications are, the more challenging are the problems faced by the designers. At Nammo AS, the research department is working on methods to better solve such problems. In this work they use a computer simulation program (G2DHeat) to estimate the transient heat transfer in rocket motor parts. Using the results from this program, they decide on design modifications and changes necessary to enhance rocket performance. When the results from the simulations become satisfactory, they perform an actual physical firing test of the rocket motor. To ensure an adequate solution, they compare temperature measurements from the firing test with the temperatures from the computer simulations. If there are too large deviations in the results, they make adjustments to the program and perform new simulations. This process can become very expensive, especially if a great number of firing tests are necessary before they achieve satisfying results (Myklebust 2008). The purpose of this master thesis is to improve the program simulations by means of including more accurate models for the physics of heat transfer.

G2DHeat has earlier been described by Riise (2008) in a project report. In this report different suggestions on how to improve G2DHeat, by including more physical material behaviours, are presented. The work in this master thesis is a continuance of the mentioned project report, and includes implementation, as well as testing of the program routines suggested in the report. The project report is recommended for proper understanding of the G2DHeat program. It also elaborates the following points which are not included in this master thesis.

- Grid configuration.
- Boundary conditions.
- The solution process (with calculations of the temperatures explicit in time).
- Kinetic models for decomposition reactions.
- Moving grid methods and issues concerning these.
- Advantages and disadvantages with CMA3 (“The Aerotherm Charring Material Thermal Response and Ablation Computer Program, Version 3”)(Schoner 1970) and FSSIM2D(Austegard 1997).
- Recommendations and suggestions for improvements of G2DHeat.

The main objective of this master thesis is to further develop the G2DHeat program to include thermal degradation reactions of ablative insulation materials, and prepare the program for implementation of future heat source/-sink reactions.

The thesis itself is divided into five parts which describe the changes, assumptions and tests of the modified G2DHeat program.

In chapter 1, an explanation of why the program is made implicit in time is provided. The implicit formulation, in addition to a source term, is then presented. Further, a presentation of the implicit solution routine is given, together with an explanation of how the different grid blocks interact with

each other in the implicit simulations. At the end of the chapter, accuracy and computation times for the implicit and the explicit solution routines are compared.

Chapter 2 contains the physical aspects of the decomposition reactions occurring within the material. This includes the kinetic model used, and how the thermodynamic and thermo-chemical properties of the materials are changed. Silica phenolic properties are given as example values.

How to numerically handle the decomposition events is presented in chapter 3. In addition to the implicit formulation in chapter 1, explicit solution of continuity and the decomposition events are included in the programs calculations. The chapter also provides a presentation of the numerical aspects of solving pyrolysis and charring ablation in G2DHeat. This includes governing equations, boundary conditions, solution routine and discussion of changes made.

Test simulations of G2DHeat are shown in chapter 4 and 5. In chapter 4 the program is compared to CMA3 and ASTHMA (Axi-Symmetric Transient Heating and Material Ablation Program) by using a simple geometry. In chapter 5 the program is compared to experimental data using a simple geometry and a more complex geometry.

The conclusions and recommendations for further work are described in chapter 6.

Chapter 1: Initial changes to the program

This chapter gives an overview of initial changes to the program code, the new program features and capabilities, and a comparison of accuracy with the old G2DHeat program and Comsol Multi Physics.

1.1 General description of the modifications

The old heat analysis program used by Nammo and explained by Riise(2008), is modified to handle the numerical calculations implicit in time, instead of explicit as before. In this manner, each cell in the grid contains the temperature for the next time step, so the transport of heat must be solved as a system of algebraic equations. The partial differential equations describing the two-dimensional problem are discretised, using the finite volume method, and solved by TDMA (Tri-diagonal matrix algorithm) in a numerical iteration procedure.

The implicit formulation in time is selected because it provides a more stable solution regardless of time step size (Versteeg and Malalasekera 1995). For the explicit solution routine to ensure the same stable solution, a substantially smaller time step size is required. In the explicit formulation, the maximum time step size required to ensure stability is strongly dependent on the size of the cell volumes and the thermodynamic properties of the materials used. Riise (2008) suggested that the thermal conductivity can be set to a very large number and the specific heat capacity to a very small number, in the cell volume, to simulate the grid cell that represents the eroded material. But this is not preferred since it results in a very small time step for the explicit solver. The time step size for the implicit formulation however, should be small enough to model the physics of the heat transfer simulated (Rian 2003).

A heat source/-sink term is included in the governing equation to allow heat contributions from endothermic and exothermic reactions within the cell volumes to be a part of the calculations.

The grid mesh that is used in the program consists of quadrilateral cell volumes that are not necessarily orthogonal on each other (Riise 2008). The layout for an example grid is shown in figure 1.1.

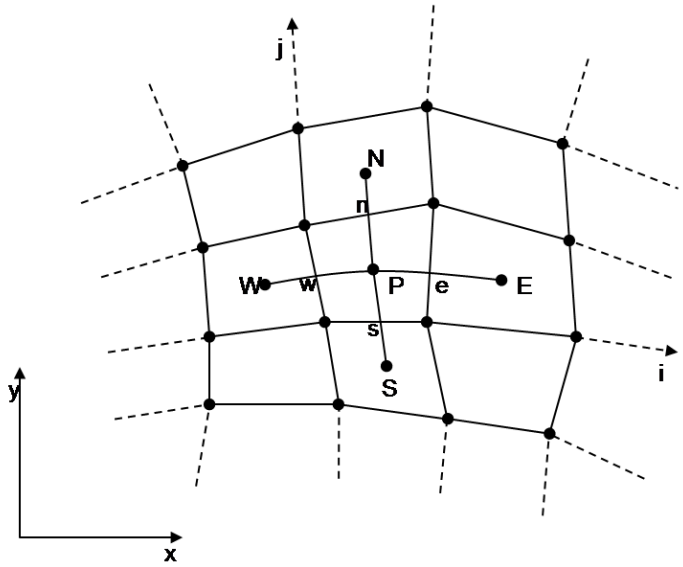


Figure 1.1: The curvilinear non-orthogonal coordinate system

From the notation given in figure 1.1, the heat flux from cell volumes “W”,”E”,”S” and “N” are calculated through the cell walls “w”,”e”,”s” and “n” into cell volume “P”.

1.1.1 The fully implicit scheme

Versteeg and Malalasekera (1995) express the transient heat equation in two dimensions of the inner cell volumes by:

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial i}\left(k \frac{\partial T}{\partial i} \right) + \frac{\partial}{\partial j}\left(k \frac{\partial T}{\partial j} \right) + \overline{S} \qquad (1.1)$$

By using the fully implicit discretisation approach on equation 1.1, a numerical approximation is found. The calculation molecule for the temperatures is shown in figure 1.2.

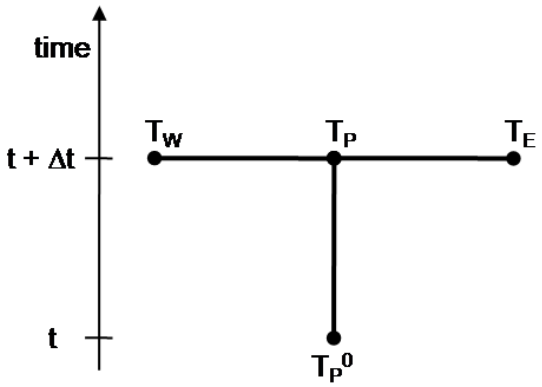


Figure 1.2: The implicit calculation molecule (Rian 2003)

A first order backward differencing scheme is used in time, while a second order central differencing scheme is used in space (Rian 2003). The discretisation procedure is outlined in appendix A and yields:

$$\underbrace{\left(\frac{\rho c \Delta V}{\Delta t}+\frac{A_e}{R_{I,e}}+\frac{A_w}{R_{I,w}}+\frac{A_n}{R_{J,n}}+\frac{A_s}{R_{J,s}}-S_p\right)}_{a_p} T_p = \underbrace{\frac{A_e}{R_{I,e}}}_{a_e} T_E + \underbrace{\frac{A_w}{R_{I,w}}}_{a_w} T_W + \underbrace{\frac{A_n}{R_{J,n}}}_{a_n} T_N + \underbrace{\frac{A_s}{R_{J,s}}}_{a_s} T_S + \underbrace{\frac{\rho c \Delta V}{\Delta t}}_{a_p^0} T_p^0 + S_u \quad (1.2)$$

Here "0" denote the value at the previous time step. For simplicity, the terms in front of the temperatures are represented by "a" and a subscript of their position related to the cell volume. This terminology is also used in the program code.

1.1.2 Source term

By introducing a source term, the modified G2DHeat program can handle more complicated problems than the old program. That includes both internal endothermic (heat sink) and exothermic (heat source) reactions which can be caused by pyrolysis of rocket materials or self-heating fuels. By including such reactions in the calculations of heat problems, more accurate results can be achieved. The existence of internal endothermic and exothermic reactions is known, but has never been looked at, except through manipulation of the conductive heat transfer coefficient, or by other less effective means (Myklebust 2008).

Linearisation of the source term in equation 1.1 gives:

$$\overline{S}\Delta V \Delta t = \left(S_P T_P + S_u\right)\Delta t \quad (1.3)$$

Here S_p and S_u are the two terms representing the actual source. S_p is the temperature dependent part of the source, while S_u is the non-temperature-dependent part.

In the modified program, an input routine for using non-temperature-dependent sources is created. This routine requires that the implicit solution routine is used and the sources are specified in the input file according to the sequence:

```
SOURCE                  -> Indicate the start of defining sources in the input file
3                       -> Number of sources
1   1 31   1 21  2500   -> Position and size of source[W.m-3], which is defined as:
2   5 10   5 10 -3600      <Grid block><start I-><end I->< start J->< end J-coordinate><source
strength>
2 15 25 15 25    700
```

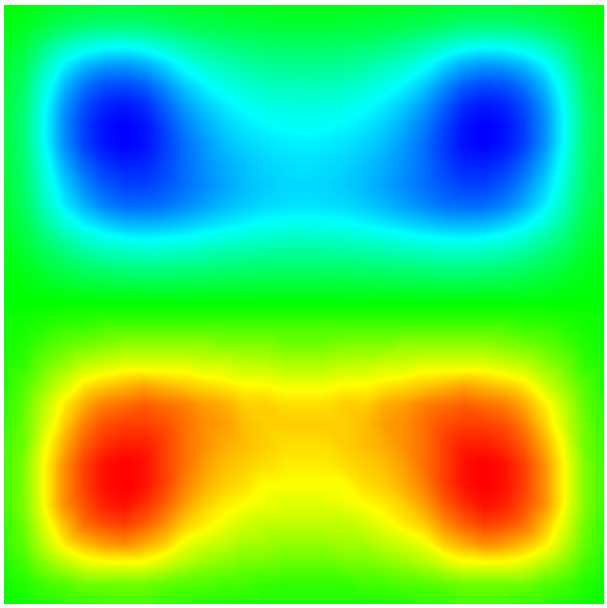


Figure 1.3: Example of heat sources and heat sinks in an insulated and initially temperature homogenous geometry

The source terms are also used to include radiation terms, as shown in chapter 1.1.4.

1.1.3 Collection of material properties

Since material properties data are stored in tables inside an executable file in the previous version of the computer program, these are difficult to adjust and require a user with insight in computer programming to do so. In the modified program, however, this has been changed. Here material properties are stored in an external file where they are easily modified. In the modified program the user only needs to specify the filename containing the material properties in the input file before starting the program. This is executed by the sequence:

```
DEFMATERIAL          -> Indicate the start of defining material properties in the input file
2                    -> Number of data files containing the material's properties
ALU.b                -> Filename with id=1
TITAN.b              -> Filename with id=2
0                    -> Number of materials that are decomposing by pyrolysis (See chapter 3.4.1)
3                    -> Number of material areas that are specified
1 2 1 81 1 11        -> Material area in the grid which is defined as:
1 3 1 11 1 91            <Filename id><grid block><start I-><end I-><start J-><end J-coordinate>
2 1 1 81 1 81
```

In appendix F, data files with material properties are shown.

1.1.4 Boundary conditions

Multiple boundary conditions are available in the old G2DHeat program (See Riise 2008), and to incorporate these in the implicit solution routine some "numerical tricks" are performed. Initially, all the grid edges are insulated ($a_* = 0$) until they are initiated with a boundary condition. In the following explanation one can imagine that there are fictive cell volumes outside the border.

The heat flux is added directly to the source term at the border:

$$S_u = \dot{Q} \qquad (1.4)$$

The convective heat contribution is added trough a modified thermal resistance at the border, and the ambient temperature is set as temperature of the fictive cell volume:

$$R_* = \frac{1}{h} + R_{conduction} \qquad (1.5)$$

$$T_* = T_\infty \qquad (1.6)$$

In the situation of radiation at the border, this is incorporated in both the temperature dependent and non-temperature dependent part of the source term:

$$S_P = \left(-A_* \sigma \varepsilon_* 4 \left(T_P^3 \right)^0 \right) \qquad (1.7)$$

$$S_u = A_* \varepsilon_* \sigma T_{surrounding}^4 + \left(A_* \sigma \varepsilon_* 3 \left(T_P^4 \right)^0 \right) \qquad (1.8)$$

The entire linearisation process of the radiation terms is shown in appendix B.

If the temperature history at the surface is known, it can be desirable to specify a temperature directly. This boundary condition is new to the program, and is easily added by:

$$R_* = R_{conduction} \qquad (1.9)$$

$$T_* = T_{spesified} \qquad (1.10)$$

1.1.5 Input specification for the implicit solution routine

In order to start the solution routine, it is required that the start and ending time, together with an appropriate time step size, are specified.

The input specification for the implicit solution routine is a bit different from the explicit. While the explicit uses a combination of time and maximum number of iterations, the implicit uses time only. The implicit solution routine is executed by:

```
IMPLICIT            -> Indicate the use of an implicit equation solver
0.01 0.0 0.1 120.0  -> Convergence and time parameters that are defined as:
                       <Convergence parameter><start time><time step><end time>
```

How to select relevant time interval length and time step size depends on the thermodynamics. Rocket missiles, for example carried by airplanes, will at first be heated by their surrounding air, then by the heat from the internal combustion when they are fired. Finally they will be cooled in the same surrounding air (Myklebust 2008). In the input file, when calling the equation solver for different time spans, the user can perform changes to the boundary conditions, and therefore manage such situations.

It is sometimes essential for accuracy that a small time step size is used (Versteeg and Malalasekera 1995). If this is done, the overall error in the solution is reduced, but comes at the expense of a longer total calculation time. The convergence parameter, however, controls the error in the iteration procedure. In detail, it checks the temperature deviation from the previous iteration and then decides if a new iteration is necessary. So by selecting small values for both of these parameters, the user is able to minimize the total error in the solutions.

1.2 Multi-block grid

Grid topology with multi-blocks is often used to describe a complex geometrical structure. The precision of the numerical calculations done with the grid, however, is strongly dependent on the user's skills to create and select a suitable grid topology (Ørbekk 1994).

Generally, the computer program uses a multi-block grid to describe the geometry. It also allows grid blocks to be partially interfaced with each other, which reduces the total number of grid blocks used. As shown in figure 1.4.

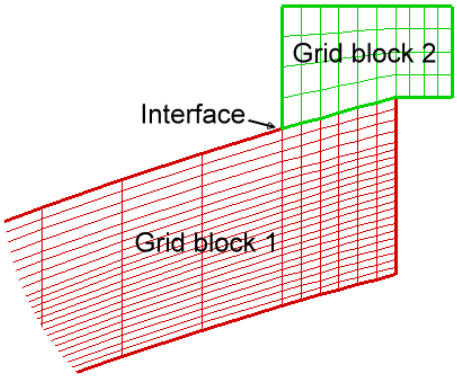


Figure 1.4: Example of two partially interfaced grid blocks

1.2.1 Shadow cells

Since the computer program works in an iterative fashion when solving the system of heat equations, grid block by grid block, shadow cells at the borders of each grid block are necessary. By using these, the temperature information from the previous iteration is passed to the interfacing grid blocks, and the next iteration procedure within the different grid blocks can start over again. When the system of heat equations in all grid blocks have converged, the program moves to the next time step. How the temperature information is exchanged can be seen in figure 1.5.

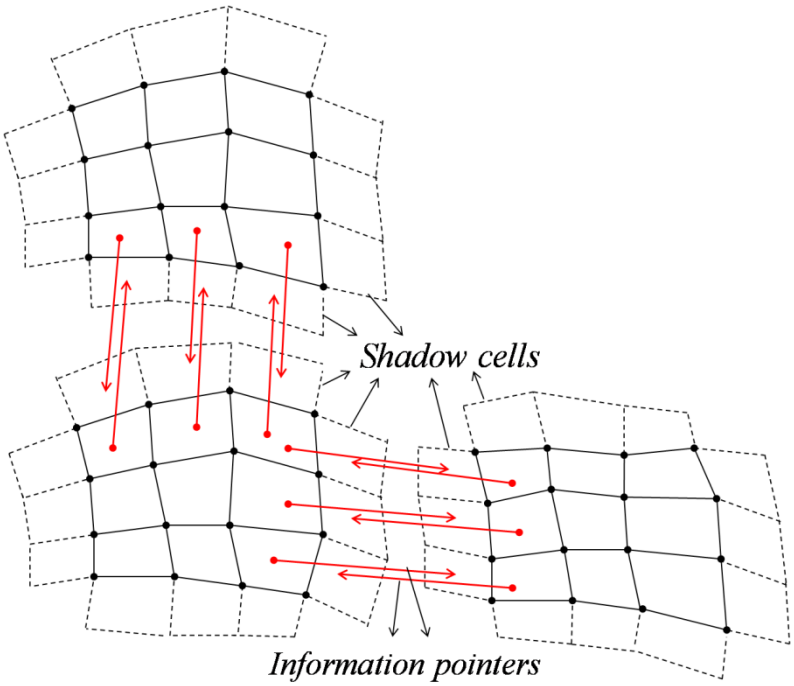


Figure 1.5: The information exchange between grid blocks

1.2.2 Specifying interfaces between grid blocks

Specifying the interfaces in the input file and which interfacing blocks the shadow cells should collect their information from, must be performed in the right manner. If not, temperature information will be collected from wrong cells and the solution result will contain discontinuities.

The user first specifies the left side or top side block before its connected right side or bottom side block, in the interface layout. Also, in which directions these are specified must be according to the directions shown figure 1.6. The interface layout for the cell block in the top left corner of figure 1.6 can be specified with neighbour block 2 (to the right) and neighbour block 3 (below) as follows:

```
INTERFACES      -> Indicate start of specifying interfaces in the input file
2               -> Number of interfaces that are specified
1  1  1 1 6     -> Interface between block 1 and 3 are specified by the two lines in the format:
3  1 11 1 6             <Grid block><start I><start J ><direction><number of steps in that direction>
1  6  1 2 5     -> Similar for the interface between block 1 and 2.
2  1  1 2 5
```

Four different values of the direction parameter are used to describe the axis directions, namely: 1 indicates positive I-direction, 2 positive J-direction, 3 negative I-direction and 4 negative J-direction.

Even though grid blocks have their own coordinate system, which may result in different axis directions for two grid blocks interfacing each other, the user can select a local direction parameter for the grid blocks and still obtain the correct directions shown with red arrows in figure 1.6.

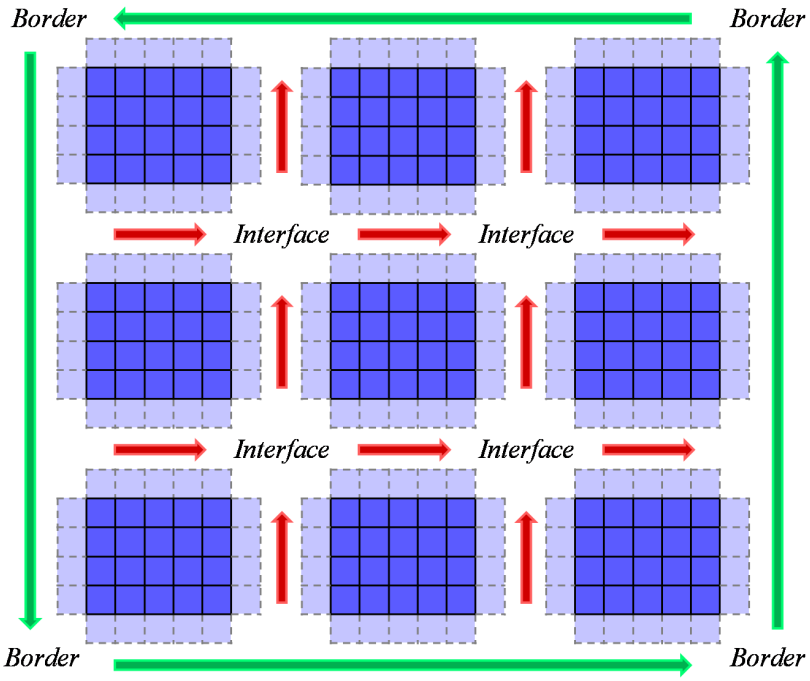


Figure 1.6: The border and interface directions

1.3 The implicit solution routine

The implicit solution routine consists of three main parts, an initialisation procedure, an update procedure and an equation solver. First the information pointers in the shadow cells are created in the initialisation procedure. This is done only once for both for the boundary edges and the interfacing grid block edges. When this is completed, the iterative part of the solution routine starts. Here a combination of both the equation solver and the update procedure is executed. Finally the results from the calculations are written to a file before the program moves to the next time step.

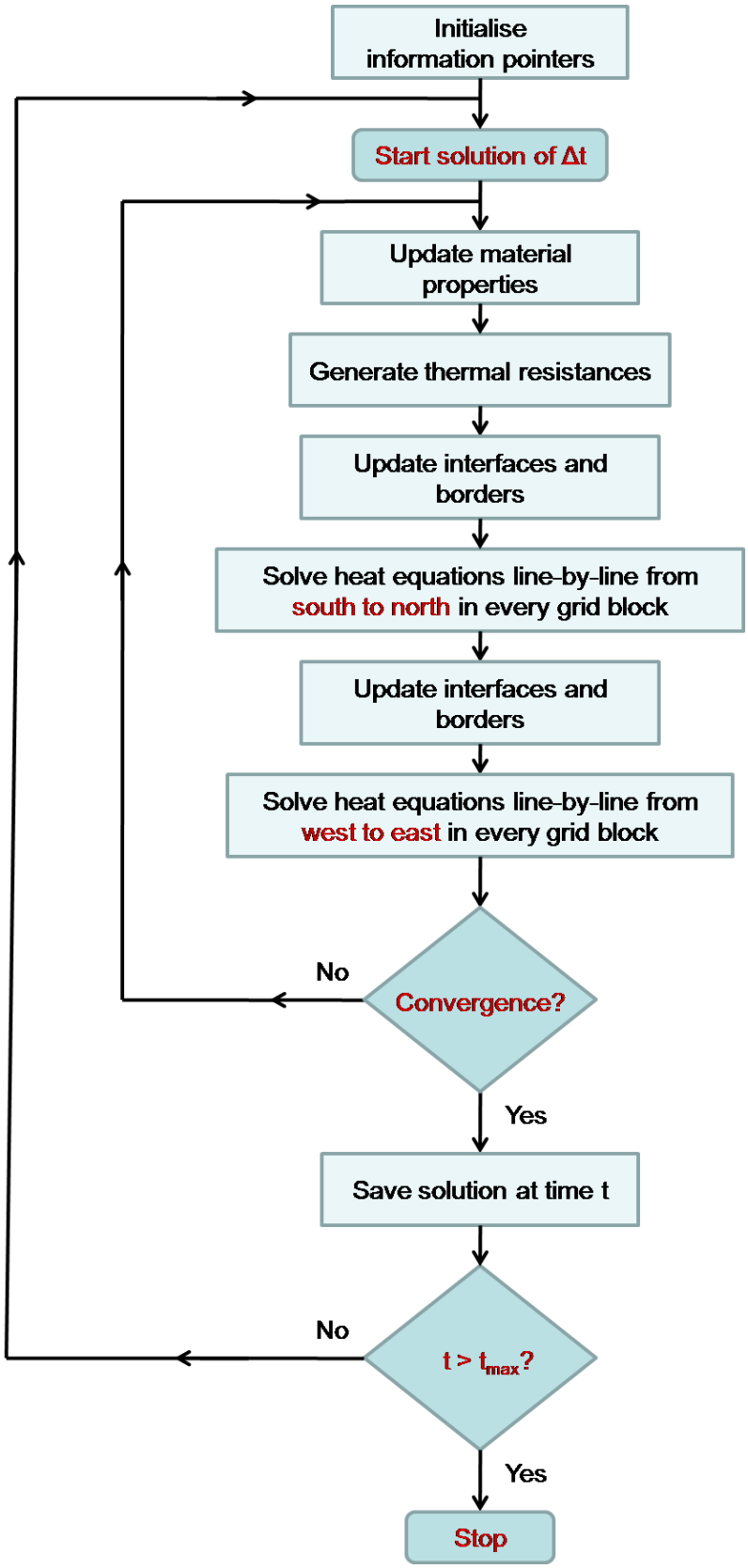


Figure 1.7: Step by step walkthrough of the implicit solution routine

1.3.1 Initialisation procedure

The initialisation procedure consists of two subroutines that are executed, INIT_IMPLICIT and BORDERS_IMPLICIT. Both are shown in appendix D with their connections to the rest of the program.

First, the pointer information for interfacing cells, together with the area of cell faces between the cells, are created and stored in shadow cells by the subroutine INIT_IMPLICIT. Then, the subroutine BORDERS_IMPLICIT uses the remaining shadow cells to store information of the boundary conditions and the face areas at the boundary.

1.3.2 Update procedure

The update procedure consists of the subroutines UPDATE_IMPLICIT, RESMAT and PICKMDATA, also shown in appendix D.

Since the equation solver solves the different grid blocks sequentially rather than solving the entire grid, an update procedure to exchange the temperatures between the grid blocks is needed.

To perform the temperature exchange, information pointers stored in the shadow cells by the initialisation procedure are used. Before the next iteration, the thermal conductivities are adjusted by the temperatures, and thermal resistances between the interfacing grid cells are created.

This temperature exchange is the primary part of the update procedure, and is executed by the subroutine UPDATE_IMPLICIT. Nevertheless, the other subroutines PICKMDATA and RESMAT are also important parts of the update procedure, as they collect the new material properties for the grid cells and generate the new thermal resistances between them.

1.3.3 Equation solver

There are two types of solution techniques for linear algebraic equations; direct methods and indirect or iterative methods (Versteeg and Malalasekera 1995). The modified computer program uses an iterative method, where an application of TDMA solves the two-dimensional domain until convergence.

TDMA is actually a direct method for one-dimensional cases, but can be applied iteratively for two-dimensional situations, in a line-by-line fashion. This is done by including the values of the equation terms from all the neighbouring lines in a source term.

Since grids of different sizes and boundary conditions of different complexness are solved by the iterative method, it sometimes requires a large number of iterations and time to reach convergence. To speed up the solution and reach convergence faster, the program uses a sweeping technique, from west to east and south to north, in the equation solver. This is shown in figure 1.8.

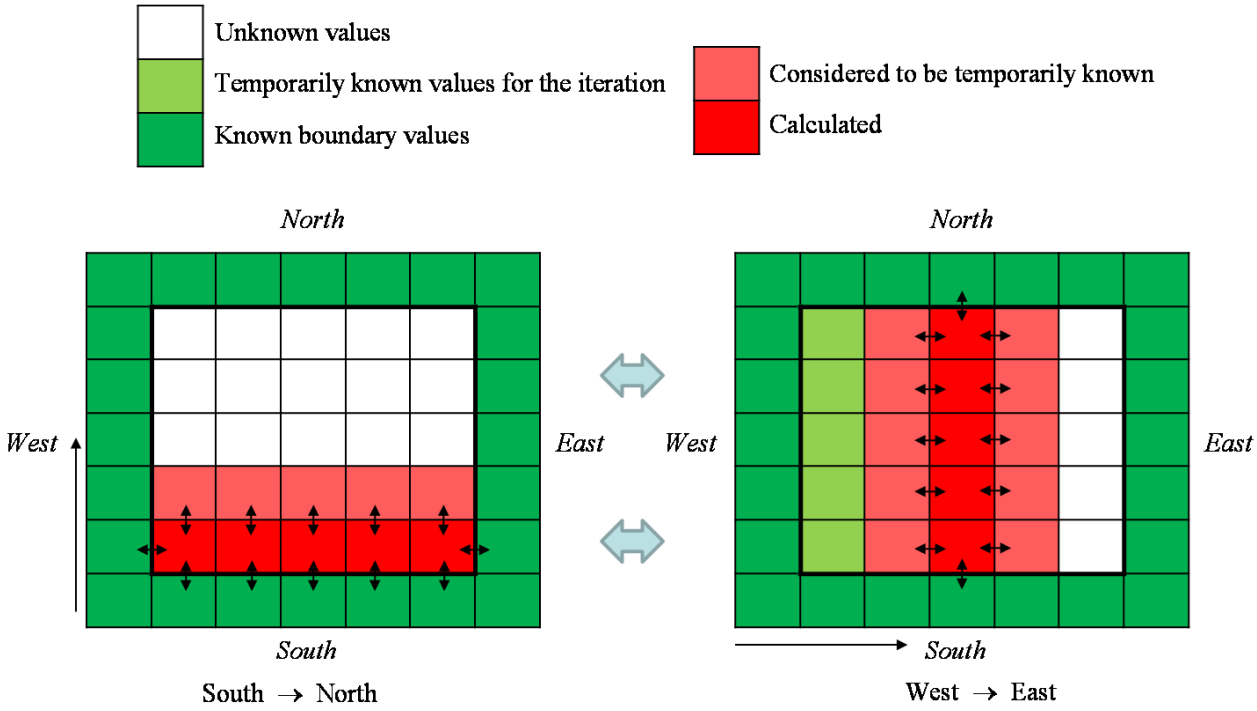


Figure 1.8: Sweeping technique used in the equation solver

1.4 Accuracy of the calculations

To validate the accuracy of the calculations done using the modified program, comparisons with the old explicit solver together with a commercial program (COMSOL Multiphysics) have been performed.

1.4.1 Implicit or explicit?

By using a simple heat problem with an analytical solution to assess the accuracy of the solution routines, the advantages and disadvantages of these routines are found by means of a comparison of the results. Hopefully, it will be easier for the user to decide which routine is the most favourable to use in other heat problems, when considering these. An overview of the initial and boundary conditions for the example problem is shown in figure 1.9.

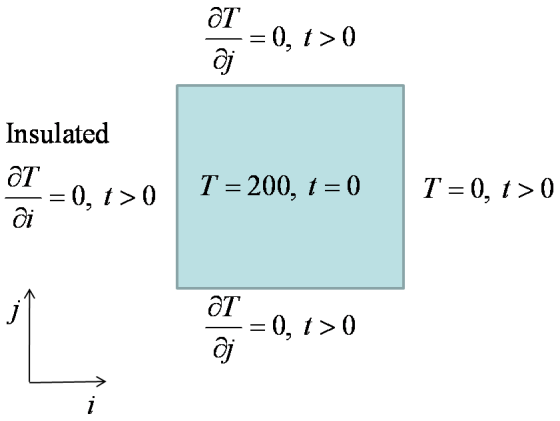


Figure 1.9: Initial and boundary conditions for the example problem

The analytical solution for the problem is given by Ozisik(Cited Versteeg and Malalasekera 1995):

$$T(i,t) = \frac{4}{\pi}\left[\sum_{n=1}^{\infty}\frac{(-1)^{n+1}}{2n-1}\exp\left(-\alpha\lambda_n^2 t\right)\cos\left(\lambda_n i\right)\right] + 200 \qquad (1.11)$$

where

$$\lambda_n = \frac{(2n-1)\pi}{2L} \qquad (1.12)$$

and

$$\alpha = \frac{k}{\rho c} \qquad (1.13)$$

To solve the problem numerically, a representative grid mesh of the plate geometry is created. In the grid mesh, all the cell volumes are created equal, with Δi=0.004m and Δj=0.004m.

The thin plate from figure 1.9 measures 0.02m by 0.02m and consists of aluminium. The material properties are assumed to be constant since the plate is a solid and the temperature variations are relatively small. Table 1.1 shows the values of aluminium.

Table 1.1: Material properties (COMSOL 3.4 2007)

Material	Thermal conductivity (W. $m^{-1}K^{-1}$)	Specific heat capacity ($J.Kg^{-1}K^{-1}$)	Density ($Kg.m^{-3}$)
Aluminium	160	900	2700
Titanium beta-21S	7,5	710	4940

The maximum time step size for the explicit equation solver is obtained using the criterion:

$$\Delta t \leq \left(\frac{\rho c \Delta V}{\frac{A_e}{R_{I,e}} + \frac{A_w}{R_{I,w}} + \frac{A_n}{R_{I,n}} + \frac{A_s}{R_{I,s}}} \right) \qquad (1.14)$$

This criterion ensures a physically correct and a convergent solution when using the explicit solution routine (Rian 2003). In this example problem the maximum time step size becomes approximately 13 seconds. The following figure shows a comparison between the implicit, explicit and exact solutions after 136 seconds, all of which 8 seconds are used as time step size.

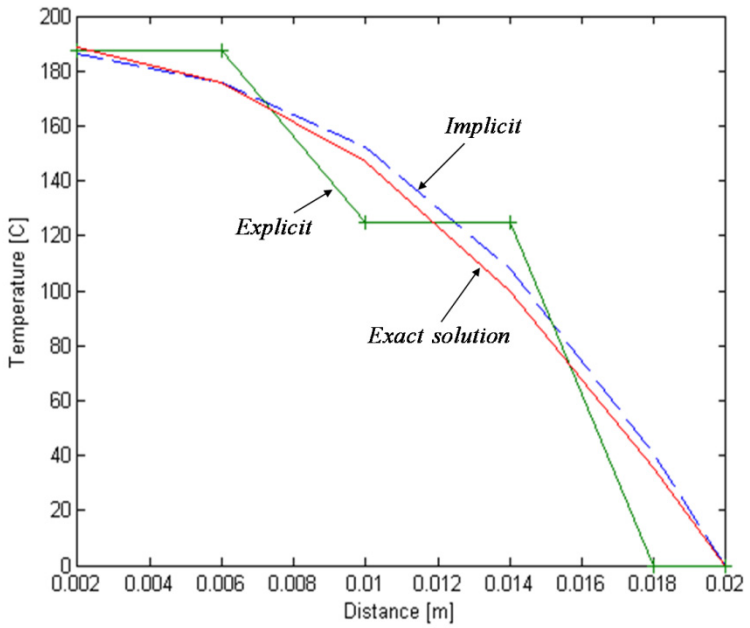


Figure 1.10: Comparison of implicit, explicit and exact solution

From figure 1.10 it can be observed that the explicit solution differs much more from the exact solution than the implicit does, which also agrees with the literature (Versteeg and Malalasekrea 1995). Here the error in the explicit solution is very clear, because a relatively large time step size and only five grid points are used to represent the temperature distribution in the I-direction. When the number of grid points is increased, the total error for both solutions is drastically reduced. If the time step is reduced instead, both solutions are improved and especially the explicit. By reducing the time step even further, both solutions approach the exact solution, as seen in figure 1.11, and eventually no differences are noticeable.

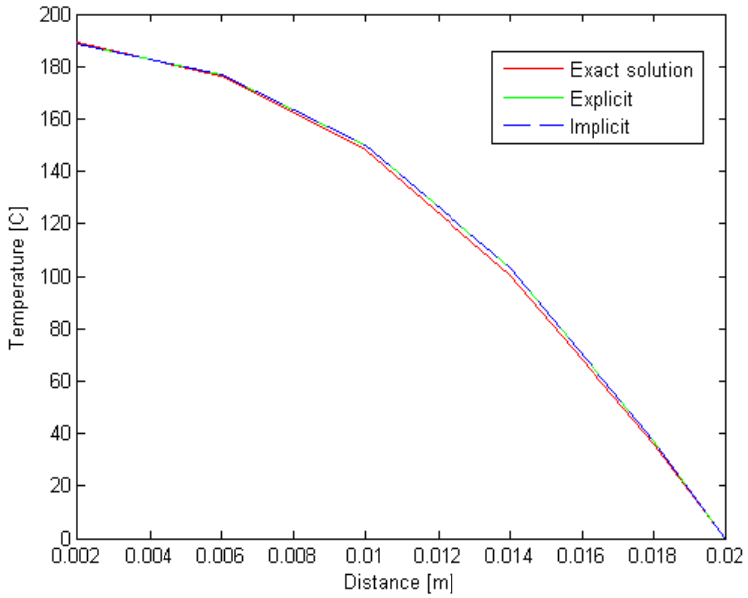


Figure 1.11: Comparison of implicit, explicit and exact solution

1.4.2 Using a multi-block grid

As previously mentioned, the grid may consist of multiple grid blocks, each with its own coordinate system. An example is shown in figure 1.12.

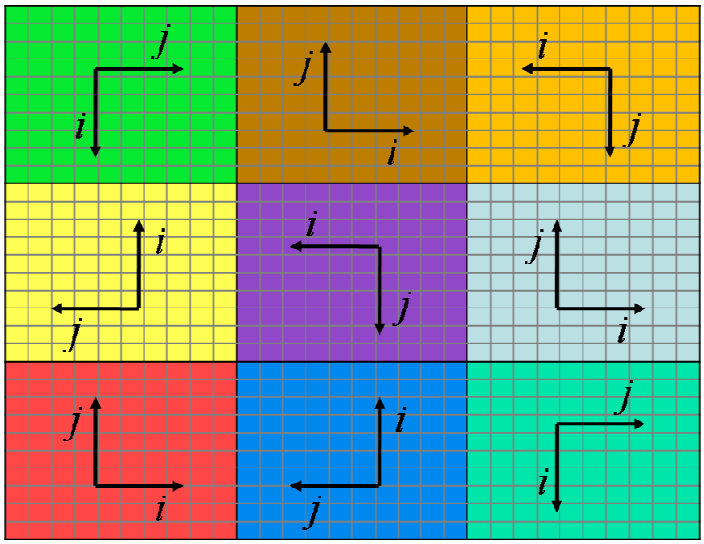


Figure 1.12: A multi-block grid with different coordinate systems

Note: Figure 1.12 shows all the grid blocks with orthogonal axis systems, even though this is not always the case, as shown in figure 1.1.

The next example is used to determine if there are any differences in accuracy when using the multi-block grid from figure 1.12, or a similar single-block grid in the implicit solution routine. Theoretically

these should give the same results, because the problem is unchanged and the same size and number of cell volumes are used.

Control points for the temperatures are created randomly, and their positions together with the chosen constant temperature boundary conditions are shown in figure 1.13. The geometry consists of aluminium which is initially set to 200 Kelvin (K), and the properties from table 1.1 are used. An indication of the temperature profile is also shown in figure 1.13. Finally the results from the implicit solutions are compared with an explicit solution where the time step is set to an infinitesimal value so it more or less represents the analytical solution.

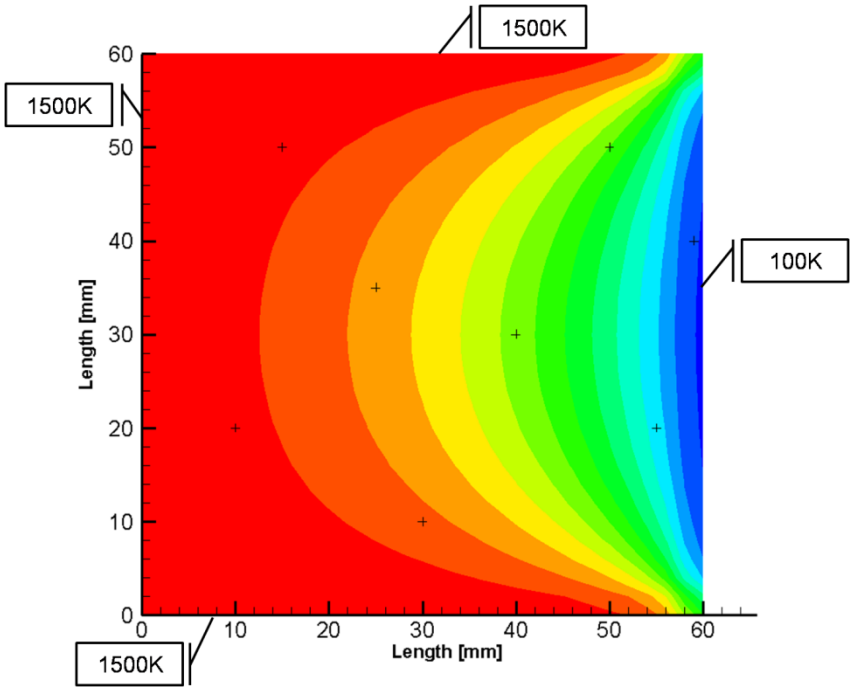


Figure 1.13: The geometry used when comparing single- and multi-block grids

Even after a simulation time of 150 seconds and using a time step size of 0.01 seconds, the results seem to be in agreement with the earlier predictions. There are no differences in accuracy when using the single- or the multi-block grid in the implicit solution routine. The results from the explicit routine however, yield some unexpected results. Even though a much smaller time step value of 0.0001 seconds is used in the explicit routine, the results from the implicit routine match perfectly. A reason for this can be the simple boundary conditions used. However, it indicates that the implicit routine can give just as good results as the explicit, even when a much larger time step is used.

1.4.3 A two-dimensional benchmark model

For a two-dimensional accuracy test of the modified G2DHeat, a benchmark model for comparison is created using COMSOL Multiphysics. This is a commercial finite-element based program for simulating heat transfer caused by convection, conduction and/or radiation (Comsol 2008). Assuming that COMSOL delivers trustworthy results, comparisons with these can be used to find how accurate the two dimensional calculations from the modified G2DHeat program are.

The model consists of a two dimensional plate where the west and south side are isolated, while the east and north side are exposed to a convective heat flux. The plate measures 0.09m by 0.09m and consists mostly of titanium, but also a layer of aluminium facing the outer heat source. The material properties are given in table 1.1 and the model is shown in figure 1.14.

Initially, the plate temperature, the convective heat transfer coefficient at the border and the external temperature are set to 350K, 8 $W.m^{-2}K^{-1}$ and 2000K. To numerically solve the problem, the plate geometry is divided into approximately 8000 cell volumes. Using these parameters, the model is simulated with a time step size of 0.01 seconds for 120 seconds both in COMSOL 3.4 and the modified G2DHeat program. These values are randomly selected, and are not necessarily physically correct.

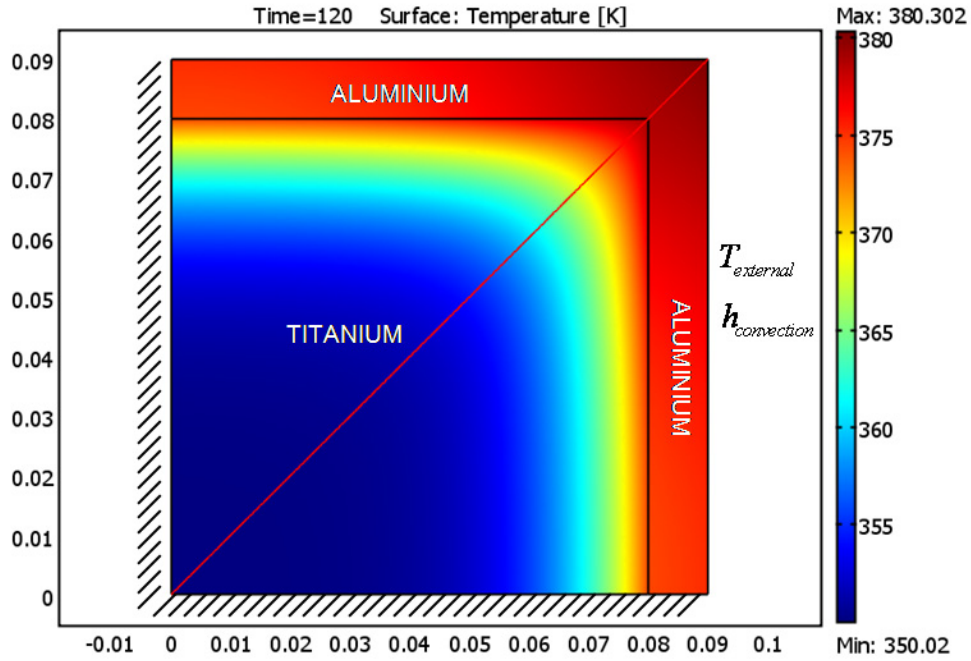


Figure 1.14: Shows the section cut where the temperatures are compared

The final results are compared using temperature values along the diagonal line shown in figure 1.14, and the results when using COMSOL and the implicit solution procedure, are both shown in figure 1.15.

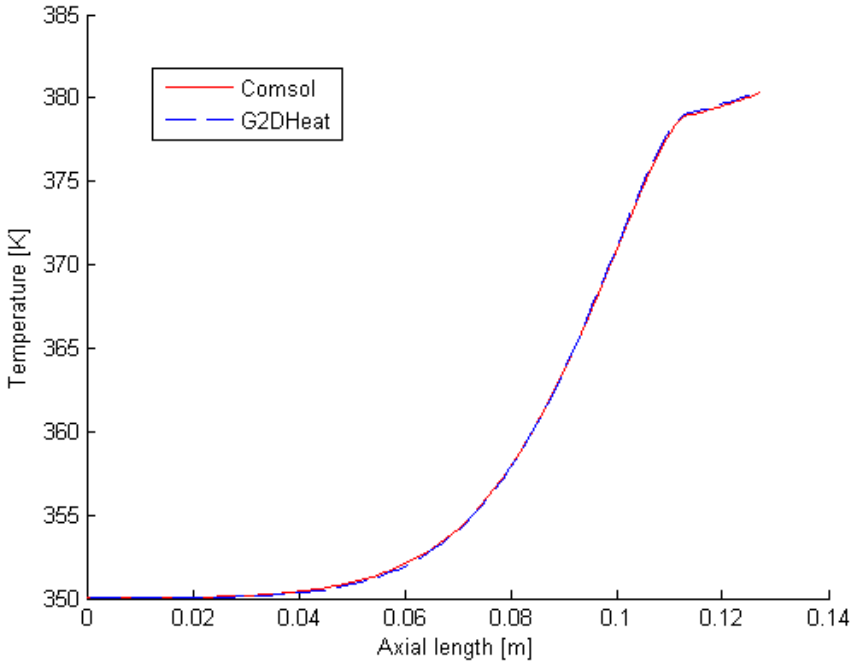


Figure 1.15: Comparison of solutions from G2DHeat using the implicit solution routine and COMSOL

When observing figure 1.15, it is hard to spot any differences between the solutions. Nevertheless, a maximal error of 0.1657K is found at the axial position 0.1124m by examining the results. And from analyzing the differences between the solutions through the entire graph interval, an average difference of 0.0746K is found.

Sometimes a small error in the solution can be accepted, but it depends on the problem. For example, a small temperature error of a few degrees in a rocket engine is of less importance than in a freezer, because it is designed to withstand greater temperature variations. But if a similar error is neglected in the freezer, the ice cream may start to melt.

There can be many reasons why some degree of error is observed in figure 1.15. For instance, since the programs use different cell layouts, this may cause an unfortunate distribution of the temperatures in the grid. Also, when an interpolation routine is used to find certain points, small errors are generated. To reduce this type of error, a better interpolation routine and/or more cell volumes in the grid can be applied (Versteeg and Malalasekera 1995). Another possible source of error can simply be a poor selection of precision for the variables in the computer programs. But this is not likely, because the different compilers and especially the newer compilers allow large number of significant digits. The reason for the error can also be the different ways the solution routines solve their grids. COMSOL uses a direct method, while G2DHeat uses an implicit iterative/indirect method or an explicit direct method. Assuming that the same number of cell volumes is used, the error can be reduced in G2DHeat by selecting a smaller time step (Rian 2003).

1.5 Measuring computation time

To determine the amount of time used by the program before a solution is reached, the program uses an intrinsic subroutine which returns the CPU time in hundredths of a second. This represents the amount of time the CPU is actually executing instructions (Chapman 2004).

For comparison, the explicit solution routine is modified from its original state to collect material properties after every time step and not in intervals of 25 time steps as previously. The implicit solution routine, however, updates these for every iteration, and hence also for every time step.

In these simulations, the geometry, the boundary conditions and the material properties from chapter 1.4.2 are used. From the geometry new grids that measure 30×30, 60×60, 90×90 and 120×120 cell volumes are created. Each is then divided into a single-block and a multi-block representation before the calculations are performed. The multi-block grids consist of nine equally large grid blocks and are similar to the grid shown in figure 1.12. Time step sizes of 0.01s and 0.0006s are used for the implicit and the explicit solution routines in all the simulations, respectively. The explicit time step size is within the limits given by the criterion in equation 1.14. The results are presented in figure 1.16.

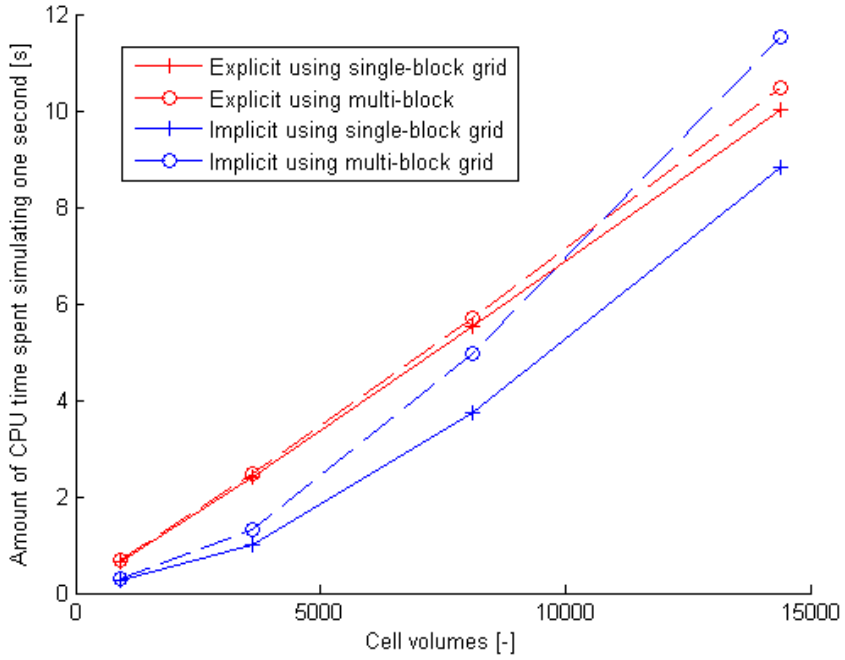


Figure 1.16: Shows the differences in computation time using the implicit solver or the explicit solver

Figure 1.16 shows the amount of CPU time spent calculating one second when using a different number of cell volumes in the solution routines. During the execution of most programs, the CPU is idle much of the time while the computer fetches data from the keyboard or disk. The CPU time of an executing program, therefore, is generally much less than the total execution time of the program (Chapman 2004).

For grids using less than 9000 cell volumes, it is seen from figure1.16 that the implicit routine has a smaller calculation time than the explicit. It can also be seen that using multi-block grids require more computational time than using single-block grids for both solution routines and all grid sizes.

This is because multi-block grids require additional subroutines to be executed. The single block calculation performed by the implicit routine remains lower than the explicit even when 14400 cell volumes are used. However, the gradient for the amount of time spent simulating one second, at this point, implies that the time spent by the implicit soon will pass the explicit.

The amount of CPU time spent on calculating one second, increases almost at a constant rate for the explicit routine, as seen in figure 1.16. This is not the case for the actual time spent by the program, since the memory, at some point, is insufficient to store information from all the cell volumes at the same time. Therefore, information must be stored on the hard drive, causing more time for the CPU to collect information, thus more time is spent simulating by the program.

Chapter 2: Pyrolysis and charring ablation

This chapter is intended to describe the physical mechanisms that involve pyrolysis and charring ablation in the modified G2DHeat program. In chapter 2.5 a proposal for further development of the boundary condition necessary for simulating charring ablation without specifying the recession rate is presented. Values for silica phenolic are used as an example of thermophysical and thermodynamic properties for an ablative material, and are also used for the simulations in chapter 4 and 5.

2.1 General

In solid rocket motors it is often used ablative materials as insulation to protect components (Sutton and Biblarz 2001). With ablative cooling, part of the ablative material is sacrificed to absorb heat and prevent heat from travelling further into the protected structure. This involves transient heat transfer processes, reaction kinetics at the surfaces, transpiration cooled boundary layer phenomena and decomposition processes within the solid. Loss of mass from the ablative materials will generally be the limiting design factor when selecting an initial thickness for the insulation (Sutton and Biblarz 2001).

Some ablative materials simply melt or sublimate only at the surface, while others partially decompose to provide a char layer. Materials such as graphite lose surface material only through chemical erosion. The G2DHeat program simulates heat transfer in composite materials where a solid residue of char is created from the pyrolysis reactions. For the most part these materials are phenolic resins containing aramid, glass, graphite or silica fibres (Rønningen 2001). The presence of a char layer in such materials imposes an additional thermal barrier without losing the good thermal absorption characteristics of decomposing materials.

The charring materials will pass through three distinct phases in their behaviour as times progresses (Rønningen 2001):

1. The temperature gets high enough for pyrolysis of the virgin material to start.
2. A char layer will form at the outer parts of the material, as the decomposition zone moves further into the material. The char layer will then be cooled by the out flow of gaseous reaction products from the decomposition zone, and to some extent reduce the heat transfer from the hot combustion gases at the surface.
3. Temperature at the surface will continue to rise, and eventually mechanical erosion, chemical erosion and/or melting of the material will begin to reduce the thickness of the char layer.

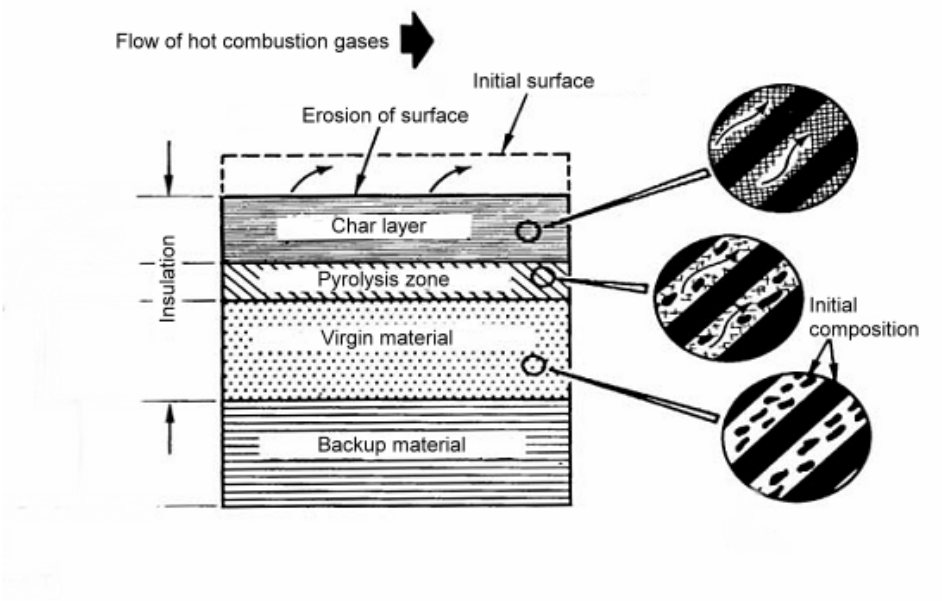


Figure 2.1: Schematic of layers associated with charring ablation (Rønningen 2001)

2.2 Kinetic model for material decomposition reactions

The thermal decomposition within the ablative material may consist of multiple reactions which separately break down the material components. These pyrolysis reactions reduce the density of the material due to gas generation, and therefore several physical considerations concerning the material must be dealt with (ASTHMA88/PC 1988):

- The change of material properties due to loss of mass
- Energy from the reactions occurring in the ablative material
- Energy exchange due to flow of pyrolysis gas

2.2.1 Independent parallel reactions

Assuming the effect of cracking in the ablative material is small, the decomposition process can be represented as a series of independent parallel reactions of the many material components (Austegard 1997).

Independent parallel reactions are described by:

$$\begin{aligned} Mass(A) &\xrightarrow{Reaction(1)} Residue(A) + Gas(A) \\ Mass(B) &\xrightarrow{Reaction(2)} Residue(B) + Gas(B) \end{aligned} \qquad (2.1)$$

Where “A” and “B” are different chemical species and the reactions “1” and “2” are independent of each other.

2.2.2 Kinetic parameters

Varhegyi and Antal(cited Austegaard 1997) describes the independent decomposition reaction of a single component by:

$$\frac{d\alpha_i}{dt} = A_i e^{-\frac{E_i}{RT}} \left(1-\alpha_i\right)^{n_i} \qquad (2.2)$$
$$i = 1,2,3,....N$$

Where

$$\alpha_i = \frac{\rho_{0i} - \rho_i}{\rho_{0i} - \rho_{ri}} \qquad (2.3)$$

The initial (ρ_{0i}) and residual (ρ_{ri}) densities of component "i" are known constants. In order to find the mass loss to volume it is necessary to rewrite equation 2.3 as follows:

$$\rho_i = \rho_{0i} - \left(\rho_{0i} - \rho_{ri}\right)\alpha_i \qquad (2.4)$$

Differentiating equation 2.4 with respect to time and assuming $\rho = \sum_i^n \rho_i$ yields the expression:

$$-\frac{d\rho}{dt} = \sum_{i=1}^{n} \left(\rho_{0i} - \rho_{ri}\right)\frac{d\alpha_i}{dt} \qquad (2.5)$$

Here the rate of change of decomposing material density ($-\frac{d\rho}{dt} = \dot{\rho}$) is found. Finally, substituting equation 2.2 into equation 2.5 yields:

$$\dot{\rho} = \sum_{i=1}^{n} \left(\rho_{0i} - \rho_{ri}\right)\left(\frac{\rho_i - \rho_{ri}}{\rho_{0i} - \rho_{ri}}\right)^{n_i} A_i e^{-\frac{E_i}{RT}} \qquad (2.6)$$

Thermal degradation kinetic parameter values for silica phenolic are shown in table 2.1.

Table 2.1: Thermal degradation kinetic parameters of components in silica phenolic (ASTHMA88/PC 1988)

	Activation energy [E] ($J.kmol^{-1}$)	Pre-exponential factor [A] ($1.s^{-1}$)	Order of reaction [n] (-)
Resin A	71.14×10^6	1.4×10^4	3
Resin B	169.98×10^6	9.75×10^8	3
Reinforcement	-	-	-

By using TGA (Thermo-gravimetric analysis) the values of the different kinetic parameters are obtained. This method determines the weight loss of the material when it is heated with constant temperature increase (Austegard 1997).

The use of the kinetic model is limited to materials in which components decompose independently of each other. If ablative materials decompose differently, for example by additional reactions with each other, inaccurate results from the simulation can occur due to the fact that this is not accounted for in the calculations. Nevertheless it is considered to be a reasonable assumption, since the same kinetic model is employed in both CMA3 and ASTHMA. These programs are extensively used in thermal performance studies of spacecraft structures, ablating heat shields and rocket nozzles (Schoner 1970). Therefore, a great deal of testing and verification of the kinetic model has been performed, and the model proves to be adequate for application in G2DHeat.

2.2.3 Definition of material densities

In the G2DHeat program the initial and residual densities of a material component can be expressed in terms of a fraction of the total material volume. This makes it possible to separate the decomposing part of the material from the non-decomposing part as shown in equation 2.7. The sum of densities, from components that can decompose form: $\rho_{decomposing\ part}$. Similar, the sum of densities from the non-decomposing components form: $\rho_{decomposing\ part}$. The non-decomposing part consists mostly of reinforcement (ASTHMA88/PC 1988).

$$\rho_{material} = \Gamma \rho_{decomposing\ part} + (1-\Gamma)\,\rho_{non-decomposing\ part} \qquad (2.7)$$

It is not possible, however, to specify a mass fraction as input. For this, the program user must convert the mass fractions into volume fractions by hand. When differentiating equation 2.7 with respects to time, the non-decomposing term vanishes. Since the only contribution to the loss of mass comes from the decomposing part of the material, equation 2.6, can be rewritten into:

$$\dot{\rho}_{material} = \Gamma \sum_{i=1}^{n} \left(\rho_{0i} - \rho_{r\,i}\right) \left(\frac{\rho_i - \rho_{r\,i}}{\rho_{0i} - \rho_{r\,i}} \right)^{n_i} A_i e^{-\frac{E_i}{RT}} \qquad (2.8)$$

The program allows an infinite number of reactions to be specified for each of the decomposing materials.

Table 2.2: Properties of components in silica phenolic (ASTHMA88/PC 1988 and Næss 1998a)

	Initial density ($Kg.m^{-3}$)	Residual density ($Kg.m^{-3}$)	Volume fraction (-)	Pyrolysis temperature (K)
Resin A	325.015	0.0	0.422	333
Resin B	973.926	518.998	0.422	550
Reinforcement	2066.380	2066.380	0.578	∞

From table 2.2 the total initial density of silica phenolic is calculated to be 1742,52 Kg/m^3. Similar, the residual density is found to be 1413,38 Kg/m^3, which is approximately 80% of the initial density. Weight loss curves for silica phenolic when using heating rates of 10, 20 and 50 K/min are shown in figure 2.2.

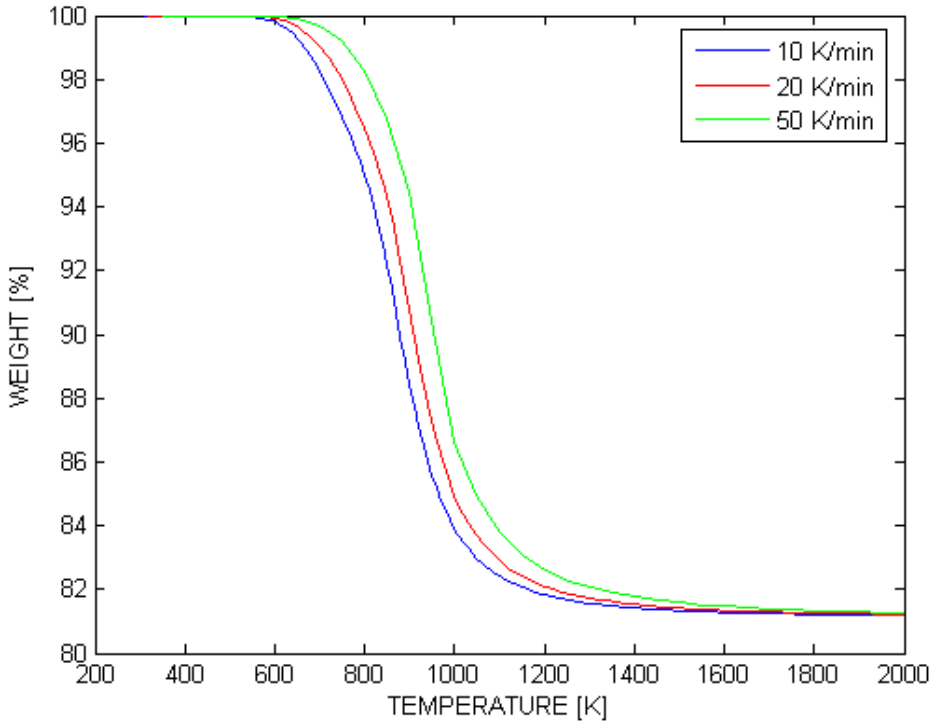


Figure 2.2: Shows weight loss when the heat flux is increased

The curves in figure 2.2 are calculated from equation 2.8 using the kinetic parameters in table 2.1 and the properties in table 2.2. The curves represent the sum of mass losses generated from the different reactions as the temperature increases. Regardless of time, these show that the pyrolysis reactions are dependent on the heat flux into the material. Physically, the start of the pyrolysis reactions may differ from one another. Therefore the program also allows the user to specify a threshold temperature (pyrolysis temperature) for the different reactions. The threshold temperatures for silica phenolic are shown in table 2.2. For material components in the non-decomposing part of the material the pyrolysis temperature is set to very high value, which is never obtained by the material.

2.3 Material properties

For an insulation material to become perfect, it should be made with components which have a low thermal conductivity, high specific heat capacity, some degree of elasticity, high heat of reaction and high threshold temperature for the pyrolysis (Rønningen 2001).

Since the decomposition process involves removal of mass from the cell volume as time progresses, the material properties will also change. To make adjustments for this, the decomposition state together with a linearly dependence between the properties of the fully charred and the virgin material is used. The fraction parameter which represents the decomposition state is given by:

$$x = \frac{\rho - \rho_r}{\rho_0 - \rho_r} \qquad (2.9)$$

The material properties then become:

$$C_p = xC_{p0} + (1-x)C_{pr} \qquad (2.10)$$

$$k = xk_0 + (1-x)k_r \qquad (2.11)$$

In addition to a thermal conductivity that is linearly dependent on the decomposition state, G2DHeat can use an anisotropic thermal conductivity. The thermal conductivities and the specific heat capacities as functions of the temperature for silica phenolic are shown in figure 2.3 and figure 2.4, respectively.

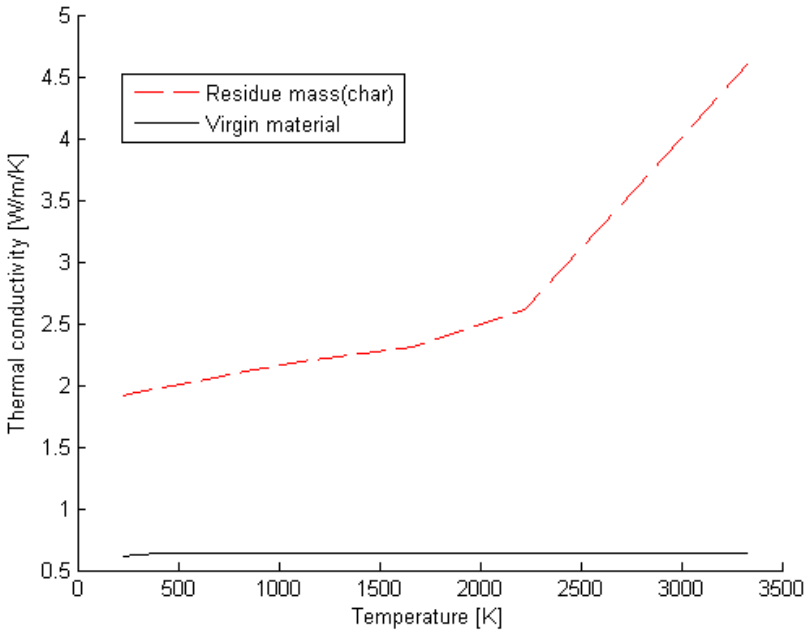


Figure 2.3: Thermal conductivity for silica phenolic (Næss 1998a)

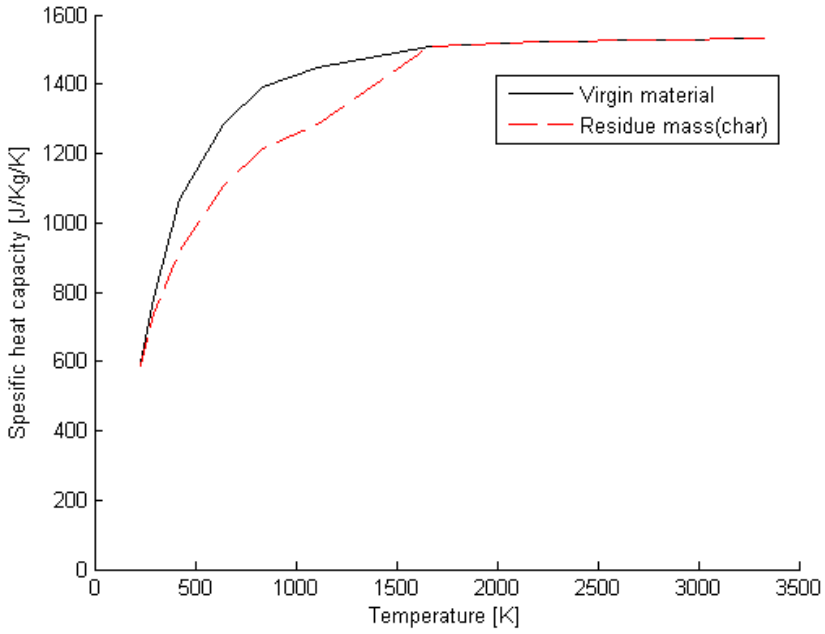


Figure 2.4: Specific heat capacities of silica phenolic (Næss 1998a)

Considering the material data provided (see appendix F), the assumption of the thermal conductivity to be a linear function of decomposition state is considered reasonable. Physically, the thermal conductivity within the ablative material is a function of heating rate into the material (ASTHMA88/PC 1988). This means that different heating conditions during decomposition gives different conductive properties for the material. CMA3 allows a functional dependence of the decomposition state to be specified (Schoner 1970). This functionality is not available in G2DHeat, but can easily be included if necessary.

2.4 Heat of pyrolysis and energy effects of transpiration

In section 2.3 the change of thermodynamic and thermophysical properties are determined on the basis of decomposition state. The corresponding energy is presented in this section. There are two primary events that are associated with the energy contribution from the pyrolysis; the energy concerning the pyrolysis reactions that occur, and the exchange of energy due to pyrolysis gases that percolate through the decomposing material. For ablative materials these are called energy absorption events (Austegard 1997).

In appendix C, the derivation of an energy balance for the material is described. The energy flux associated with the pyrolysis reactions is given by:

$$\dot{Q}_{pyr} = \dot{m}_{pyr} \Delta h_{pyr} \qquad (2.12)$$

Where $\dot{m}_{pyr}$ is the rate of material pyrolysing, and Δh_{pyr} the energy that is used for producing gases, called the “heat of pyrolysis”. The heat of pyrolysis can be expressed in terms of an enthalpy difference between the gases and the solids:

$$\Delta h_{pyr} = \left(h_g - \overline{u} \right) \qquad (2.13)$$

This term is determined by measurements using differential scanning calorimeter (DSC). This involves an apparatus which heats a sample of the material. The heat flux into the sample is kept at rates giving a constant increase in temperature in the sample and its holder. The heat flux is then logged together with time and the temperature of the sample holder. Since the sample is placed in a holder while it is heated, another similar test with an empty holder must be performed in order to find the base line (the reference holder). When there are no reactions left to occur in the sample, the heat flux into sample holder and the reference holder are equal when the temperatures inside them are kept constant. From the plots of heat flux versus time and versus temperature, the heat of pyrolysis, together with the specific heat capacity for the sample are obtained (Austegard 1997).

Sometimes the enthalpies of formation for each of the material components are known. Then the heat of pyrolysis is found from the enthalpy difference in equation 2.13 and the following calculations (ASTHMA88/PC 1988):

$$\overline{u} = \frac{\rho_0 h_0 - \rho_r h_r}{\rho_0 - \rho_r} \qquad (2.14)$$

This represents the amount of energy per mass which can decompose. The enthalpies of formation for the virgin material ($h^0_{f,0}$) and the residue products ($h^0_{f,r}$) at a reference temperature can, together with the specific enthalpy difference, determine enthalpies at other temperatures. Assuming there is a constant pressure within the material, the specific enthalpy difference can be represented by the specific heat capacity and a temperature difference (Moran and Shapiro 2004). In an equation form, this is expressed as:

$$h_0 = h^0_{f,0} + C_{p0}\Delta T \qquad (2.15)$$

$$h_r = h^0_{f,r} + C_{pr}\Delta T \qquad (2.16)$$

Where

$$\Delta T = T_{evaluated} - T_{reference}$$

The heat of formation for virgin silica phenolic and its residue products are at a reference temperature of 297.78K, -11.764×10^5 J/Kg and -123.115×10^5 J/Kg, respectively (Næss 1998a). In programs such as CMA3 and ASTHMA88, it is required that the pyrolysis gas specific enthalpy is specified as a function of temperature in their input file. These are then added to the heat of formation of pyrolysis gas which also is specified in the input file (Schoner 1970 and ASTHMA88/PC 1988).

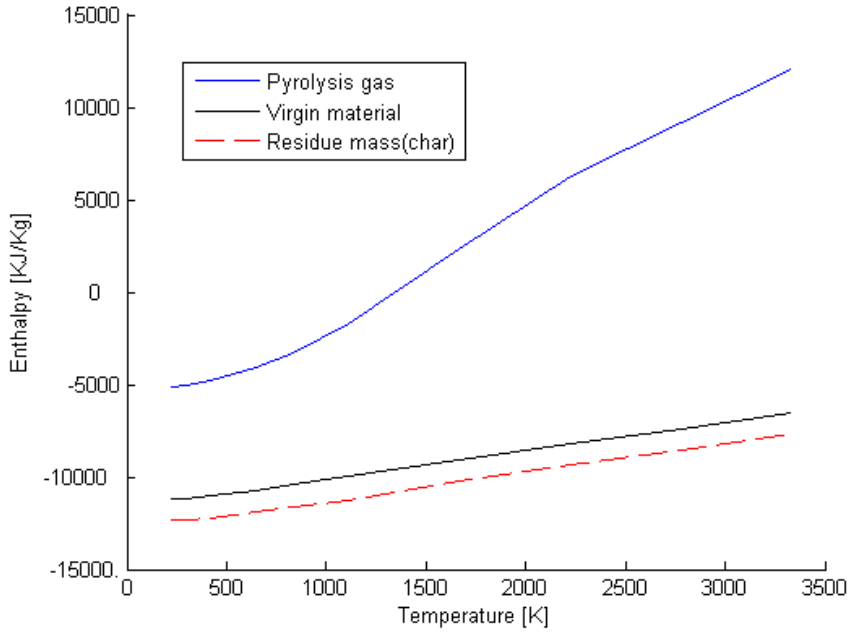


Figure 2.5: Enthalpy values for silica phenolic (Næss 1998a)

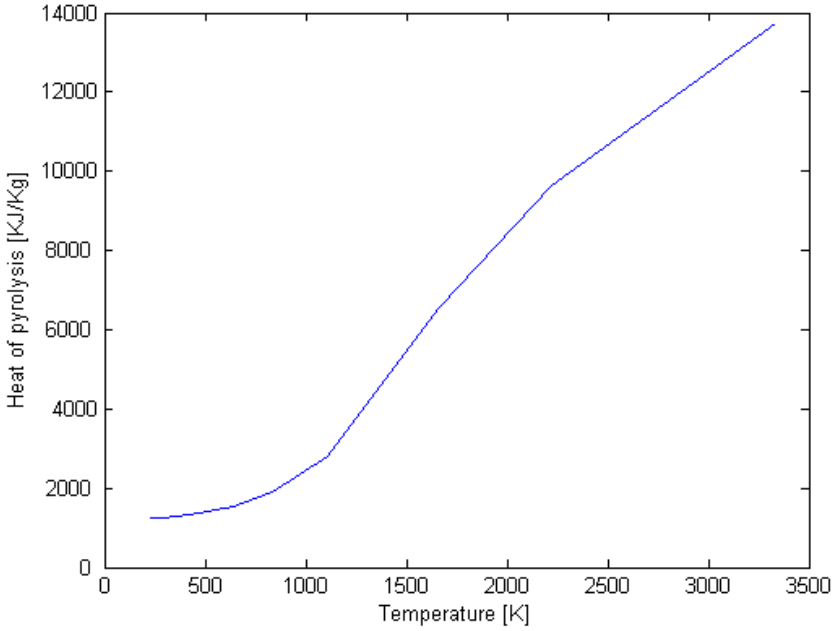


Figure 2.6: Heat of pyrolysis for silica phenolic (Næss 1998a)

As the temperature increases within the silica phenolic, it is seen from figure 2.5 that there is a slight increase of the virgin and residue enthalpies. In the same figure the pyrolysis gas enthalpy is observed to increase significantly more rapid than the others. From this, and the definition in equation 2.13, it is apparent that the heat of pyrolysis in figure 2.6 increases considerably due to the

increasing difference between pyrolysis gas and solid material. Therefore, the preliminary studies of the gas enthalpy are important to assure accuracy within reasonable limits when using the heat of pyrolysis in the calculations.

Assuming the pressure in the material to be constant, the pyrolysis gas enthalpy differences can be approximated with a temperature difference and the specific heat capacity for the pyrolysis gas. For silica phenolic the specific heat capacity of pyrolysis gas is linearly approximated from its enthalpy (Næss 1998a) and shown in figure 2.7.

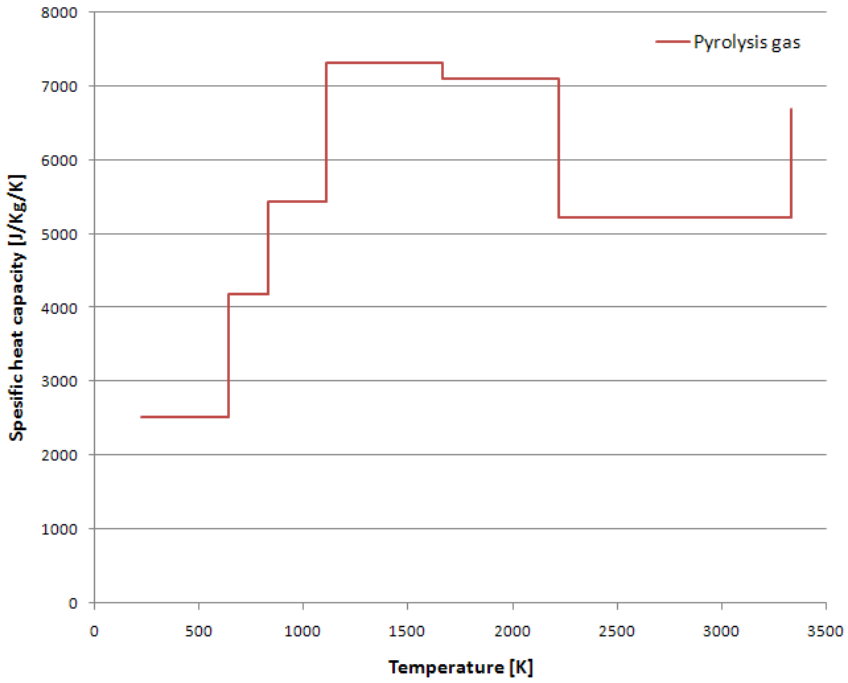


Figure 2.7: Specific heat capacity of pyrolysis gas from silica phenolic

To include the energy absorbed by pyrolysis gas when it is flowing through the material, the specific heat capacity of the gas is used. To calculate this, an analysis of the entire gas as a whole, or components alone, must be conducted (Grønli 1996).

2.5 Heat of ablation

In chapter 2.1 the behaviour of the ablative materials as time progresses is described. In the case when a char layer is present at the outer parts of the decomposing material, an increasing surface temperature can start different erosion events (Rønningen 2001). The G2DHeat program can simulate some of these events, but requires a recession rate to be specified. An explanation of these events is presented in chapter 3.2. In this chapter, however, a suggestion for how to obtain the recession rate is described.

Assuming the ablation of the char layer to occur at a constant known temperature where the char layer “melts” (ablation temperature), and the heat of ablation to be known through experiments, then the recession rate can be explicitly calculated from the energy balance shown in figure 2.8.

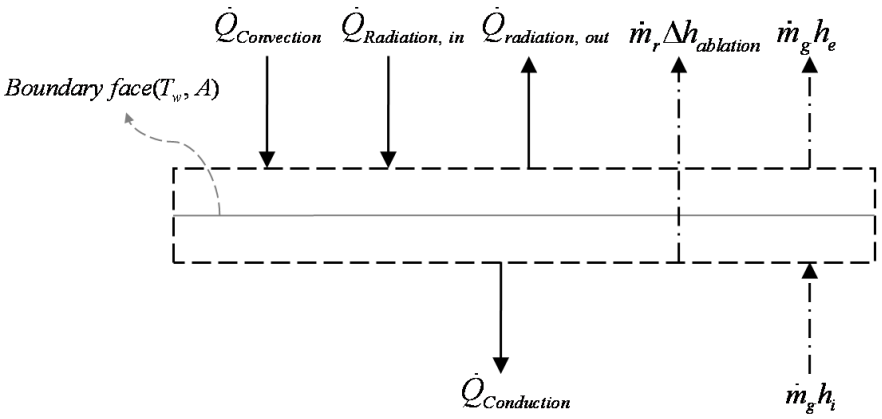


Figure 2.8: Control volume at the boundary

On equation form the energy balance in figure 2.8 becomes:

$$\dot{Q}_{Convection} - \dot{Q}_{Conduction} + \dot{Q}_{Radiation,\ in} - \dot{Q}_{Radiation,\ out} - \dot{m}_g (h_e - h_i) - \dot{m}_r \Delta h_{abl} = 0 \qquad (2.17)$$

When there is no surface recession present, the term $\Delta h_{ablation}$ (Heat of ablation) is zero. It is also absent when only a mechanical erosion rate is specified. The term $\dot{m}_r$ represents the rate of residual mass leaving the control volume. The term $\dot{m}_g$ represents the rate of pyrolysis gases leaving the control volume, and together with the enthalpy difference $(h_e - h_i)$ it expresses the rate at which energy is absorbed by the pyrolysis gases. This has just a minor impact on the overall heat transfer, and can therefore be neglected from equation 2.17, and instead be accounted for by a small reduction to the convective heat coefficient (Austegard 1997). Assuming the heat fluxes only are functions of the surface temperature within a time step, the following expressions are obtained:

Using Fourier's law (Moran and Shapiro 2004) for a temperature gradient in the x-direction the conduction heat transfer becomes:

$$\dot{Q}_{Conduction} = A_{surface} k \frac{T_{solid} - T_{surface}}{x_{solid} - x_{surface}} = f\left(T_{surface}\right) \qquad (2.18)$$

Here T_{solid} from the previous time step is used in the explicit method, while T_{solid} from the previous iteration in the implicit method. Similar procedure also applies for the rest of the terms.

The convective heat transfer becomes (Incropera and DeWitt 2002):

$$\dot{Q}_{Convection} = A_{surface} \overline{h} \left(T_{\infty} - T_{surface}\right) = f(T_{surface}) \qquad (2.19)$$

The radiation heat transfer becomes (Incropera and DeWitt 2002):

$$\dot{Q}_{radiation,\ in} = A_{surface} \varepsilon_{surface} \sigma \varepsilon_{surrounding} T^4_{surrounding} = constant \qquad (2.20)$$

$$\dot{Q}_{Radiation,\ out} = A_{surface} \sigma \varepsilon_{surface} T^4_{surface} = f(T^4_{surface}) \qquad (2.21)$$

From these terms the surface temperature can be found iteratively. When the surface temperature is lower than the ablation temperature, the heat is transferred as normal. But when the ablation temperature is reached, the temperature is assumed to be fixed, and the remaining energy in the energy balance is used in the ablation process. Since the temperature at the surface is fixed, the only unknown in equation 2.17 is the term $\dot{m}_r$. The recession rate ($\dot{r}$) is then found from solving this equation and the assumption:

$$\dot{m}_r = \rho_r A \left(\frac{\partial r}{\partial t} \right) \qquad (2.22)$$

Where ρ_r is the char density and A the surface area. How the recession rate is handled numerically is explained in chapter 3.2.2.

Chapter 3: Program for simulating charring ablation

In this chapter, a presentation of the numerical aspects of solving pyrolysis and charring ablation in G2DHeat is provided. This includes governing equations, boundary conditions, a solution routine and a discussion of changes made.

3.1 Conservation of energy for inner cell volumes

The conservation of energy for a control volume is by Incropera and DeWitt (2002) stated as follows:

> "The amount of thermal and mechanical energy that enters a control volume, plus the amount of thermal energy generated within the control volume, minus the amount of thermal and mechanical energy that leaves the control volume must equal the increase in the amount of energy stored in the control volume".

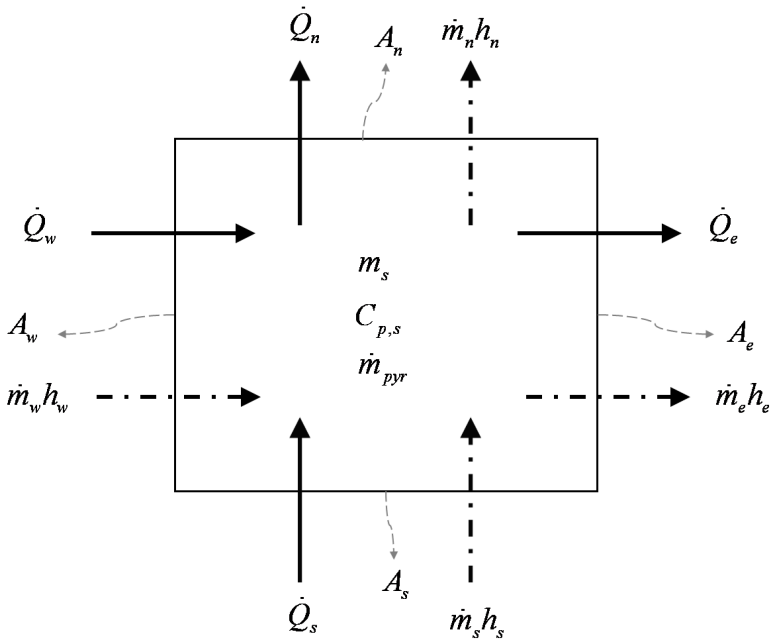


Figure 3.1: Shows the conservation of energy for inner cell volumes

One should consider the control volume shown in figure 3.1 and the same notation on the cell faces and cell volumes as in figure 1.1 when observing the derivations in this chapter.

The following simplifying assumptions are made (Austegard 1997):

- Thermal equilibrium exists between the solid material and the decomposition gases. The out-flowing gases have the same temperature as the surrounding material.
- No gases are accumulated in the cell volumes.
- Decomposition gases are non-reactive.
- The pressure throughout the ablative material is constant.

The energy equation on differential form can be expressed as (Rian 2003):

$$\frac{\partial \rho \overline{u}}{\partial t}+\frac{\partial \rho hu}{\partial i}+\frac{\partial \rho hv}{\partial j}=\frac{\partial}{\partial i}\left(k\frac{\partial T}{\partial i}\right)+\frac{\partial}{\partial j}\left(k\frac{\partial T}{\partial j}\right)+\overline{S} \qquad (3.1)$$

Where the first term represents the amount of energy stored in the control volume, and the remaining terms the amount of energy entering or leaving the control volume.

The equation of mass transfer on differential form can be expressed as (Rian 2003):

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial i}+\frac{\partial \rho v}{\partial j}=0 \qquad (3.2)$$

To numerically solve equation 3.1, it is discretised using the finite volume method. Here an integration of energy contributions over a control volume is performed. To calculate the temperature gradients at the control volume faces, an approximate distribution of properties between neighbouring cell volumes is used. The approximation can for some situations cause unstable and oscillating solutions (Austegard 1997). To prevent this, the upwind differencing scheme is used on the diffusive terms. This takes into account the flow direction of pyrolysis gas when determining the temperature at the cell face. The entire discretisation process is shown in appendix C.

The governing equation solved by G2DHeat becomes:

$$\underbrace{\left(\frac{\rho C_{v,s}\Delta V}{\Delta t}+a_e+a_w+a_n+a_s-S_p\right)}_{a_p}T_p=\underbrace{\left(\frac{A_e}{R_{I,e}}+\max\left(0,-C_{p,g}\dot{m}_e\right)\right)}_{a_e}T_E+\underbrace{\left(\frac{A_w}{R_{I,w}}+\max\left(0,C_{p,g}\dot{m}_w\right)\right)}_{a_w}T_W$$
$$+\underbrace{\left(\frac{A_n}{R_{J,n}}+\max\left(0,-C_{p,g}\dot{m}_n\right)\right)}_{a_n}T_N+\underbrace{\left(\frac{A_s}{R_{J,s}}+\max\left(0,C_{p,g}\dot{m}_s\right)\right)}_{a_s}T_S+\underbrace{\frac{\rho C_{v,s}\Delta V}{\Delta t}}_{a_p^0}T_p^0+\dot{m}_{pyr}\Delta h_{pyr}+S_u \qquad (3.3)$$

The solution process for equation 3.3 is explained in chapter 3.3.

3.2 Boundary conditions for ablative materials

In the G2DHeat program there are several boundary conditions available for both decomposing and non-decomposing materials (Backup materials) (Riise 2008). For materials that decompose, some additional conditions are available, to include the physics presented in chapter 2.

The program allows the following boundary conditions for the ablative material to be specified:

- Melting ablation with specified recession rate and surface temperature.
- Internal decomposition with specified heat flux and mechanical erosion.
- Internal decomposition with specified heat flux and no recession.

These options are somewhat limited in terms of ablation, since a specification of the recession rate is required. For option 1 the mass is removed at a known rate to sustain the temperature at the border. The boundary in Option 2 is subjected to erosion which simply removes pieces of material from the surface. In option 3, recession of the material is absent, but mass losses due to out-flowing pyrolysis gas are present.

The grid mesh and boundary conditions remains fixed throughout the simulation. The heat flux and the constant surface temperature are handled in the same way as in chapter 1.1.4. To cope with the movement of the decomposition- and the erosion front, these are calculated explicitly. An illustration is shown in figure 3.2.

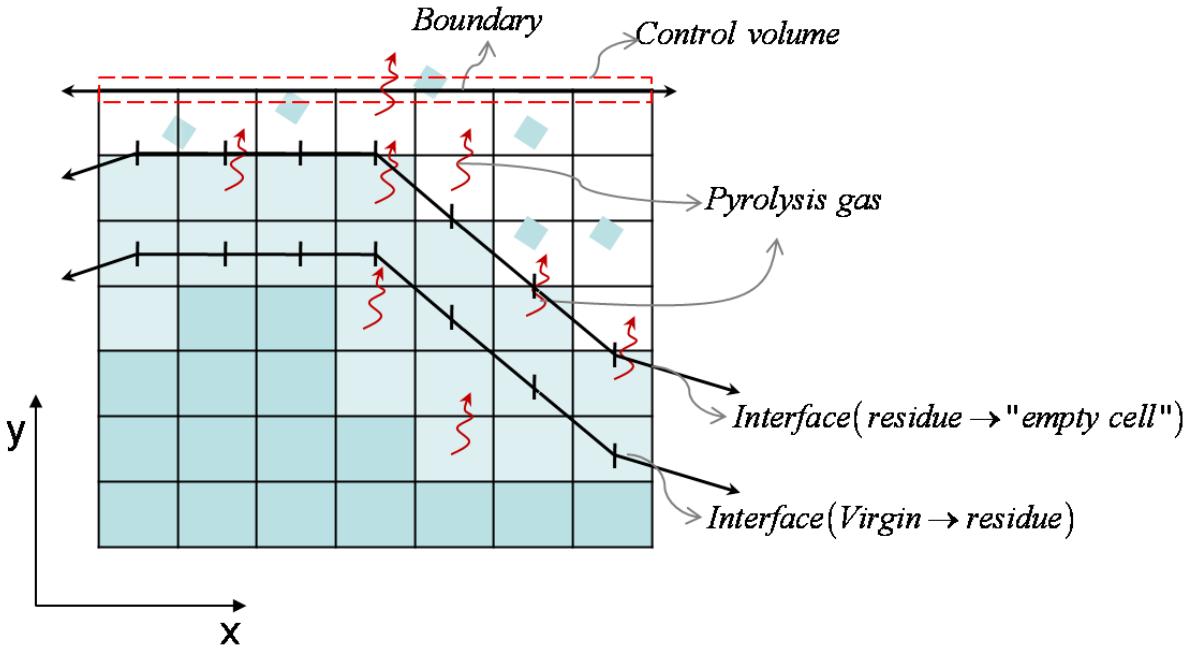


Figure 3.2: Interface definitions in the program

To model the empty cell volumes, a very high value for the thermal conductivity and a low value for the specific heat capacity are used.

3.2.1 The virgin to residue interface

The motive for finding the interfaces, especially the interface between virgin and residue, is so that directional vectors for the pyrolysis gas and the mechanical erosion are found.

Identification parameters (ID) are used to save computational time and determine the current state of a cell volume. Initially, these are specified with a positive or negative value to separate cell volumes with ablative materials from cell volumes with backup materials. Events such as the decomposition and the erosion are only calculated for ablative cell volumes. Hence, less computation time is needed for the simulation. If a cell volume of the ablative material decomposes and becomes residue, the identification parameter is increased by 1 in this cell volume. To initiate the interface, the id is set to 3 for cell volumes at the original boundary which is specified by the user. In short terms:

$$\begin{aligned} &id < 0 \rightarrow Backup\ material \\ &id > 0 \rightarrow Ablative\ material \\ &1 = Virgin\ or\ decomposing\ material \\ &2 = Residue\ or\ eroded\ material \\ &3 = Material\ at\ the\ specified\ boundary \end{aligned} \qquad (3.4)$$

The different ablative identification parameters are used to determine the virgin to residue interface shown in figure 3.2. The cell volume centres closest to the decomposition zone are found by using a geometrical routine. For every cell volume with id=1, the directional vector from the centre of the cell volume points towards the closest part of the interface that is composed of these cell volume centres.

To decide whether or not a cell volume is char or virgin, the fraction parameter(x) in chapter 2.3 is used. The standard decision value is set to 0.02, but can be adjusted by the user in the input file.

$$\begin{aligned} &x \geq 0.02 \ \rightarrow \ virgin\ or\ decomposing\ material \\ &x < 0.02 \ \rightarrow \ residue \end{aligned} \qquad (3.5)$$

It is important to notice that this only makes adjustments to the ID of the cell volume, thereby still allowing decomposition reactions within the cell volume to finish. This is done so that the energy is conserved.

3.2.2 Mechanical erosion and recession rate

A routine for calculating the recession of material on the basis of known recession rates is created. The recession rates are specified as functions of the simulation time, and are given as depth eroded material per second ($m.s^{-1}$). It is assumed that only residue (char) material is removed as the surface recedes. This states that the decomposition front must recede faster or equal to the erosion front.

Initially, by using the direction vector in the cell volume, the routine calculates the length (l) to where erosion is started. This is shown in figure 3.3. Then, from this length and the known recession rates, the time (t) until erosion of the cell volume starts is calculated. The time to end of erosion is found in a similar way.

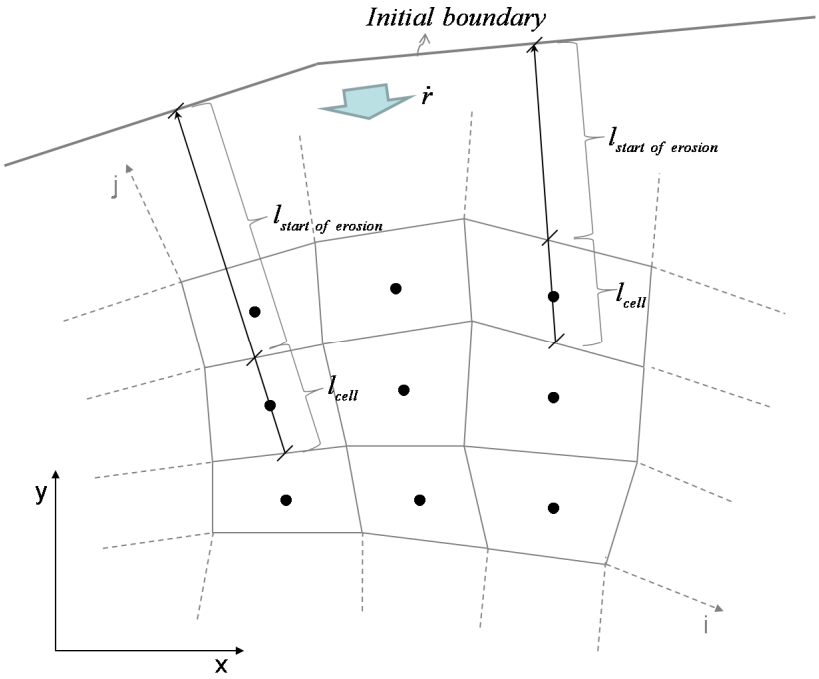


Figure 3.3: Shows the length which is used to calculate the erosion rate in the cell volume

$$t_{start\_erosion} = \frac{l_{start\ of\ erosion}}{\bar{\dot{r}}_{start\ of\ erosion}} \qquad (3.6)$$

Here the term $\bar{\dot{r}}_{start\ of\ erosion}$ is the mean recession rate for the present time interval of the $l_{start\ of\ erosion}$.

To cope with simulation time steps of different values, the recession rate through the cell volume is assumed to be constant. When the current time of the simulation reaches the time "start of erosion" ($t_{current} \geq t_{start\_erosion}$), the recession rate of the cell volume becomes:

$$\bar{\dot{r}}_{cell} = \frac{l_{cell}}{t_{end\ of\ erosion} - t_{current}} \qquad (3.7)$$

From this and the assumption of constant mass removal through the cell volume, the rate of density changes due to the recession of material is found:

$$\left(\frac{\partial \bar{\rho}}{\partial t}\right)_{cell} = -\frac{\bar{\dot{r}}_{cell}}{l_{cell}}\bar{\rho} \qquad (3.8)$$

Here $\bar{\rho}$ is the current density at "start of erosion" in the cell volume, and as earlier, it is assumed to be equal to the residue density.

In chapter 2.3 the material properties of a decomposing material is stated to be linearly dependent on the virgin and residual property. Similar linear approximation for the material properties of eroding cell volumes is assumed, only that the current property of the cell volume at "start of erosion" and the property value for emulating empty cell volumes are used instead.

Mechanical erosion of backup materials could be performed in the same manner. The only requirement is that the materials are specified as an ablative material together with a very high threshold temperature for the pyrolysis, which is never obtained by the material.

In chapter 2.5 a suggestion for finding the recession rate is outlined. The time to start of erosion in the cell volumes could be calculated similarly as in this chapter, only that it would be necessary to recalculate when there is a change in the recession rate.

3.3 Solution routine

In G2DHeat some additional computation events are necessary when ablative materials are present. For the program, each computational step is described by the main events:

- Internal decomposition of the ablative material (pyrolysis) and calculation of surface recession.
- Updating of direction vectors and iteratively solving of the continuity equation (mass transfer).
- Energy balance for the entire system including the surface.

From these events new values of densities, temperatures and pyrolysis gas production rates are found. In preparation of next computational step, specific heat capacities of solid and gas, thermal resistances and heats of pyrolysis are updated. The detail of the solution process is shown in figure 3.4, while the source code of the program is shown in appendix J.

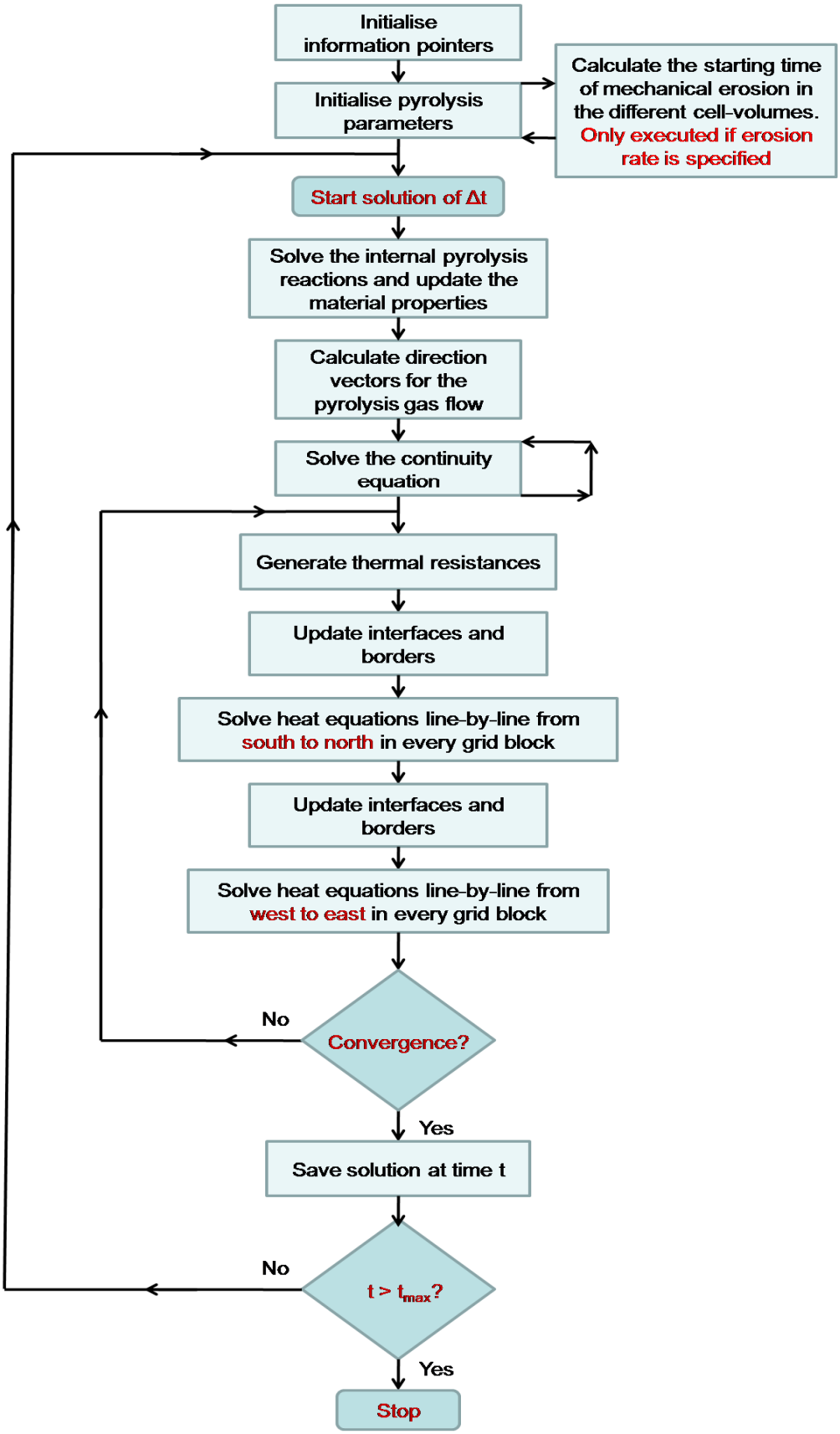


Figure 3.4: Flow chart for the solution routine

3.3.1 Numerically solving the pyrolysis

In chapter 2.2 decomposition reactions of ablative materials are explained. Numerically these events are computed explicit in time. Hence, the previous temperature within the cell volume is used in the kinetic model to calculate the amount of pyrolysis gas generated. This is performed in a reaction-by-reaction fashion. The sum of gas contributions become the total amount pyrolysis gas that is generated during a time step, and from this, the density of the cell volume is adjusted and the current gas production rate for use in the continuity equation is created.

The fraction parameter in equation 2.9 is then created from the new cell volume density. With this parameter, material properties are adjusted in accordance with the temperature, equation 2.10 and equation 2.11. The specific heat capacity of pyrolysis gas and the heat of pyrolysis are adjusted only by means of the current temperature. The same temperature is also used for updating properties for backup materials.

Next, the identification parameter for the cell volume is determined as described in chapter 3.2.1. If the cell volume is part of the interface between virgin and residue, it is inserted into a vector together with the rest of the cell volumes which also represent parts of the interface.

3.3.2 Solving the continuity equation using vectors

The pyrolysis gas in the ablative material is assumed to percolate in the direction of the vector calculated using the earlier described geometrical routine. Since cell volumes are not shaped according to the gas flow, the flow must be decomposed as shown in figure 3.5.

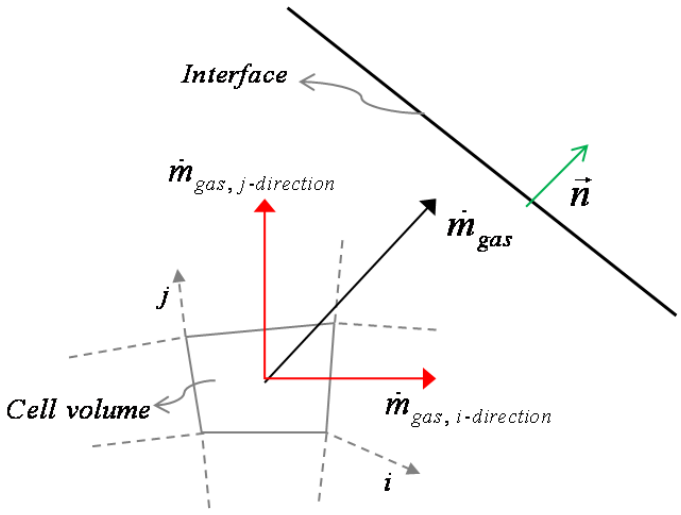


Figure 3.5: Shows the decomposition in i- and j-direction of the gas direction vector

To include the gas flow in the energy balance, it must be expressed in terms of the i- and j-coordinate. The angle between the direction vector of gas and the i-axis of the current cell volume is assumed to be a fair representation for finding the fraction parameters λ_i (i-direction) and λ_j (j-direction). This allows for the amount of out-flowing gas in the different directions to be found.

The mass transfer on differential form is shown in equation 3.2. By using the finite volume method with backward differencing in time and central differencing in space the following equation is obtained:

$$\dot{m}_{pyr} + (\rho uA)_e - (\rho uA)_w + (\rho vA)_n - (\rho vA)_s = 0 \qquad (3.9)$$

Here $\dot{m}_{pyr}$ is the pyrolysis gas leaving the control volume (Note that this is negative). To deal with vector directions instead of pressure or other controlling means, some corrections to equation 3.9 must be performed. Assuming that no gas is accumulated within the control volume, the amount of gas flowing into the control volume, plus the amount of gas generated within the control volume, must equal the gas flowing out of the control volume. This can be expressed as:

$$\dot{m}_{pyr} + \underbrace{x_1\dot{m}_e - x_2\dot{m}_w + x_3\dot{m}_n - x_4\dot{m}_s}_{-\dot{m}_{in}} + \dot{m}_{out} = 0 \qquad (3.10)$$

x_1, x_2, x_3 and x_4 are controlled by the vector direction of gas in the cell volume. Values of these for different angles are shown in table 3.1. The amount of gas leaving the control volume in i- and j-direction can be expressed by:

$$\dot{m}_{out} = \underbrace{\dot{m}_{out}\lambda_i}_{\dot{m}_i} + \underbrace{\dot{m}_{out}\lambda_j}_{\dot{m}_j} \qquad (3.11)$$

And

$$|\lambda_i| + |\lambda_j| = 1 \qquad (3.12)$$

Using previous expressions, the mass transport for a control volume can be calculated by:

$$\begin{aligned}
\dot{m}_{out} &= x_2\dot{m}_w - x_1\dot{m}_e + x_4\dot{m}_s - x_3\dot{m}_n - \dot{m}_{pyr} \\
\dot{m}_e &= x_1\dot{m}_e + x_2\lambda_i\dot{m}_{out} \\
\dot{m}_w &= x_2\dot{m}_w + x_1\lambda_i\dot{m}_{out} \\
\dot{m}_n &= x_3\dot{m}_n + x_4\lambda_j\dot{m}_{out} \\
\dot{m}_s &= x_4\dot{m}_s + x_3\lambda_j\dot{m}_{out}
\end{aligned} \qquad (3.13)$$

This solution sequence is performed in every cell volume of the ablative material, and is calculated iteratively until convergence.

Table 3.1: Direction variables for the continuity equation

Angle	X1	X2	X3	X4	λ i-direction	λ j-direction
0 (360)	0	1	1	1	1	0
90	1	1	0	1	0	1
180	1	0	1	1	1	0
270	1	1	1	0	0	1
0< angle <90	0	1	0	1	(+)	(+)
90< angle <180	1	0	0	1	(-)	(+)
180< angle <270	1	0	1	0	(-)	(-)
270< angle <360	0	1	1	0	(+)	(-)

The subroutines executed by G2DHeat are shown in appendix E.

3.4 New input modifications

For handling the ablative materials in G2DHeat, some additional input specifications have been created.

3.4.1 Material properties of decomposing materials

For the ablative materials that undergo changes due to pyrolysis reactions, an additional input file describing the different reactions must be supplied. As functions of the temperature, the properties of gas and residue product from the pyrolysis reactions are included in the material properties file. For simplicity, the heats of pyrolysis are also included in this file.

In the input file it is important that the user specifies the decomposing materials prior to the backup materials. The files with pyrolysis kinetics must be specified in the same order as its connected material properties file. Example:

```
DEFMATERIAL          -> Indicate the start of defining material properties in the input file
2                    -> Number of data files containing a material's properties
SIPH.b               -> Filename with id=1
ALU.b                -> Filename with id=2
1                    -> Number of materials that are decomposing by pyrolysis
SIPHpyr.b            -> The file containing pyrolysis data's that are attached with material id=1
3                    -> Number of material area's that are specified
1 2 1 81 1 11        -> Material area which is defined as:
1 3 1 11 1 91            <Filename id><grid block><start I-><end I-><start J-><end J-coordinate>
2 1 1 81 1 81
```

Here aluminium (ALU) as backup material and silica phenolic (SiPh) as ablative material are applied. For ablative materials, the program automatically calculates the pyrolysis reactions without further specifications.

3.4.2 Adjustments of the virgin to residue interface

In chapter 3.2.1 the use of a decision variable to find the virgin to residue interface is explained. In the program it is possible to adjust this value by the input:

```
PYROLYSIS            -> Indicate start of defining the decision value
 0.1                 -> The new decision value
```

The default value of this decision parameter is 0.02, but chapter 3.4.4 describes various situations where it is desirable to change the parameter.

3.4.3 Mechanical erosion

The specification of recession rate or mechanical erosion is performed by the input sequence:

```
M_EROSION                 -> Indicate start of specifying erosion/recession
5                         -> Number of erosion/recession rates
1.0     0.0               -> Recession/erosion rates as function of time are specified by:
2.0     0.000713540294       <End time for current recession>< recession rate [m.s-1]>
3.0     0.000510695946
4.0     0.000480400863
100.0   0.0
```

The specification is performed only once in the input file and applies to the entire simulation time.

3.4.4 Printouts

To visualize the results at certain times in the simulation, the visualization tool Tecplot is used. For saving densities, temperatures and pyrolysis gas production rates to file in the program, the following input specification is used:

```
SAVERHO                   ->Saves densities, temperatures and gas production rates to file.
plot_1.b                  -> Filename
```

This file must first be converted by using another tool (g2dnoden) before Tecplot is able to read the data.

3.5 Discussion

There has been a great deal of assumptions in the chapter. Mostly, these have been based on other literature, but also some of them have come naturally from the methods selected and used. It is important to have in mind that the numerical approximations will only give results which are in direct reflection of the theoretical models that are used to describe the physics. Even so, there are possibilities for improving the results by using better numerical approximations.

3.5.1 Energy considerations

The initial assumptions made for the energy balance in chapter 3.1, are only to some extent true. When the ablative material decomposes there is likely that cracks will form. Achieving thermal equilibrium between the solids and pyrolysis gas in a cell volume can then be a problem, since the gas flow possibly will tend to flow in the cracks where it is less resistance.

In some cases the out-flowing gases accumulate within parts of the ablative material (Austegard 1997). If this happens, the temperature in parts of the material could increase because the heat sink effects caused by pyrolysis gases flowing through the material cease to exist in these parts. If this occurs close to the surface, a pressure build-up here could cause parts of the material to be blown off.

Since different ablative materials produce different reaction products and gases, it is likely that these gases are still reactive when leaving the cell volume. How much these remaining reactions influence the calculations is unknown. But it is considered to be a fair assumption since the main reactions of creating the gases are finished.

3.5.2 Limitations on Time Step Size

For calculating the temperatures in G2DHeat, an implicit method is used. Generally, the implicit method is unconditionally stable regardless of the time step size (Versteeg and Malalasekera 1995). However, in G2DHeat the time step must be reduced when ablative materials are included in the simulation. This is because of the explicit solution of decomposition reactions and the continuity equation. Currently there are no available routines in G2DHeat for adjusting the time step size. By using common sense and experience, the program user should be able to perform this initially in the input file.

3.5.3 Numerical techniques

The erosion/recession method for fixed grids that is outlined in chapter 3.2.2, has some weaknesses. In figure 3.6 the recession of the material is shown at three different times in the simulation. For cell volume “A”, the ending time occurs at $t+\Delta t_2$. Physically, there is still material left in the cell volume when it is numerically assumed to be empty. The recession starts at $t+\Delta t_1$ for cell volume “B”. Physically, this is also inaccurate. Since the cell volume at this time is already reduced.

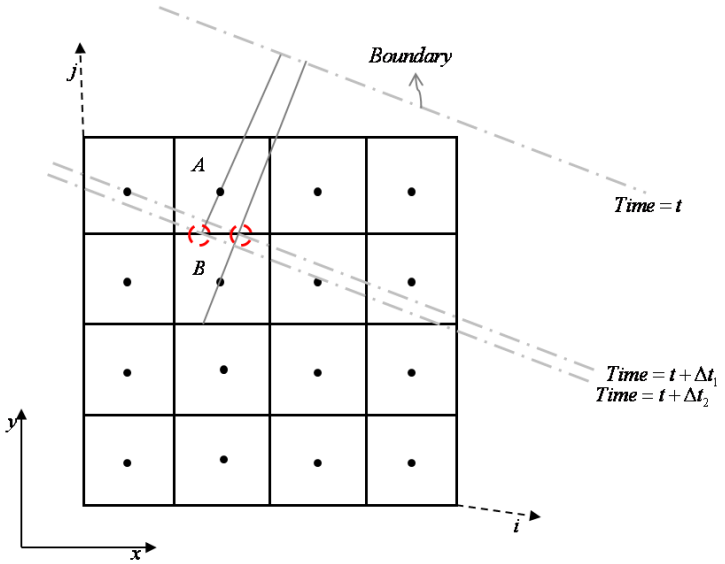


Figure 3.6: Assumption made for the mechanical erosion

Somewhat, this error can be reduced, since the grid in G2DHeat can form itself according to the geometry.

When calculating the flow of pyrolysis gas in the continuity equation, a problem with pyrolysis gas flowing in wrong directions can occur. To illustrate this, a rectangular geometry consisting of aluminium and silica phenolic is exposed to heat flux on the left and bottom side, while the top and right side is insulated. This is shown in figure 3.7 with an indication of how the heat distributes itself in the geometry. In this illustration case, the heat flux is set relatively high to achieve decomposition and gas generation within the silica phenolic. Since the aluminium conduct heat much better than the silica phenoic (see appendix F for parameter values), a decomposition front in the silica phenolic is moving from the border on the left side inwards in the material against the aluminium. For a better visualisation, the decomposition is present in all parts of the silica phenolic except the dark blue area shown in figure 3.7.

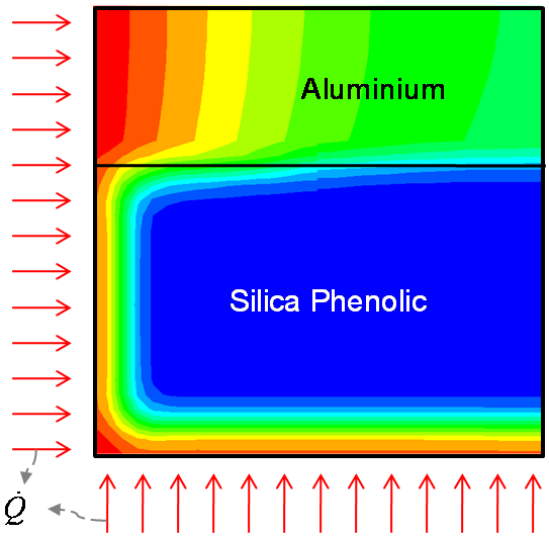


Figure 3.7: Test case used to illustrate the problem with direction vectors

When using the standard decision value of 0.02 for the interface in the simulation, some gas is flowing in the wrong directions, as seen in figure 3.8. The decision value for the interface is explained in chapter 3.2.1. Physically, this gas is supposed to percolate through the partially decomposed material towards the surface on the left side of the geometry. It is the direction vectors, which are geometrically calculated, that cause gas to flow in the wrong directions. Because the routine calculating these vectors only accounts for the position of the virgin to residue interface, and not the pressure or the porosity of the material. Therefore the shortest distance to the interfaced is used. Since the interface is defined by cell volumes with id greater than 1, the decision value can be increased to allow a more suitable movement of the interface.

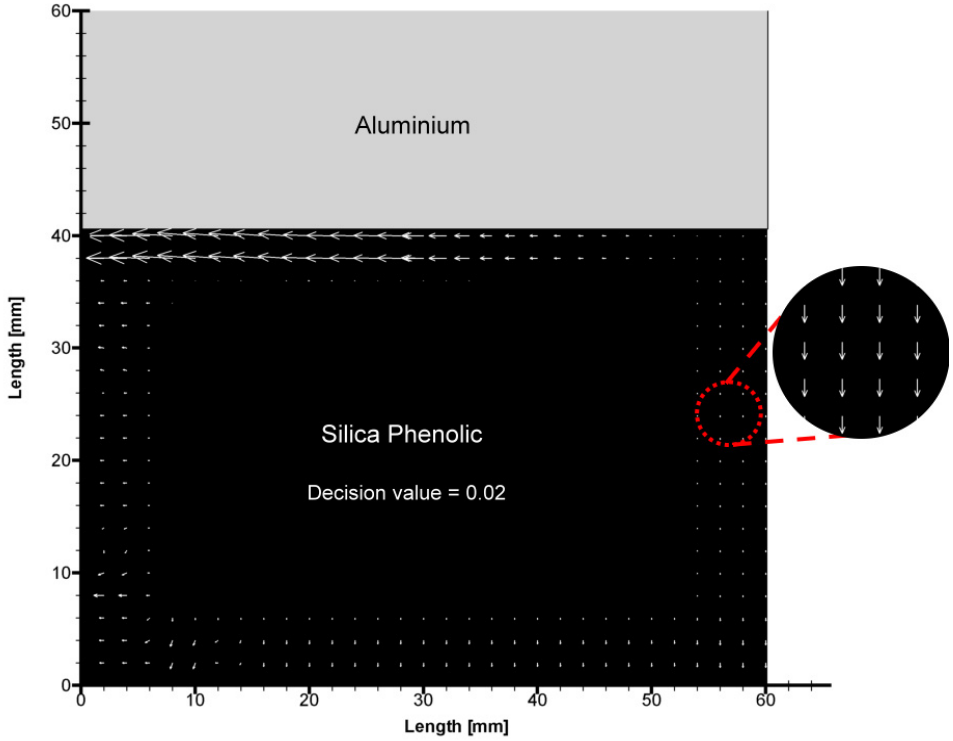


Figure 3.8: Error caused by the vector routine

By increasing the decision value to 0.1, the new interface is defined by cell volumes which have 10 percent or less material left which can decompose. This adjustment to the decision value, results in correct directions for the gas flow, as shown in figure 3.9.

To fully avoid this type of problems, a better method for calculating direction vectors or a method for including calculations of pressure and velocity fields in the material should be considered.

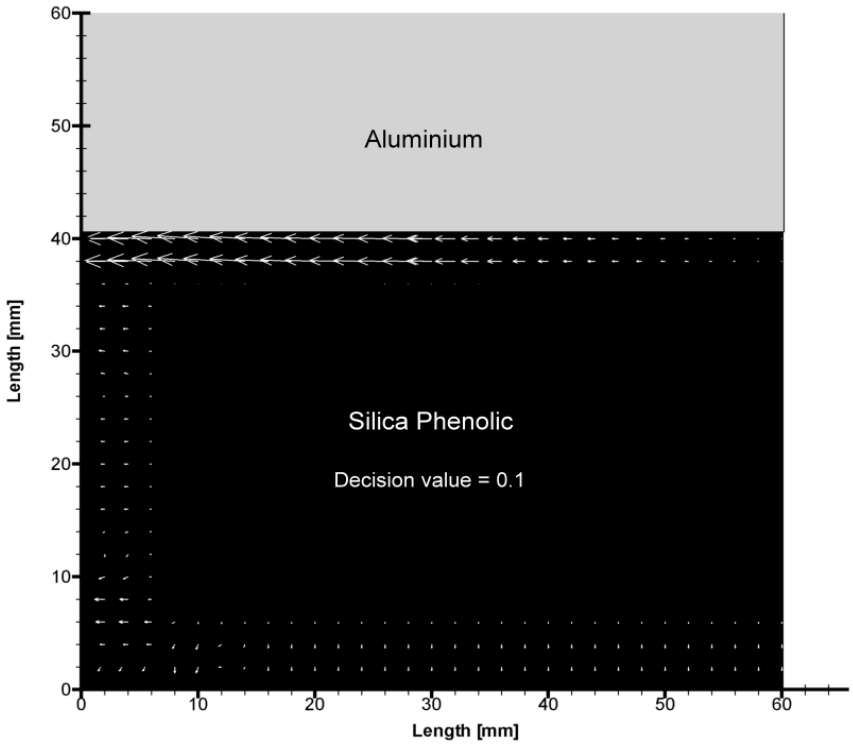


Figure 3.9: Pyrolysis gas flow with corrected vector directions

Chapter 4: Test simulation

In order to validate the program, this chapter provides a comparison of a test simulation performed in two commercial programs (CMA3 and ASTHMA) in addition to G2DHeat. First, a short introduction to what separates G2DHeat from the two other programs is presented. Second, a description of the chosen model geometry is given, including the necessary input preparations and boundary conditions used in the simulations. Finally, the results from the simulations are discussed.

4.1 Simulation programs for decomposing materials

To ensure that G2DHeat is valid for engineering applications, it has been performed code-to-code comparison with the commercial programs CMA3 and ASTHMA. The implementation of models, the numerical approximations and other assumptions are verified to some extent by these comparisons. However, a proper validation of the physical phenomenon is only accomplished by comparing the simulation results with experimental data. This is performed in chapter 5.

Before comparisons of the simulation results can be made, a fundamental question must be put forth:

- What separates G2DHeat from CMA3 and ASTHMA?

In answer to the question, both the numerical aspects and the physical aspects are handled a bit differently by the programs, and therefore detailed comments are necessary. The numerical aspects of interest are summarized and shown in table 4.1, while the physical aspects of interest are shown in table 4.2.

Table 4.1: Numerical aspects of interest for G2DHeat, CMA3 and ASTHMA

Aspects	G2DHeat	CMA3	ASTHMA
Grid mesh	-Two-dimensional grid mesh. -Quadrilateral cell volumes not necessarily orthogonal. -Multi-block grid configuration. -2D and Axi-symmetrical options. -Nodal scheme with nodes in the centre of the cells. -Eulerian method for moving grid. -Maximum number of grid nodes is currently 35000, but can be increased.	-One-dimensional grid mesh. -Employ sub-mesh with nodelets (or sub-nodes) within each cell element. -Simple representation of plate-, sylinder-, sphere- or tube like geometries. -Lagrangian method for moving grid. -Nodal scheme with nodes in the centre of the cells, and one at the cell face on the surface. -Maximum number of grid nodes is 100.	-Two-dimensional grid mesh. -Quadrilateral cell volumes not necessarily orthogonal. -Axi-symmetrical option. -Two nodal schemes, nodes at the back-face or in the centre of the cells. -Lagrangian method for moving grid in a column-by-column fashion. -Maximum number of grid nodes is 600.
Solution routines	-Implicit solution of the internal energy balance. -Explicit linkage to decomposition events and the mass conservation. -Implicit solution of the surface energy balance. Note: Don't include calculation of recession rate. -Finite volume type solution procedure.	-Implicit solution of the internal energy balance. - Explicit linkage to the decomposition events and the mass conservation. -Implicit solution of the surface energy balance, except the recession rate which is calculated explicitly. -Finite difference type solution procedure.	-Alternating-direction implicit, classical explicit, column-implicit/row-explicit or column-explicit/row-implicit solution of the internal energy balance. -Explicit or mixture of implicit-explicit linkage to the decomposition events and the mass conservation. -Explicit or implicit solution to the surface energy balance. -Finite difference type solution procedure.

Table 4.2: Physical aspects of interest for G2DHeat, CMA3 and ASTHMA

Aspects	G2DHeat	CMA3	ASTHMA
Internal events	-Kinetic model with series of independent parallel reactions of the material. -Maximum number of reactions is 20 for each material. -Allows different threshold temperature for the reactions.	-Kinetic model with series of independent parallel reactions of the material. -Maximum number of reactions is three for each material. -Allows different threshold temperature for the reactions.	-Kinetic model with series of independent parallel reactions of the material. -Maximum number of reactions is five for each material. -Fixed threshold temperature for the reactions (333 K). -Radiation within material gaps and contact thermal resistance.
Boundary events	-Melting ablation with specified recession rate and surface temperature. -Include radiation, convection and other specified heat fluxes. - Similar heat fluxes with specified mechanical erosion. -Adjusts/corrects the convective heat coefficient on basis of an isentropic flow at the surface. -Include calculation of recovery temperature.	-Melting ablation with specified recession rate and surface temperature. -Include radiation, convection and other specified heat fluxes. -General convective heating and thermo-chemical erosion. -Adjusts/corrects the convective heat coefficient for radius changes, char swell and transpiration effects at the surface (blowing). -Allows cracking or fissuring of the surface char layer.	-Melting ablation with specified recession rate and surface temperature. -Include radiation, convection and other specified heat fluxes. -General convective heating and thermo-chemical erosion or mechanical erosion. Adjusts/corrects the convective heat coefficient for transpiration effects at the surface (blowing).

4.2 Problem description

This example treats one-dimensional heat transfer in the radial direction of a simple model geometry subjected to heat causing melting ablation and internal decomposition of the ablative material. The model is taken as a sectional cut of a larger cylinder geometry that is similar to the blast pipe of a rocket motor. The model consists of silica phenolic as ablative material and aluminium as backup material. In the radial direction these measure 6,35mm and 1,8mm, respectively. Since the geometry is a long cylinder, the heat transfer in the axial direction is neglected. Hence, the left side and right side of the model geometry is assumed to be insulated. The schematic of the model geometry is shown in figure 4.1.

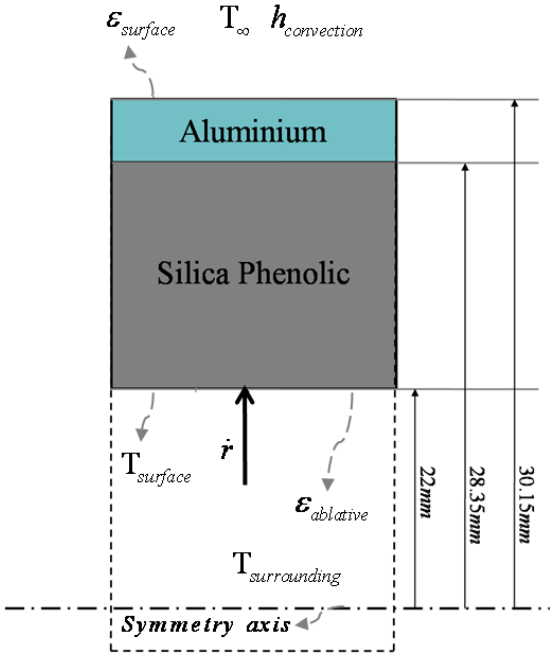


Figure 4.1: Schematic of geometry and boundary conditions

The programs have only one boundary condition in common that accounts for ablation reactions at the surface. This requires a recession rate and ablation temperature to be specified by the user. Physically, this means that the amount of material being sacrificed to sustain the fixed ablation temperature at the surface during thermal stress is known. In this simulation, the model is first exposed to heating/ ablation of the insulation side for a period of 5 seconds, and then cooled for 95 seconds. The boundary conditions are outlined in chapter 4.2.2.

4.2.1 Input preparations for the simulation

In order to simulate the example problem, large amounts of input to the programs are required. A great deal of this input has been found by examining old input files used in CMA3 and ASTHMA, and conversions and examples provided in ASTHMA 88/PC (1988). All data have been supplied by Nammo.

The properties of silica phenolic are presented in chapter 2, while the properties for aluminium are given in appendix F. In G2DHeat these are stored in external files collected by the program from filenames specified in the input file. In ASTHMA and CMA3 they are specified directly in the input file of the program.

Grid geometry must also be made for each of the programs individually. For the grid in ASTHMA, the corner points of the cell volumes are specified in the input file. In total, 17 cell volumes in the radial direction and 3 in the axial direction are used for the representation of the model. In CMA3 a one-dimensional grid layout is used. For emulating the axi-symmetrical geometry, the cell volumes are created by multiplying the width of the cell with the relative area being proportional to the radius. In this grid, 18 cell nodes/volumes with 10 nodelets (sub-nodes/-volumes) are used. G2DHeat is using 37 cell volumes in the radial direction and 11 in the axial direction.

The estimate for the recession rate (the rate material is mechanically removed) is found using an empirical correlation with the pressure present on the outside of the insulation (SiPh)(Næss 1998b). Whether or not the pressure values and correlation are valid for this example is unknown. However, it is assumed to be a fair approximation since the same rates are employed in all the programs. Therefore the errors will be equally large. The recession rates used in this simulation are shown in figure 4.2.

The input files used by CMA3, ASTHMA and G2DHeat are shown in appendix G, H and I, respectively.

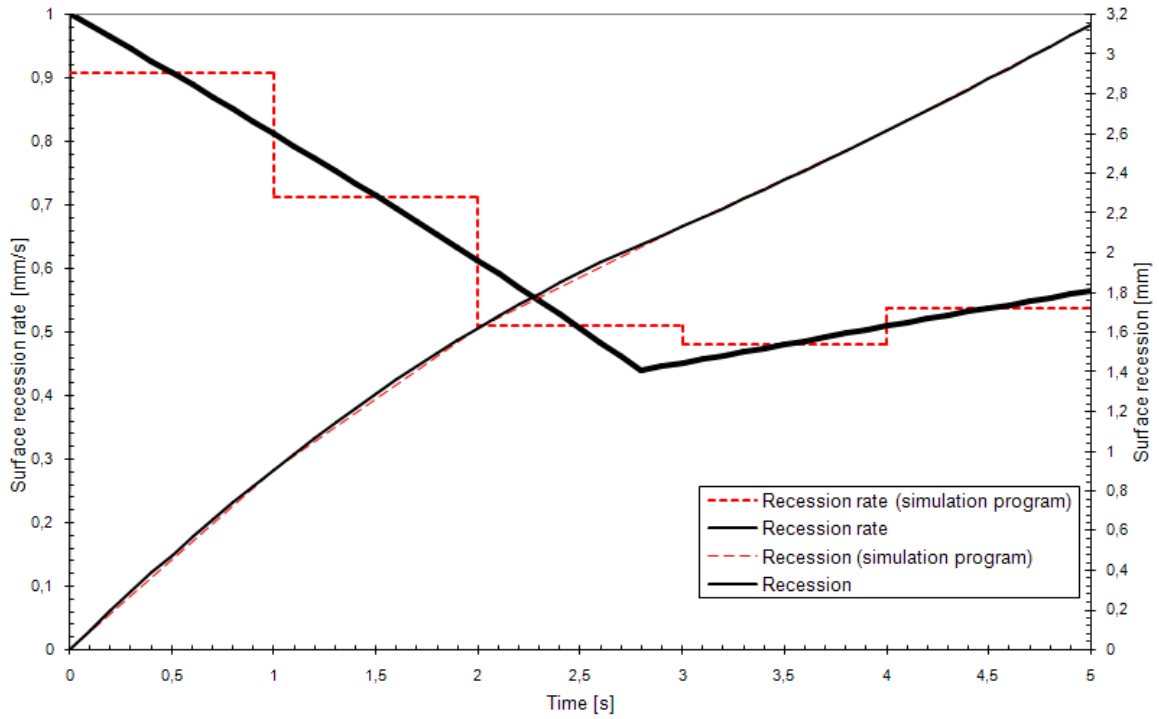


Figure 4.2: The recession rates used in the simulations

4.2.2 Boundary conditions

Initially, the model geometry has a uniform temperature of 299.15 K. For the entire simulation time, the boundary condition on the outer face of the aluminium is exposed to free convection and radiation to the surrounding air. The boundary condition on the inside of the model geometry is more complex due to the moving boundary. Assuming that the heat transfer within the model to be relatively low before the surface reaches the ablation temperature, the surface temperature is initially set to the ablation temperature in the programs. This temperature remains fixed during the ablation period of five seconds, before radiation cooling of the surface is enabled. The boundary conditions associated with this example, are shown as the simulation time proceeds in table 4.3.

Table 4.3: Boundary conditions as simulation time proceeds (Myklebust 2008)

Time [s]	Ablating surface (SiPh)	External surface (Aluminium)
0.0 – 5.0	$T_{surface} = 2473K$ $\dot{r} = f(Time)$	$T_{\infty} = 293.15K$ $h_{convection} = 26\ W/m^2K$ $\varepsilon_{surface} = 0.05$
5.0 – 100.0	$\varepsilon_{ablative} = 0.6$ $T_{surrounding} = 473.15K$	$T_{\infty} = 293.15K$ $h_{convection} = 26\ W/m^2K$ $\varepsilon_{surface} = 0.05$

4.3 Results

The accuracy of the program is first assessed by comparing the predicted temperatures from the programs. Measurements are taken at the outer surface of the aluminium, and the temperature histories are shown in figure 4.3.

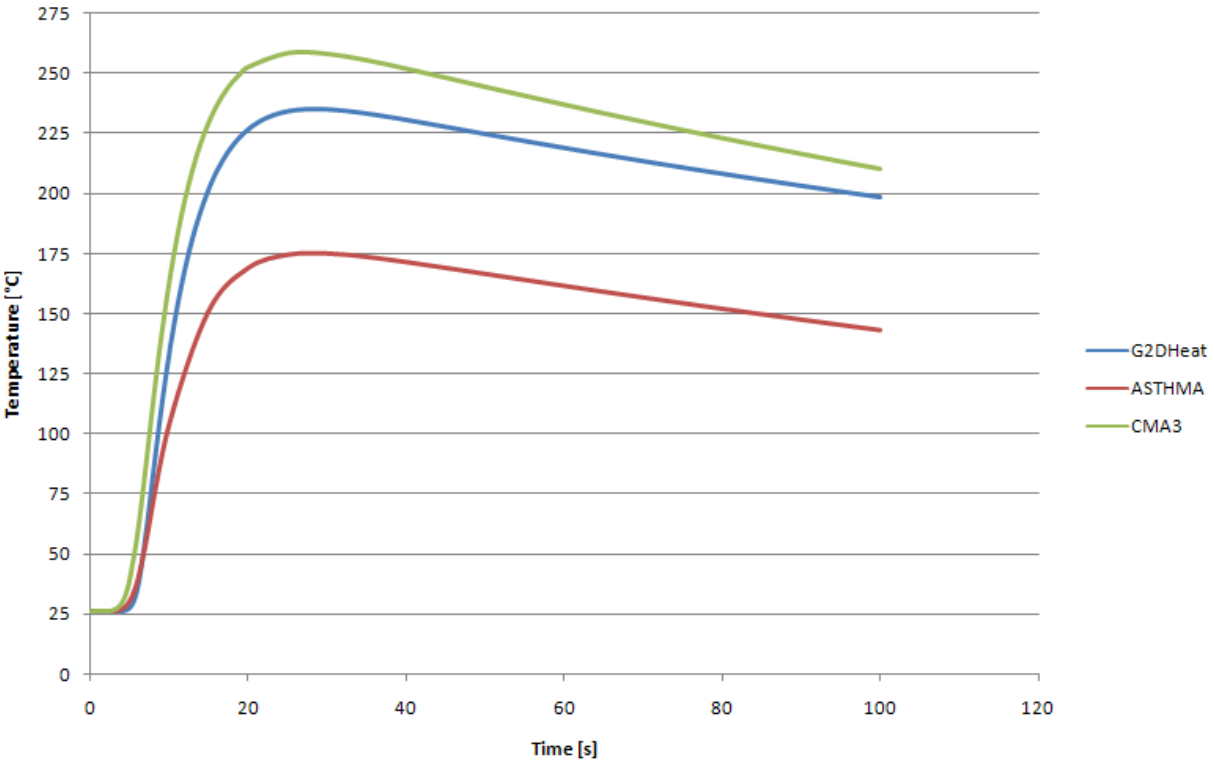


Figure 4.3: Comparison of temperature history at the outer surface

As can be seen from figure 4.3, the results from ASTHMA differ greatly from those obtained using G2DHeat and CMA3. This big difference suggests that something is wrong with the program, the specification of input and/or the way the program performs its calculations. In ASTHMA 88/PC (1988), it is commented on discovered and corrected coding errors. This gives an indication of a program not fully tested. When trying to find a suitable case to be executed in all programs, without returning fatal program errors, ASTHMA limited the options. This is because even small changes to the emissivity caused ASTHMAs surface solution routine to fail. Some uncertainties around the input specifications therefore exist. Especially, this was a problem when combining the time dependent and temperature dependent parts of the boundary conditions, to achieve analogous input to the programs. Because of these arguments, and the fact that the results from G2DHeat and CMA3 are relatively close, CMA3 is considered to be more trustworthy than ASTHMA in this comparison.

Since CMA3 is considered most reliable, G2DHeat produces conflicting results as shown in figure 4.3. Provided that the same amount of heat has entered the system, it seems to be a bigger heat sink in G2DHeat than in CMA3. This could also be the case for ASTHMA, but the heat sink would be much larger compared to the heat sink in G2DHeat. In the figure it is seen that the profiles of the temperature curves are quite similar, except the fact that they are at different temperatures. Also seen from the figure, ASTHMA and G2DHeat start their initial temperature increases at a later point than CMA3. This supports the probability that a greater heat sink or a greater thermal resistance is present in G2DHeat and ASTHMA than in CMA3.

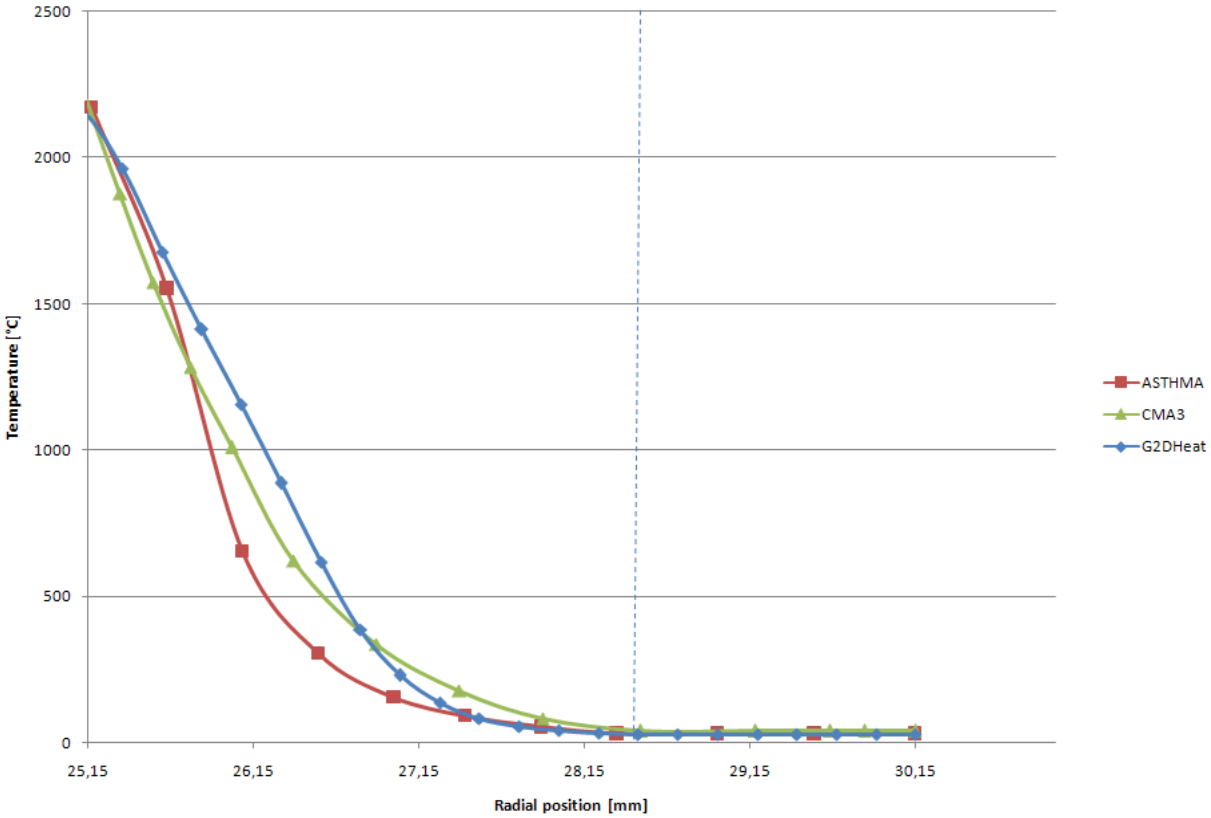


Figure 4.4: Comparison of temperature profile at 5 seconds

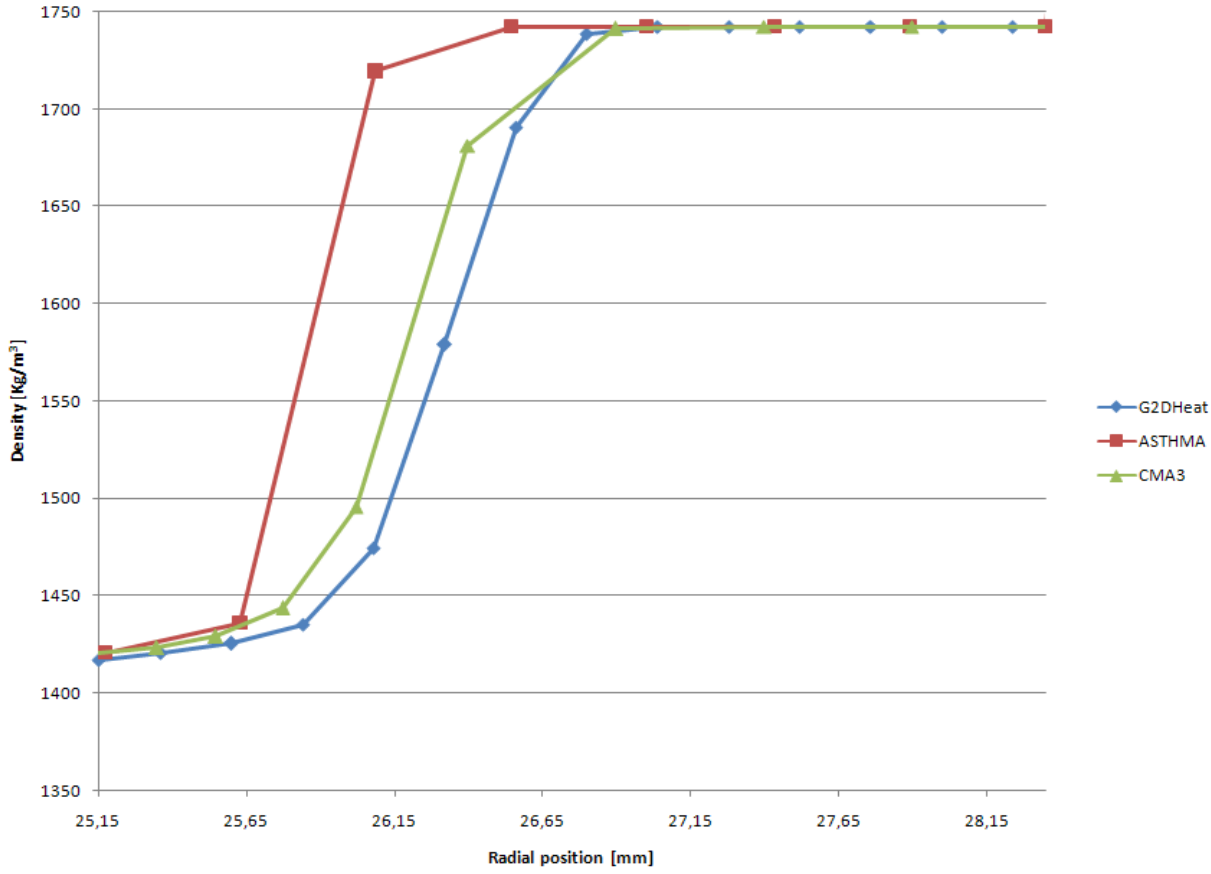


Figure 4.5: Comparison of density in the insulation at 5 seconds

To clarify what might cause differences in the results, the period after heat input (from t=5s) is considered. During this period the heat which has entered the system distributes itself in the model geometry, or is lost to the surrounding air by radiation and/or convection. The temperature distribution for the entire model and the density changes for the insulation (SiPh), at the end of the heat input period (t=5s), are used to illustrate some of the reasons why the differences in the results occur. Temperature distribution is shown in figure 4.4, while density changes are displayed in figure 4.5.

In the outer most part of the insulation, 25.15 mm to 25.65 mm, the simulated temperatures are relatively high, while the densities are low. As seen in figure 4.4, the temperatures simulated by G2DHeat are greater than the ones simulated by CMA3 and ASTHMA, but the densities are less, as displayed in figure 4.5. The density gradient is small in the radial direction in this part because the decomposition events stagnate, only leaving residue products (char) behind. Since decomposition events are controlling the rate at which mass is removed, and the mass loss increases as the temperature is rising, G2DHeats results are consistent with the results from CMA3. More detailed, a higher temperature is reached and more mass is removed, for this part of the insulation, in G2DHeats simulation than it is in CMA3s simulation. ASTHMAs simulation is inconsistent with CMA3s simulation, since a greater temperature is reached, while less mass is removed from this part of the insulation. The differences are small, but could be explained by a higher heat flux present in ASTHMA than in CMA3, because the mass loss is a function of heat input to the system as shown in figure 2.2. However, this is uncomprehending, due to the fact that the temperature at the surface remains fixed during the heat input period, hence the heat flux should initially be equal in all the programs.

In the part spanning from 25.65 mm to 26.8 mm, ASTHMAs temperature sinks rapidly within the insulation compared to the temperatures of CMA3 and G2DHeat. ASTHMAs density, on the other hand, is larger since less insulation has decomposed. It seems that in ASTHMAs simulation there is a larger heat barrier at the outer parts of the insulation, since less heat is allowed to enter the inner parts, which again prevents decomposition events. In this part of the insulation, the programs share the fact that all the densities are reduced from the insulations virgin state towards the residue (char) state. The difference in density becomes very eminent in this part, as shown in figure 4.5. A difference between CMA3 and G2DHeat is also observed, where more insulation is removed in G2DHeats simulation.

In the remaining part of the insulation the temperature simulated by CMA3 is the highest. At the same point the density of the insulation is close to the virgin state in all the simulations. This indicates that less thermal resistance is present in the outer parts of the insulation in CMA3 than in G2DHeat and ASTHMA. It is also important to notice that this allows more thermal energy to be stored, since the density is much larger.

The deviations in temperature at the outer surface of the aluminium simulated by G2DHeat and CMA3 are also shown in figure 4.3. These can be explained by the temperature and density distribution in the insulation, at the end of heat input (t=5s), for the different programs. G2DHeats temperature, which is higher than CMA3s in the outer parts of the insulation, is insufficient to heat the aluminium to the same temperature as that of CMA3. The total amount of energy used to heat aluminium is less for G2DHeat than for CMA3. This is because the density in the outer parts of the

insulation is lower in G2DHeats case, while temperature in CMA3s case is higher in the parts where the density is greater.

It is clear that there is a larger heat sink in G2DHeat than in CMA3. For some reason, more insulation material is decomposed to provide this larger heat sink. The cause of this could be; different handling of the boundary condition, numerical handling of decomposition events, errors in the program code and/or the usage of incorrect input values.

In G2DHeat the surface remains fixed and numerical manipulation is employed to model the receding surface as explained in chapter 3.2, while in CMA3 the grid follows the receding surface. The different ways the surface is handled could result in errors, causing a larger heat sink. It is unknown how big the errors can be or if they are present at all.

When the cell volume has started to recede in G2DHeat, it is assumed that decomposition events are insignificant in this cell volume, since the material is close to or already fully decomposed in this cell volume, and therefore neglected. It is not likely that this alone causes the entire error, because there is too little mass remaining in the cell volume that can decompose and result in a significant heat sink. However, the error can be reduced by selecting smaller cell volumes for the representation of the model geometry.

A great deal of modifications has been made to G2DHeat, and it is not unlikely that logical errors can exist in the program, since the program is not tested as much as CMA3. However, this error, if it exists, might give an additional contribution to the heat sink present in the insulation during decomposition events. More testing is necessary to discover such errors, because the FORTRAN compiler does not recognise these.

Unit conversion of material properties and reformulation of kinetic parameters due to different definitions of these in the programs, are also possible sources of error. A misinterpretation can easily occur since the documentation describing the conversions (ASTHMA 88/PC 1988) is divergent.

To support what has been stated as cause of differences in temperature, using simulation results at the end of heat input, simulation results from a later point in time (t = 10s) is considered. The temperature profile at this time is displayed in figure 4.6 and the density distribution in the insulation is shown in figure 4.7.

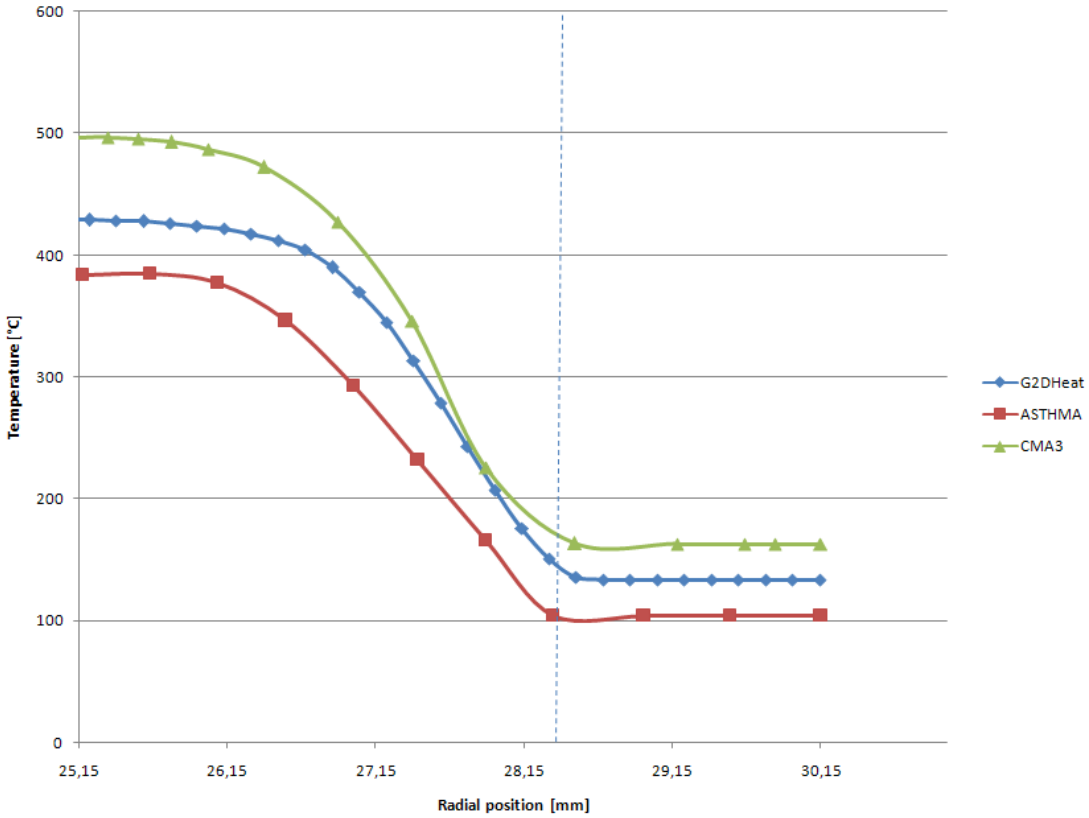


Figure 4.6: Comparison of temperature profile at 10 seconds

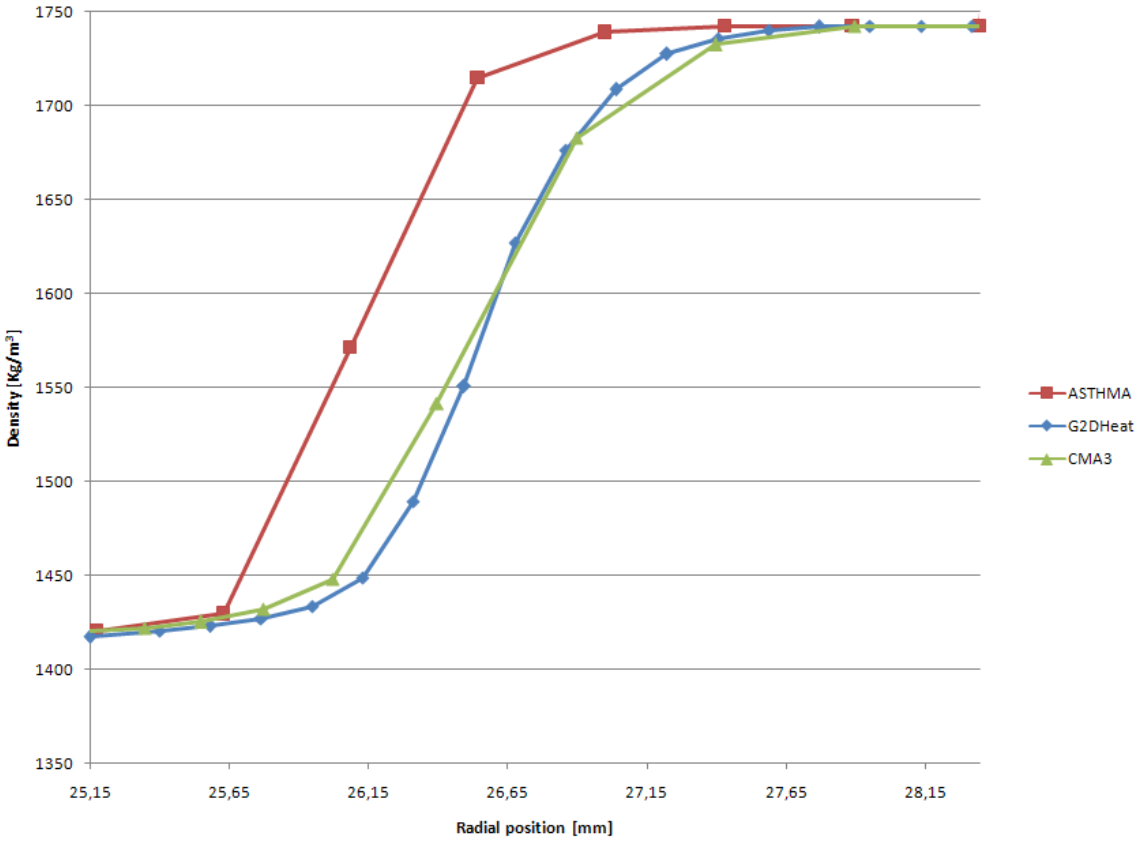


Figure 4.7: Comparison of density of the insulation at 10 seconds

From figure 4.6 it is seen that the heat has started to distribute itself in the model geometry, the temperature in the aluminium increases while the temperature in the insulations outer parts decreases. At the same time the amount of energy remaining in the model for the different programs after the heat input becomes more eminent. The highest simulated temperature is obtained using CMA3, followed by G2DHeat and then ASTHMA.

Since the temperature in the CMA3 simulation is higher than for G2DHeat, especially in the inner part of the insulation, decomposition events are more active, which can be seen from the change in density between figure 4.5 and figure 4.7. The density in the outer part gets closer to the density of G2DHeats, while in the inner parts of the insulation it becomes less than G2DHeat. This complies with the fact that there is a greater mass loss with increasing temperature in the inner parts of the insulation. However, there is not enough insulation decomposing or heat sink effect to lower the temperature in the aluminium to the same level as the temperature in G2DHeats simulation. Even though the density in CMA3 is closing in on the one simulated by G2Dheat, the decomposition reactions are occurring at a lower temperature than they did in G2DHeat, and less heat sink effect is generated. Because, as it can be seen from the curve for heat of pyrolysis in figure 2.6, it increases at a higher rate than the temperature, thus more energy is consumed by the same amount decomposed material, the higher the temperature becomes.

At 10 seconds the temperature simulated by ASTHMA is lower than both CMA3 and G2DHeat, as shown in figure 4.6. This is quite compatible with the simulated temperature at 5 seconds, since from this point heat is lost to the surroundings and/or is used to decompose parts of the insulation. Hence, it becomes more apparent that there has been a larger heat sink, or that heat has been prevented to enter the system in some other way. When considering ASTHMA, the density is large in the outer parts of the insulation (25.65 mm to 26.15 mm), where the simulated temperatures are at their highest, as displayed in figure 4.4 and figure 4.6. These temperatures are not so high that the aluminium can be heated sufficiently to reach the same temperature as simulated by CMA3. The deviation is enhanced since the remaining insulation, in this part, continues to decompose, which provides a more powerful heat sink, compared to CMA3.

Chapter 5: Comparisons to experimental observations

In this chapter two different geometric models are used to compare the program simulations with experimental observations. A simple model is applied to assess the simulated results for temperature and char depth. A more complex model is then used to show that the program can perform two dimensional simulation of heat transfer in decomposing materials. At the end, the simulated char depths are compared to those experimentally measured.

5.1 The simple model

The simple model described in chapter 5.1.1 is used in this comparison of simulation results from G2DHeat to experimental observations. Nammo has performed firing tests of the rocket motor, where the model geometry is taken from, to obtain these experimental observations. Here the rocket motor was mounted horizontally to a static vehicle. During the testing they monitored temperatures on the motors outer surface, before they examined the fired motor and measured the char depths of the insulation (SiPh).

The transient heat transfer from the hot combustion gases to the insulation material is simulated by assuming an isentropic flow through the motor. Local convective heat transfer coefficient and recovery temperature are then calculated from the properties of the combustion gases together with stagnation pressure and stagnation temperature. How this is performed is described by Riise(2008).

5.1.1 Model definition

The model is assumed to be insulated on its right and left side for the same reasons as stated in chapter 4.2. Since ablation temperature in this case is unknown for silica phenolic, it is only performed simulations with pyrolysis effects and mechanical erosion present. The model is shown in figure 5.1.

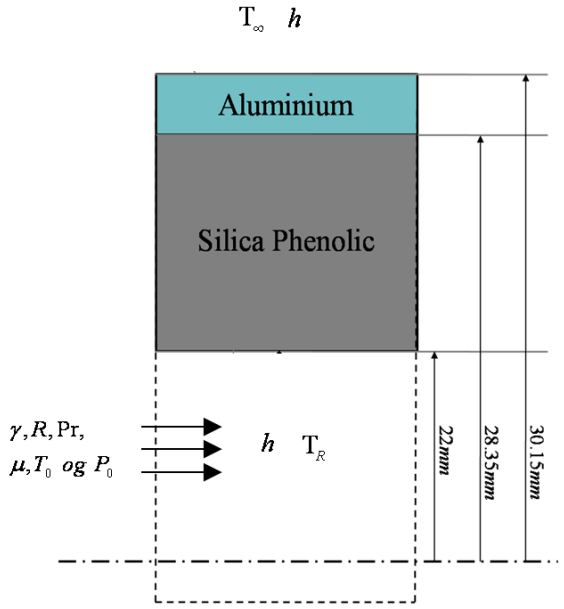


Figure 5.1: Schematic of the simple model

Total time for the simulation is 60 seconds. The boundary condition for the outer surface is free convection to the surrounding air, with a convective heat coefficient value of 26 $W.m^{-2}K^{-1}$ and ambient temperature of 288.15 K. Inside the motor the boundary condition is divided into three time periods. In the two first time periods (0 - 4.4s and 4.4 - 5s), a subsonic flow of combustion gases through the motor is used to calculate the convective heat transfer coefficient and the recovery temperature (a temperature adjusted or corrected for the mechanism of decelerated combustion gases in the surface boundary layer)(Riise 2008). Properties of the combustion gas together with stagnation pressure for the two time periods are shown in table 5.1. In the cooling phase, the third time period (5 - 60s), a convective heat transfer coefficient of 50 $W.m^{-2}K^{-1}$ and ambient temperature of 473K are used. All values are supplied by Nammo.

Table 5.1: Property values used for calculating recovery temperatures and convective heat transfer coefficients

Property	Unit	Value
Gas constant (R)	$J.Kg^{-1}K^{-1}$	313.18
Adiabatic constant (γ)	-	1.1553
Stagnation pressure (P_0) – time[s] 0 - 4.4	Pa	9.5×10^{6}
Stagnation pressure (P_0) – time[s] 4.4 - 5	Pa	4.5×10^{6}
Stagnation temperature (T_0)	K	3165.9
Prandtl number (Pr)	-	0.41
Viscosity (μ)	$Ns.m^{-2}$	0.76×10^{-4}
Radius in the nozzle throat (r)	m	0.01855

Material properties of silica phenolic and aluminium are shown in appendix F, while the mechanical erosion rate used in the simulation is shown in figure 4.2.

5.1.2 Results

The temperature histories at the top and bottom of the horizontally mounted motor were provided by Nammo, and show temperature variations on its outer surface. Figure 5.2 compares the temperature history on the external surface obtained from experimental observations and results from the computer simulations.

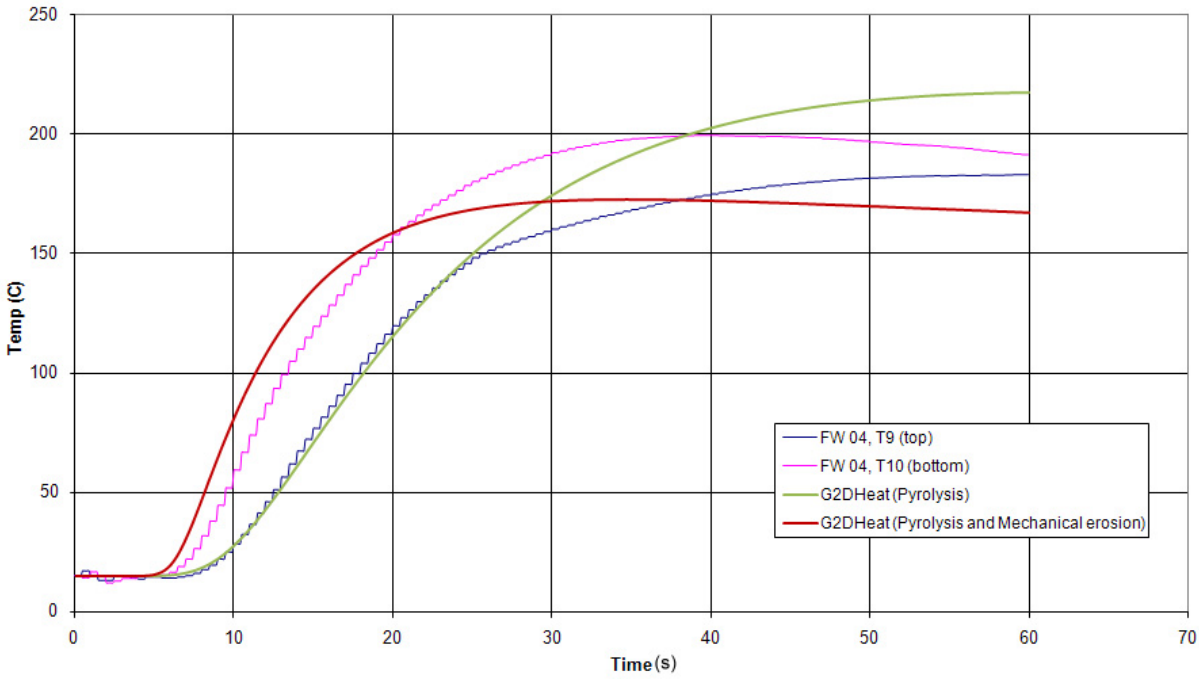


Figure 5.2: Temperature history on the external surface

From figure 5.2 it can be seen that there was a considerable difference in the measured temperatures on the top of the motor compared to those measured at the bottom of the motor. This difference might be reduced during flight, since the rocket no longer is restrained to the static vehicle, thus causing less accumulation of heat beneath the rocket motor. Considering this effect, it is probable that the temperature would have ended up somewhere in-between these measured results. Since such effects are absent in the simulation program, the simulated results also should have ended up in this region.

During motor burn (0 – 5s) the simulated temperatures on the outer surface remain unchanged, which is consistent with the measurements. At burn out (t=5s), the insulation depth for the case with mechanical erosion is reduced, while in the case where only pyrolysis is considered, the insulation depth is constant. The reduction of insulation depth should give an increase of the outer surface temperature at an earlier point than in the case where it remains constant. This is because the same temperature front now has a shorter distance to travel. The total temperature increase should also be lower in this case, since the removal of mass will give reductions to the amount of stored energy. The simulation results displayed in figure 5.2 is in agreement with these predictions.

Comparing the simulated temperatures and the experimentally obtained temperatures shown in figure 5.2 it is clear that there are deviations. In the simulation including mechanical erosion the increase in temperature starts early as predicted, but looking at the two measured temperature

curves it is evident that the simulation results are not within the desired region. Choosing a different erosion rate might shift the simulated temperatures closer to the measured values. The total temperature increase will also be influenced by a change in the erosion rate, this by allowing more or less heat to enter the parts of the material remaining after the erosion.

Considering the other simulated case, the temperature history for the top side of the rocket motor is a good match up to approximately 25 seconds. After the 25 seconds, the simulated temperatures increase even further. This is probably because energy is present in parts of the material which is eroded in the experimental case.

Other reasons for the differences between the simulated and experimental results could be:

- Unrealistic values of material properties used in the simulations.
- Poor estimates of the heat of pyrolysis and specific heat capacities for the pyrolysis gas.
- Too coarse assumptions made to the physical models implemented in the program.
- Inadequate numerical approximations.
- Insufficient specification of the boundary conditions.
- Physical effects not accounted for in the program, such as chemical reactions on the material surface.
- Experimental errors.

The values for the material properties are collected from old input files used in CMA3. The origin of these values is unknown and could therefore be faulty. If the composition of silica phenolic used in this rocket motor is inconsistent with the properties collected from the input files, the simulation results could contain errors. Another possible source of error is the conversion of units between CMA3 and G2DHeat.

The heat of pyrolysis and enthalpies of the pyrolysis gas are determined from analysing the pyrolysis gas as described in chapter 2.4. For the simulations performed in this chapter, relevant values are gathered from the old input files used in CMA3. The heat of pyrolysis is assumed to be a function of temperature only, or more detailed, a function of the pyrolysis gas enthalpy and the enthalpy of the decomposing material when neglecting the pressure changes in the material. This approximation and the values gathered from the input files might cause errors in the simulations, but the size of the errors is unknown.

If the decomposition reactions that occur in silica phenolic are dependent on each other, it can be inadequate to use the kinetic model currently implemented in the program. The reason for this is the assumption that the decomposition reactions are independent of each other. Provided that the reactions are dependent on more variables than temperature alone, a different kinetic model for decomposition reactions should be considered.

The numerical approximations made in the program could be insufficient for simulating the physical events occurring when the insulation material is decomposing. Suggested improvements are described in chapter 6.2.

In the rocket motor, the flow of exhaust gases is assumed to be isentropic. Using this assumption in the program calculations of the heat transfer coefficient and recovery temperature, would result in a simulation of an ideal situation. But in real life, such ideal situations seldom occur, due to energy

losses such as chemical reactions, energy stored in solid particles etc. Assuming the temperature after the firing of the motor to be constant at 473K, is a rough estimate since the temperature will decrease from a very high temperature towards the ambient temperature on the outside of the rocket. To better approximate the boundary conditions, the boundary layer near the surface could be investigated. This could lead to better estimates of the actual heat flux entering the material, but the models are often more complicated and need accurate user specified values for improving the simulation results.

If other physical effects such as fissuring of the material, chemical reactions at the surface and/or accumulated pressure in the material occur, are not accounted for by the program. This could lead to poorer quality of the simulation results.

Experimental errors caused by the measuring equipment or the layout of how the firing tests are performed, could lead to inaccurate results. However, this is not likely, since there has been performed several firing tests with qualified personnel monitoring the tests.

The remaining thickness of silica phenolic is 3.2 mm in the physically tested rocket, and 1.1 mm of this is char. In figure 5.3 and 5.4, the material densities for the simulated cases are shown at the end of the simulation.

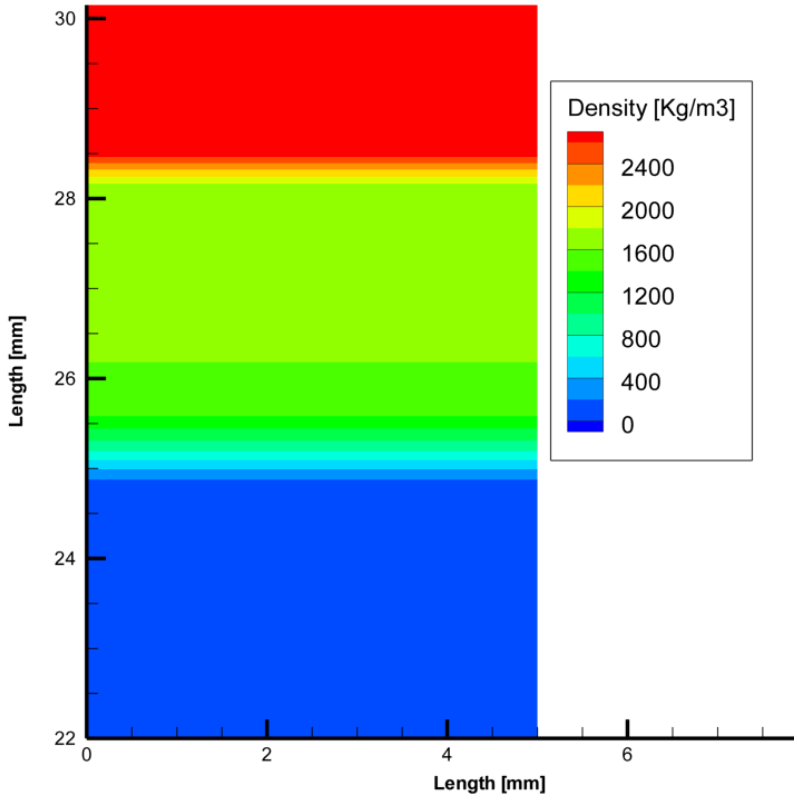


Figure 5.3: Densities in the case with pyrolysis and mechanical erosion (Time= 60 s)

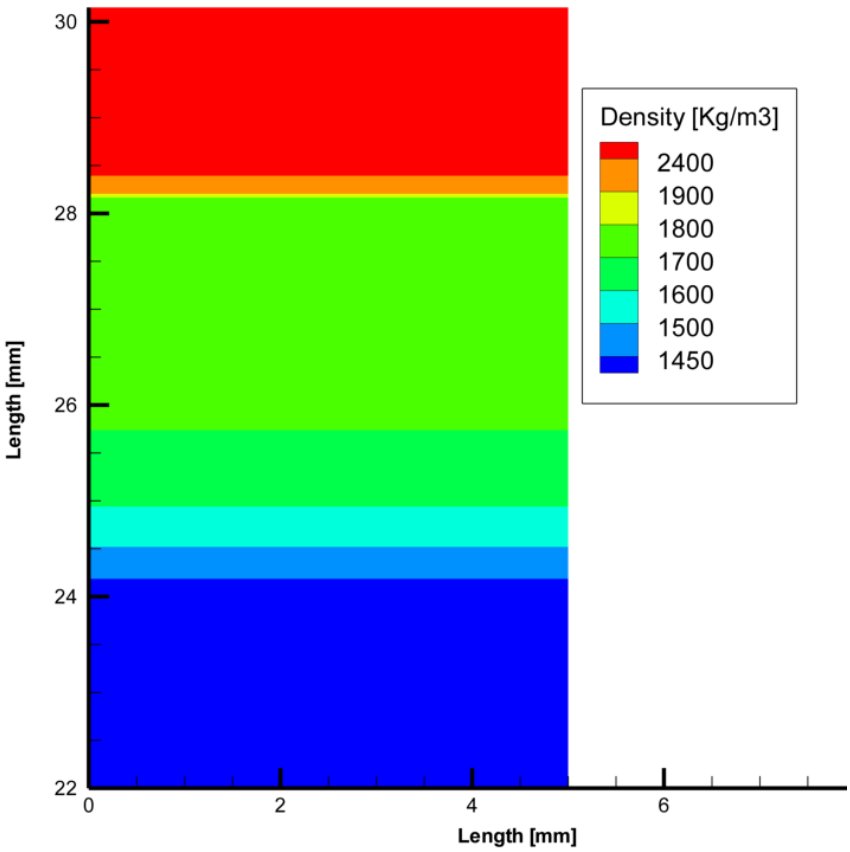


Figure 5.4: Densities in the case with pyrolysis (Time= 60 s)

For the situation with mechanical erosion, it is seen in figure 5.3 that the char depth is near the measured depth of 1.1 mm. The dark green area is more or less char in this figure, and it is easy to see how the material density in the partially decomposed region is varying. For the observed char depth in the tested motor, it can be hard to determine the actual density of the material or the variation of density in the partially decomposed region. To be certain that the simulated char depth is valid with the measured one, different tests with various char depths remaining after the tests should be compared with corresponding computer simulations.

For the case with pyrolysis shown in figure 5.4, the thickness of the char layer is observed to be much grater than the measured one, but not as deep. This is probably because a great deal of the char layer is removed by erosion in the experimental tests, thus allowing the decomposition front to reach further into the material.

5.2 The complex model

One of the unique features of G2DHeat is the possibility of simulating two-dimensional heat transfer in the decomposing materials. Therefore, more complex model geometry is selected so this can be visualised. Similar to the simple model, Nammo has performed analogue firing tests with the rocket motor where the geometry is taken from.

5.2.1 Model definition

The model includes an axial two dimensional cut of the blast pipe and nozzle of the rocket motor. The model consists of silica phenolic as insulation with a graphite nozzle throat insert to withstand erosion when high velocity combustion gases are flowing through the nozzle. The motor case consists of aluminium. The left side of the model is assumed to be insulated, since the axial heat transfer from the rest of the structure is neglected. The schematic of the model is shown in figure 5.5.

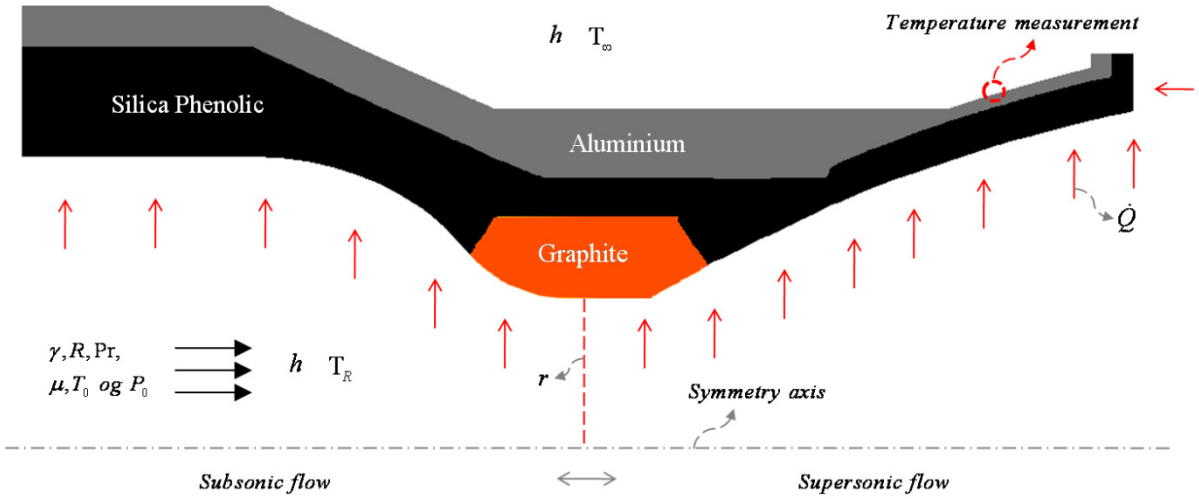


Figure 5.5: Schematic of blast pipe and nozzle

The total simulation time is 300 seconds. The boundary condition on the outer surface is free convection with a heat transfer coefficient of 20 $W.m^{-2}K^{-1}$ and an ambient temperature of 288 K. The motor is burned for a period of 15 seconds before it is cooled for a period of 285 seconds. For the boundary conditions on the inside of the motor, an isotropic flow is used to calculate the local convective heat transfer coefficient and recovery temperature until burn out of the motor (t=15s). The flow is assumed to be subsonic up to the nozzle throat, and supersonic after. Computer calculations are described by Riise(2008). Values for this simulation are given in table 5.2. In the first part of the cooling period (15 – 40s) the convective heat transfer coefficient 50 $W.m^{-2}K^{-1}$ and the ambient temperature is 473 K, while for the last part of the cooling period (40 – 300s) the convective heat transfer coefficient is 20 $W.m^{-2}K^{-1}$ and the ambient temperature is 288 K. All values are supplied by Nammo.

Material properties for the silica phenolic, aluminium and graphite are given in appendix F.

Table 5.2: Property values used for calculating recovery temperatures and convective heat transfer coefficients

Property	Unit	Value
Gas constant (R)	$J.Kg^{-1}K^{-1}$	317.57
Adiabatic constant (γ)	-	1.1553
Stagnation pressure (P_0)	Pa	6.5×10^{6}
Stagnation temperature (T_0)	K	2927
Prandtl number (Pr)	-	0.41
Viscosity (μ)	$Ns.m^{-2}$	0.76×10^{-4}
Radius in the nozzle throat (r)	m	0.0115

5.2.2 Results

To verify that the simulation temperatures are representative for what happens physically, they are compared to a temperature measurement from the firing test at the surface point shown in figure 5.5. These temperature histories are shown for both situations in figure 5.6.

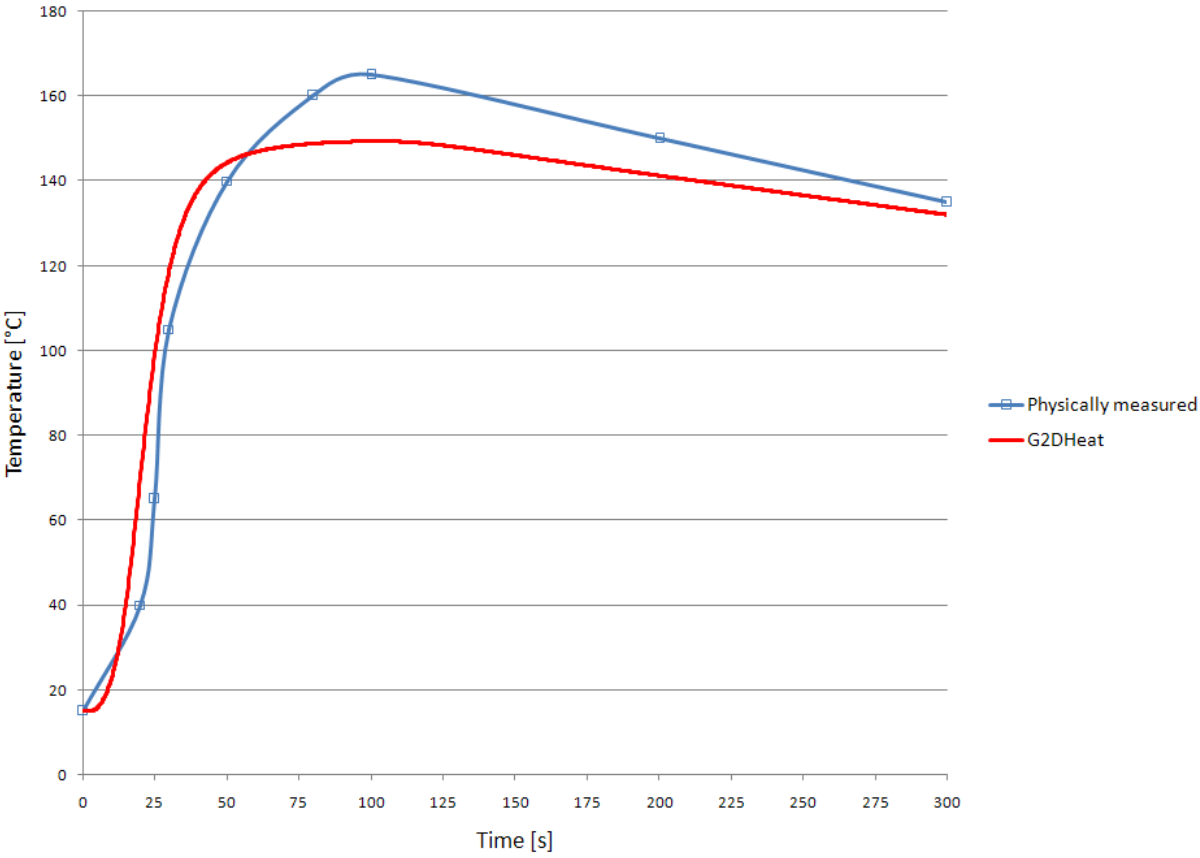


Figure 5.6: Temperature history for the complex model

During motor burn (0 – 15s), it is seen from the figure 5.6 that the simulated temperature increases slower than the experimental. It is likely that the experimental temperature also should increase at a slower rate in the same time period. This is because it takes some time for the heat to reach the

point where the measurement is taken. Looking at the curve for the experimentally obtained temperatures, the marked points represent the eight measurements available. Thus, this only provides a rough indication of how the temperature changes.

As figure 5.6 shows, the maximum temperature reached by the simulation does not agree with the one experimentally measured. The insulation depth where the point of measurement is projected to the symmetry axis is initially 8.3 mm, while after burn out (t = 15s) the depth is 6.5 mm. This erosion is not accounted for in the simulation. A reduction in the insulation depth will cause heat to reach the aluminium faster. Due to the long burn time this allows the aluminium to achieve a higher temperature than when the insulation is intact, and thus increasing the thermal resistance.

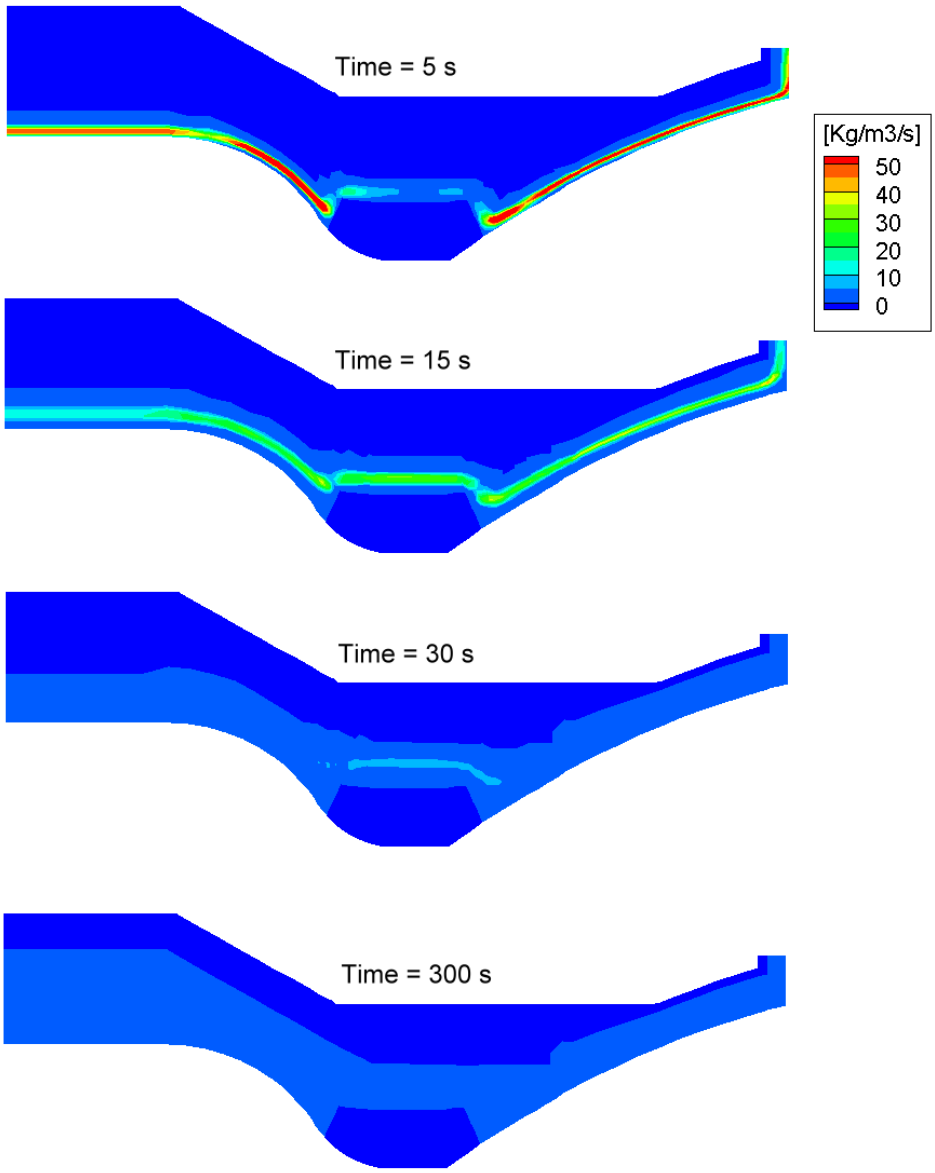


Figure 5.7: Gas production rates as time proceeds

To illustrate the rate at which the material is decomposing, gas production rates for the simulation are presented in figure 5.7. The decomposition rate is at its climax at the start of motor burn, and then decreases from burn out and throughout the cooling period. After about 30 seconds of

simulation time, insignificant amounts of gas generation occur. Corresponding density changes are shown in figure 5.8.

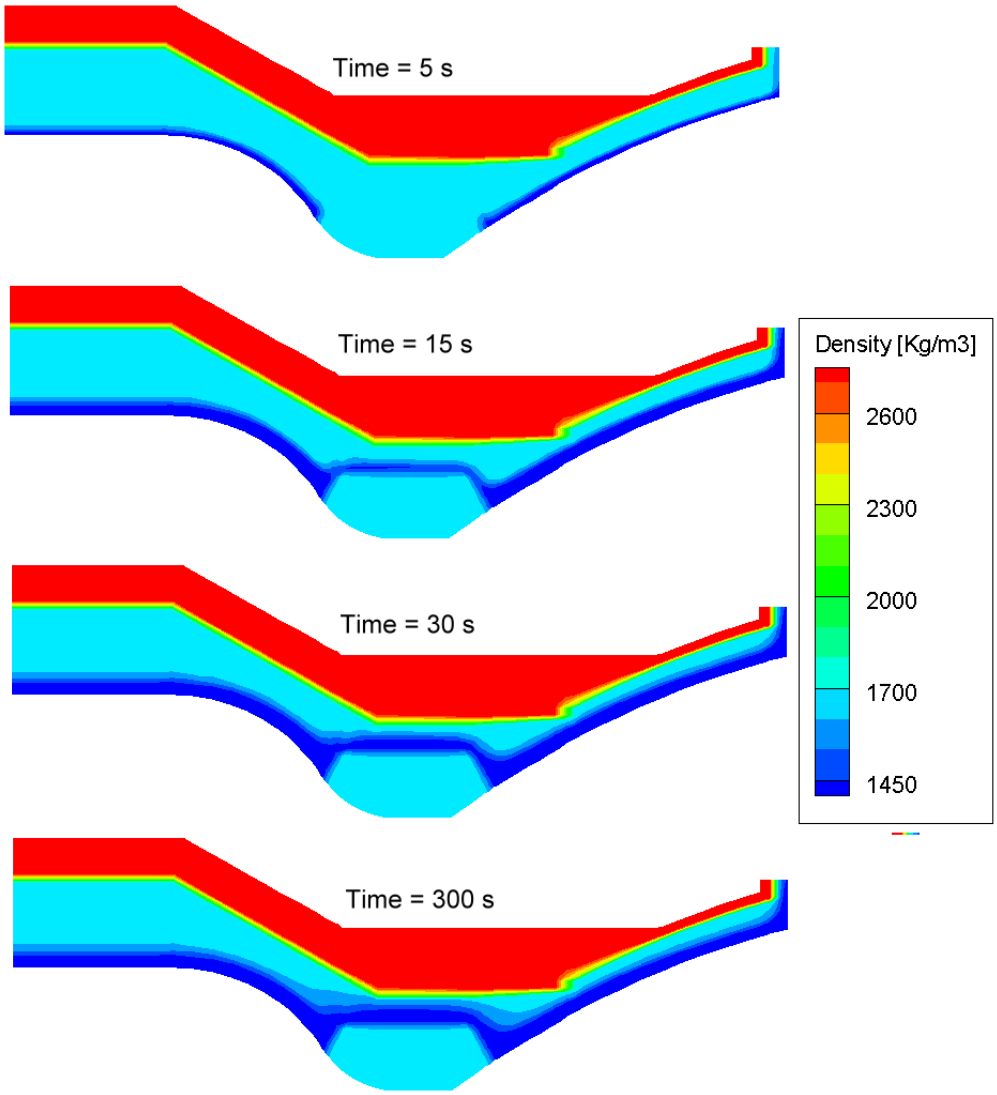


Figure 5.8: Material densities as time proceeds

From the start of motor burn till approximately 30 seconds have passed, the difference between char and virgin is eminent, because the decomposition zone is very narrow. During the cooling period the latent heat causes the decomposition zone to increase, this is especially visible after 300 seconds nearby the nozzle throat insert in figure 5.8. Studying the kinetic values given in table 2.2, it is natural for this phenomenon to happen, since the temperature, even at the end of the simulation, is greater than what is necessary for the decomposition reactions to occur.

After the firing test, measurements of char depths were performed at the positions A, B, C and D shown in figure 5.9.

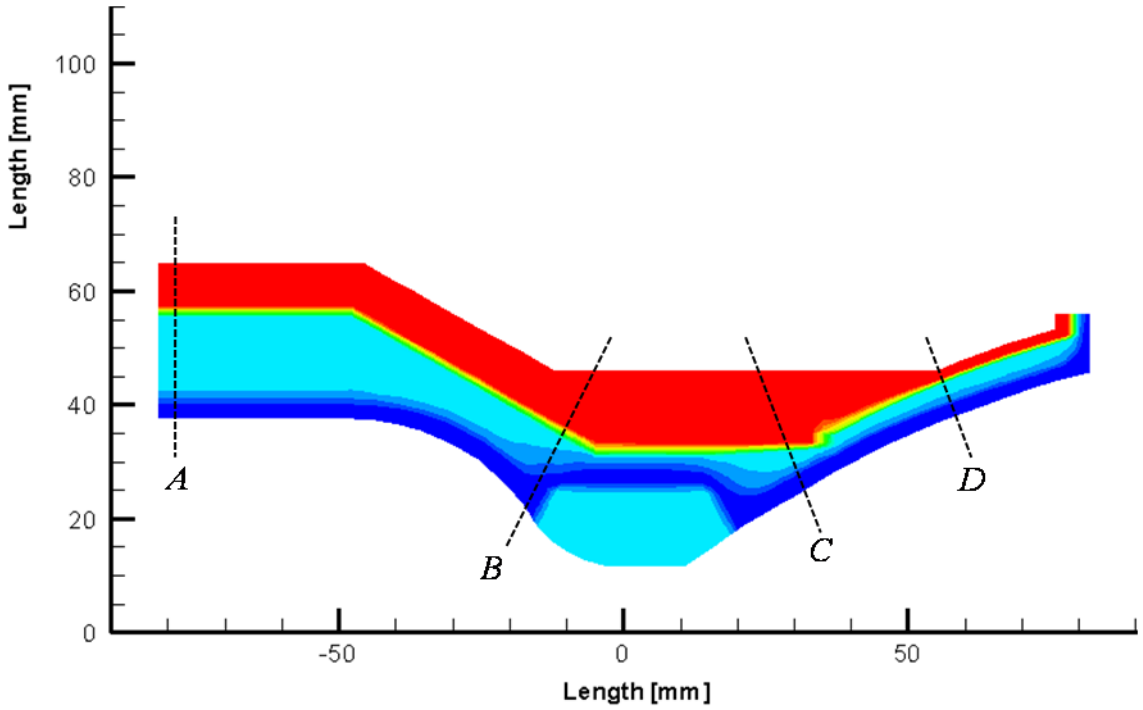


Figure 5.9: Positions where the char depth is measured

When comparing the measured char depths with the simulated ones in table 5.3, it is assumed that material density up till approximately 1550 Kg.m^3 of the initial 1742 Kg.m^3 is defined as char compared to the experimentally measured. It is also important to have in mind that the experimentally measured depth is the actual char depth without including the depth of receded insulation, which for some of the positions is relatively large.

Table 5.3: Char depths measured from the simulated and experimental results

Position	Simulated [mm]	Experimental [mm]
A	3.4	3.4
B	9.3	8
C	4.7	4.8
D	3	3.1

There is some deviation between the simulated and experimental char depths in table 5.3. One possible reason for this is the absence of erosion in the simulated case. Depending on what rate the insulation recedes during motor burn, the final char depth can become different. Therefore more physical models should be included in the program to calculate the recession, or studies of how the material recedes at different positions should be carried out.

At position C in figure 5.9 the material has not receded at all in the experimental case, therefore measurements at this point is representative for comparison with at the identical position in the simulated case. The predicted char depth in position C is in close range with the experimental results. Decomposition events in this part of the material are mainly influenced by temperature differences because the surface remains fixed, and is therefore not chemically or mechanically consumed.

Many other sources of errors due to the assumptions made can also significantly affect the results. These are mentioned in chapter 5.1.2.

Chapter 6: Conclusions and further work

In this chapter the present program development and results are concluded. Then, recommendation for further work to improve the program is presented.

6.1 Conclusions

From the development and testing of G2DHeat the following conclusions can be drawn:

- The program is modified to solve the energy equation implicit in time for two-dimensional multi-block grids.
- The implicit solution routine performs well compared to the explicit routine, especially when considering the possibility of using a larger time step in the simulation. In the explicit routine it is necessary to check, and possibly adjust, the time step size to ensure a convergent solution, while the implicit routine avoids this, and therefore saves computational time. The accuracy of the implicit routine, when only calculating conduction in solids, is in close range with the commercially available program COMSOL Multiphysics. Allowing the use of different time step to achieve adequate results, the computation time for the implicit routine is lower compared to the explicit when grids containing less than approximately 9000 cell volumes are used.
- A source term that includes internal energy effects from endothermic and exothermic reactions in the heat transfer calculations is added to the energy balance in the program. In the present program this is used for endothermic pyrolysis reactions of ablative materials.
- The pyrolysis gas flow is calculated explicitly in time by solving the continuity equation using direction vectors pointing from the center of the cell volume towards the interface between the residue zone and the decomposition zone of the ablative material.
- A routine for calculating charring ablation with known recession rate is developed. While mechanical erosion of ablative materials is included as an additional input option when using other boundary conditions.
- Comparing the results from the test simulations performed with the commercial programs, CMA3 and ASTHMA, to G2DHeat, it is evident that a greater heat sink is present in G2DHeat and ASTHMA, than in CMA3. Although the programs use the same kinetic model, significantly differences in results are observed. To determine the cause of error, extensively work with fault localisation is necessary.
- Comparing the experimentally obtained measurements with simulations performed with G2DHeat in chapter 5, the char depths after simulation agree well with the experimental results. Even though a great deal of simplifying assumptions is made in the program, the results give an indication of a program being able to cope with two-dimensional heat transfer calculations for ablative materials. However, the boundary conditions currently available in G2DHeat are not sufficient to properly handle the ablative phenomenon, since the temperature history, as well as recession rate, must be included in the input file. These are difficult to predict, because, physically, they are functions of the heat transfer occurring within the material in addition to the events adjacent to the surface.
- The many uncertainties that arises from the comparisons of results, suggests that more testing is necessary before G2DHeat can be considered trustworthy.

6.2 Recommendations for further work

A considerable amount of changes and improvements have been made to G2DHeat during this work. However, there is still a great deal of improvements and testing necessary before the program can be considered trustworthy and fully qualified to simulate the physics of ablation.

In chapter 4.3, the possibility of an additional heat sink present in G2DHeat is commented. To locate or determine its existence, the following suggestion is given. First, a case should be generated for CMA3, then verified with experimental results, before the same case is employed in ASTHMA. If these approximations yield satisfying results and can be supported by the experimental results, verification and troubleshooting of G2DHeat can be initiated. If an additional heat sink exists, the source of error should be found by comparing the input and output of the programs. The short time available in this thesis for testing and attempting to find the cause of the deviation in the results, thus the reason for believing that there exists an additional heat sink, has been hindered by too many uncertainties concerning CMA3 and ASTHMA.

Further work recommended for improving G2DHeat is given throughout the rest of the chapter.

6.2.1 Improving the calculation of pyrolysis gas flow

To compute the continuity of pyrolysis gas in G2DHeat, a generation of direction vectors to decide the directions of the gas flow are required. This determination of gas flow is not entirely correct, but is assumed to be a fair approximation, considering the lack of information on pressure relations within the decomposing materials. However, in further work this simplified assumption should be improved by including a calculation of pressure and velocity field of the pyrolysis gas. If the gas flow is governed by the pressures inside the material, there is no need for complicated routines for calculating the interfaces or direction vectors any more. The flow direction is given directly by the pressure differences within the material.

To prevent the pressure controlled gas flow from entering impenetrable materials when flowing through the geometry, permeability values for the materials could be used. Where, values between 1 and 0 represent the partially penetrable material and 1 the impenetrable. In some way, these values perhaps could be determined by the decomposition state of the ablative material.

Another possibility, but less accurate, is to improve the geometrical routine to handle more complex situations as discussed in chapter 3.4.3.

6.2.2 Improvements to the boundary conditions

A method for calculating the local convective heat coefficient inside the rocket motor, assuming an isentropic flow adjacent the surface, is present in G2DHeat. This uses the radius at different axial positions in the motor, the pressure, and properties of the gases flowing at the surface to obtain the coefficients. However, the calculation is performed only once, which is in the initiation procedure of the program. To improve the method, this should be executed for every time step to include the effects of an increasing radius, as the ablative material is receding.

To simulate charring ablation, the present G2DHeat program requires a recession rate and an ablation temperature to be specified in the input file. To cope with situations where the recession rate and/or ablation temperature are unknown, other methods should be developed. The method suggested in chapter 2.5 for instance, is a simple way of calculating the recession rate. Instead of

solving the chemical reactions, the heat of ablation is known and the recession rate is calculated from the energy balance at the surface. There are also methods that include the chemical reactions at the surface. A short introduction to how these methods can operate is given in the following suggestion.

The chemical reactions at the surface are often complicated and require greater amount of information of the ablative material, reaction gases, and the boundary layer close to the surface (Schoner 1970). Programs, such as CMA3 and ASTHMA, include these chemical reactions in their surface boundary conditions.

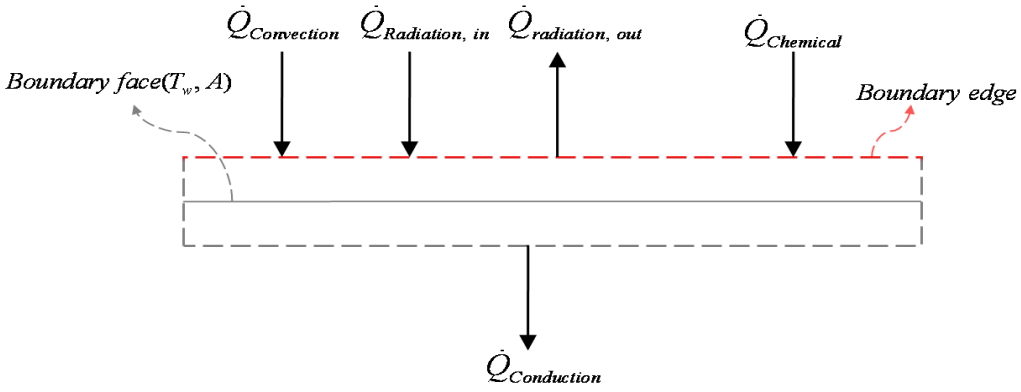


Figure 6.1: Energy balance of the surface

Developing a routine for calculating the energy balance at the surface should be the first step for introducing chemical reactions to the G2DHeat program. The purpose of the routine is to provide a new surface temperature and a new recession rate for the internal energy balance. Numerically, this could be handled explicit or implicit in time. The present internal decomposition events and the internal mass transfer are using the explicit approach, and therefore it should be considered for the chemical reactions too. An energy balance of the surface using the film coefficient model is given by (Schoner 1970):

$$\underbrace{\rho_e u_e C_H \left(H_r - h_e\right)}_{\dot{Q}_{convection}} + \underbrace{\rho_e u_e C_M \left(\sum_i \left(Z^*_{i,e} - Z^*_{i,w}\right) h^{T_w}_{f,i} \right) + \dot{m}_r \left(h_r - h_w\right) + \dot{m}_g \left(h_g - h_w\right)}_{\dot{Q}_{chemical}} \qquad (6.1)$$

$$+\dot{Q}_{radiation,in} - \dot{Q}_{radiation,out} - \dot{Q}_{conduction} = 0$$

This is actually a generalized case of the energy equation and includes unequal mass diffusion coefficients (ASTHMA3 1972 and Schoner 1970). Parameters with subscript “e” are defined at the boundary layer outer edge shown in figure 6.1, while parameters with subscript “w” are defined at the boundary surface. The convective term in equation 6.1 represents the diffusive heat flux from the gas phase to the surface, and excludes chemical energies. The second term represents the net of chemical energy fluxes at the surface. The rest of the terms are described in chapter 2.5. More information concerning specific terms is given in ASTHMA88/PC (1988).

In common, CMA3 and ASTHMA are using three sorts of pre-calculated tables for obtaining the parameters in the energy balance, namely:

- **Time tables** (Parameters as functions of simulation time).

  Heat transfer coefficient($\rho_e u_e C_H$)

  Pressure (P)

  Recovery enthalpy (H_r)

  Radiation flux ($\dot{Q}_{radiation,in}$)

  Mass transfer coefficient ($\rho_e u_e C_M$)(or the constant ratio C_M / C_H)

- **Thermo-chemistry tables** (Parameters as function of pressure, dimensionless ablation rate ($B_c^{'} = \dot{m}_r / \rho_e u_e C_M$) and dimensionless pyrolysis gas rate ($B_g^{'} = \dot{m}_g / \rho_e u_e C_M$).

  Surface temperature ($T_w = f(P,\ \mathrm{B}_c^{'} \text{ and } \mathrm{B}_g^{'})$)

  Energy caused by diffusion driving force("e") ($\sum_i Z_{i,e}^* h_{f,i}^{T_w} = f(P,\ \mathrm{B}_c^{'} \text{ and } \mathrm{B}_g^{'})$

  Enthalpy of gases adjacent to the surface ($h_w = f\left(P,\ \mathrm{B}_c^{'} \text{ and } \mathrm{B}_g^{'}\right)$)

  Energy caused by diffusion driving force("w") ($\sum_i Z_{i,w}^* h_{f,i}^{T_w} = f(P \text{ and } T_w)$)

  Enthalpy of gases at outer boundary layer edge ($h_e = f(P \text{ and } T_w)$)

- **Material property tables** (Parameters as functions of surface temperature).

  Enthalpy of residue products (char) (h_r)

  Enthalpy of pyrolysis gas (h_g)

  Surface emissivity ($\varepsilon_{surface}$)

These are generated by experiments and/or by separate chemistry programs, such as EST (Aerotherm Equilibrium Surface Thermochemistry Program, version 1, 2 or 3), ACE (Aerotherm Chemical Equilibrium Program) and GASKET (Aerotherm Graphite Surface Kinetics Computer Program). To create the thermo-chemistry tables, mass transfer coefficients (within the time range of the simulation)($\rho_e u_e C_M$) must be supplied to the chemistry program. Alternatively, heat transfer coefficients ($\rho_e u_e C_H$) and a ratio between the heat and mass transfer coefficients (C_M / C_H) instead. In addition, some of the programs must also be supplied with pre-exponential factors, which are used for kinetically controlling the surface reactions. Input of material and external gas compositions is obvious, and the remaining input is only used for limiting the output of the program. For instance, specifications of pressure, pyrolysis gas flow rate and ablation rate range (ASTHMA3 1972).

To obtain the physical data necessary for simulating the surface in G2DHeat, this tabular approach should be considered. In this way, there are enough properties available for solving the surface energy balance at the current time step. For further development of G2DHeat, the following surface solution process (for a time step) is suggested:

1. Solve the internal decomposition events to obtain the mass flow of pyrolysis gas ($\dot{m}_g$).
2. Gather necessary values from the time tables at the current time in the simulation, and if necessary, perform corrections to the heat transfer coefficient for various effects adjacent to the surface (Schoner 1970). If the mass transfer coefficient is absent from the time tables, use its ratio to the heat transfer coefficient instead.
3. Calculate the dimensionless pyrolysis gas rate.
4. From an initial guess of the ablation rate($\dot{m}_r$), iteratively solve the energy balance in the sequence:
    - Calculate the dimensionless ablation rate.
    - Obtain the surface temperature and other parameters from the thermo-chemistry tables.
    - Obtain parameters from the material property tables using the surface temperature.
    - Calculate the radiative and the conductive heat flux using the surface temperature (See chapter 2.5).
    - Insert values into the energy balance and solve. If there is departure from zero, select a better guess for the ablation rate and start over. Otherwise, the ablation rate and surface temperature is determined, so move to the next solution step.
5. Calculate the recession rate from the ablation rate (See chapter 2.5) and make adjustments to the cell volumes time “start of erosion” (See chapter 3.2.2).
6. Solve the internal energy balance using the obtained surface temperature.

6.2.3 Including slow cook-off calculations

In tactical missile applications it is often important to characterise the thermal behaviour of the solid propellant, and consequently its sensitivity. If the missile is subjected to unplanned stimuli such as an external fire, it can be useful to predict the detonation or ignition time of the missile. To determine the response of the propellant, slow cook-off tests can be performed. These tests involve uniformly heating of the missile structure including its propellant at low heating rates. After some time, the propellant reaches a certain temperature where exothermic reactions (due to decomposition of the propellant) give an additional increase of the temperature. When this temperature is reached, the heating rate becomes self sustained, and eventually causing the propellant to ignite (Victor 1990).

With the new feature of having a source term in G2Dheat, it is possible to simulate these exothermic reactions in the program. The reaction kinetics of the propellant can be implemented in the same way as the pyrolysis kinetics in the present program.

6.2.4 Improving physical models

The kinetic model and boundary conditions in the program do not include all the relevant physics involved in charring ablation, but they provide a basis for further development of the program. In time, more of the physics can be added as experimental data becomes available. Some suggestions for improvements are:

- Allowing decomposition of multiple materials, each producing a pyrolysis gas that can react with the other gases and the solid material it percolates through.
- Allowing thermal and chemical non-equilibrium.
- Ablation reactions at the surface of the material, not being fully charred.
- Thermo chemical reactions at the surface of the material.

6.2.5 Miscellaneous improvements

When using a large amount of cell volumes in G2DHeats grid (greater than 9000 cell volumes), a parallelisation of the computer code would offer significant computational economies, since the work load is divided on multiple CPUs (Riise 2008). Especially an improvement for the implicit solution routine as it is seen from the comparison of computational time with the explicit solver in figure 1.16.

Simplifying the input specification to the program can be performed by creating a graphical user interface. Potentially, this also reduces the human error when the input files are created. A visualization tool could also be integrated or connected to the user interface for making it more convenient for the user to verify the simulated results.

To handle a complex motor structure in a two-dimensional grid representation can be difficult, since the motor structure can consist of an irregular geometry. Therefore in further development of the program, a three-dimensional expansion should be considered.

References

Austegard, Anders (1997). “An experimental and numerical study of a jetfire stop material and a new helical flow heat exchanger”. “Doctoral thesis”. Trondheim: NTNU.

ASTHMA3 (1972). “USER'S MANUAL, AEROTHERM AXI-SYMMETRIC TRANSIENT HEATING AND MATERIAL ABLATION COMPUTER PROGRAM”. California: Air Force Rocket Propulsion Laboratory.

ASTHMA88/PC (1988). “Installation and User’s Guide, Axi-Symmetric Transient Heating and Material Ablation Program ”. California: Galary Applied Engineering, Inc.

CFD-online (2008). “Source term linearization, Picard's Method”. Collected 25.06.2008 from http://www.cfd-online.com/Wiki/Source_term_linearization

Chapman, Stephen J. (2004). “Fortran 90/95 for Scientists and Engineers, second edition”. New York: McGraw-Hill.

Comsol (2008). “PRODUCTS, COMSOL MULTIPHSYSICS”. Collected 12.05.2008 from http://www.comsol.com/products/.

COMSOL 3.4 (2007). “Material/coefficients Library in COMSOL Multiphysics”. Collected 20.05.2008 from version 3.4.0.248.

Grønli, Morten (1996). “A Theoretical And Experimental Study Of The Thermal Degradation Of Biomass”. “Doctoral thesis”. Trondheim: NTNU.

Incropera, Frank P and David P. DeWitt (2002). ”Fundamentals of Heat and Mass Transfer, Fifth Edition”. New York: John Wiley & Sons, INC.

Moran, Michael J. and Howard N. Shapiro (2004). ”Fundamentals of Engineering Thermodynamics”. New York: John Wiley & Sons, INC.

Myklebust, John (2008). ”Personal Communication”. Raufoss: Nammo AS.

Næss, Erling (1998a). “Input file, CMA3” Dated 02.07.1998. Raufoss: Nammo AS.

Næss, Erling (1998b). “CMA3calculations, Excel document” Dated 07.07.1998. Raufoss: Nammo AS.

Rian, Kjell Erik (2003). ”Numerisk Varme- og Strømningsteknikk, forelesningskompendium”. Trondheim: NTNU.

Riise, Jørn (2008). “Termisk analyse av rakettmotorer”. Project report. Trondheim: NTNU.

Rønningen, Jan-Erik (2001). ”Rakett-teknikk, innføring i grunnleggende rakettmotor teori, 2 utgave”. Andøya: Nasjonalt Senter For Romrelatert Opplæring.

Schoner, Robert J. (1970). ”User`s Manual, Aerotherm Charring Material Thermal Response and Abation Program, Version 3”. California: Air Force Rocket Propulsion Laboratory.

Sutton, George P. and Oscar Biblarz (2001). ”Rocket Propulsion Elements, Seventh Edition”. New York: John Wiley & Sons, INC.

Versteeg, H. K. and W. Malalasekera (1995). ”An introduction to Computational Fluid Dynamics, the finite volume method”. Harlow(England): Pearson Education Limited.

Victor, Andrew C. (1990). “Insensitive Munitions Considerations”. Washington: American institute of Aeronautics and Astronautics, Inc.

Ørbekk, Erland (1994). “Algebraic and elliptic grid generation for CFD applications”. “Doctoral thesis”. Trondheim: NTH

Appendix A: Discretisation of the Heat Balance Equation

The transient heat transfer in two-dimensions can be expressed by (Versteeg and Malalasekera 1995):

$$\rho c \frac{\partial T}{\partial t} = \frac{\partial}{\partial i}\left(k \frac{\partial T}{\partial i} \right) + \frac{\partial}{\partial j}\left(k \frac{\partial T}{\partial j} \right) + \overline{S}$$

First, the heat contribution is integrated over the control volume in time and space:

$$\int_t^{t+\Delta t} \int_{CV} \rho c \frac{\partial T}{\partial t} dV = \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial i}\left(k \frac{\partial T}{\partial i} \right) dV dt + \int_t^{t+\Delta t} \int_{CV} \frac{\partial}{\partial j}\left(k \frac{\partial T}{\partial j} \right) dV dt + \int_t^{t+\Delta t} \int_{CV} \overline{S} dV dt$$

Then, the Gauss-theorem (from volume to face integral) is employed:

$$\int_{CV} \left[\int_t^{t+\Delta t} \rho c \frac{\partial T}{\partial t} dt \right] dV = \int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial i} \right)_e - \left(kA \frac{\partial T}{\partial i} \right)_w \right] dt + \int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial j} \right)_n - \left(kA \frac{\partial T}{\partial j} \right)_s \right] dt + \int_t^{t+\Delta t} \int_{CV} \overline{S} dV dt$$

By using forward Euler in time and central differencing in space, the following terms are obtained:

$$\int_{CV} \left[\int_t^{t+\Delta t} \rho c \frac{\partial T}{\partial t} dt \right] dV = \rho c (T_p - T_p^0) \Delta V$$

$$\int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial i} \right)_e - \left(kA \frac{\partial T}{\partial i} \right)_w \right] dt = \int_t^{t+\Delta t} \left[\left(k_e A_e \frac{T_E - T_P}{\delta i_{PE}} \right) - \left(k_w A_w \frac{T_P - T_W}{\delta i_{WP}} \right) \right] dt$$

$$\int_t^{t+\Delta t} \left[\left(kA \frac{\partial T}{\partial j} \right)_n - \left(kA \frac{\partial T}{\partial j} \right)_s \right] dt = \int_t^{t+\Delta t} \left[\left(k_n A_n \frac{T_N - T_P}{\delta j_{PN}} \right) - \left(k_s A_s \frac{T_P - T_S}{\delta j_{SP}} \right) \right] dt$$

$$\int_t^{t+\Delta t} \int_{CV} \overline{S} dV dt = \int_t^{t+\Delta t} \overline{S} \Delta V dt$$

Combining these using the fully implicit scheme, yields:

$$\rho c (T_p - T_p^0) \Delta V = \left[\left(k_e A_e \frac{T_E - T_P}{\delta i_{PE}} \right) - \left(k_w A_w \frac{T_P - T_W}{\delta i_{WP}} \right) \right] \Delta t + \left[\left(k_n A_n \frac{T_N - T_P}{\delta j_{PN}} \right) - \left(k_s A_s \frac{T_P - T_S}{\delta j_{SP}} \right) \right] \Delta t + \overline{S} \Delta V \Delta t$$

Introducing harmonic mean thermal conductivity defined as:

$$Q_e = \left(k_e \frac{T_E - T_P}{\delta i_{PE}} \right) = \left(k_E \frac{T_E - T_e}{\delta i_{Ee}} \right) = \left(k_P \frac{T_e - T_P}{\delta i_{eP}} \right)$$

$$\Rightarrow \left(\frac{T_E - T_e + T_e - T_P}{\frac{\delta i_{Ee}}{k_E} + \frac{\delta i_{eP}}{k_P}} \right) = \left(\frac{T_E - T_P}{R_{I,e}} \right)$$

Performing linearisation of the source term:

$$\overline{S} \Delta V \Delta t = \left(S_P T_P + S_u \right) \Delta t$$

Rearranging and inserting terms into the final equation:

$$\underbrace{\left(\frac{\rho c \Delta V}{\Delta t} + \frac{A_e}{R_{I,e}} + \frac{A_w}{R_{I,w}} + \frac{A_n}{R_{J,n}} + \frac{A_s}{R_{J,s}} - S_p \right)}_{a_p} T_p = \underbrace{\frac{A_e}{R_{I,e}}}_{a_e} T_E + \underbrace{\frac{A_w}{R_{I,w}}}_{a_w} T_W + \underbrace{\frac{A_n}{R_{J,n}}}_{a_n} T_N + \underbrace{\frac{A_s}{R_{J,s}}}_{a_s} T_S + \underbrace{\frac{\rho c \Delta V}{\Delta t}}_{a_p^0} T_p^0 + S_u$$

$T^0 = Temperature(time = t)$
$T = Temperature(time = t + \Delta t)$
$R = Thermal\ resistance$
$A = Lateral\ surface$
$\rho = Density$
$c = Heat\ capacity$
$\Delta V = Cell\ volume$

Appendix B: Linearisation of radiative heat transfer terms

In order to include radiation heat transfer in the implicit solution routine, linearisation of the radiation terms from the explicit routine is performed (CFD-online 2008):

$$\dot{Q}_{radiation,\,in} = A_* \varepsilon_* \sigma T_{surrounding}^4 = constant$$

$$\dot{Q}_{Radiation,\,out} = A_* \sigma \varepsilon_* T_P^4 = f(T_P^4)$$

Combining the radiation in and out:

$$S = \dot{Q}_{Radiation,\,tot} = A_* \varepsilon_* \sigma T_{surrounding}^4 - A_* \sigma \varepsilon_* T_P^4$$

A Taylor series expansion of the source is expressed by:

$$S = S^0 + \left(\frac{\partial S}{\partial T}\right)^0 \left(T_P - T_P^0\right)$$

Where

$$\left(\frac{\partial S}{\partial T}\right)^0 = -A_* \sigma \varepsilon_* 4 \left(T_P^3\right)^0$$

Here ”0” denote values at the previous iteration. The source term becomes:

$$S = \left(A_* \varepsilon_* \sigma T_{surrounding}^4 - A_* \sigma \varepsilon_* T_P^4\right)^0 + \left(-A_* \sigma \varepsilon_* 4 \left(T_P^3\right)^0\right)\left(T_P - T_P^0\right)$$

Dividing the temperature dependent part from the non-temperature dependent part yields:

$$S_P = \left(-A_* \sigma \varepsilon_* 4 \left(T_P^3\right)^0\right)$$

$$S_u = A_* \varepsilon_* \sigma T_{surrounding}^4 + \left(A_* \sigma \varepsilon_* 3 \left(T_P^4\right)^0\right)$$

These are used in the implicit solution routine.

Appendix C: Discretisation of the Energy Balance Equation

The energy equation on differential form can be expressed as (Rian 2003):

$$\frac{\partial \rho \overline{u}}{\partial t}+\frac{\partial \rho hu}{\partial i}+\frac{\partial \rho hv}{\partial j}=\frac{\partial}{\partial i}\left(k\frac{\partial T}{\partial i}\right)+\frac{\partial}{\partial j}\left(k\frac{\partial T}{\partial j}\right)+\overline{S}$$

Rewriting the transient term:

$$\rho\frac{\partial \overline{u}}{\partial t}+\overline{u}\frac{\partial \rho}{\partial t}+\frac{\partial \rho uh}{\partial i}+\frac{\partial \rho vh}{\partial j}+=\frac{\partial}{\partial i}\left(k\frac{\partial T}{\partial i}\right)+\frac{\partial}{\partial j}\left(k\frac{\partial T}{\partial j}\right)+\overline{S}$$

Assuming constant volume in the solid and employ specific heat capacity:

$$\rho\frac{\partial \overline{u}}{\partial t}=\rho C_{v,s}\frac{\partial T}{\partial t}$$

The equation then becomes:

$$\rho C_{v,s}\frac{\partial T}{\partial t}+\overline{u}\frac{\partial \rho}{\partial t}+\frac{\partial \rho uh}{\partial i}+\frac{\partial \rho vh}{\partial j}=\frac{\partial}{\partial i}\left(k\frac{\partial T}{\partial i}\right)+\frac{\partial}{\partial j}\left(k\frac{\partial T}{\partial j}\right)+\overline{S}$$

Discretise this by using the finite volume method (Rian 2003):

$$\int_t^{t+\Delta t}\int_{CV}\rho C_{v,s}\frac{\partial T}{\partial t}dV+\int_t^{t+\Delta t}\int_{CV}\overline{u}\frac{\partial \rho}{\partial t}dVdt+\int_t^{t+\Delta t}\int_{CV}\frac{\partial \rho uh}{\partial i}dVdt+\int_t^{t+\Delta t}\int_{CV}\frac{\partial \rho vh}{\partial j}dVdt$$
$$=\int_t^{t+\Delta t}\int_{CV}\frac{\partial}{\partial i}\left(k\frac{\partial T}{\partial i}\right)dVdt+\int_t^{t+\Delta t}\int_{CV}\frac{\partial}{\partial j}\left(k\frac{\partial T}{\partial j}\right)dVdt+\int_t^{t+\Delta t}\int_{CV}\overline{S}dVdt$$

Where

$$\int_{CV}\left[\int_t^{t+\Delta t}\rho C_{v,s}\frac{\partial T}{\partial t}dt\right]dV+\int_{CV}\left[\int_t^{t+\Delta t}\overline{u}\frac{\partial \rho}{\partial t}dt\right]dV$$
$$+\int_t^{t+\Delta t}\left[\left(\rho uAh\right)_e-\left(\rho uAh\right)_w\right]dt+\int_t^{t+\Delta t}\left[\left(\rho vAh\right)_n-\left(\rho vAh\right)_s\right]dt$$
$$=\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial i}\right)_e-\left(kA\frac{\partial T}{\partial i}\right)_w\right]dt+\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial j}\right)_n-\left(kA\frac{\partial T}{\partial j}\right)_s\right]dt+\int_t^{t+\Delta t}\int_{CV}\overline{S}dVdt$$

Introduce the continuity equation to improve the numerical approximation (Rian 2003):

$$\frac{\partial \rho}{\partial t}+\frac{\partial \rho u}{\partial i}+\frac{\partial \rho v}{\partial j}=0$$

The enthalpy for cell volume "P" is multiplied on both sides of the equal sign, and then the finite volume method is employed:

$$\int_t^{t+\Delta t}\int_{CV}\frac{\partial \rho}{\partial t}dVdt+\int_t^{t+\Delta t}\int_{CV}(\rho u)dVdt+\int_t^{t+\Delta t}\int_{CV}(\rho v)dVdt=0 \;\Big|\cdot h_P$$

$$\int_{CV}\left[\int_t^{t+\Delta t}h_P\frac{\partial \rho}{\partial t}dt\right]dV+\int_t^{t+\Delta t}h_P\left[(\rho uA)_e-(\rho uA)_w\right]dt+\int_t^{t+\Delta t}h_P\left[(\rho vA)_n-(\rho vA)_s\right]dt=0$$

Subtracting this from the Energy equation:

$$\int_{CV}\left[\int_t^{t+\Delta t}\rho C_{v,s}\frac{\partial T}{\partial t}dt\right]dV+\int_{CV}\left[\int_t^{t+\Delta t}(\overline{u}-h_P)\frac{\partial \rho}{\partial t}dt\right]dV$$
$$+\int_t^{t+\Delta t}\left[(\rho uA)_e(h_e-h_P)-(\rho uA)_w(h_w-h_P)\right]dt+\int_t^{t+\Delta t}\left[(\rho vA)_n(h_n-h_P)-(\rho vA)_s(h_s-h_P)\right]dt$$
$$=\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial i}\right)_e-\left(kA\frac{\partial T}{\partial i}\right)_w\right]dt+\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial j}\right)_n-\left(kA\frac{\partial T}{\partial j}\right)_s\right]dt+\int_t^{t+\Delta t}\int_{CV}\overline{S}dVdt$$

Assuming constant pressure and introduce specific heat capacities:

$$C_{p,g}(T_w-T_P)=(h_w-h_P)$$

$$C_{p,g}(T_e-T_P)=(h_e-h_P)$$

$$C_{p,g}(T_n-T_P)=(h_n-h_P)$$

$$C_{p,g}(T_s-T_P)=(h_s-h_P)$$

The heat of pyrolysis is defined by (Austegard 1997):

$$\Delta h_{pyr}=(h_P-\overline{u})$$

Inserting these into the energy equation:

$$\int_{CV}\left[\int_t^{t+\Delta t}\rho C_{v,s}\frac{\partial T}{\partial t}dt\right]dV+\int_{CV}\left[\int_t^{t+\Delta t}-\Delta h_{pyr}\frac{\partial \rho}{\partial t}dt\right]dV$$
$$+\int_t^{t+\Delta t}C_{p,g}\left[(\rho uA)_e(T_e-T_P)-(\rho uA)_w(T_w-T_P)\right]dt+\int_t^{t+\Delta t}C_{p,g}\left[(\rho vA)_n(T_n-T_P)-(\rho vA)_s(T_s-T_P)\right]dt$$
$$=\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial i}\right)_e-\left(kA\frac{\partial T}{\partial i}\right)_w\right]dt+\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial j}\right)_n-\left(kA\frac{\partial T}{\partial j}\right)_s\right]dt+\int_t^{t+\Delta t}\int_{CV}\overline{S}dVdt$$

Using the fully implicit time discretisation approach (Versteeg and Malalasekera 1995):

$$\int_{CV}\left[\int_t^{t+\Delta t}\rho C_{v,s}\frac{\partial T}{\partial t}dt\right]dV=\rho C_{v,s}(T_p-T_p^0)\Delta V$$

$$\int_{CV}\left[\int_t^{t+\Delta t}-\Delta h_{pyr}\frac{\partial \rho}{\partial t}dt\right]dV=-\Delta h_{pyr}\left(\rho_P-\rho_P^0\right)\Delta V$$

$$\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial i}\right)_e-\left(kA\frac{\partial T}{\partial i}\right)_w\right]dt=\left[\left(k_eA_e\frac{T_E-T_P}{\delta i_{PE}}\right)-\left(k_wA_w\frac{T_P-T_W}{\delta i_{WP}}\right)\right]\Delta t$$

$$\int_t^{t+\Delta t}\left[\left(kA\frac{\partial T}{\partial j}\right)_n-\left(kA\frac{\partial T}{\partial j}\right)_s\right]dt=\left[\left(k_nA_n\frac{T_N-T_P}{\delta j_{PN}}\right)-\left(k_sA_s\frac{T_P-T_S}{\delta j_{SP}}\right)\right]\Delta t$$

$$\int_t^{t+\Delta t}\int_{CV}\overline{S}dVdt=\overline{S}\Delta V\Delta t$$

Apply the use of a harmonic mean thermal conductivity, represented by R, and estimated as:

$$Q_e=\left(k_e\frac{T_E-T_P}{\delta i_{PE}}\right)=\left(k_E\frac{T_E-T_e}{\delta i_{Ee}}\right)=\left(k_P\frac{T_e-T_P}{\delta i_{eP}}\right)$$
$$\Rightarrow\left(\frac{T_E-T_e+T_e-T_P}{\frac{\delta i_{Ee}}{k_E}+\frac{\delta i_{eP}}{k_P}}\right)=\left(\frac{T_E-T_P}{R_{I,e}}\right)$$

Applying the upwind differencing sheme:

$$u = positive \;\Rightarrow\; T_e = T_P \;and\; T_w = T_W$$
$$u = negative \;\Rightarrow\; T_e = T_E \;and\; T_w = T_P$$
$$v = positive \;\Rightarrow\; T_n = T_P \;and\; T_s = T_S$$
$$v = negative \;\Rightarrow\; T_n = T_N \;and\; T_s = T_P$$
$$\dot{m}_e = \left(\rho u A\right)_e$$
$$\dot{m}_w = \left(\rho u A\right)_w$$
$$\dot{m}_n = \left(\rho v A\right)_n$$
$$\dot{m}_s = \left(\rho v A\right)_s$$

The pyrolysis gas production rate denotes:

$$\dot{m}_{pyr} = \frac{\left(\rho_P - \rho_P^0\right)\Delta V}{\Delta t}$$

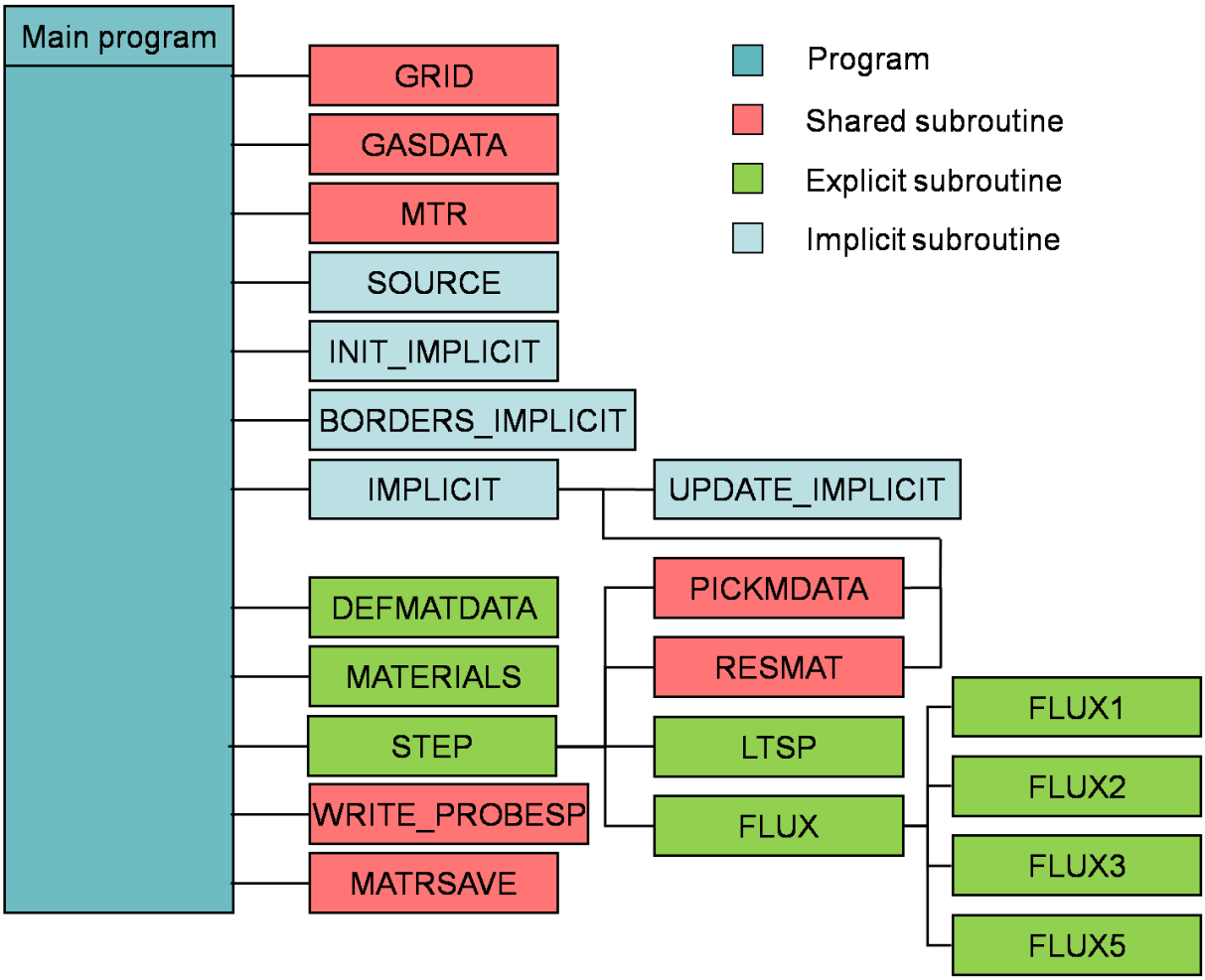
Linearisation of the source term gives:

$$\overline{S}\Delta V \Delta t = \left(S_P T_P + S_u\right)\Delta t$$

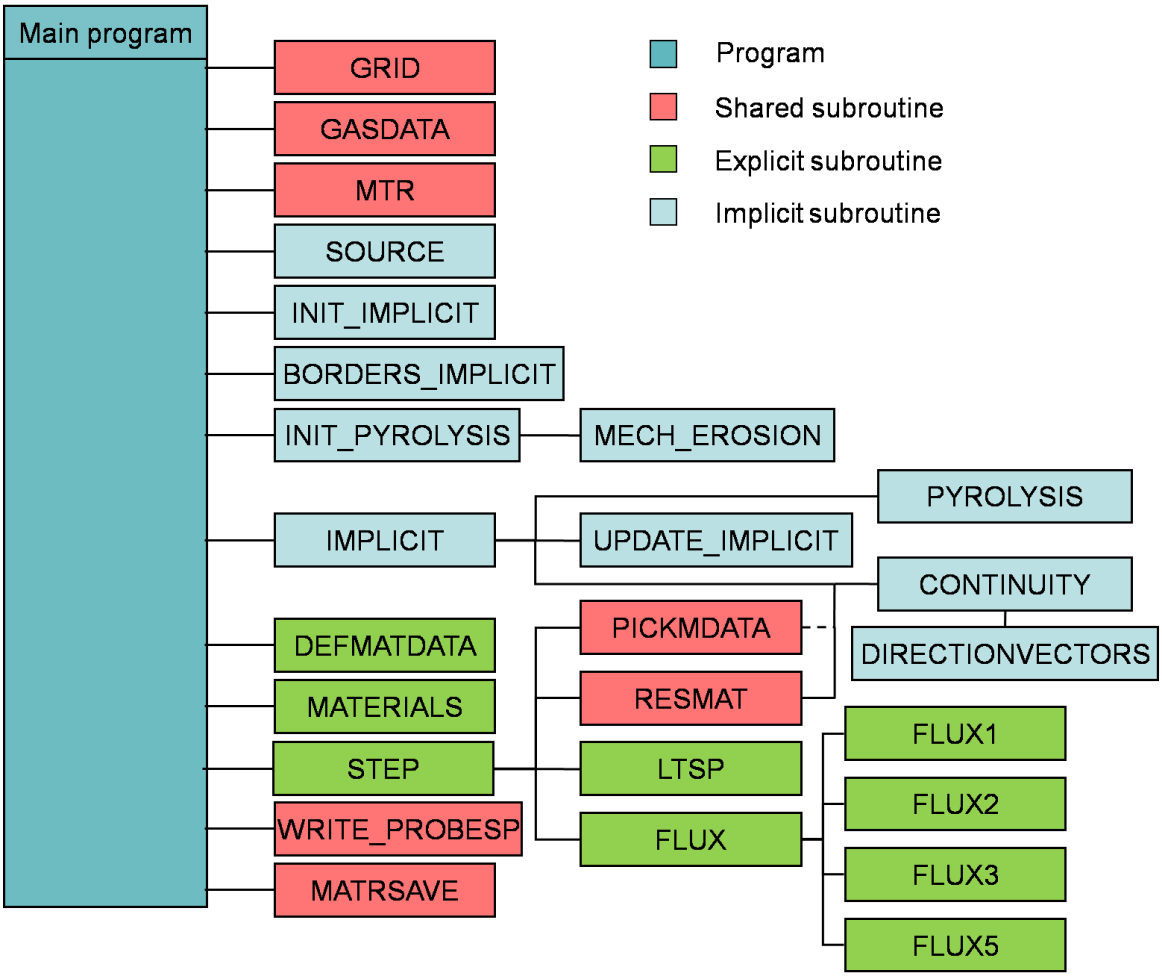
Combining these terms the governing equation solved by G2DHeat becomes:

$$\underbrace{\left(\frac{\rho C_{v,s}\Delta V}{\Delta t} + a_e + a_w + a_n + a_s - S_p\right)}_{a_p} T_p = \underbrace{\left(\frac{A_e}{R_{I,e}} + \max\left(0, -C_{p,g}\dot{m}_e\right)\right)}_{a_e} T_E + \underbrace{\left(\frac{A_w}{R_{I,w}} + \max\left(0, C_{p,g}\dot{m}_w\right)\right)}_{a_w} T_W$$
$$+ \underbrace{\left(\frac{A_n}{R_{J,n}} + \max\left(0, -C_{p,g}\dot{m}_n\right)\right)}_{a_n} T_N + \underbrace{\left(\frac{A_s}{R_{J,s}} + \max\left(0, C_{p,g}\dot{m}_s\right)\right)}_{a_s} T_S + \underbrace{\frac{\rho C_{v,s}\Delta V}{\Delta t}}_{a_p^0} T_p^0 + \dot{m}_{pyr}\Delta h_{pyr} + S_u$$

Appendix D: Block diagram of subroutines



Appendix E: Block diagram of subroutines



Appendix F: Material properties

Silica phenolic

TEMPERATURE	Cp	TCI	TCJ	RHO	CpGas	CpResidue	TCIResidue	TCJResidue	DHPYR
227.780E+00	594.500E+00	000.617E+00	000.617E+00	174.252E+01	251.210E+01	586.200E+00	001.919E+00	001.919E+00	122.910E+04
294.440E+00	782.900E+00	000.629E+00	000.629E+00	174.252E+01	251.210E+01	736.900E+00	001.938E+00	001.938E+00	123.010E+04
422.220E+00	106.760E+01	000.636E+00	000.636E+00	174.252E+01	251.210E+01	921.100E+00	001.981E+00	001.981E+00	133.670E+04
644.440E+00	128.530E+01	000.642E+00	000.642E+00	174.252E+01	251.210E+01	110.530E+01	002.050E+00	002.050E+00	154.140E+04
833.330E+00	139.000E+01	000.642E+00	000.642E+00	174.252E+01	418.690E+01	121.420E+01	002.118E+00	002.118E+00	189.710E+04
111.111E+01	144.860E+01	000.642E+00	000.642E+00	174.252E+01	544.280E+01	128.530E+01	002.193E+00	002.193E+00	280.920E+04
166.667E+01	150.720E+01	000.642E+00	000.642E+00	174.252E+01	732.680E+01	150.720E+01	002.305E+00	002.305E+00	656.500E+04
222.222E+01	152.400E+01	000.642E+00	000.642E+00	174.252E+01	711.760E+01	152.400E+01	002.617E+00	002.617E+00	964.950E+04
277.777E+01	152.820E+01	000.642E+00	000.642E+00	174.252E+01	523.350E+01	152.820E+01	003.601E+00	003.601E+00	117.000E+05
333.333E+01	153.240E+01	000.642E+00	000.642E+00	174.252E+01	523.350E+01	153.240E+01	004.617E+00	004.617E+00	137.460E+05

Temperature [K]

Thermal conductivity(TCI,TCJ,TCIResidue, TCJResidue) [W/mK]

Heat of pyrolysis(DHPYR) [J/Kg]

Spesific heat capacity(Cp,CpGas,CpResidue) [J/Kg/K]

Density(RHO) [Kg/m3]

RHOO[Kg/m3]	RHOR[Kg/m3]	A[1/s]	(E(i)/R)[K]	n[-]	Treac[K]	VolFrac
325.015E+00	000.000E+00	140.000E+02	855.600E+01	003.000E+00	333.000E+00	000.422E+00
973.926E+00	518.998E+00	975.297E+06	204.440E+02	003.000E+00	550.000E+00	000.422E+00
206.638E+01	206.638E+01	000.000E+00	000.000E+00	000.000E+00	100.000E+07	000.578E+00

Aluminium

TEMPERATURE	Cp	TCI	TCJ	RHO
172.220E+00	954.590E+00	109.659E+00	109.659E+00	280.032E+01
293.330E+00	960.033E+00	130.220E+00	130.220E+00	280.032E+01
373.330E+00	962.964E+00	147.666E+00	147.666E+00	280.032E+01
473.330E+00	967.151E+00	167.604E+00	167.604E+00	280.032E+01
573.330E+00	983.898E+00	188.165E+00	188.165E+00	280.032E+01

Graphite

TEMPERATURE	Cp	TCI	TCJ	RHO
273.150E+00	751.600E+00	117.900E+00	117.900E+00	177.000E+01
373.150E+00	977.700E+00	106.400E+00	106.400E+00	177.000E+01
473.150E+00	120.370E+01	097.000E+00	097.000E+00	177.000E+01
573.150E+00	138.780E+01	087.100E+00	087.100E+00	177.000E+01
673.150E+00	150.840E+01	079.300E+00	079.300E+00	177.000E+01
773.150E+00	162.890E+01	072.300E+00	072.300E+00	177.000E+01
873.150E+00	168.870E+01	066.600E+00	066.600E+00	177.000E+01
973.150E+00	171.130E+01	061.200E+00	061.200E+00	177.000E+01
107.315E+01	173.390E+01	057.100E+00	057.100E+00	177.000E+01
117.315E+01	175.650E+01	053.000E+00	053.000E+00	177.000E+01
127.315E+01	177.920E+01	049.300E+00	049.300E+00	177.000E+01
137.315E+01	180.110E+01	046.400E+00	046.400E+00	177.000E+01
277.315E+01	191.900E+01	041.500E+00	041.500E+00	177.000E+01

Appendix G: Input file used in CMA3

```
     1.0       1.0
Case kap 4 SiPh.6.35mm, Alu.1.8mm, Indre dia. 44.00mm
Tinit 26 C Cm/Ch=N.A. Forced erosion rates
Jørn Riise  2008-07-14  fri konv. (26 W/m2K) og emis.0.05                  18
  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18
A              20.29      0.00    14000.       3.0   15400.0     600.
B              60.80     32.40   4.47 E9       3.0   36800.0    1000.
C             129.00    129.00       0.0       0.0       0.0    8000.
10 18            0.0     100.0      0.20      1.00       5.0      1.0        0.4
      5.00      20.0   0.01000    -4805.    -5293.             0.422      536.0
 1    538.47            0.3937-2    0.8661
 1    538.47            0.7874-2
 1    538.47            0.7874-2
 1    538.47            0.9843-2
 1    538.47            0.9843-2
 1    538.47            1.9685-2
 1    538.47            1.9685-2
 1    538.47            1.9685-2
 1    538.47            1.9685-2
 1    538.47            3.9370-2
 1    538.47            3.9370-2
 1    538.47            1.9685-2
 1    538.47            1.9685-2
 1    538.47            1.3780-2
 3    538.47              .02726
 3    538.47              .02726
 3    538.47              .00817
 3    538.47              .00817
  0.001287      0.05    527.67      0.02      0.98
        410.      .142  0.99 E-4       .6
        530.      .187  1.01 E-4       .6
        760.      .255  1.02 E-4       .6
       1160.      .307  1.03 E-4       .6
       1500.      .332  1.03 E-4       .6
       2000.      .346  1.03 E-4       .6
       3000.      .360  1.03 E-4       .6
       4000.      .364  1.03 E-4       .6
       5000.      .365  1.03 E-4       .6
-1     6000.      .366  1.03 E-4       .6
        410.      .140  3.08 E-4       .6
        530.      .176  3.11 E-4       .6
        760.      .220  3.18 E-4       .6
       1160.      .264  3.29 E-4       .6
       1500.      .290  3.40 E-4       .6
       2000.      .307  3.52 E-4       .6
       3000.      .360  3.70 E-4       .6
       4000.      .364  4.20 E-4       .6
       5000.      .365  5.78 E-4       .6
-1     6000.      .366  7.41 E-4       .6
 3   175.0
        310.    0.2280  .0176000       .0
        528.    0.2293  .0209000       .0
        672.    0.2300  .0237000       .0
        852.    0.2310  .0269000       .0
+1     1032.    0.2350  .0302000       .0
1     500.     1000.     1500.     2000.     3000.     4000.     6000.     6500.
    -2200.    -1900.    -1400.     -750.     1000.     2700.     5200.     6000.
       0.0   4451.40    35.74
      1.00   4451.40    35.74
      1.00   4451.40    28.09
      2.00   4451.40    28.09
      2.00   4451.40    20.10
      3.00   4451.40    20.10
      3.00   4451.40    18.91
      4.00   4451.40    18.91
      4.00   4451.40    21.15
```

```
        5.00   4451.40    21.15
        5.00      1.0     0.15
2     100.00      1.0     0.15
```

Appendix H: Input file used in ASTHMA

```
C  ASTHMA88 v1.00 Jørn Riise 2008-06-30
C  SIMPL1 firing; simulation of blast pipe mid position
C  Materials: SiPh (6.35mm)/ Alu (1.8 mm)
C  CASE 299.15 C  Cm/Ch=N.A.
C  Specified erosion profile (option2) and surface temp.= C
C  Fri konveksjon og emissivitet=0.0 bakvegg
C  1-D model; fixed erosion rate; Start time=0.0; End time=100.0 s
   17    3   0.0     100.0               -1.0E-3   -.75
  .001287  0.05      527.67     0.0            00101000
   0.2       1.0
   0.5       5.0
   0.5      10.0
  -5.0     100.0
 1.187E+00      0.00
 1.163E+00      0.00
 1.140E+00      0.00
 1.116E+00      0.00
 1.098E+00      0.00
 1.080E+00      0.00
 1.063E+00      0.00
 1.045E+00      0.00
 1.027E+00      0.00
 1.009E+00      0.00
 9.911E-01      0.00
 9.733E-01      0.00
 9.554E-01      0.00
 9.376E-01      0.00
 9.197E-01      0.00
 9.019E-01      0.00
 8.840E-01      0.00
 8.661E-01      0.00
 1.187E+00      1.00
 1.163E+00      1.00
 1.140E+00      1.00
 1.116E+00      1.00
 1.098E+00      1.00
 1.080E+00      1.00
 1.063E+00      1.00
 1.045E+00      1.00
 1.027E+00      1.00
 1.009E+00      1.00
 9.911E-01      1.00
 9.733E-01      1.00
 9.554E-01      1.00
 9.376E-01      1.00
 9.197E-01      1.00
 9.019E-01      1.00
 8.840E-01      1.00
 8.661E-01      1.00
 1.187E+00      2.00
 1.163E+00      2.00
 1.140E+00      2.00
 1.116E+00      2.00
 1.098E+00      2.00
 1.080E+00      2.00
 1.063E+00      2.00
 1.045E+00      2.00
 1.027E+00      2.00
 1.009E+00      2.00
 9.911E-01      2.00
 9.733E-01      2.00
 9.554E-01      2.00
 9.376E-01      2.00
 9.197E-01      2.00
```

9.019E-01	2.00
8.840E-01	2.00
8.661E-01	2.00
1.187E+00	3.00
1.163E+00	3.00
1.140E+00	3.00
1.116E+00	3.00
1.098E+00	3.00
1.080E+00	3.00
1.063E+00	3.00
1.045E+00	3.00
1.027E+00	3.00
1.009E+00	3.00
9.911E-01	3.00
9.733E-01	3.00
9.554E-01	3.00
9.376E-01	3.00
9.197E-01	3.00
9.019E-01	3.00
8.840E-01	3.00
8.661E-01	3.00

02130 0	538.47	0.0	1.0	1.0
02100 0	538.47	0.0	0.0	1.0
02100 0	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01110 1	538.47	0.0	0.0	1.0
02130 0	538.47	0.0	1.0	1.0
02100 0	538.47	0.0	0.0	1.0
02100 0	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01110 1	538.47	0.0	0.0	1.0
02130 0	538.47	0.0	1.0	1.0
02100 0	538.47	0.0	0.0	1.0
02100 0	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0
01100 1	538.47	0.0	0.0	1.0

```
01100 1                538.47                0.0       0.0       1.0
01100 1                538.47                0.0       0.0       1.0
01100 1                538.47                0.0       0.0       1.0
01110 1                538.47                0.0       0.0       1.0
 2 1
 2  0.422    129.     81.09      0.6
 14000.     15400.     3.        0.417
  9.753E+8  36800.     3.        0.583
 8              0.0
      500.     1000.     1500.     2000.     3000.     4000.     6000.     6500.
    -2200.    -1900.    -1400.     -750.     1000.     2700.     5200.     6000.
   -4805.0
   -5293.0      0.0
      410.      .142   .000099                .6
      530.      .187   .000101                .6
      760.      .255   .000102                .6
     1160.      .307   .000103                .6
     1500.      .332   .000103                .6
     2000.      .346   .000103                .6
     3000.      .360   .000103                .6
     4000.      .364   .000103                .6
     5000.      .365   .000103                .6
1    6000.      .366   .000103                .6
      410.      .140   .000308                .6
      530.      .176   .000311                .6
      760.      .220   .000318                .6
     1160.      .264   .000329                .6
     1500.      .290   .000340                .6
     2000.      .307   .000352                .6
     3000.      .360   .000370                .6
     4000.      .364   .000420                .6
     5000.      .365   .000578                .6
1    6000.      .366   .000741                .6
      175.
      310.    0.2280  .0176000
      528.    0.2293  .0209000
      672.    0.2300  .0237000
      852.    0.2310  .0269000
     1032.    0.2350  .0302000
1    2032.    0.2350  .0302000
      0.00   4451.40      0.00
      1.00   4451.40     35.74
      2.00   4451.40     63.83
      3.00   4451.40     83.93
      4.00   4451.40    102.84
      5.00   4451.40    123.99
      5.00               0.15
+1  100.00               0.15
```

Appendix I: Input file used in G2DHeat

```
### Simulation case chapter 4
GRID
simpl1.b

SCALEMESH
.001

THICKNESS
1.

METRICS
AXI

INTERFACES
0

DEFMATERIAL
2
SIPH.b
ALU.b
1
SIPHpyr.b
2
1 1  1 11 1 27
2 1  1 11 27 37

PROBES_T
simpl1_T.d
.1
10
1 5 1
1 5 2
1 5 8
1 5 12
1 5 16
1 5 20
1 5 27
1 5 30
1 5 33
1 5 36

INOUTBC
2
  1  1  1 1 11  7 0
    2473.
  1  11  37 3 11  1 1
   26.   293.15
   0.05  293.15

TINIT
  299.15

M_EROSION
6
1.   0.000907796
2.   0.000713486
3.   0.000510540
4.   0.000480314
5.   0.000537210
100. 0.

IMPLICIT
0.0001 0.0 0.001 5.0
```

```
SAVESOL
smod_05.b

SAVERHO
rmod_05.b

INOUTBC
2
  1  1  1 1 11  2 1
   0.0
   0.6  473.15
  1  11  37 3 11  1 1
   26.   293.15
   0.05  293.15

IMPLICIT
0.0001 5.0 0.01 10.0

SAVESOL
smod_10.b

SAVERHO
rmod_10.b

IMPLICIT
0.0001 10.0 0.01 15.0

SAVESOL
smod_15.b

SAVERHO
rmod_15.b

IMPLICIT
0.0001 15.0 0.01 30.0

SAVESOL
smod_30.b

SAVERHO
rmod_30.b

IMPLICIT
0.0001 30.0 0.01 60.0

SAVESOL
smod_60.b

SAVERHO
rmod_60.b

IMPLICIT
0.0001 60.0 0.01 100.0

SAVESOL
smod_100.b

SAVERHO
rmod_100.b
END
```

Appendix J: The source code

```
 !Name:GlobaleVariable
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: This module contains the globale variables used throughout the
program
 !Variables of importance:  PROP=PROPERTIES  TC=TERMAL CONDUCTANCE  C=SPESIFIC HEAT
 MODULE GlobaleVariable
      IMPLICIT NONE
      !Variables for the implicit routine:
      INTEGER, DIMENSION(5,1000,50)          :: boundNU,boundSU,boundWU,boundEU
      DOUBLE PRECISION, DIMENSION(7,1000,50) :: boundN,boundS,boundW,boundE
      DOUBLE PRECISION                       :: konvergenskrit,precisionP
      DOUBLE PRECISION, DIMENSION(35000)     :: TEMPo
      !Variables for the Pyrolysis and recession:
      INTEGER                                ::
NIFACE,NBFACE,NMEK,Pyrini,MREG,pyrolyse,M_EROSION
      INTEGER,DIMENSION(50)                  :: AM_ANTPYR
      INTEGER, DIMENSION(35000)              :: ANTPYR,IFACE,IFACEBULK,MEKSTART
      DOUBLE PRECISION, DIMENSION(200)       :: MEKTIMES,MEK
      DOUBLE PRECISION, DIMENSION(20,35000)  ::
RHO,RHOO,RHOR,APYR,NPYR,TREAC,EPYR,VOLFRAC
      DOUBLE PRECISION                       :: IFACEVALUE
      DOUBLE PRECISION, DIMENSION(35000)     ::
Angles,XCP,YCP,VX,VY,VMX,VMY,MI,MJ,MPYR,CPG,DHPYR,MEKRHOTOT, &

MEKALFA,LENIBFACE,LENIBCELL,IBFACEX,IBFACEY,RHOFRAC,RHOOTOT, &
                                             RHORTOT,MEKTIME,MEKTIMED,MEKEND
      !Variables for input
      INTEGER, DIMENSION(20,30)              :: M_DATA,M_TEMP,M_TCI,M_TCJ,M_CP,M_RO
      DOUBLE PRECISION, DIMENSION(20,30)     ::
AM_TEMP,AM_TCI,AM_CP,AM_RO,AM_TCJ,AM_RHOO,AM_RHOR,AM_APYR,AM_EPYR, &

AM_NPYR,AM_TREAC,AM_CPG,AM_DHPYR,AM_VOLFRAC,AM_CPR, &
                                             AM_TCIR,AM_TCJR
      !Variables for time,grid,material properties ect.
      DOUBLE PRECISION                       ::
TIME,TSF,CFL,DTIME,TSTART,TSTOP,TIMEP,DTPROBE,TIMEPP,DTPROBEP
      INTEGER, DIMENSION(20)                 :: IMPT,I0PT,J0PT,IDPT,NPPT
      INTEGER                                :: NMESH,NPROBE,MODE,NREG,NIF,NBND
      INTEGER, DIMENSION(50)                 ::
NMP,NI,NJ,IMAIF,I0AIF,J0AIF,IDAIF,NPAIF,IMBIF,     &
                                             I0BIF,J0BIF,IDBIF,NPBIF
      DOUBLE PRECISION, DIMENSION(200)       :: AM_DATA,TPROBE
      DOUBLE PRECISION, DIMENSION(35000)     ::
XM,YM,SIX,SIY,SJX,SJY,AI,AJ,DI,DJ,VOL,RO,TCI,TCJ,C,RI,RJ,Su, &
                                             TEMP,DTEMP,DQ
      INTEGER, DIMENSION(200)                :: IMP,IP,JP,IMR,IR1,IR2,JR1,JR2,IMAT
      INTEGER, DIMENSION(35000)              :: MAT
      INTEGER, DIMENSION(200)                ::
IMBND,I0BND,J0BND,IDBND,NPBND,ITBND,IRADBND
      DOUBLE PRECISION, DIMENSION(50,500)    ::
ROBND,UBND,VISCBND,CPBND,PRBND,TRBND,HCEBND,ALBND
      DOUBLE PRECISION, DIMENSION(500)       ::
HCBND,TUBND,QBND,DTHCBND,DTTUBND,EMSBND,TRADBND
      DOUBLE PRECISION                       :: GCNT,RGAS,PRND,VISC,SCALEMESH
      INTEGER                                :: ISUP,NPROBEP
      DOUBLE PRECISION                       :: PTOT,TTOT,ATHROAT,PI
      CHARACTER                              :: MATRFIL*32

END MODULE GlobaleVariable

 !Name: G2DHeat
 !Author: Jørn Riise, *
 !Date: 30-06-2008
 !Description: Simulation program for heat transfer in solids.
```

```
      PROGRAM G2DHeat
      USE GlobaleVariable
      implicit none
      !Local variables
      INTEGER           :: ICOUNTP,ILOG,IPROBEP,IFIL,I,J,K,IMESH,NMPL,  &
                           NIL,NJL,LP,NSTEP,ISTEP,K1,istat,istatt,steps,MatID,IM, &
                           Istart,Iend,Jstart,Jend,IREG,t_start,t_stop,PREG
      DOUBLE PRECISION  ::
RTHROAT,TINITIAL,T_TEMP,T_TCI,T_CP,T_RO,T_TCJ,kildeSu,T_CPR, &

T_RHOO,T_RHOR,T_APYR,T_EPYR,T_NPYR,T_TREAC,T_CPG,T_DHPYR,T_VOLFRAC, &
                           T_TCIR,T_TCJR
      REAL              :: TEMPtmp,TCItmp,TCJtmp,RHOtmp,xmm,MPYRtmp,VXtmp,VYtmp
      CHARACTER EVENT*32,TEXT*32

      !Initialize variables
      NIF=0
      NPROBE=0
      NPROBEP=0
      ICOUNTP=0
      ILOG=0
      Su=Su*0
      SCALEMESH=1.
      PI=4.*ATAN(1.)
      !Pyrolysis:Decision variable for the interface
      IFACEVALUE=0
      !Identification parameters for the ablative material
      IFACEBULK=IFACEBULK*0
      !Pyrolysis gas flow in I- and J-direction
      MJ=MJ*0
      MI=MI*0

      !START LOOP FOR INPUT DATA
      call timer ( t_start)
      DO WHILE(.TRUE.)

      READ(5,'(A32)')EVENT
      IF(EVENT(1:1) .NE. '#')WRITE(6,'(A32)')EVENT

      IF(EVENT .EQ. 'END')THEN
        call timer ( t_stop )
        write(6,*) 'Elapsed CPU time = ', t_stop - t_start
        IF(NPROBEP.GT.0)THEN
          DO IPROBEP=1,NPROBEP
            IFIL=11+IPROBEP
            CLOSE(IFIL)
          ENDDO
          WRITE(6,*)'UPDATE MANUALLY NUMBER OF TIME POINTS ON DATAFILE'
          WRITE(6,*)'Number of time points, ICOUNTP =',ICOUNTP
        ENDIF
        CLOSE(10)
        STOP 'FINISHED'
      ENDIF

      IF(EVENT .EQ. 'SCALEMESH')THEN
        READ(5,*) SCALEMESH
      END IF

      IF(EVENT .EQ. 'GRID')THEN
        CALL GRID
      ENDIF

      IF(EVENT .EQ. 'METRICS')THEN
        READ(5,'(A32)')EVENT
        IF(EVENT .EQ. '2D')  MODE=0
        IF(EVENT .EQ. 'AXI') MODE=1
        CALL MTR
```

```
      ENDIF

!      IF(EVENT .EQ. 'THICKNESS')THEN
!        READ(5,*)DEPTH
!      ENDIF

      IF(EVENT.EQ.'LOGFILE')THEN
        ILOG=1
        READ(5,'(A32)') TEXT
        OPEN(10,FILE=TEXT,STATUS='UNKNOWN',FORM='FORMATTED')
      ENDIF

      IF(EVENT .EQ. 'DEFMAT')THEN
        READ(5,*)NREG
        DO 1 I=1,NREG
          READ(5,'(A32)')TEXT
          IF(TEXT .EQ. 'ALUMINIUM'    )IMAT(I)=1
          IF(TEXT .EQ. 'STEEL'        )IMAT(I)=2
          IF(TEXT .EQ. 'SIPH'         )IMAT(I)=3
          IF(TEXT .EQ. 'SIPH_COOL'    )IMAT(I)=11
          IF(TEXT .EQ. 'MOLYBDEN'     )IMAT(I)=4
          IF(TEXT .EQ. 'EPDM'         )IMAT(I)=5
          IF(TEXT .EQ. 'HOTGAS'       )IMAT(I)=6
          IF(TEXT .EQ. 'ARAMIDE-EPOXY')IMAT(I)=7
          IF(TEXT .EQ. 'PROPELLANT'   )IMAT(I)=8
          IF(TEXT .EQ. 'TITAN'        )IMAT(I)=9
          IF(TEXT .EQ. 'GRAPHITE'     )IMAT(I)=10
          IF(TEXT .EQ. 'CARBON'       )IMAT(I)=12
          IF(TEXT .EQ. 'TZM'          )IMAT(I)=13
          IF(TEXT .EQ. 'WL10'         )IMAT(I)=14
          IF(TEXT .EQ. 'GRAPHITE-IG11')IMAT(I)=15
          IF(TEXT .EQ. 'DLR_C-C/SiC')IMAT(I)=16

          READ(5,*)IMR(I),IR1(I),IR2(I),JR1(I),JR2(I)
    1   CONTINUE
        CALL DEFMATDATA        !BOR KANSKJE VARE EN EGEN FIL
        CALL MATERIALS
      ENDIF
      !****************************************
      !Initializing material properties from file.
      !****************************************
      IF(EVENT .EQ. 'DEFMATERIAL')THEN
         pyrolyse=0
         !Material properties
         READ(5,*)MREG
         DO I=1,MREG
           READ(5,'(A32)')TEXT
           OPEN(19,FILE=TEXT,STATUS='UNKNOWN',FORM='FORMATTED',ACTION='READ')
           istat=0
           steps=0
           READ(19,*)EVENT
           WRITE(6,*) 'Event: ',EVENT
           DO

READ(19,91,IOSTAT=istat)T_TEMP,T_CP,T_TCI,T_TCJ,T_RO,T_CPG,T_CPR,T_TCIR,T_TCJR,T_DH
PYR
              IF(istat.LT.0) EXIT
              steps=steps+1
              AM_TEMP(I,steps)=T_TEMP
              AM_CP(I,steps)=T_CP
              AM_TCI(I,steps)=T_TCI
              AM_TCJ(I,steps)=T_TCJ
              AM_RO(I,steps)=T_RO
              AM_CPG(I,steps)=T_CPG
              AM_TCIR(I,steps)=T_TCIR
              AM_TCJR(I,steps)=T_TCJR
              AM_DHPYR(I,steps)=T_DHPYR
              AM_CPR(I,steps)=T_CPR
```

```
          END DO
         AM_DATA(I)=steps
      91
FORMAT(G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,
TR1,G11.3,TR1,G11.3)
         CLOSE (19)
         END DO

         !Kinetic parameters for use in decomposition reactions
         READ(5,*)PREG
         DO I=1,PREG
           READ(5,'(A32)')TEXT
           OPEN(18,FILE=TEXT,STATUS='UNKNOWN',FORM='FORMATTED',ACTION='READ')
           istatt=0
           steps=0
           READ(18,*)EVENT
           WRITE(6,*)EVENT
           DO

READ(18,491,IOSTAT=istatt)T_RHOO,T_RHOR,T_APYR,T_EPYR,T_NPYR,T_TREAC,T_VOLFRAC
              IF(istatt.LT.0) EXIT
              steps=steps+1
              AM_RHOO(steps,I)=T_RHOO
              AM_RHOR(steps,I)=T_RHOR
              AM_APYR(steps,I)=T_APYR
              AM_EPYR(steps,I)=T_EPYR
              AM_NPYR(steps,I)=T_NPYR
              AM_TREAC(steps,I)=T_TREAC
              AM_VOLFRAC(steps,I)=T_VOLFRAC
           END DO
         AM_ANTPYR(I)=steps
     491 FORMAT(G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3,TR1,G11.3)
         CLOSE (18)
         pyrolyse=1
         Pyrini=1
         END DO
         !Initializing identification parameters
         READ(5,*)NREG
         DO IREG=1,NREG
          READ(5,*)MatID,IM,Istart,Iend,Jstart,Jend
          NMPL=NMP(IM)
          NIL = NI(IM)
          DO J=Jstart,Jend-1
           DO I=Istart,Iend-1
             K=I+NIL*(J-1)+NMPL
             MAT(K)=MatID
             !(1 ->Ablative, -1 -> backup)
             IF(MatID.GT.PREG) THEN
              IFACEBULK(K)=-1
             ELSE
              IFACEBULK(K)=1
             END IF
           END DO
          END DO
         END DO

       END IF

       IF(EVENT .EQ. 'INTERFACES')THEN
         READ(5,*)NIF
         IF(NIF.NE.0)THEN
           DO 2 I=1,NIF
             READ(5,*)IMAIF(I),I0AIF(I),J0AIF(I),IDAIF(I),NPAIF(I)
             READ(5,*)IMBIF(I),I0BIF(I),J0BIF(I),IDBIF(I),NPBIF(I)
     2     CONTINUE
         END IF
       END IF
```

```
      IF(EVENT .EQ. 'INOUTBC')THEN
        READ(5,*)NBND
        IF(NBND.NE.0)THEN
          DO 3 I=1,NBND
            READ(5,*)IMBND(I),I0BND(I),J0BND(I),IDBND(I),NPBND(I),      &
                     ITBND(I),IRADBND(I)
            IF(ITBND(I).EQ.1) READ(5,*)HCBND(I),TUBND(I)        !
Varmeovergangstall + Gasstemperatur
            IF(ITBND(I).EQ.2)READ(5,*)QBND(I)                   ! Varmefluks
            IF(ITBND(I).EQ.3)THEN                               !
              READ(5,'(A32)')TEXT
              OPEN(1,FILE=TEXT,FORM='FORMATTED',STATUS='UNKNOWN')
              CALL GASDATA(1,I)
            ENDIF
            IF(ITBND(I).EQ.4)THEN
              READ(5,'(A32)')TEXT
              OPEN(1,FILE=TEXT,FORM='FORMATTED',STATUS='UNKNOWN')
              CALL GASDATA(2,I)
            ENDIF
            IF(ITBND(I).EQ.5)READ(5,*)HCBND(I),TUBND(I),DTHCBND(I),DTTUBND(I)
!V.overg. tall+gasstemp+rampe v.ovg.tall+rampe Tgass
            IF(IRADBND(I).EQ.1)READ(5,*)EMSBND(I),TRADBND(I)     ! Emissivitet +
strålingstemperatur

!     1D isentropic nozzle flow used as input to heat coeff. computations
            IF(ITBND(I).EQ.6)THEN                               ! Isentropisk
dysestrøm
!       subsonic,   isup=0
!       supersonic, isup=1

              READ(5,*)PTOT,TTOT,RTHROAT,ISUP
              ATHROAT=PI*RTHROAT**2
              CALL GASDATA(3,I)
            ENDIF
            !Constant surface temperature
            IF(ITBND(I).EQ.7)READ(5,*)TUBND(I)
    3     CONTINUE
        END IF
      END IF

      IF(EVENT .EQ. 'GASDATA')THEN
        READ(5,*)GCNT,RGAS,PRND,VISC
      END IF

      IF(EVENT .EQ. 'TINIT')THEN
        READ(5,*)TINITIAL
          DO 4 IMESH=1,NMESH
          NMPL=NMP(IMESH)
          NIL = NI(IMESH)
          NJL = NJ(IMESH)
          DO 5 J=1,NJL-1
          DO 5 I=1,NIL-1
            K=I+NIL*(J-1)+NMPL
            TEMP(K)=TINITIAL

    5     CONTINUE
    4   CONTINUE
      ENDIF

      IF(EVENT .EQ. 'PROBES_T')THEN
        READ(5,'(A32)')TEXT
        OPEN(11,FILE=TEXT,FORM='FORMATTED',STATUS='UNKNOWN')
        READ(5,*) DTPROBE
        READ(5,*)NPROBE
        DO LP=1,NPROBE
          READ(5,*) IMP(LP),IP(LP),JP(LP)
        ENDDO
      ENDIF
```

```
    IF(EVENT .EQ. 'PROBES_PT')THEN
      READ(5,*) DTPROBEP
      READ(5,*)NPROBEP
      DO I=1,NPROBEP
        READ(5,'(A32)')TEXT
        IFIL=11+I
        OPEN(IFIL,FILE=TEXT,FORM='FORMATTED',STATUS='UNKNOWN')
        WRITE(IFIL,*) 'TITLE = "HEAT 2D data"'
        WRITE(IFIL,*) 'VARIABLES = "YM", "TIME", "TEMP"'
        READ(5,*)IMPT(I),I0PT(I),J0PT(I),IDPT(I),NPPT(I)
        WRITE(IFIL,*) 'ZONE I=',NPPT(I)-1,', J=',0,', F=POINT'
      END DO
    ENDIF

    IF(EVENT .EQ. 'CFL')THEN
      READ(5,*)CFL,TSTART,DTIME,TSTOP
      TIME=TSTART
      TIMEP=TSTART
      TIMEPP=TSTART
    ENDIF

    IF(EVENT .EQ. 'TRANSIENT')THEN
      READ(5,*)NSTEP
      DO 8 ISTEP=1,NSTEP
        CALL STEP(ISTEP)
        TIME=TIME+TSF*CFL
        !WRITE(6,*)ISTEP,TIME,TSF*CFL
        IF(NPROBE .GT. 0)THEN
          IF(TIME.GE.TIMEP)THEN
            TIMEP=TIMEP+DTPROBE
            DO LP=1,NPROBE
              I=IP(LP)
              J=JP(LP)
              IMESH=IMP(LP)
              K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
              TPROBE(LP+1)=TEMP(K1)
            END DO
            TPROBE(1)=TIME
            WRITE(11,555)(TPROBE(I),I=1,NPROBE+1)
          ENDIF
        ENDIF

        IF(NPROBEP .GT. 0)THEN
          IF(TIME.GE.TIMEPP)THEN
            TIMEPP=TIMEPP+DTPROBEP
            CALL WRITE_PROBESP(TIME)
           ICOUNTP=ICOUNTP+1
          ENDIF
        ENDIF
555     FORMAT(F10.3,200F9.4)
        IF(TIME.GT.TSTOP) GOTO 99
  8   CONTINUE
      IF(ILOG.EQ.1)WRITE(10,*)                                          &
         'INCREASE NUMBER OF ITERATIONS, TIME,TSTOP=',TIME,TSTOP
      WRITE(6,*)'INCREASE NUMBER OF ITERATIONS, TIME,TSTOP=',TIME,TSTOP
      STOP
 99   CONTINUE
    ENDIF

    !*****************************
    !Initializing recession rates.
    !*****************************
    IF(EVENT .EQ. 'M_EROSION')THEN
      READ(5,*)NMEK
      DO I=1,NMEK
      READ(5,*)MEKTIMES(I),MEK(I)
      END DO
```

```
      M_EROSION=1
    END IF


    !**************************
    !Implicit solution routine.
    !**************************
    IF (EVENT .EQ. 'IMPLICIT') THEN
     READ(5,*)konvergenskrit,TSTART,DTIME,TSTOP
     !Initializing time parameters
     TIME=TSTART
     TIMEP=TSTART
     TIMEPP=TSTART

     !Initializing interfaces and boundaries
     CALL INIT_IMPLICIT

     !Initializing boundary conditions
     CALL BORDERS_IMPLICIT

     !Initializing Pyrolysis parameters
     IF(pyrolyse.EQ.1) CALL INIT_PYROLYSIS

     !Solution routine
     DO

        CALL IMPLICIT_
        TIME=TIME+DTIME

        !Saving data
        IF(NPROBE .GT. 0)THEN
          IF(TIME.GE.TIMEP)THEN
            TIMEP=TIMEP+DTPROBE
            DO LP=1,NPROBE
              I=IP(LP)
              J=JP(LP)
              IMESH=IMP(LP)
              K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
              TPROBE(LP+1)=TEMP(K1)
            END DO
            TPROBE(1)=TIME
            WRITE(11,551)(TPROBE(I),I=1,NPROBE+1)
          ENDIF
        ENDIF

        IF(NPROBEP .GT. 0)THEN
          IF(TIME.GE.TIMEPP)THEN
            TIMEPP=TIMEPP+DTPROBEP
            CALL WRITE_PROBESP(TIME)
           ICOUNTP=ICOUNTP+1
          ENDIF
        ENDIF
551     FORMAT(F10.3,200F9.4)

        IF (TIME.GE.TSTOP) EXIT

     END DO
    END IF

    !************************************************
    !Initializing selection parameter for the interface.
    !************************************************
    IF (EVENT .EQ. 'PYROLYSIS') THEN
      READ(5,*)IFACEVALUE
    END IF

    !**********************************
    !Initializing heat source/sink term.
```

```
   !**********************************
   IF(EVENT .EQ. 'SOURCE')THEN
   READ(5,*)NREG
      DO IREG=1,NREG
       READ(5,*)IM,Istart,Iend,Jstart,Jend,kildeSu
       NMPL=NMP(IM)
       NIL = NI(IM)
       DO J=Jstart,Jend-1
        DO I=Istart,Iend-1
          K=I+NIL*(J-1)+NMPL
          Su(K)=kildeSu
        END DO
       END DO
      END DO


   END IF

   IF(EVENT .EQ. 'READSOL')THEN
     READ(5,'(A32)')TEXT
     OPEN(UNIT=8,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
     DO 23 IMESH=1,NMESH
       NMPL=NMP(IMESH)
       NIL = NI(IMESH)
       NJL = NJ(IMESH)
       DO 24 J=1,NJL-1
       DO 24 I=1,NIL-1
         K=I+NIL*(J-1)+NMPL
         READ(8) TEMPtmp,TCItmp,TCJtmp,xmm
         TEMP(K)=REAL(TEMPtmp,8)
         TCI(K)=REAL(TCItmp,8)/1000.
         TCJ(K)=REAL(TCJtmp,8)/1000.
         mat(k)=ifix(xmm)
24     CONTINUE
23   CONTINUE
     CLOSE(8)
   ENDIF


   IF (EVENT .EQ. 'SAVEMATR') THEN
      read(5,'(A32)')MATRFIL
      CALL MATRSAVE
   END IF

   IF(EVENT .EQ. 'SAVESOL')THEN
     READ(5,'(A32)')TEXT
     OPEN(UNIT=9,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
     DO 13 IMESH=1,NMESH
       NMPL=NMP(IMESH)
       NIL = NI(IMESH)
       NJL = NJ(IMESH)
       DO 14 J=1,NJL-1
       DO 14 I=1,NIL-1
         K=I+NIL*(J-1)+NMPL
         xmm=REAL(mat(k))
         TEMPtmp=REAL(TEMP(K),4)
         TCItmp=REAL(TCI(K),4)*1000.
         TCJtmp=REAL(TCJ(K),4)*1000.
         WRITE(9)TEMPtmp,TCItmp,TCJtmp,xmm
14     CONTINUE
13   CONTINUE
   ENDIF

   !**************************************************
   !Saving the solution:DENSITIES, TEMPERATURES and Mpyr.
   !**************************************************
   IF(EVENT .EQ. 'SAVERHO')THEN
     READ(5,'(A32)')TEXT
```

```
  OPEN(UNIT=15,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
  DO IMESH=1,NMESH
    NMPL=NMP(IMESH)
    NIL = NI(IMESH)
    NJL = NJ(IMESH)
    DO J=1,NJL-1
     DO I=1,NIL-1
      K=I+NIL*(J-1)+NMPL
      xmm=REAL(mat(k))
      RHOtmp=REAL(RO(K),4)
      TEMPtmp=REAL(TEMP(K),4)
      MPYRtmp=REAL(((-1)*(MPYR(K)/VOL(K))),4)
      WRITE(15)RHOtmp,TEMPtmp,MPYRtmp,xmm
     END DO
    END DO
   END DO
END IF

!*****************************************************************
!Saving the solution:VECTOR DIRECTIONS[RO,RU](Techplot) and Mpyr.
!*****************************************************************
IF(EVENT .EQ. 'SAVEVECTOR')THEN
  READ(5,'(A32)')TEXT
  OPEN(UNIT=15,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
  DO IMESH=1,NMESH
    NMPL=NMP(IMESH)
    NIL = NI(IMESH)
    NJL = NJ(IMESH)
    DO J=1,NJL-1
     DO I=1,NIL-1
      K=I+NIL*(J-1)+NMPL
      xmm=REAL(mat(k))
      VXtmp=REAL(VX(K),4)
      VYtmp=REAL(VY(K),4)
      MPYRtmp=REAL(MPYR(K),4)
      WRITE(15)VXtmp,VYtmp,MPYRtmp,xmm
     END DO
    END DO
   END DO
END IF

!*****************************************************************
!Saving the solution:VECTOR DIRECTIONS[RO,RU](Techplot) and Mpyr.
!*****************************************************************
IF(EVENT .EQ. 'SAVEM_EROSION')THEN
  READ(5,'(A32)')TEXT
  OPEN(UNIT=15,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
  DO IMESH=1,NMESH
    NMPL=NMP(IMESH)
    NIL = NI(IMESH)
    NJL = NJ(IMESH)
    DO J=1,NJL-1
     DO I=1,NIL-1
      K=I+NIL*(J-1)+NMPL
      xmm=REAL(mat(k))
      VXtmp=REAL(VMX(K),4)
      VYtmp=REAL(VMY(K),4)
      MPYRtmp=REAL(MPYR(K),4)
      WRITE(15)VXtmp,VYtmp,MPYRtmp,xmm
     END DO
    END DO
   END DO
END IF

!*********************************************************
!Saving the solution: Gas flow I- and J-direction and Mpyr.
!*********************************************************
IF(EVENT .EQ. 'SAVEGAS')THEN
```

```
      READ(5,'(A32)')TEXT
      OPEN(UNIT=15,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
      DO IMESH=1,NMESH
        NMPL=NMP(IMESH)
        NIL = NI(IMESH)
        NJL = NJ(IMESH)
        DO J=1,NJL-1
         DO I=1,NIL-1
          K=I+NIL*(J-1)+NMPL
          xmm=REAL(mat(k))
          VXtmp=REAL(MI(K),4)
          VYtmp=REAL(MJ(K),4)
          MPYRtmp=REAL(MPYR(K),4)
          WRITE(15)VXtmp,VYtmp,MPYRtmp,xmm
         END DO
        END DO
       END DO
    END IF



    ENDDO

    CLOSE(11)
    STOP
    END !End input routine

    SUBROUTINE WRITE_PROBESP(tid)
    USE GlobaleVariable
    implicit none
    !Local variables
    INTEGER          :: ID,JD,IPROBEP,IFIL,IMS,I0S,J0S,IDS,NPIS,IPP,I, &
                        J,K,K1,K2,K3,K4
    DOUBLE PRECISION :: YC,tid!xc,
    DIMENSION ID(4),JD(4)
    DATA ID / 1, 0,-1, 0/
    DATA JD / 0, 1, 0,-1/
    DO 1 IPROBEP=1,NPROBEP
      IFIL=11+IPROBEP
      IMS=IMPT(IPROBEP)
      I0S=I0PT(IPROBEP)
      J0S=J0PT(IPROBEP)
      IDS=IDPT(IPROBEP)
      NPIS=NPPT(IPROBEP)
      DO 2 IPP=1,NPIS-1
        I=I0S+ID(IDS)*(IPP-1)
        J=J0S+JD(IDS)*(IPP-1)
        K=I+NI(IMS)*(J-1)+NMP(IMS)
        K1=I  +NI(IMS)*(J-1)+NMP(IMS)
        K2=I+1+NI(IMS)*(J-1)+NMP(IMS)
        K3=I  +NI(IMS)*(J-0)+NMP(IMS)
        K4=I+1+NI(IMS)*(J-0)+NMP(IMS)
        !XC=.25*(XM(K1)+XM(K2)+XM(K3)+XM(K4))
        YC=.25*(YM(K1)+YM(K2)+YM(K3)+YM(K4))
        WRITE(IFIL,*) REAL(YC*1000.,4),REAL(tid,4),REAL(TEMP(K))
2   CONTINUE
1 CONTINUE
    RETURN
    END

    SUBROUTINE GRID
    USE GlobaleVariable
    implicit none

    !Local variables
    INTEGER                    :: I,NTOT,NMESHtmp
    INTEGER, DIMENSION (50)    :: NMPtmp,NItmp,NJtmp
    REAL, DIMENSION (35000)    :: Xtmp,Ytmp
```

```
      CHARACTER TEXT*32

      READ(5,'(A32)') TEXT
      OPEN(7,FILE=TEXT,FORM='UNFORMATTED',STATUS='UNKNOWN')
      READ(7) NMESHtmp
      READ(7) (NMPtmp(I),NItmp(I),NJtmp(I),I=1,NMESHtmp)
      NTOT=NMPtmp(NMESHtmp)+NItmp(NMESHtmp)*NJtmp(NMESHtmp)
      READ(7)(Xtmp(I),Ytmp(I),I=1,NTOT)
      DO I=1,NTOT
        XM(I)=REAL(Xtmp(I),8)
        YM(I)=REAL(Ytmp(I),8)
      END DO
      CLOSE(7)
      NMESH=NMESHtmp
      DO I=1,NMESHtmp
         NMP(I)=NMPtmp(I)
         NI(I)=NItmp(I)
         NJ(I)=NJtmp(I)
      END DO
      RETURN
      END

      SUBROUTINE MTR
      USE GlobaleVariable
      implicit none

      !Local variables
      INTEGER          :: IMESH,I,J,K1,K2,K3,K4,IMMIN,IMIN,JMIN,IMMAX,  &
                          IMAX,JMAX
      DOUBLE PRECISION :: VMIN,VMAX
      VMIN=+1.E20
      VMAX=-1.E20
      DO 1 IMESH=1,NMESH
        DO 10 J=1,NJ(IMESH)                    ! scale grid
        DO 10 I=1,NI(IMESH)
          K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
          XM(K1)=XM(K1)*SCALEMESH
          YM(K1)=YM(K1)*SCALEMESH
 10   CONTINUE

      DO 2 J=1,NJ(IMESH)-1                     ! 2D METRICS
      DO 2 I=1,NI(IMESH)
        K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
        K3=K1+NI(IMESH)
        SIX(K1)= YM(K3)-YM(K1)
        SIY(K1)=-XM(K3)+XM(K1)
        AI(K1)=SQRT(SIX(K1)**2+SIY(K1)**2)
 2    CONTINUE
      DO 3 J=1,NJ(IMESH)
      DO 3 I=1,NI(IMESH)-1
        K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
        K2=K1+1
        SJX(K1)=-YM(K2)+YM(K1)
        SJY(K1)= XM(K2)-XM(K1)
        AJ(K1)=SQRT(SJX(K1)**2+SJY(K1)**2)
 3    CONTINUE

      DO 4 J=1,NJ(IMESH)-1                     ! DI,DJ BASED
      DO 4 I=1,NI(IMESH)-1                     ! ON 2D METRICS
        K1=I+NI(IMESH)*(J-1)+NMP(IMESH)          !+ volumes
        K2=K1+1
        K3=K1+NI(IMESH)
        K4=K3+1
        DJ(K1)=0.5*(SQRT(SIX(K1)**2+SIY(K1)**2)+SQRT(SIX(K2)**2+SIY(K2)**2))
        DI(K1)=0.5*(SQRT(SJX(K1)**2+SJY(K1)**2)+SQRT(SJX(K3)**2+SJY(K3)**2))

        IF(MODE.EQ.1)THEN
!          VOL(K1)=DEPTH*1./6.*((XM(K4)-XM(K1))*(YM(K3)**2-YM(K2)**2)  &
```

```
            VOL(K1)=1./6.*((XM(K4)-XM(K1))*(YM(K3)**2-YM(K2)**2)         &
                          -(XM(K3)-XM(K2))*(YM(K4)**2-YM(K1)**2)         &
                          +(XM(K4)*YM(K4)-XM(K1)*YM(K1))*(YM(K3)-YM(K2)) &
                          -(XM(K3)*YM(K3)-XM(K2)*YM(K2))*(YM(K4)-YM(K1)))
          ELSE
!           VOL(K1)=DEPTH*.5*((XM(K4)-XM(K1))*(YM(K3)-YM(K2))            &
            VOL(K1)=0.5*((XM(K4)-XM(K1))*(YM(K3)-YM(K2))                 &
                        -(XM(K3)-XM(K2))*(YM(K4)-YM(K1)))
          ENDIF
          IF(VOL(K1) .LT. VMIN)THEN
            VMIN=VOL(K1)
            IMMIN=IMESH
            IMIN=I
            JMIN=J
          ENDIF
          IF(VOL(K1) .GT. VMAX)THEN
            VMAX=VOL(K1)
            IMMAX=IMESH
            IMAX=I
            JMAX=J
          ENDIF

    4   CONTINUE

        IF(MODE.EQ.1)THEN                          ! AXI- SYMMETRICAL
          DO 5 J=1,NJ(IMESH)-1                     ! OPTION
          DO 5 I=1,NI(IMESH)
            K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
            K3=K1+NI(IMESH)
            SIX(K1)= .5*(YM(K1)+YM(K3))*(YM(K3)-YM(K1))
            SIY(K1)=-.5*(YM(K1)+YM(K3))*(XM(K3)-XM(K1))
            AI(K1)=SQRT(SIX(K1)**2+SIY(K1)**2)
    5     CONTINUE
          DO 6 J=1,NJ(IMESH)
          DO 6 I=1,NI(IMESH)-1
            K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
            K2=K1+1
            SJX(K1)=-.5*(YM(K1)+YM(K2))*(YM(K2)-YM(K1))
            SJY(K1)= .5*(YM(K1)+YM(K2))*(XM(K2)-XM(K1))
            AJ(K1)=SQRT(SJX(K1)**2+SJY(K1)**2)
    6     CONTINUE
        ENDIF

    1 CONTINUE
      WRITE(6,*)' MIN VOLUME:  ',IMMIN,IMIN,JMIN,VMIN
      WRITE(6,*)' MAX VOLUME:  ',IMMAX,IMAX,JMAX,VMAX
      RETURN
      END

      SUBROUTINE GASDATA(IOPT,IBOUND)
      USE GlobaleVariable
      implicit none
      INTEGER          :: IOPT,IBOUND

      !Local variables
      INTEGER          :: ID,JD,IBND,IM,I0,IPP,J0,IDR,NP,K0,KD,    &
                          K1,K2,I
      DOUBLE PRECISION :: ERRMAX,EPS,CPGAS,RFACT,FACT,WGFLOW,XPOS,    &
                          RADIUS,AREA,ARATIO,AM,AMN,DIFF,TC,PC,ROC,VEL, &
                          RE,F,ST

      DIMENSION ID(4),JD(4)
      DATA ID / 1, 0,-1, 0/
      DATA JD / 0, 1, 0,-1/

      PI=4.*ATAN(1.)
      ERRMAX=1.D-06
      EPS=1.D-06
```

```
  IF(IOPT.EQ.1)THEN
    WRITE(6,*)' IP     RO        V      VISC*1.E06   CP        PR ',&
            '       LENGTH'
    DO 1 IBND=1,NBND
      DO 2 IPP=1,NPBND(IBND)-1
        READ(1,*)ROBND(IBND,IPP), UBND(IBND,IPP),VISCBND(IBND,IPP),    &
                 CPBND(IBND,IPP),PRBND(IBND,IPP),  ALBND(IBND,IPP)
        WRITE(6,100)IPP,ROBND(IBND,IPP),UBND(IBND,IPP),                &
                    VISCBND(IBND,IPP)*1.E06,CPBND(IBND,IPP),          &
                    PRBND(IBND,IPP),ALBND(IBND,IPP)
2     CONTINUE
1   CONTINUE
  ENDIF
  IF(IOPT.EQ.2)THEN
    WRITE(6,*)' IP     TR       HC'
    IBND=IBOUND
  ENDIF

  !1D isentropic nozzle flow(input)
  IF(IOPT.EQ.3)THEN
    WRITE(6,*)' XPOS   AM        PC     TR       HC'
    CPGAS=GCNT*RGAS/(GCNT-1.)
    RFACT=PRND**.33333333
    FACT=.5*(GCNT+1.)/(GCNT-1.)
    WGFLOW=(2./(GCNT+1.))**FACT*SQRT(GCNT/RGAS/TTOT)*PTOT*ATHROAT
    IBND=IBOUND
    IM =IMBND(IBND)
    I0 =I0BND(IBND)
    J0 =J0BND(IBND)
    IDR=IDBND(IBND)
    NP =NPBND(IBND)
    K0 =I0         +NI(IM)*(J0         -1)+NMP(IM)
    !K0C=I0+IC(IDR)+NI(IM)*(J0+JC(IDR)-1)+NMP(IM)
    !K0I=I0         +NI(IM)*(J0+JC(IDR)-1)+NMP(IM)
    !K0J=I0+IC(IDR)+NI(IM)*(J0         -1)+NMP(IM)
    KD= ID(IDR)+NI(IM)* JD(IDR)
    DO 5 IPP=1,NP-1
      K1=K0+KD*(IPP-1)
      K2=K0+KD* IPP
      XPOS=.5*(XM(K1)+XM(K2))
      RADIUS=.5*(YM(K1)+YM(K2))
      AREA=PI*RADIUS**2
      ARATIO=AREA/ATHROAT
      IF(ARATIO.LT.1.0+EPS)THEN
        AM=1.0
        GOTO 10
      ENDIF

      !Subsonic
      IF(ISUP.EQ.0)THEN
        am=.5
        DO 6 I=1,200
          amn=(1.+.5*(gcnt-1.)*am**2)**fact*sqrt(rgas*ttot/gcnt)*   &
              wgflow/ptot/area
          diff=abs(amn-am)/am
          am=amn
          IF(DIFF.LT.ERRMAX)GOTO 7
6       CONTINUE
       WRITE(6,*)'ITERATION FAILED, SUBSONIC'
7       CONTINUE
      ENDIF

      !Supersonic
      IF(ISUP.EQ.1)THEN
        am=1.5
        DO 8 I=1,200
          amn=sqrt(2./(gcnt-1.)*((am/wgflow*ptot*area*              &
```

```
                  sqrt(gcnt/rgas/ttot))**(1./fact)-1.))
              diff=abs(amn-am)/am
              am=amn
             IF(DIFF.LT.ERRMAX)GOTO 9
    8        CONTINUE
            WRITE(6,*)'ITERATION FAILED, SUPERSONIC'
    9        CONTINUE
           ENDIF
   10      CONTINUE

           tc=ttot/(1.+.5*(gcnt-1.)*am**2)
           pc=ptot/(1.+.5*(gcnt-1.)*am**2)**(gcnt/(gcnt-1.))
           roc=pc/rgas/tc
           vel=wgflow/area/roc

           !Viscosity power law from SPP
           VISCBND(IBND,IPP)=VISC*(TC/TTOT)**.6728
           TRBND(IBND,IPP)=RFACT*TTOT+(1.-RFACT)*TC
           TRBND(IBND,IPP)=TRBND(IBND,IPP)-273.
           RE=WGFLOW*2/(PI*RADIUS*VISCBND(IBND,IPP))
           F=0.0791/(RE**0.25)
           ST=F/2/(1+1.99*RE**(-0.125)*(PRND-1))
           HCEBND(IBND,IPP)=ROC*VEL*ST*CPGAS

           WRITE(6,*)XPOS*1000.,AM,PC/1.E6,TRBND(IBND,IPP),HCEBND(IBND,IPP)
    5    CONTINUE
       ENDIF

       CLOSE(1)
  100  FORMAT(I4,6F10.3)
       RETURN
       END

       SUBROUTINE LTSP
       USE GlobaleVariable
       implicit none
       !Local variables
       INTEGER          :: IMESH,I,J,K1,K2,K3,printer
       DOUBLE PRECISION :: TMAX,TS
       printer=1
       TS=DTIME
       DO IMESH=1,NMESH
         DO J=1,NJ(IMESH)-1
         DO I=1,NI(IMESH)-1
           K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
           K2=K1+1
           K3=K1+NI(IMESH)

TMAX=RO(K1)*C(K1)*VOL(K1)/((AI(K1)/RI(K1))+(AI(K2)/RI(K2))+(AJ(K1)/RJ(K1))+(AJ(K3)/
RJ(K3)))
           IF (TS>TMAX) THEN
             TS=TMAX
             IF (printer.EQ.1) THEN
              WRITE(6,*) 'Time step has been changed, dt= ',TS
              printer=0
             END IF
           END IF
         END DO
         END DO
       END DO
       TSF=TS
       RETURN
       END

 !Name:IMPLICIT
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: Solves the governing equation using TDMA line-by-line 2D.
```

```
       SUBROUTINE IMPLICIT_
       USE GlobaleVariable
       implicit none

       !Local variables
       INTEGER                              :: IMESH,K1,K2,K3,I,pos,J,K4,K5,dummy
       DOUBLE PRECISION, DIMENSION(1000)    :: ma,mb,mc,md,P,Q
       DOUBLE PRECISION                     ::
tempP,tempPo,tempE,tempN,tempS,aw,ae,an,as,ap,kildeP,kildeU,apo, &

denomiator,tempOLD,SIGMA,RADIATION,EMMIS
       LOGICAL                              :: ferdig

       SIGMA=5.67*1.E-08
       !Initializing convergence variable
       ferdig = .FALSE.

       !Pyrolysis decomposition reactions
       IF (pyrolyse.EQ.1) THEN
        CALL PYROLYSIS
        CALL CONTINUITY
       END IF

       !Temporary storage of temperatures
       TEMPo(:)=TEMP(:)

       DO

        dummy=1

        IF(ferdig) EXIT

        ferdig=.TRUE.

        !Updating material properties
        IF (pyrolyse.EQ.0) THEN
        CALL PICKMDATA
        END IF
        CALL RESMAT


       !Updating shadow cells
       CALL UPDATE_IMPLICIT

       !Sweeping(from South to North) in j-direction
       DO IMESH=1,NMESH
       DO J=1,NJ(IMESH)-1
             DO I=1,NI(IMESH)-1
                        kildeP=0
                        kildeU=0
                        pos=I+1
                        !Coordinates in 1D-vector
                        K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
                        K2=K1+1
                        K3=K1+NI(IMESH)
                        K4=K1-NI(IMESH)
                        IF (J.EQ.1) THEN
                           !South edge
                           an=(AJ(K3)/RJ(K3))+(CPG(K1)*MAX(0.,(-MJ(K3))))
                           as=boundS(3,I,IMESH)

kildeU=boundS(5,I,IMESH)+(Su(K1)*VOL(K1))+(MPYR(K1)*DHPYR(K1))
                           !kildeP=0
                           RADIATION=boundS(6,I,IMESH)
                           EMMIS=boundS(7,I,IMESH)
                           IF (RADIATION.GT.0) THEN
```

```
                              kildeP=(-
1)*boundS(1,I,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundS(1,I,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF

                              tempN=TEMP(K3)
                              tempS=boundS(4,I,IMESH)
                              tempPo=TEMPo(K1)
                           ELSE IF (J.EQ.(NJ(IMESH)-1)) THEN
                              !North edge
                              an=boundN(3,I,IMESH)
                              as=(AJ(K1)/RJ(K1))+(CPG(K1)*MAX(0.,MJ(K1)))

kildeU=boundN(5,I,IMESH)+(Su(K1)*VOL(K1))+(MPYR(K1)*DHPYR(K1))
                              RADIATION=boundN(6,I,IMESH)
                              EMMIS=boundN(7,I,IMESH)
                              IF (RADIATION.GT.0) THEN
                               kildeP=(-
1)*boundN(1,I,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundN(1,I,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF

                              tempN=boundN(4,I,IMESH)
                              tempS=TEMP(K4)
                              tempPo=TEMPo(K1)

                           ELSE
                              !Internal cell volumes
                              an=(AJ(K3)/RJ(K3))+(CPG(K1)*MAX(0.,(-MJ(K3))))
                              as=(AJ(K1)/RJ(K1))+(CPG(K1)*MAX(0.,MJ(K1)))
                              kildeU=(Su(K1)*VOL(K1))+(MPYR(K1)*DHPYR(K1))

                              tempN=TEMP(K3)
                              tempS=TEMP(K4)
                              tempPo=TEMPo(K1)
                           END IF

                           IF (I.EQ.1) THEN
                              !West edge
                              aw=boundW(3,J,IMESH)
                              ae=(AI(K2)/RI(K2))+(CPG(K1)*MAX(0.,(-MI(K2))))
                              kildeU=kildeU+boundW(5,J,IMESH)

                              RADIATION=boundW(6,J,IMESH)
                              EMMIS=boundW(7,J,IMESH)
                              IF (RADIATION.GT.0) THEN
                               kildeP=(-
1)*boundW(1,J,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundW(1,J,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF

                           ELSE IF (I.EQ.(NI(IMESH)-1)) THEN
                              !East edge
                              aw=(AI(K1)/RI(K1))+(CPG(K1)*MAX(0.,MI(K1)))
                              ae=boundE(3,J,IMESH)
                              kildeU=kildeU+boundE(5,J,IMESH)

                              RADIATION=boundE(6,J,IMESH)
                              EMMIS=boundE(7,J,IMESH)
                              IF (RADIATION.GT.0) THEN
                               kildeP=(-
1)*boundE(1,J,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundE(1,J,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF
```

```
                ELSE
                   aw=(AI(K1)/RI(K1))+(CPG(K1)*MAX(0.,MI(K1)))
                   ae=(AI(K2)/RI(K2))+(CPG(K1)*MAX(0.,(-MI(K2))))
                END IF

                !Cell + source
                apo=(RO(K1)*C(K1)*VOL(K1))/DTIME
                ap=apo+aw+ae+an+as-kildeP


                !TDMA Part 1 (Forward substitution)

                !Initial temperature West
                IF (I.EQ.1) THEN
                   P(1)=0
                   Q(1)=boundW(4,J,IMESH)
                END IF

                !Internal coefficients
                ma(pos)=ap
                mb(pos)=ae
                mc(pos)=aw
                md(pos)=(an*tempN)+(as*tempS)+(apo*tempPo)+kildeU

                !Forward substitution
                denomiator=ma(pos)-(mc(pos)*P(pos-1))
                P(pos)=mb(pos)/denomiator
                Q(pos)=((mc(pos)*Q(pos-1))+md(pos))/denomiator

          END DO !End i and TDMA Part 11


          !TDMA Part 2 (Backward substitution)
          DO I=NI(IMESH)-1, 1, -1
              pos=I+1
              !Coordinates in 1D-vector
              K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
              K2=K1+1
              K3=K1+NI(IMESH)
              K4=K1-NI(IMESH)

              !Initial temperature East
              IF(I.EQ.(NI(IMESH)-1)) THEN
              tempE=boundE(4,J,IMESH)
              ELSE
              tempE=temp(K2)
              END IF

              !Backward substitution
              tempP=(P(pos)*tempE)+Q(pos)
              temp(K1)=tempP

          END DO !End TDMA Part 2

          END DO !End j
END DO !End imesh

!Updating shadow cells
CALL UPDATE_IMPLICIT

!Sweeping(from West to East) in i-direction
DO IMESH=1,NMESH
DO I=1,NI(IMESH)-1
      DO J=1,NJ(IMESH)-1
                kildeP=0
                kildeU=0
                pos=J+1
                !Coordinates in 1D-vector
```

```
                           K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
                           K2=K1+1
                           K3=K1+NI(IMESH)
                           K5=K1-1

                           IF (I.EQ.1) THEN
                              !North edge
                              an=boundW(3,J,IMESH)
                              as=(AI(K2)/RI(K2))+(CPG(K1)*MAX(0.,(-MI(K2))))

kildeU=boundW(5,J,IMESH)+(Su(K1)*VOL(K1))+(MPYR(K1)*DHPYR(K1))

                              RADIATION=boundW(6,J,IMESH)
                              EMMIS=boundW(7,J,IMESH)
                              IF (RADIATION.GT.0) THEN
                               kildeP=(-
1)*boundW(1,J,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundW(1,J,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF

                              tempN=boundW(4,J,IMESH)
                              tempS=TEMP(K2)
                              tempPo=TEMPo(K1)
                           ELSE IF (I.EQ.(NI(IMESH)-1)) THEN
                              !South edge
                              an=(AI(K1)/RI(K1))+(CPG(K1)*MAX(0.,MI(K1)))
                              as=boundE(3,J,IMESH)

kildeU=boundE(5,J,IMESH)+(Su(K1)*VOL(K1))+(MPYR(K1)*DHPYR(K1))

                              RADIATION=boundE(6,J,IMESH)
                              EMMIS=boundE(7,J,IMESH)
                              IF (RADIATION.GT.0) THEN
                               kildeP=(-
1)*boundE(1,J,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundE(1,J,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF

                              tempN=TEMP(K5)
                              tempS=boundE(4,J,IMESH)
                              tempPo=TEMPo(K1)
                           ELSE
                              !Internal cell volumes
                              an=(AI(K1)/RI(K1))+(CPG(K1)*MAX(0.,MI(K1)))
                              as=(AI(K2)/RI(K2))+(CPG(K1)*MAX(0.,(-MI(K2))))
                              kildeU=(Su(K1)*VOL(K1))+(MPYR(K1)*DHPYR(K1))

                              tempN=TEMP(K5)
                              tempS=TEMP(K2)
                              tempPo=TEMPo(K1)
                           END IF

                           IF (J.EQ.1) THEN
                              !West edge
                              ae=(AJ(K3)/RJ(K3))+(CPG(K1)*MAX(0.,(-MJ(K3))))
                              aw=boundS(3,I,IMESH)
                              kildeU=kildeU+boundS(5,I,IMESH)

                              RADIATION=boundS(6,I,IMESH)
                              EMMIS=boundS(7,I,IMESH)
                              IF (RADIATION.GT.0) THEN
                               kildeP=(-
1)*boundS(1,I,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundS(1,I,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                              END IF
```

```
                        ELSE IF (J.EQ.(NJ(IMESH)-1)) THEN
                           !East edge
                           ae=boundN(3,I,IMESH)
                           aw=(AJ(K1)/RJ(K1))+(CPG(K1)*MAX(0.,MJ(K1)))
                           kildeU=kildeU+boundN(5,I,IMESH)

                           RADIATION=boundN(6,I,IMESH)
                           EMMIS=boundN(7,I,IMESH)
                           IF (RADIATION.GT.0) THEN
                            kildeP=(-
1)*boundN(1,I,IMESH)*SIGMA*EMMIS*(4*(TEMPo(K1)**3))

kildeU=kildeU+(boundN(1,I,IMESH)*SIGMA*EMMIS*(RADIATION+(3*(TEMPo(K1)**4))))
                           END IF
                        ELSE
                           ae=(AJ(K3)/RJ(K3))+(CPG(K1)*MAX(0.,(-MJ(K3))))
                           aw=(AJ(K1)/RJ(K1))+(CPG(K1)*MAX(0.,MJ(K1)))
                        END IF

                        !Cell + source
                        apo=(RO(K1)*C(K1)*VOL(K1))/DTIME
                        ap=apo+aw+ae+an+as-kildeP

                        !TDMA Part 1 (Forward substitution)

                        !Initial temperature West
                        IF (J.EQ.1) THEN
                           P(1)=0
                           Q(1)=boundS(4,I,IMESH)
                        END IF

                        !Internal coefficients
                        ma(pos)=ap
                        mb(pos)=ae
                        mc(pos)=aw
                        md(pos)=(an*tempN)+(as*tempS)+(apo*tempPo)+kildeU

                        !Forward substitution
                        denomiator=(ma(pos)-(mc(pos)*P(pos-1)))
                        P(pos)=mb(pos)/denomiator
                        Q(pos)=((mc(pos)*Q(pos-1))+md(pos))/denomiator


                    END DO !End j and TDMA part 1

                    !TDMA Part 2 (Backward substitution)
                    DO J=NJ(IMESH)-1, 1, -1
                        pos=J+1
                        !Coordinates in 1D-vector
                        K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
                        K2=K1+1
                        K3=K1+NI(IMESH)
                        K5=K1-1

                        !Initial temperature East
                        tempOLD=temp(K1)
                        IF(J.EQ.(NJ(IMESH)-1)) THEN
                        tempE=boundN(4,I,IMESH)
                        ELSE
                        tempE=temp(K3)
                        END IF

                        !Backward substitution
                        tempP=(P(pos)*tempE)+Q(pos)
                        temp(K1)=tempP

                        !Check for convergence
                        IF(ABS(tempOLD-tempP).GT.konvergenskrit) THEN
```

```
                        ferdig=.FALSE.
                    END IF !Check for convergence
                END DO !End TDMA Part 2

         END DO !End i
        END DO !End imesh


        END DO !End convergence

        END SUBROUTINE !End Implicit

 !Name: MECH_EROSION
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: Calculates the starting and ending time of the recession

     SUBROUTINE MECH_EROSION
     USE GlobaleVariable
     implicit none

     DOUBLE PRECISION                    ::
LENIFACE,TLENIFACE,IDX,IDY,deltaSF,deltaCP,xcross,ycross,ykCP,ykSF, &

xcross1,ycross1,xcross2,ycross2,IDXX,IDYY,TLEN1,TLEN2,SUMMEK,MAXLEN, &
                                          LENIBFACED
     INTEGER                             ::
K1,K2,K3,K4,I,J,K,IMESH,R,B,IBPKT,donemek,cross,TNMEK,startMEK,FIRST1,FIRST2
     DOUBLE PRECISION, DIMENSION(4)      :: dyB,dxB,XB,YB


     DO IMESH=1,NMESH
      DO I=1,NI(IMESH)-1
       DO J=1,NJ(IMESH)-1
          K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
          !For decomposing cell volumes
          IF(IFACEBULK(K1).GT.0) THEN
          !Calculates the recession vectors
            K2=K1+1
            K3=K1+NI(IMESH)
            K4=K3+1
            IDX=IBFACEX(1)-XCP(K1)
            IDY=IBFACEY(1)-YCP(K1)
            LENIFACE=SQRT((IDX**2)+(IDY**2))+1
            DO K=1,NBFACE
             IDX=IBFACEX(K)-XCP(K1)
             IDY=IBFACEY(K)-YCP(K1)
             TLENIFACE=SQRT((IDX**2)+(IDY**2))
             IF(TLENIFACE.LT.LENIFACE) THEN
              VMX(K1)=IDX
              VMY(K1)=IDY
              IBPKT=K
              LENIFACE=TLENIFACE
             END IF
            END DO
            !Center points on cell volume edges 1=south,2=east,3=north and 4=west
            dxB(1)=XM(K2)-XM(K1)
            dyB(1)=YM(K2)-YM(K1)
            dxB(2)=XM(K4)-XM(K2)
            dyB(2)=YM(K4)-YM(K2)
            dxB(3)=XM(K3)-XM(K4)
            dyB(3)=YM(K3)-YM(K4)
            dxB(4)=XM(K1)-XM(K3)
            dyB(4)=YM(K1)-YM(K3)
            XB(1)=XM(K1)
            YB(1)=YM(K1)
            XB(2)=XM(K2)
            YB(2)=YM(K2)
```

```
XB(3)=XM(K4)
YB(3)=YM(K4)
XB(4)=XM(K3)
YB(4)=YM(K3)


FIRST1=0
FIRST2=0
DO R=1,4
 !Find the crossing points for recession length in cell volumes
 cross=0
 IF (ABS(VMX(K1)).GT.0) THEN
  deltaCP=(VMY(K1)/VMX(K1))
  ykCP=YCP(K1)-(deltaCP*XCP(K1))
  IF (ABS(dxB(R)).GT.0) THEN
   deltaSF=(dyB(R)/dxB(R))
   ykSF=YB(R)-(deltaSF*XB(R))
   IF (ABS(deltaSF-deltaCP).GT.0) THEN
    xcross=(ykSF-ykCP)/(deltaCP-deltaSF)
    ycross=ykSF+(deltaSF*xcross)
    cross=1
   END IF
  ELSE
    xcross=XB(R)
    ycross=ykCP+(deltaCP*xcross)
    cross=1
  END IF
 ELSE
  IF (ABS(dxB(R)).GT.0) THEN
   deltaSF=(dyB(R)/dxB(R))
   ykSF=YB(R)-(deltaSF*XB(R))
   xcross=XCP(K1)
   ycross=ykSF+(deltaSF*xcross)
   cross=1
  END IF
 END IF

 !If crossing point is found:
 IF (cross.EQ.1) THEN
   IF (ABS(VMX(K1)).GT.0) THEN
    IF (xcross.LT.XCP(K1)) THEN
      IF (FIRST1.EQ.0) THEN
       xcross1=xcross
       ycross1=ycross
       FIRST1=1
      ELSE IF (xcross.GT.xcross1) THEN
       xcross1=xcross
       ycross1=ycross
      END IF
    ELSE
      IF (FIRST2.EQ.0) THEN
       xcross2=xcross
       ycross2=ycross
       FIRST2=1
      ELSE IF (xcross.LT.xcross2) THEN
       xcross2=xcross
       ycross2=ycross
      END IF
    END IF
   ELSE
    IF (ycross.GT.YCP(K1)) THEN
      IF (FIRST1.EQ.0) THEN
       xcross1=xcross
       ycross1=ycross
       FIRST1=1
      ELSE IF (ycross.GT.ycross1) THEN
       xcross1=xcross
       ycross1=ycross
```

```
                END IF
              ELSE
                IF (FIRST2.EQ.0) THEN
                 xcross2=xcross
                 ycross2=ycross
                 FIRST2=1
                ELSE IF (ycross.LT.ycross2) THEN
                 xcross2=xcross
                 ycross2=ycross
                END IF
              END IF
            END IF
          END IF

        END DO

        !Crossing points: xcross1 og xcross2
        IDX=IBFACEX(IBPKT)-xcross1
        IDY=IBFACEY(IBPKT)-ycross1
        TLEN1=SQRT((IDX**2)+(IDY**2))
        IDXX=IBFACEX(IBPKT)-xcross2
        IDYY=IBFACEY(IBPKT)-ycross2
        TLEN2=SQRT((IDXX**2)+(IDYY**2))
        !Calculates the length to start of recession and the length inside cell
volume
        IF (TLEN1.LT.TLEN2) THEN
         LENIBFACE(K1)=TLEN1
         LENIBCELL(K1)=TLEN2-TLEN1
        ELSE
         LENIBFACE(K1)=TLEN2
         LENIBCELL(K1)=TLEN1-TLEN2
        END IF
        !Calculates starting and ending time of recession, erosjon VMEK()=m/s,
NMEK=number of recession rates

          !**************
          !Starting time.
          !**************
          donemek=0
          startMEK=1
          TNMEK=NMEK !N ending times
          DO
           IF (donemek.EQ.1) EXIT
             !Mean recession rate
             SUMMEK=MEK(startMEK)*(MEKTIMES(startMEK))
             IF (ABS(TNMEK-startMEK).GE.2) THEN
              DO B=(startMEK+1),(TNMEK-1)
               SUMMEK=SUMMEK+(MEK(B)*(MEKTIMES(B)-MEKTIMES(B-1)))
              END DO
             END IF
             IF(TNMEK.EQ.1) THEN
                MAXLEN=SUMMEK
             ELSE
                MAXLEN=SUMMEK+((MEKTIMES(TNMEK)-MEKTIMES(TNMEK-1))*MEK(TNMEK))
             END IF
             IF (MAXLEN.GT.LENIBFACE(K1)) THEN
               IF(MEK(TNMEK).EQ.0) THEN
                MEKTIME(K1)=0
               ELSE IF(TNMEK.EQ.1) THEN
                MEKTIME(K1)=(LENIBFACE(K1)/MEK(TNMEK))
               ELSE
                MEKTIME(K1)=((LENIBFACE(K1)-SUMMEK)/MEK(TNMEK))+MEKTIMES(TNMEK-
1)
               END IF
             ELSE
               MEKTIME(K1)=1.0E15
               donemek=1
             END IF
```

```
                  !Check for current recession
                  IF (MEKTIME(K1).LT.MEKTIMES(TNMEK-1)) THEN
                   TNMEK=TNMEK-1
                  ELSE
                   donemek=1
                  END IF

               END DO

               !************
               !Ending time.
               !************
               donemek=0
               TNMEK=NMEK !N ending times
               LENIBFACED=LENIBFACE(K1)+LENIBCELL(K1)
               DO
                  IF (donemek.EQ.1) EXIT
                  !Mean recession rate
                  SUMMEK=MEK(startMEK)*(MEKTIMES(startMEK))
                  IF (ABS(startMEK-TNMEK).GE.2) THEN
                   DO B=(startMEK+1),(TNMEK-1)
                    SUMMEK=SUMMEK+(MEK(B)*(MEKTIMES(B)-MEKTIMES(B-1)))
                   END DO
                  END IF
                  IF (TNMEK.EQ.1) THEN
                     MAXLEN=SUMMEK
                  ELSE
                     MAXLEN=SUMMEK+((MEKTIMES(TNMEK)-MEKTIMES(TNMEK-1))*MEK(TNMEK))
                  END IF
                  IF (MAXLEN.GT.LENIBFACED) THEN
                    IF(MEK(TNMEK).EQ.0) THEN
                     MEKTIMED(K1)=0
                    ELSE IF(TNMEK.EQ.1) THEN
                     MEKTIMED(K1)=(LENIBFACED/MEK(TNMEK))
                     MEKEND(K1)=0.
                    ELSE
                     MEKTIMED(K1)=((LENIBFACED-SUMMEK)/MEK(TNMEK))+MEKTIMES(TNMEK-1)
                     MEKEND(K1)=0.
                    END IF
                  ELSE
                   IF(MEK(TNMEK).EQ.0) THEN
                    MEKTIMED(K1)=0
                   ELSE
                    MEKTIMED(K1)=((MAXLEN-SUMMEK)/MEK(TNMEK))+MEKTIMES(TNMEK-1)
                    donemek=1
                    MEKEND(K1)=(((LENIBFACED-SUMMEK)/MEK(TNMEK))+MEKTIMES(TNMEK-1))-
MEKTIMED(K1)
                   END IF
                  END IF

                  !Check for current recession
                  IF (MEKTIMED(K1).LT.MEKTIMES(TNMEK-1)) THEN
                   TNMEK=TNMEK-1
                  ELSE
                   donemek=1
                  END IF
               END DO
            END IF !End decomposing cell volumes

          END DO
         END DO
        END DO


        END SUBROUTINE !End MECH_EROSION
```

```
!Name: DIRECTIONVECTORS
!Author: Jørn Riise
!Date: 30-06-2008
!Description: Calculates the direction vectors for the decompoosing material.
    SUBROUTINE directionVectors
    USE GlobaleVariable
    implicit none

    DOUBLE PRECISION                    :: LENIFACE,TLENIFACE,IDX,IDY
    INTEGER                             :: K1,I,J,K,IMESH


    DO IMESH=1,NMESH
     DO I=1,NI(IMESH)-1
      DO J=1,NJ(IMESH)-1
         K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
         !For decomposing cell volumes
        IF(IFACEBULK(K1).EQ.1) THEN
          IDX=XCP(IFACE(K))-XCP(K1)
          IDY=YCP(IFACE(K))-YCP(K1)
          LENIFACE=SQRT((IDX**2)+(IDY**2))+1
          DO K=1,NIFACE
           IDX=XCP(IFACE(K))-XCP(K1)
           IDY=YCP(IFACE(K))-YCP(K1)
           TLENIFACE=SQRT((IDX**2)+(IDY**2))
           IF(TLENIFACE.LT.LENIFACE) THEN
             VX(K1)=IDX
             VY(K1)=IDY
             LENIFACE=TLENIFACE
           END IF
          END DO
         ELSE IF(IFACEBULK(K1).GE.2 .AND. ABS(IFACEVALUE-0.02).LT.0.001) THEN
           IDX=IBFACEX(1)-XCP(K1)
           IDY=IBFACEY(1)-YCP(K1)
           LENIFACE=SQRT((IDX**2)+(IDY**2))+1
           DO K=1,NBFACE
            IDX=IBFACEX(K)-XCP(K1)
            IDY=IBFACEY(K)-YCP(K1)
            TLENIFACE=SQRT((IDX**2)+(IDY**2))
            IF(TLENIFACE.LT.LENIFACE) THEN
             VX(K1)=IDX
             VY(K1)=IDY
             LENIFACE=TLENIFACE
            END IF
           END DO
         END IF !End decomposing cell volumes

      END DO
     END DO
    END DO

    END SUBROUTINE !End directionVectors

!Name: Continuity
!Author: Jørn Riise
!Date: 30-06-2008
!Description: Solves the continuity equation using vectors.
    SUBROUTINE CONTINUITY
    USE GlobaleVariable
    implicit none

    INTEGER                             :: IMESH,K1,K2,K3,I,J,done,W1,E1,S1,N1
    DOUBLE PRECISION                    :: mOUT,Angle,iy,ix
    INTEGER,DIMENSION(4)                :: aDIR
    DOUBLE PRECISION, DIMENSION(35000)  :: lamdaX,lamdaY,mOLD
    INTEGER,DIMENSION(35000,4)          :: bDIR
```

```
done=0
MI=MI*0
MJ=MJ*0
CALL directionVectors

DO IMESH=1,NMESH
 DO I=1,NI(IMESH)-1
  DO J=1,NJ(IMESH)-1
   !Coordinates in 1D-vector
   K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
   K2=K1+1
   !For decomposing cell volumes
   IF(IFACEBULK(K1)>0) THEN
    !Gas direction
    ix=XM(K2)-XM(K1)
    iy=YM(K2)-YM(K1)
    Angle=ATAN2(VY(K1),VX(K1))-ATAN2(iy,ix)
    IF (Angle.LT.0) THEN
       Angle=Angle+(2*PI)
    END IF

    IF (Angle.LT.PI) THEN
       IF (Angle.LT.(PI/2)) THEN
          IF(Angle.EQ.0) THEN
          aDIR=(/0,1,1,1/)
          ELSE
          aDIR=(/0,1,0,1/)
          END IF
          lamdaY(K1)=2*Angle/PI
          lamdaX(K1)=1-lamdaY(K1)
       ELSE
          IF(ABS(Angle-(PI/2)).LT.0.001) THEN
          aDIR=(/1,1,0,1/)
          ELSE
          aDIR=(/1,0,0,1/)
          END IF
          lamdaX(K1)=2*((PI/2)-Angle)/PI
          lamdaY(K1)=1+lamdaX(K1)
       END IF
    ELSE
       IF (Angle.LT.(3*PI/2)) THEN
          IF (ABS(Angle-PI).LT.0.001) THEN
          aDIR=(/1,0,1,1/)
          ELSE
          aDIR=(/1,0,1,0/)
          END IF
          lamdaY(K1)=2*(PI-Angle)/PI
          lamdaX(K1)=(-1)-lamdaY(K1)
       ELSE
          IF (ABS(Angle-(3*PI/2)).LT.0.001) THEN
          aDIR=(/1,1,1,0/)
          ELSE IF (ABS(Angle-(2*PI)).LT.0.001) THEN
          aDIR=(/0,1,1,1/)
          ELSE
          aDIR=(/0,1,1,0/)
          END IF
          lamdaX(K1)=2*(Angle-(3*PI/2))/PI
          lamdaY(K1)=lamdaX(K1)-1
       END IF
    END IF
    bDIR(K1,:)=aDIR(:)
   END IF !End decomposing cell volumes
  END DO !End J
 END DO !End I
END DO !End IMESH

  !Iteration routine
  DO
```

```
        IF(done.EQ.1) EXIT
        done=1

        DO IMESH=1,NMESH
          DO I=1,NI(IMESH)-1
            DO J=1,NJ(IMESH)-1
              !Coordinates in 1D-vector
              K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
              !For decomposing cell volumes
              IF(IFACEBULK(K1)>0) THEN
               K2=K1+1
               K3=K1+NI(IMESH)

               mOUT=((-1)*MPYR(K1))-(bDIR(K1,1)*MI(K2))+(bDIR(K1,2)*MI(K1))- &
                    (bDIR(K1,3)*MJ(K3))+(bDIR(K1,4)*MJ(K1))

               IF (I.EQ.1) THEN
                  W1=boundWU(1,J,IMESH)
                  IF (W1.EQ.1) THEN

MJ(boundWU(2,J,IMESH))=(bDIR(K1,2)*MI(K1))+(bDIR(K1,1)*lamdaX(K1)*mOUT) !WEST
                  ELSEIF (W1.EQ.2) THEN

MI(boundWU(2,J,IMESH))=(bDIR(K1,2)*MI(K1))+(bDIR(K1,1)*lamdaX(K1)*mOUT) !WEST
                  END IF
               ELSE IF (I.EQ.(NI(IMESH)-1)) THEN
                  E1=boundEU(1,J,IMESH)
                  IF (E1.EQ.1) THEN

MJ(boundEU(2,J,IMESH))=(bDIR(K1,1)*MI(K2))+(bDIR(K1,2)*lamdaX(K1)*mOUT) !East
                  ELSEIF (E1.EQ.2) THEN

MI(boundEU(2,J,IMESH))=(bDIR(K1,1)*MI(K2))+(bDIR(K1,2)*lamdaX(K1)*mOUT) !East
                  END IF
               END IF

               IF (J.EQ.1) THEN
                  S1=boundSU(1,I,IMESH)
                  IF (S1.EQ.1) THEN

MJ(boundSU(2,I,IMESH))=(bDIR(K1,4)*MJ(K1))+(bDIR(K1,3)*lamdaY(K1)*mOUT) !SOUTH
                  ELSEIF (S1.EQ.2) THEN

MI(boundSU(2,I,IMESH))=(bDIR(K1,4)*MJ(K1))+(bDIR(K1,3)*lamdaY(K1)*mOUT) !SOUTH
                  END IF
               ELSE IF (J.EQ.(NJ(IMESH)-1)) THEN
                  N1=boundNU(1,I,IMESH)
                  IF (N1.EQ.1) THEN

MJ(boundNU(2,I,IMESH))=(bDIR(K1,3)*MJ(K3))+(bDIR(K1,4)*lamdaY(K1)*mOUT) !NORTH
                  ELSEIF (N1.EQ.2) THEN

MI(boundNU(2,I,IMESH))=(bDIR(K1,3)*MJ(K3))+(bDIR(K1,4)*lamdaY(K1)*mOUT) !NORTH
                  END IF
               END IF

               MI(K1)=(bDIR(K1,2)*MI(K1))+(bDIR(K1,1)*lamdaX(K1)*mOUT) !WEST
               MI(K2)=(bDIR(K1,1)*MI(K2))+(bDIR(K1,2)*lamdaX(K1)*mOUT) !EAST
               MJ(K1)=(bDIR(K1,4)*MJ(K1))+(bDIR(K1,3)*lamdaY(K1)*mOUT) !SOUTH
               MJ(K3)=(bDIR(K1,3)*MJ(K3))+(bDIR(K1,4)*lamdaY(K1)*mOUT) !NORTH

               !Check for convergence
               IF(ABS(mOLD(K1)-mOUT).GT.1E-12) THEN
                  done=0
               END IF
               mOLD(K1)=mOUT
             END IF !End decomposing cell volumes
```

```
          END DO !End J
        END DO !End I
      END DO !End MESH

     END DO !End iterasjon

    END SUBROUTINE !End Continuity


 !Name: INIT_PYROLYSIS
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: Solves the continuity equation using vectors.
     SUBROUTINE INIT_PYROLYSIS
     USE GlobaleVariable
     implicit none

     INTEGER                                 ::
K1,K2,K3,K4,II,JJ,IIMESH,PYRMAT,REAC,switch
     DOUBLE PRECISION                        ::
xp1,yp1,xp2,yp2,xp3,yp3,xp4,yp4,a1to3,b1to3, &
                                              a4to2,b4to2,tempx1,tempx2
       !Decision variable for the interface
       IF(IFACEVALUE.EQ.0) THEN
         IFACEVALUE=0.02
       END IF

       !Initializing parameters
       IF (Pyrini.EQ.1) THEN
            Pyrini=0
            RHORTOT=RHORTOT*0
            RHOOTOT=RHOOTOT*0
            VX=VX*0
            VY=VY*0
            VMX=VMX*0
            VMY=VMY*0
            MEKEND=MEKEND*0
            precisionP=1.0E2
         DO IIMESH=1,NMESH
           DO JJ=1,NJ(IIMESH)-1
             DO II=1,NI(IIMESH)-1
              K1=II+NI(IIMESH)*(JJ-1)+NMP(IIMESH)
              !For decomposing cell volumes
              IF(IFACEBULK(K1)>0) THEN
               PYRMAT=MAT(K1)
               ANTPYR(K1)=AM_ANTPYR(PYRMAT)
               DO REAC=1,ANTPYR(K1)
                 VOLFRAC(REAC,K1)=AM_VOLFRAC(REAC,PYRMAT)
                 RHO(REAC,K1)=AM_RHOO(REAC,PYRMAT)*VOLFRAC(REAC,K1)
                 RHOO(REAC,K1)=AM_RHOO(REAC,PYRMAT)*VOLFRAC(REAC,K1)
                 RHOR(REAC,K1)=AM_RHOR(REAC,PYRMAT)*VOLFRAC(REAC,K1)
                 APYR(REAC,K1)=AM_APYR(REAC,PYRMAT)
                 EPYR(REAC,K1)=AM_EPYR(REAC,PYRMAT)
                 NPYR(REAC,K1)=AM_NPYR(REAC,PYRMAT)
                 TREAC(REAC,K1)=AM_TREAC(REAC,PYRMAT)
                 RHOOTOT(K1)=RHOOTOT(K1)+RHOO(REAC,K1)
                 RHORTOT(K1)=RHORTOT(K1)+RHOR(REAC,K1)
               END DO
               RHOFRAC(K1)=1.
               RO(K1)=RHOOTOT(K1)
               MEKSTART(K1)=0
               MEKTIME(K1)=1.0E15
               MEKTIMED(K1)=1.0E15

              END IF
             END DO
           END DO
         END DO
```

```
        END IF

        !Initializing parameters
        NBFACE=0
        IBFACEX=IBFACEX*0
        IBFACEY=IBFACEY*0
        DO IIMESH=1,NMESH
          DO JJ=1,NJ(IIMESH)-1
            DO II=1,NI(IIMESH)-1
              !Defining centerpoints in the cell volume
              K1=II+NI(IIMESH)*(JJ-1)+NMP(IIMESH)
              !For decomposing cell volumes
              IF(IFACEBULK(K1)>0) THEN
               K2=K1+1
               K3=K1+NI(IIMESH)
               K4=K3+1

               !Edge coordinated xp=x-point,yp=y-point, 1=south, 2=east, 3=north and
4=west
               xp1=XM(K1)+((XM(K2)-XM(K1))/2)
               yp1=YM(K1)+((YM(K2)-YM(K1))/2)
               xp2=XM(K2)+((XM(K4)-XM(K2))/2)
               yp2=YM(K2)+((YM(K4)-YM(K2))/2)
               xp3=XM(K4)+((XM(K3)-XM(K4))/2)
               yp3=YM(K4)+((YM(K3)-YM(K4))/2)
               xp4=XM(K3)+((XM(K1)-XM(K3))/2)
               yp4=YM(K3)+((YM(K1)-YM(K3))/2)
               !Line 1->3 and 4->2 makes an intersection (y=ax+b)
               switch=1
               tempx1=(xp3-xp1)
               IF(ABS(tempx1).GT.0.0) THEN
                 a1to3=(yp3-yp1)/tempx1
                 b1to3=yp3-(a1to3*xp3)
               ELSE
                 switch=0
                 XCP(K1)=xp3
               END IF
                 tempx2=(xp2-xp4)
               IF(ABS(tempx2).GT.0.0) THEN
                 a4to2=(yp2-yp4)/tempx2
                 b4to2=yp4-(a4to2*xp4)
               ELSE
                 switch=0
                 XCP(K1)=xp2
               END IF


               !Center point in cell volume k1: XCP(K1),YCP(K1)
               IF (switch.EQ.1) THEN
                IF(ABS(a1to3).GT.0.0) THEN
                 XCP(K1)=(b4to2-b1to3)/(a1to3-a4to2)
                 YCP(K1)=(a1to3*XCP(K1))+b1to3
                ELSE
                 XCP(K1)=(b4to2-b1to3)/(a1to3-a4to2)
                 YCP(K1)=(a4to2*XCP(K1))+b4to2
                END IF
               ELSE
                IF(ABS(tempx1).GT.0.0) THEN
                   YCP(K1)=b1to3+(a1to3*XCP(K1))
                 ELSE
                   YCP(K1)=b4to2+(a4to2*XCP(K1))
                 END IF
               END IF

               IF(II.EQ.1) THEN
                  IF(boundW(2,JJ,IIMESH).GT.0) THEN
                    WRITE(6,*) 'west'
                    IFACEBULK(K1)=3
```

```
                VX(K1)=xp4-XCP(K1)
                VY(K1)=yp4-YCP(K1)
                NBFACE=NBFACE+1
                IBFACEX(NBFACE)=xp4
                IBFACEY(NBFACE)=yp4
              END IF
            ELSEIF(II.EQ.(NI(IIMESH)-1)) THEN
              IF (boundE(2,JJ,IIMESH).GT.0) THEN
                WRITE(6,*) 'east'
                IFACEBULK(K1)=3
                VX(K1)=xp2-XCP(K1)
                VY(K1)=yp2-YCP(K1)
                NBFACE=NBFACE+1
                IBFACEX(NBFACE)=xp2
                IBFACEY(NBFACE)=yp2
                END IF
            END IF

            IF (JJ.EQ.1) THEN
              IF (boundS(2,II,IIMESH).GT.0) THEN
                WRITE(6,*) 'south'
                IFACEBULK(K1)=3
                VX(K1)=xp1-XCP(K1)
                VY(K1)=yp1-YCP(K1)
                NBFACE=NBFACE+1
                IBFACEX(NBFACE)=xp1
                IBFACEY(NBFACE)=yp1
              END IF
              IF(ABS(boundS(2,II,IIMESH)-1).LT.0.0001) THEN
                precisionP=1.0E14
              ELSE
                precisionP=1.0E2
              END IF
            ELSEIF (JJ.EQ.(NJ(IIMESH)-1)) THEN
              IF (boundN(2,II,IIMESH).GT.0) THEN
                WRITE(6,*) 'north'
                IFACEBULK(K1)=3
                VX(K1)=xp3-XCP(K1)
                VY(K1)=yp3-YCP(K1)
                NBFACE=NBFACE+1
                IBFACEX(NBFACE)=xp3
                IBFACEY(NBFACE)=yp3
              END IF
            END IF

          END IF !End decomposing cell volumes
        END DO !End J
      END DO !End I
    END DO !End Mesh

    !If recession rates are specified
    IF (M_EROSION.EQ.1) THEN
     CALL MECH_EROSION
     M_EROSION=0
    END IF

  END SUBROUTINE !End INIT_PYROLYSIS


 !Name: PYROLYSIS
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: Solves the internal decomposition reactions and adjust material
properties.
  SUBROUTINE PYROLYSIS
  USE GlobaleVariable
  implicit none
```

```
      DOUBLE PRECISION                        ::
DALFA,RHOTEMP,RHORTEMP,RHODIFF,ALFADIFF,ALFATEMP,T,TMIN,TMAX,XSI,T1,T2,XFRAC, &
                                                MEKDTIME,MFRAC
      INTEGER                                 ::
NREAC,REAC,K1,K2,K3,K4,K5,I,J,IMESH,IIMAT,IT,N

      NIFACE=0
      IFACE=IFACE*0

      DO IMESH=1,NMESH
         DO I=1,NI(IMESH)-1
            DO J=1,NJ(IMESH)-1
             K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
             K3=K1+NI(IMESH)
             !For decomposing cell volumes
             IF(IFACEBULK(K1).GT.0) THEN
              !Check for starting time of recession (default=MEKTIME>TIME)
             IF (TIME.GT.MEKTIME(K1)) THEN
                IF (TIME.GE.MEKTIMED(K1)) THEN
                 IF (MEKEND(K1).LE.0) THEN
                    C(K1)=0.01
                    TCI(K1)=1*precisionP
                    TCJ(K1)=1*precisionP
                    RO(K1)=0.1
                    MPYR(K1)=0.
                    RHOFRAC(K1)=0.
                    CPG(K1)=0.
                    DHPYR(K1)=0.
                 END IF
                ELSE
                  !Calculates the recession rate of the cell volume
                  IF(MEKSTART(K1).EQ.0) THEN
                   MEKDTIME=(MEKTIMED(K1)+MEKEND(K1))-TIME
                   MEKRHOTOT(K1)=RO(K1)
                   MEKALFA(K1)=MEKRHOTOT(K1)/MEKDTIME
                   MEKSTART(K1)=1
                  END IF
                  RO(K1)=RO(K1)-(MEKALFA(K1)*DTIME)
                  MPYR(K1)=0.
                  MFRAC=RO(K1)/MEKRHOTOT(K1)
                  IF (MFRAC.LE.0.0) THEN
                    RO(K1)=0.1
                    MFRAC=0.
                  END IF

                  XFRAC=RHOFRAC(K1)

                  !Update material properties
                  T=TEMP(K1)
                      IIMAT=MAT(K1)
                      N=NINT(AM_DATA(IIMAT))
                      TMIN=AM_TEMP(IIMAT,1)
                      TMAX=AM_TEMP(IIMAT,N)
                      IF (T.LE.TMIN) THEN
                       T1=TMIN
                       T2=TMIN
                       TCI(K1)=((AM_TCI(IIMAT,1)*XFRAC)+(AM_TCIR(IIMAT,1)*(1-
XFRAC))*MFRAC)+((1-MFRAC)*1.0*precisionP)
                       TCJ(K1)=((AM_TCJ(IIMAT,1)*XFRAC)+(AM_TCJR(IIMAT,1)*(1-
XFRAC))*MFRAC)+((1-MFRAC)*1.0*precisionP)
                       C(K1)  =(((AM_CP(IIMAT,1)*XFRAC)+(AM_CPR(IIMAT,1)*(1-
XFRAC)))*MFRAC)+((1-MFRAC)*(0.1E-1))
                    CPG(K1)=AM_CPG(IIMAT,1)
                    DHPYR(K1)=AM_DHPYR(IIMAT,1)
                       ELSE IF (T.GE.TMAX) THEN
                       T1=TMAX
                       T2=TMAX
```

```
                    TCI(K1)=((AM_TCI(IIMAT,N)*XFRAC)+(AM_TCIR(IIMAT,N)*(1-
XFRAC))*MFRAC)+((1-MFRAC)*1.0*precisionP)
                TCJ(K1)=((AM_TCJ(IIMAT,N)*XFRAC)+(AM_TCJR(IIMAT,N)*(1-
XFRAC))*MFRAC)+((1-MFRAC)*1.0*precisionP)
                    C(K1)   =(((AM_CP(IIMAT,N)*XFRAC)+(AM_CPR(IIMAT,N)*(1-
XFRAC)))*MFRAC)+((1-MFRAC)*(0.1E-1))
                  CPG(K1)=AM_CPG(IIMAT,N)
                  DHPYR(K1)=AM_DHPYR(IIMAT,N)
                     ELSE
                      DO IT=1,N-1
                       T1=AM_TEMP(IIMAT,IT)
                       T2=AM_TEMP(IIMAT,IT+1)
                       XSI=(T-T1)/(T2-T1)
                       IF(T2.GT.T)EXIT
               END DO
                  TCI(K1)=((((((1.-
XSI)*AM_TCI(IIMAT,IT))+(XSI*AM_TCI(IIMAT,IT+1)))*XFRAC)+ &
                           ((((1.-
XSI)*AM_TCIR(IIMAT,IT))+(XSI*AM_TCIR(IIMAT,IT+1)))*(1-XFRAC)))*MFRAC)+ &
                           ((1-MFRAC)*1.0*precisionP)
                  TCJ(K1)=((((((1.-
XSI)*AM_TCJ(IIMAT,IT))+(XSI*AM_TCJ(IIMAT,IT+1)))*XFRAC)+ &
                           ((((1.-
XSI)*AM_TCJR(IIMAT,IT))+(XSI*AM_TCJR(IIMAT,IT+1)))*(1-XFRAC)))*MFRAC)+ &
                           ((1-MFRAC)*1.0*precisionP)
               C(K1)  =(((((1.-XSI)*AM_CP(IIMAT,IT) +XSI*AM_CP(IIMAT,IT+1))*XFRAC)+
&
                            (((1.-XSI)*AM_CPR(IIMAT,IT)+XSI*AM_CPR(IIMAT,IT+1))*(1-
XFRAC)))* &
                           MFRAC)+((1-MFRAC)*(0.1E-1))
                  CPG(K1)=((1.-XSI)*AM_CPG(IIMAT,IT)) +(XSI*AM_CPG(IIMAT,IT+1))
                  DHPYR(K1)=((1.-XSI)*AM_DHPYR(IIMAT,IT))
+(XSI*AM_DHPYR(IIMAT,IT+1))
              END IF
               END IF

             !End M_EROSION
             ELSE
               NREAC=ANTPYR(K1)
               !Calculates the pyrolysis reactions
               ALFATEMP=0.
               DO REAC=1,NREAC
                 RHORTEMP=RHOR(REAC,K1)
                 RHOTEMP=RHO(REAC,K1)
                 RHODIFF=RHOO(REAC,K1)-RHORTEMP
                 IF (TEMP(K1).GT.TREAC(REAC,K1)) THEN
                    DALFA=DTIME*RHODIFF*((((RHOTEMP-
RHORTEMP))/RHODIFF)**NPYR(REAC,K1))* &
                      APYR(REAC,K1)*EXP(-EPYR(REAC,K1)/TEMP(K1))
                   IF (RHOTEMP.GT.RHORTEMP) THEN
                      ALFADIFF=RHOTEMP-DALFA
                      IF (ALFADIFF.GT.RHORTEMP) THEN
                         ALFATEMP=ALFATEMP+DALFA
                         RHO(REAC,K1)=ALFADIFF
                      ELSE
                         ALFATEMP=ALFATEMP+(RHOTEMP-RHORTEMP)
                         RHO(REAC,K1)=RHORTEMP
                      END IF
                   END IF
                 END IF
               END DO


                 !Update the density and pyrolysis gas rate
                 MPYR(K1)=((-ALFATEMP)*VOL(K1))/DTIME
                 RO(K1)=RO(K1)-ALFATEMP
                 IF (RO(K1).LE.0) THEN
                   RO(K1)=0.1
```

```
                  END IF

                  !Fraction parameter
                  XFRAC=(RO(K1)-RHORTOT(K1))/(RHOOTOT(K1)-RHORTOT(K1))
                  RHOFRAC(K1)=XFRAC

                  !Update material properties
                  T=TEMP(K1)
               IIMAT=MAT(K1)
               N=NINT(AM_DATA(IIMAT))
               TMIN=AM_TEMP(IIMAT,1)
               TMAX=AM_TEMP(IIMAT,N)
               IF (T.LE.TMIN) THEN
                T1=TMIN
                T2=TMIN
                TCI(K1)=(AM_TCI(IIMAT,1)*XFRAC)+(AM_TCIR(IIMAT,1)*(1-XFRAC))
                TCJ(K1)=(AM_TCJ(IIMAT,1)*XFRAC)+(AM_TCJR(IIMAT,1)*(1-XFRAC))
                C(K1)  =(AM_CP(IIMAT,1)*XFRAC)+(AM_CPR(IIMAT,1)*(1-XFRAC))
                   CPG(K1)=AM_CPG(IIMAT,1)
                   DHPYR(K1)=AM_DHPYR(IIMAT,1)
               ELSE IF (T.GE.TMAX) THEN
                T1=TMAX
                T2=TMAX
                TCI(K1)=(AM_TCI(IIMAT,N)*XFRAC)+(AM_TCIR(IIMAT,N)*(1-XFRAC))
                TCJ(K1)=(AM_TCJ(IIMAT,N)*XFRAC)+(AM_TCJR(IIMAT,N)*(1-XFRAC))
                C(K1)  =(AM_CP(IIMAT,N)*XFRAC)+(AM_CPR(IIMAT,N)*(1-XFRAC))
                   CPG(K1)=AM_CPG(IIMAT,N)
                   DHPYR(K1)=AM_DHPYR(IIMAT,N)
               ELSE
                DO IT=1,N-1
                 T1=AM_TEMP(IIMAT,IT)
                 T2=AM_TEMP(IIMAT,IT+1)
                 XSI=(T-T1)/(T2-T1)
                 IF(T2.GE.T)EXIT
                END DO
                   TCI(K1)=((((1.-
XSI)*AM_TCI(IIMAT,IT))+(XSI*AM_TCI(IIMAT,IT+1)))*XFRAC)+ &
                           ((((1.-
XSI)*AM_TCIR(IIMAT,IT))+(XSI*AM_TCIR(IIMAT,IT+1)))*(1-XFRAC))
                   TCJ(K1)=((((1.-
XSI)*AM_TCJ(IIMAT,IT))+(XSI*AM_TCJ(IIMAT,IT+1)))*XFRAC)+ &
                           ((((1.-
XSI)*AM_TCJR(IIMAT,IT))+(XSI*AM_TCJR(IIMAT,IT+1)))*(1-XFRAC))
               C(K1)  =(((1.-XSI)*AM_CP(IIMAT,IT) +XSI*AM_CP(IIMAT,IT+1))*XFRAC)+ &
                           (((1.-XSI)*AM_CPR(IIMAT,IT)+XSI*AM_CPR(IIMAT,IT+1))*(1-
XFRAC))
                   CPG(K1)=AM_CPG(IIMAT,IT) !((1.-XSI)*AM_CPG(IIMAT,IT))
+(XSI*AM_CPG(IIMAT,IT+1))
                   DHPYR(K1)=((1.-XSI)*AM_DHPYR(IIMAT,IT))
+(XSI*AM_DHPYR(IIMAT,IT+1))
                END IF
              END IF !End material properties for decomposing cell volumes


              !Updates the identification parameter
              IF (RHOFRAC(K1).LT.IFACEVALUE) THEN
                IF(IFACEBULK(K1).LT.3) THEN
                IFACEBULK(K1)=2
                END IF
              END IF

              !For residue or "empty" cell volumes
              IF(IFACEBULK(K1).GE.2) THEN
                 !Find the cell volumes on the interface between virgin/decomposing
material and residue material
                 !-> K2=east <-> K4=west <-> K3=north <-> K5=south <- of K1, check
if id=1 is a neighbour
                 IF (I.EQ.1) THEN
```

```
      !West
      K4=boundWU(5,J,IMESH)
      K2=K1+1
   ELSE IF (I.EQ.(NI(IMESH)-1)) THEN
      !East
      K4=K1-1
      K2=boundEU(5,J,IMESH)
   ELSE
      !Internal
      K4=K1-1
      K2=K1+1
   END IF

   IF (J.EQ.1) THEN
      !South
      K5=boundSU(5,I,IMESH)
      K3=K1+NI(IMESH)
   ELSE IF (J.EQ.(NJ(IMESH)-1)) THEN
      !North
      K5=K1-NI(IMESH)
      K3=boundNU(5,I,IMESH)
   ELSE
      K5=K1-NI(IMESH)
      K3=K1+NI(IMESH)
   END IF

   IF(K2.GT.0) THEN
     IF(RHOFRAC(K2).GT.IFACEVALUE) THEN
       NIFACE=NIFACE+1
       IFACE(NIFACE)=K1
       K3=0
       k4=0
       k5=0
     END IF
   END IF
   IF(K3.GT.0) THEN
     IF(RHOFRAC(K3).GT.IFACEVALUE) THEN
       NIFACE=NIFACE+1
       IFACE(NIFACE)=K1
       K4=0
       K5=0
     END IF
   END IF
   IF(K4.GT.0) THEN
     IF(RHOFRAC(K4).GT.IFACEVALUE) THEN
       NIFACE=NIFACE+1
       IFACE(NIFACE)=K1
       K5=0
     END IF
   END IF
   IF(K5.GT.0) THEN
     IF(RHOFRAC(K5).GT.IFACEVALUE) THEN
       NIFACE=NIFACE+1
       IFACE(NIFACE)=K1
     END IF
   END IF

 END IF !End decomposing cell volumes


ELSE  !Update properties of backup material

   T=TEMP(K1)
   IIMAT=MAT(K1)
   N=NINT(AM_DATA(IIMAT))
   TMIN=AM_TEMP(IIMAT,1)
   TMAX=AM_TEMP(IIMAT,N)
```

```
              IF (T.LE.TMIN) THEN
                T1=TMIN
                T2=TMIN
                RO(K1) =AM_RO(IIMAT,1)
                TCI(K1)=AM_TCI(IIMAT,1)
                TCJ(K1)=AM_TCJ(IIMAT,1)
                C(K1)  =AM_CP(IIMAT,1)
              ELSE IF (T.GE.TMAX) THEN
                T1=TMAX
                T2=TMAX
                RO(K1) =AM_RO(IIMAT,N)
                TCI(K1)=AM_TCI(IIMAT,N)
                TCJ(K1)=AM_TCJ(IIMAT,N)
                C(K1)  =AM_CP(IIMAT,N)
              ELSE
                DO IT=1,N-1
                 T1=AM_TEMP(IIMAT,IT)
                 T2=AM_TEMP(IIMAT,IT+1)
                 XSI=(T-T1)/(T2-T1)
                 IF(T2.GT.T)EXIT
                END DO
                RO(K1) =(1.-XSI)*AM_RO(IIMAT,IT) +XSI*AM_RO(IIMAT,IT+1)
                TCI(K1)=(1.-XSI)*AM_TCI(IIMAT,IT)+XSI*AM_TCI(IIMAT,IT+1)
                TCJ(K1)=(1.-XSI)*AM_TCJ(IIMAT,IT)+XSI*AM_TCJ(IIMAT,IT+1)
                C(K1)  =(1.-XSI)*AM_CP(IIMAT,IT) +XSI*AM_CP(IIMAT,IT+1)
              END IF

           END IF

         END DO !End J
      END DO !End I
   END DO !End MESH


   END SUBROUTINE !End pyrolysis


!Name: UPDATE_IMPLICIT
!Author: Jørn Riise
!Date: 30-06-2008
!Description: Updates the shadow cells, exchange information between grid blocks.
     SUBROUTINE UPDATE_IMPLICIT
     USE GlobaleVariable
     implicit none

     !Local variables
     INTEGER                              :: IMESH,J,I,E1,E2,W1,W2,N1,N2,S1,S2
     DOUBLE PRECISION                     :: REA,REB,RWA,RWB,RNA,RNB,RSA,RSB
     DOUBLE PRECISION                     :: GEA,GWA,GNA,GSA



     DO IMESH=1,NMESH
       DO J=1,NJ(IMESH)-1
          !East edge

          !Choice 1
          E1=boundEU(1,J,IMESH)

          IF (E1.EQ.0) THEN
             REA=0
             GEA=0
             !SourceTot=source * area
             boundE(5,J,IMESH)=boundE(2,J,IMESH)*boundE(1,J,IMESH)
          ELSE IF (E1.EQ.1) THEN
             !Temp and resistance J-direction
             REA=RJ(boundEU(2,J,IMESH))
             GEA=MJ(boundEU(2,J,IMESH))
```

```
               boundE(4,J,IMESH)=TEMP(boundEU(5,J,IMESH))
            ELSE IF (E1.EQ.2) THEN
               !Temp and resistance I-direction
               REA=RI(boundEU(2,J,IMESH))
               GEA=MI(boundEU(2,J,IMESH))
               boundE(4,J,IMESH)=TEMP(boundEU(5,J,IMESH))
            ELSE IF (E1.EQ.3) THEN
               GEA=0
               !Convective resistance
               REA=(1/boundE(2,J,IMESH))
            ELSE
               REA=0
               GEA=0
            END IF

            !Choice 2
            E2=boundEU(3,J,IMESH)

            IF (E2.EQ.1) THEN
               REB=RJ(boundEU(4,J,IMESH))
            ELSE IF (E2.EQ.2) THEN
               REB=RI(boundEU(4,J,IMESH))
            ELSE
               REB=0
            END IF

            REA=REA+REB
            !Calculates Area/Resistance
            IF (REA.NE.0) THEN

boundE(3,J,IMESH)=(boundE(1,J,IMESH)/REA)+(CPG(boundEU(5,J,IMESH))*MAX(0.,(-GEA)))
            END IF
            !End East edge

            !West edge
            W1=boundWU(1,J,IMESH)
            IF (W1.EQ.0) THEN
               RWA=0
               GWA=0
               boundW(5,J,IMESH)=boundW(2,J,IMESH)*boundW(1,J,IMESH)
            ELSE IF (W1.EQ.1) THEN
               RWA=RJ(boundWU(2,J,IMESH))
               GWA=MJ(boundWU(2,J,IMESH))
               boundW(4,J,IMESH)=TEMP(boundWU(5,J,IMESH))
            ELSE IF (W1.EQ.2) THEN
               RWA=RI(boundWU(2,J,IMESH))
               GWA=MI(boundWU(2,J,IMESH))
               boundW(4,J,IMESH)=TEMP(boundWU(5,J,IMESH))
            ELSE IF (W1.EQ.3) THEN
               GWA=0
               RWA=1/boundW(2,J,IMESH)
            ELSE
               RWA=0
               GWA=0
            END IF

            W2=boundWU(3,J,IMESH)
            IF (W2.EQ.1) THEN
               RWB=RJ(boundWU(4,J,IMESH))
            ELSE IF (W2.EQ.2) THEN
               RWB=RI(boundWU(4,J,IMESH))
            ELSE
               RWB=0
            END IF

            RWA=RWA+RWB
            IF (RWA.NE.0) THEN
```

```
boundW(3,J,IMESH)=(boundW(1,J,IMESH)/RWA)+(CPG(boundWU(5,J,IMESH))*MAX(0.,GWA))
            END IF
            !End West edge

         END DO !End West and East

         DO I=1,NI(IMESH)-1
            !North edge
            N1=boundNU(1,I,IMESH)
            IF (N1.EQ.0) THEN
               RNA=0
               GNA=0
               boundN(5,I,IMESH)=boundN(2,I,IMESH)*boundN(1,I,IMESH)
            ELSE IF (N1.EQ.1) THEN
               RNA=RJ(boundNU(2,I,IMESH))
               GNA=MJ(boundNU(2,I,IMESH))
               boundN(4,I,IMESH)=TEMP(boundNU(5,I,IMESH))
            ELSE IF (N1.EQ.2) THEN
               RNA=RI(boundNU(2,I,IMESH))
               GNA=MI(boundNU(2,I,IMESH))
               boundN(4,I,IMESH)=TEMP(boundNU(5,I,IMESH))
            ELSE IF (N1.EQ.3) THEN
               GNA=0
               RNA=1/boundN(2,I,IMESH)
            ELSE
               RNA=0
               GNA=0
            END IF

            N2=boundNU(3,I,IMESH)
            IF (N2.EQ.1) THEN
               RNB=RJ(boundNU(4,I,IMESH))
            ELSE IF (N2.EQ.2) THEN
               RNB=RI(boundNU(4,I,IMESH))
            ELSE
               RNB=0
            END IF

            RNA=RNA+RNB
            IF (RNA.NE.0) THEN

boundN(3,I,IMESH)=(boundN(1,I,IMESH)/RNA)+(CPG(boundNU(5,I,IMESH))*MAX(0.,(-GNA)))
            END IF
            !End North edge

            !South edge
            S1=boundSU(1,I,IMESH)
            IF (S1.EQ.0) THEN
               RSA=0
               GSA=0
               boundS(5,I,IMESH)=boundS(2,I,IMESH)*boundS(1,I,IMESH)
            ELSE IF (S1.EQ.1) THEN
               RSA=RJ(boundSU(2,I,IMESH))
               GSA=MJ(boundSU(2,I,IMESH))
               boundS(4,I,IMESH)=TEMP(boundSU(5,I,IMESH))
            ELSE IF (S1.EQ.2) THEN
               RSA=RI(boundSU(2,I,IMESH))
               GSA=MI(boundSU(2,I,IMESH))
               boundS(4,I,IMESH)=TEMP(boundSU(5,I,IMESH))
            ELSE IF (S1.EQ.3) THEN
               GSA=0
               RSA=1/boundS(2,I,IMESH)
            ELSE
               RSA=0
               GSA=0
            END IF
```

```
            S2=boundSU(3,I,IMESH)
            IF (S2.EQ.1) THEN
               RSB=RJ(boundSU(4,I,IMESH))
            ELSE IF (S2.EQ.2) THEN
               RSB=RI(boundSU(4,I,IMESH))
            ELSE
               RSB=0
            END IF

            RSA=RSA+RSB
            IF (RSA.NE.0) THEN

boundS(3,I,IMESH)=(boundS(1,I,IMESH)/RSA)+(CPG(boundSU(5,I,IMESH))*MAX(0.,GSA))
            END IF
            !End South edge
          END DO !End North and South

        END DO !End imesh

        END SUBROUTINE !End UPDATE_IMPLICIT



 !Name: BORDERS_IMPLICIT
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: Initializing boundary conditions.
        SUBROUTINE BORDERS_IMPLICIT

        USE GlobaleVariable
        implicit none

        !Local variables
        INTEGER           :: ID,JD,IC,JC,IRS,JRS,IM,I0,J0,IDR,NP,K0,    &
                             K0I,K0J,KD,K1,KRSI,KRSJ,IBND,IPP
        DOUBLE PRECISION :: T,RADIUS,RE,ST
        DIMENSION ID(4),JD(4),IC(4),JC(4),IRS(4),JRS(4)
        INTEGER :: valg,step,pos

        DATA ID / 1, 0,-1, 0/
        DATA JD / 0, 1, 0,-1/
        DATA IC / 0,-1,-1, 0/
        DATA JC / 0, 0,-1,-1/
        DATA IRS/ 1, 0, 1, 0/
        DATA JRS/ 0, 1, 0, 1/

        DO IBND=1,NBND
          IM =IMBND(IBND) !Block
          I0 =I0BND(IBND) !Start I
          J0 =J0BND(IBND) !Start J
          IDR=IDBND(IBND) !direction
          NP =NPBND(IBND) !Number of nodes in the direction

          !Pointer information for the cell volumes along the edge
          K0 =I0        +NI(IM)*(J0        -1)+NMP(IM)
          K0I=I0        +NI(IM)*(J0+JC(IDR)-1)+NMP(IM)
          K0J=I0+IC(IDR)+NI(IM)*(J0        -1)+NMP(IM)
          KD= ID(IDR)+NI(IM)* JD(IDR)

          !Decides if RI or RJ is used in the thermal resistance
          valg=JRS(IDR)+1

          !I-direction(Startposition=I0, step=ID(IDR))
          IF (IRS(IDR).EQ.1) THEN
            !North
            IF (J0.GT.(NJ(IM)/2)) THEN
                 DO IPP=1,NP-1
                   K1  =K0 +KD*(IPP-1)
```

```
                KRSI=K0I+KD*(IPP-1)
                KRSJ=K0J+KD*(IPP-1)

                !Describe the movement in the edge vector
                step=(IPP)*ID(IDR)
                pos=I0+step

               IF(ITBND(IBND).EQ.1 .OR. ITBND(IBND).EQ.5)THEN
                 IF(ITBND(IBND).EQ.5)THEN
                    T=(TIME-TSTART)/(TSTOP-TSTART)
                    !Create pointers and save information(area,HC ect.) in vectors
                    boundNU(1,pos,IM)=3
                    boundNU(3,pos,IM)=valg
                    boundNU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                    boundN(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                    boundN(2,pos,IM)=HCBND(IBND)+T*DTHCBND(IBND)
                    boundN(4,pos,IM)=TUBND(IBND)+T*DTTUBND(IBND)
                 ELSE
                    boundNU(1,pos,IM)=3
                    boundNU(3,pos,IM)=valg
                    boundNU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                    boundN(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                    boundN(2,pos,IM)=HCBND(IBND)
                    boundN(4,pos,IM)=TUBND(IBND)
                 ENDIF

               ENDIF

               IF(ITBND(IBND).EQ.2)THEN
                 boundN(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                 boundN(2,pos,IM)=QBND(IBND)
               ENDIF

               IF(ITBND(IBND).EQ.3)THEN     !internal heat transfer
                 RADIUS=YM(K1)
                 RE=UBND(IBND,IPP)*RADIUS/VISCBND(IBND,IPP)
                 ST=0.0791/RE**0.25*0.5/                                        &
                 (1.+1.99*RE**(-0.125)*(PRBND(IBND,IPP)-1))

                 boundNU(1,pos,IM)=3
                 boundNU(3,pos,IM)=valg
                 boundNU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                 boundN(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)

boundN(2,pos,IM)=ROBND(IBND,IPP)*UBND(IBND,IPP)*CPBND(IBND,IPP)*ST
                 boundN(4,pos,IM)=TUBND(IBND)
               ENDIF
               IF(ITBND(IBND).EQ.4.OR.ITBND(IBND).EQ.6)THEN
                 boundNU(1,pos,IM)=3
                 boundNU(3,pos,IM)=valg
                 boundNU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                 boundN(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                 boundN(2,pos,IM)=HCEBND(IBND,IPP)
                 boundN(4,pos,IM)=TRBND(IBND,IPP)
               ENDIF

               IF(IRADBND(IBND).EQ.1)THEN   !add radiation heat transfer
                 boundN(6,pos,IM)=(TRADBND(IBND)**4)
                 boundN(7,pos,IM)=EMSBND(IBND)
               ENDIF
               IF(ITBND(IBND).EQ.7)THEN
                 boundNU(1,pos,IM)=4
                 boundNU(3,pos,IM)=valg
                 boundNU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                 boundN(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                 boundN(2,pos,IM)=1 !because of the pyrolysis, it has no
functionality.
                 boundN(4,pos,IM)=TUBND(IBND)
```

```
          ENDIF

        END DO !End North


      ELSE !South
        DO IPP=1,NP-1
          K1  =K0 +KD*(IPP-1)
          KRSI=K0I+KD*(IPP-1)
          KRSJ=K0J+KD*(IPP-1)

          !Describe the movement in the edge vector
          step=(IPP-1)*ID(IDR)
          pos=I0+step


          IF(ITBND(IBND).EQ.1 .OR. ITBND(IBND).EQ.5)THEN
            IF(ITBND(IBND).EQ.5)THEN
              T=(TIME-TSTART)/(TSTOP-TSTART)
              !Create pointers and save information(area,HC ect.) in vectors
              boundSU(1,pos,IM)=3
              boundSU(3,pos,IM)=valg
              boundSU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
              boundS(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
              boundS(2,pos,IM)=HCBND(IBND)+T*DTHCBND(IBND)
              boundS(4,pos,IM)=TUBND(IBND)+T*DTTUBND(IBND)
            ELSE
              boundSU(1,pos,IM)=3
              boundSU(3,pos,IM)=valg
              boundSU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
              boundS(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
              boundS(2,pos,IM)=HCBND(IBND)
              boundS(4,pos,IM)=TUBND(IBND)
            ENDIF

          ENDIF

          IF(ITBND(IBND).EQ.2)THEN
            boundS(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
            boundS(2,pos,IM)=QBND(IBND)
          ENDIF

          IF(ITBND(IBND).EQ.3)THEN     !internal heat transfer
            RADIUS=YM(K1)
            RE=UBND(IBND,IPP)*RADIUS/VISCBND(IBND,IPP)
            ST=0.0791/RE**0.25*0.5/                                          &
            (1.+1.99*RE**(-0.125)*(PRBND(IBND,IPP)-1))

            boundSU(1,pos,IM)=3
            boundSU(3,pos,IM)=valg
            boundSU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
            boundS(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)

boundS(2,pos,IM)=ROBND(IBND,IPP)*UBND(IBND,IPP)*CPBND(IBND,IPP)*ST
            boundS(4,pos,IM)=TUBND(IBND)
          ENDIF
          IF(ITBND(IBND).EQ.4.OR.ITBND(IBND).EQ.6)THEN
            boundSU(1,pos,IM)=3
            boundSU(3,pos,IM)=valg
            boundSU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
            boundS(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
            boundS(2,pos,IM)=HCEBND(IBND,IPP)
            boundS(4,pos,IM)=TRBND(IBND,IPP)
          ENDIF

          IF(IRADBND(IBND).EQ.1)THEN   !add radiation heat transfer
            boundS(6,pos,IM)=(TRADBND(IBND)**4)
            boundS(7,pos,IM)=EMSBND(IBND)
```

```
                ENDIF

                IF(ITBND(IBND).EQ.7)THEN
                  boundSU(1,pos,IM)=4
                  boundSU(3,pos,IM)=valg
                  boundSU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                  boundS(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                  boundS(2,pos,IM)=1 !because of the pyrolysis, it has no
functionality.
                  boundS(4,pos,IM)=TUBND(IBND)
                ENDIF

               END DO !End South

          END IF !End North and South

        ELSE !J-Direction (Startposition=J0, step=JD(IDR))

          !East
          IF (I0.GT.(NI(IM)/2)) THEN
               DO IPP=1,NP-1
                 K1  =K0 +KD*(IPP-1)
                 KRSI=K0I+KD*(IPP-1)
                 KRSJ=K0J+KD*(IPP-1)

                 !Describe the movement in the edge vector
                 step=(IPP-1)*JD(IDR)
                 pos=J0+step


                IF(ITBND(IBND).EQ.1 .OR. ITBND(IBND).EQ.5)THEN
                  IF(ITBND(IBND).EQ.5)THEN
                     T=(TIME-TSTART)/(TSTOP-TSTART)
                     !Create pointers and save information(area,HC ect.) in vectors
                     boundEU(1,pos,IM)=3
                     boundEU(3,pos,IM)=valg
                     boundEU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                     boundE(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                     boundE(2,pos,IM)=HCBND(IBND)+T*DTHCBND(IBND)
                     boundE(4,pos,IM)=TUBND(IBND)+T*DTTUBND(IBND)
                  ELSE
                     boundEU(1,pos,IM)=3
                     boundEU(3,pos,IM)=valg
                     boundEU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                     boundE(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                     boundE(2,pos,IM)=HCBND(IBND)
                     boundE(4,pos,IM)=TUBND(IBND)
                  ENDIF

                ENDIF

                IF(ITBND(IBND).EQ.2)THEN
                     boundE(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                     boundE(2,pos,IM)=QBND(IBND)
                ENDIF

                IF(ITBND(IBND).EQ.3)THEN     !internal heat transfer
                  RADIUS=YM(K1)
                  RE=UBND(IBND,IPP)*RADIUS/VISCBND(IBND,IPP)
                  ST=0.0791/RE**0.25*0.5/                                          &
                  (1.+1.99*RE**(-0.125)*(PRBND(IBND,IPP)-1))

                  boundEU(1,pos,IM)=3
                  boundEU(3,pos,IM)=valg
                  boundEU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                  boundE(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)

boundE(2,pos,IM)=ROBND(IBND,IPP)*UBND(IBND,IPP)*CPBND(IBND,IPP)*ST
```

```
                  boundE(4,pos,IM)=TUBND(IBND)
                ENDIF
                IF(ITBND(IBND).EQ.4.OR.ITBND(IBND).EQ.6)THEN
                  boundEU(1,pos,IM)=3
                  boundEU(3,pos,IM)=valg
                  boundEU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                  boundE(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                  boundE(2,pos,IM)=HCEBND(IBND,IPP)
                  boundE(4,pos,IM)=TRBND(IBND,IPP)
                ENDIF


                IF(IRADBND(IBND).EQ.1)THEN   !add radiation heat transfer
                  boundE(6,pos,IM)=(TRADBND(IBND)**4)
                  boundE(7,pos,IM)=EMSBND(IBND)
                ENDIF

                IF(ITBND(IBND).EQ.7)THEN
                  boundEU(1,pos,IM)=4
                  boundEU(3,pos,IM)=valg
                  boundEU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                  boundE(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                  boundE(2,pos,IM)=1 !because of the pyrolysis, it has no
functionality.
                  boundE(4,pos,IM)=TUBND(IBND)
                ENDIF

              END DO !End East


          ELSE !West
               DO IPP=1,NP-1
                 K1  =K0 +KD*(IPP-1)
                 KRSI=K0I+KD*(IPP-1)
                 KRSJ=K0J+KD*(IPP-1)

                 !Describe the movement in the edge vector
                 step=IPP*JD(IDR)
                 pos=J0+step


                IF(ITBND(IBND).EQ.1 .OR. ITBND(IBND).EQ.5)THEN
                  IF(ITBND(IBND).EQ.5)THEN
                     T=(TIME-TSTART)/(TSTOP-TSTART)
                     !Create pointers and save information(area,HC ect.) in vectors
                     boundWU(1,pos,IM)=3
                     boundWU(3,pos,IM)=valg
                     boundWU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                     boundW(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                     boundW(2,pos,IM)=HCBND(IBND)+T*DTHCBND(IBND)
                     boundW(4,pos,IM)=TUBND(IBND)+T*DTTUBND(IBND)
                  ELSE
                     boundWU(1,pos,IM)=3
                     boundWU(3,pos,IM)=valg
                     boundWU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                     boundW(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                     boundW(2,pos,IM)=HCBND(IBND)
                     boundW(4,pos,IM)=TUBND(IBND)
                  ENDIF

                ENDIF

                IF(ITBND(IBND).EQ.2)THEN
                  boundW(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                  boundW(2,pos,IM)=QBND(IBND)
                ENDIF

                IF(ITBND(IBND).EQ.3)THEN     !internal heat transfer
```

```
                    RADIUS=YM(K1)
                    RE=UBND(IBND,IPP)*RADIUS/VISCBND(IBND,IPP)
                    ST=0.0791/RE**0.25*0.5/                                          &
                    (1.+1.99*RE**(-0.125)*(PRBND(IBND,IPP)-1))

                    boundWU(1,pos,IM)=3
                    boundWU(3,pos,IM)=valg
                    boundWU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                    boundW(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)

boundW(2,pos,IM)=ROBND(IBND,IPP)*UBND(IBND,IPP)*CPBND(IBND,IPP)*ST
                    boundW(4,pos,IM)=TUBND(IBND)
                  ENDIF
                  IF(ITBND(IBND).EQ.4.OR.ITBND(IBND).EQ.6)THEN
                    boundWU(1,pos,IM)=3
                    boundWU(3,pos,IM)=valg
                    boundWU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                    boundW(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                    boundW(2,pos,IM)=HCEBND(IBND,IPP)
                    boundW(4,pos,IM)=TRBND(IBND,IPP)
                  ENDIF

                  IF(IRADBND(IBND).EQ.1)THEN   !add radiation heat transfer
                    boundW(6,pos,IM)=(TRADBND(IBND)**4)
                    boundW(7,pos,IM)=EMSBND(IBND)
                  ENDIF

                  IF(ITBND(IBND).EQ.7)THEN
                    boundWU(1,pos,IM)=4
                    boundWU(3,pos,IM)=valg
                    boundWU(4,pos,IM)=KRSJ*IRS(IDR)+KRSI*JRS(IDR)
                    boundW(1,pos,IM)=AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)
                    boundW(2,pos,IM)=1 !because of the pyrolysis, it has no
functionality.
                    boundW(4,pos,IM)=TUBND(IBND)
                  ENDIF

                END DO !End West

            END IF !End East and West


          END IF !End directions


        END DO !End Boundary condition

   100 FORMAT(2I4,6F10.3)

        END SUBROUTINE !BORDERS_IMPLICIT



!Name: INIT_IMPLICIT
 !Author: Jørn Riise
 !Date: 30-06-2008
 !Description: Initializing pointer information for the interfaces and boundaries
        SUBROUTINE INIT_IMPLICIT
        USE GlobaleVariable
        implicit none

        !Local variables
        INTEGER          :: IIF,IMA,I0A,J0A,&
                            IDA,NPA,IMB,I0B,J0B,IDB,K0A,K0B,K0IA,K0JA,    &
                            K0IB,K0JB,KDA,KDB,KCA,KCB,KRSIA,KRSJA,KRSIB,  &
                            KRSJB,IPP,valgA,valgB,pos,step,stepS
        INTEGER, DIMENSION(4) :: ID,JD,ICA,JCA,ICB,JCB,IRS,JRS
```

```
DATA ID/ 1, 0,-1, 0/
DATA JD/ 0, 1, 0,-1/
DATA ICA/ 0,-1,-1, 0/
DATA JCA/ 0, 0,-1,-1/
DATA ICB/ 0, 0,-1,-1/
DATA JCB/-1, 0, 0,-1/
DATA IRS/ 1, 0, 1, 0/
DATA JRS/ 0, 1, 0, 1/

!Initializing shadow cells: Flux=0 -> insulated
boundN=boundN*0
boundS=boundS*0
boundW=boundW*0
boundE=boundE*0
boundNU=boundNU*0
boundSU=boundSU*0
boundWU=boundWU*0
boundEU=boundEU*0

!Bestemme interfasegrenser

DO 1 IIF=1,NIF
  IMA=IMAIF(IIF) !Block A
  I0A=I0AIF(IIF) !Start I
  J0A=J0AIF(IIF) !Start J
  IDA=IDAIF(IIF) !Direction
  NPA=NPAIF(IIF) !Number of nodes in the direction

  IMB=IMBIF(IIF) !Block B
  I0B=I0BIF(IIF) !Start I
  J0B=J0BIF(IIF) !Start J
  IDB=IDBIF(IIF) !Direction


  !Pointer information for the cell volumes along the edge
  K0A =I0A+ICA(IDA)+NI(IMA)*(J0A+JCA(IDA)-1)+NMP(IMA)
  K0B =I0B+ICB(IDB)+NI(IMB)*(J0B+JCB(IDB)-1)+NMP(IMB)
  K0IA=I0A         +NI(IMA)*(J0A+JCA(IDA)-1)+NMP(IMA)
  K0JA=I0A+ICA(IDA)+NI(IMA)*(J0A         -1)+NMP(IMA)
  K0IB=I0B         +NI(IMB)*(J0B+JCB(IDB)-1)+NMP(IMB)
  K0JB=I0B+ICB(IDB)+NI(IMB)*(J0B         -1)+NMP(IMB)

  !Similar coordinates in the 1D-vector along the edge
  KDA =ID(IDA)+NI(IMA)*JD(IDA)
  KDB =ID(IDB)+NI(IMB)*JD(IDB)

  !Declare vector layout
  valgA=JRS(IDA)+1
  valgB=JRS(IDB)+1

  !********
  !Block A.
  !********

  !I-direction (Startposition=I0A, step=ID(IDA))
  IF (IRS(IDA).EQ.1) THEN
         IF (J0A .GT. ( NJ(IMA)/2 )) THEN
                 !Decide startposition in the edge vector
                 !Positive if ID(IDA) > 0
                 IF (ID(IDA) .GT. 0 ) THEN
                    stepS=1
                 ELSE
                    stepS=0
                 END IF

                 DO IPP=1,NPA-1
```

```
                              !Pointer information of cell volumes and the edge face
in 1D-vector
                              KCA  =K0A +KDA*(IPP-1)
                              KCB  =K0B +KDB*(IPP-1)
                              KRSIA=K0IA+KDA*(IPP-1)
                              KRSJA=K0JA+KDA*(IPP-1)
                              KRSIB=K0IB+KDB*(IPP-1)
                              KRSJB=K0JB+KDB*(IPP-1)

                              step=(IPP-stepS)*ID(IDA)
                              pos=I0A+step
                              !Save pointer information in vectors along the edge
                              boundNU(1,pos,IMA)=valgA
                              boundNU(2,pos,IMA)=KRSJA
                              boundNU(3,pos,IMA)=valgB
                              boundNU(4,pos,IMA)=KRSJB*IRS(IDB)+KRSIB*JRS(IDB)
                              boundNU(5,pos,IMA)=KCB
                              boundN(1,pos,IMA)=AJ(KRSJA)
                        END DO

                ELSE
                        IF (ID(IDA) .GT. 0 ) THEN
                           stepS=1
                        ELSE
                           stepS=0
                        END IF
                        DO IPP=1,NPA-1
                              !Pointer information of cell volumes and the edge face
in 1D-vector
                              KCA  =K0A +KDA*(IPP-1)
                              KCB  =K0B +KDB*(IPP-1)
                              KRSIA=K0IA+KDA*(IPP-1)
                              KRSJA=K0JA+KDA*(IPP-1)
                              KRSIB=K0IB+KDB*(IPP-1)
                              KRSJB=K0JB+KDB*(IPP-1)

                              step=(IPP-stepS)*ID(IDA)
                              pos=I0A+step
                              !Save pointer information in vectors along the edge
                              boundSU(1,pos,IMA)=valgA
                              boundSU(2,pos,IMA)=KRSJA
                              boundSU(3,pos,IMA)=valgB
                              boundSU(4,pos,IMA)=KRSJB*IRS(IDB)+KRSIB*JRS(IDB)
                              boundSU(5,pos,IMA)=KCB
                              boundS(1,pos,IMA)=AJ(KRSJA)
                        END DO

                END IF !End I-direction

        !J-direction (Startposition=J0A, step=JD(IDA))
        ELSE
                IF (I0A .GT. ( NI(IMA)/2 )) THEN
                        IF (JD(IDA) .GT. 0 ) THEN
                           stepS=1
                        ELSE
                           stepS=0
                        END IF
                        DO IPP=1,NPA-1
                              !Pointer information of cell volumes and the edge face
in 1D-vector
                              KCA  =K0A +KDA*(IPP-1)
                              KCB  =K0B +KDB*(IPP-1)
                              KRSIA=K0IA+KDA*(IPP-1)
                              KRSJA=K0JA+KDA*(IPP-1)
                              KRSIB=K0IB+KDB*(IPP-1)
                              KRSJB=K0JB+KDB*(IPP-1)

                              step=(IPP-stepS)*JD(IDA)
```

```
                        pos=J0A+step
                        !Save pointer information in vectors along the edge
                        boundEU(1,pos,IMA)=valgA
                        boundEU(2,pos,IMA)=KRSIA
                        boundEU(3,pos,IMA)=valgB
                        boundEU(4,pos,IMA)=KRSJB*IRS(IDB)+KRSIB*JRS(IDB)
                        boundEU(5,pos,IMA)=KCB
                        boundE(1,pos,IMA)=AI(KRSIA)
                    END DO
            ELSE
                    IF (JD(IDA) .GT. 0 ) THEN
                       stepS=1
                    ELSE
                       stepS=0
                    END IF
                    DO IPP=1,NPA-1
                        !Pointer information of cell volumes and the edge face
in 1D-vector
                        KCA  =K0A +KDA*(IPP-1)
                        KCB  =K0B +KDB*(IPP-1)
                        KRSIA=K0IA+KDA*(IPP-1)
                        KRSJA=K0JA+KDA*(IPP-1)
                        KRSIB=K0IB+KDB*(IPP-1)
                        KRSJB=K0JB+KDB*(IPP-1)

                        step=(IPP-stepS)*JD(IDA)
                        pos=J0A+step
                        !Save pointer information in vectors along the edge
                        boundWU(1,pos,IMA)=valgA
                        boundWU(2,pos,IMA)=KRSIA
                        boundWU(3,pos,IMA)=valgB
                        boundWU(4,pos,IMA)=KRSJB*IRS(IDB)+KRSIB*JRS(IDB)
                        boundWU(5,pos,IMA)=KCB
                        boundW(1,pos,IMA)=AI(KRSIA)
                    END DO

            END IF !End J-direction

        END IF !End Block A

        !********
        !Block B.
        !********
        !I-direction (Startposition=I0B, step=ID(IDB))
        IF (IRS(IDB).EQ.1) THEN
               IF (J0B .GT. ( NJ(IMB)/2 )) THEN
                    IF (ID(IDB) .GT. 0 ) THEN
                       stepS=1
                    ELSE
                       stepS=0
                    END IF
                    DO IPP=1,NPA-1
                        !Pointer information of cell volumes and the edge face
in 1D-vector
                        KCA  =K0A +KDA*(IPP-1)
                        KCB  =K0B +KDB*(IPP-1)
                        KRSIA=K0IA+KDA*(IPP-1)
                        KRSJA=K0JA+KDA*(IPP-1)
                        KRSIB=K0IB+KDB*(IPP-1)
                        KRSJB=K0JB+KDB*(IPP-1)

                        step=(IPP-stepS)*ID(IDB)
                        pos=I0B+step
                        !Save pointer information in vectors along the edge
                        boundNU(1,pos,IMB)=valgA
                        boundNU(2,pos,IMB)=KRSJA*IRS(IDA)+KRSIA*JRS(IDA)
                        boundNU(3,pos,IMB)=valgB
                        boundNU(4,pos,IMB)=KRSJB
```

```
                              boundNU(5,pos,IMB)=KCA
                              boundN(1,pos,IMB)=AJ(KRSJA)*IRS(IDA)+AI(KRSIA)*JRS(IDA)
                        END DO
                ELSE
                        IF (ID(IDB) .GT. 0 ) THEN
                           stepS=1
                        ELSE
                           stepS=0
                        END IF
                        DO IPP=1,NPA-1
                              !Pointer information of cell volumes and the edge face
in 1D-vector
                              KCA  =K0A +KDA*(IPP-1)
                              KCB  =K0B +KDB*(IPP-1)
                              KRSIA=K0IA+KDA*(IPP-1)
                              KRSJA=K0JA+KDA*(IPP-1)
                              KRSIB=K0IB+KDB*(IPP-1)
                              KRSJB=K0JB+KDB*(IPP-1)

                              step=(IPP-stepS)*ID(IDB)
                              pos=I0B+step
                              !Save pointer information in vectors along the edge
                              boundSU(1,pos,IMB)=valgA
                              boundSU(2,pos,IMB)=KRSJA*IRS(IDA)+KRSIA*JRS(IDA)
                              boundSU(3,pos,IMB)=valgB
                              boundSU(4,pos,IMB)=KRSJB
                              boundSU(5,pos,IMB)=KCA
                              boundS(1,pos,IMB)=AJ(KRSJA)*IRS(IDA)+AI(KRSIA)*JRS(IDA)
                        END DO

                END IF !End I-direction

        !J-direction (Startposition=J0B, step=JD(IDB))
        ELSE
                IF (I0B .GT. ( NI(IMB)/2 )) THEN
                        IF (JD(IDB) .GT. 0 ) THEN
                           stepS=1
                        ELSE
                           stepS=0
                        END IF
                        DO IPP=1,NPA-1
                              !Pointer information of cell volumes and the edge face
in 1D-vector
                              KCA  =K0A +KDA*(IPP-1)
                              KCB  =K0B +KDB*(IPP-1)
                              KRSIA=K0IA+KDA*(IPP-1)
                              KRSJA=K0JA+KDA*(IPP-1)
                              KRSIB=K0IB+KDB*(IPP-1)
                              KRSJB=K0JB+KDB*(IPP-1)

                              step=(IPP-stepS)*JD(IDB)
                              pos=J0B+step
                              !Save pointer information in vectors along the edge
                              boundEU(1,pos,IMB)=valgA
                              boundEU(2,pos,IMB)=KRSJA*IRS(IDA)+KRSIA*JRS(IDA)
                              boundEU(3,pos,IMB)=valgB
                              boundEU(4,pos,IMB)=KRSIB
                              boundEU(5,pos,IMB)=KCA
                              boundE(1,pos,IMB)=AJ(KRSJA)*IRS(IDA)+AI(KRSIA)*JRS(IDA)
                        END DO
                ELSE
                        IF (JD(IDB) .GT. 0 ) THEN
                           stepS=1
                        ELSE
                           stepS=0
                        END IF
                        DO IPP=1,NPA-1
```

```
                              !Pointer information of cell volumes and the edge face
in 1D-vector
                              KCA  =K0A +KDA*(IPP-1)
                              KCB  =K0B +KDB*(IPP-1)
                              KRSIA=K0IA+KDA*(IPP-1)
                              KRSJA=K0JA+KDA*(IPP-1)
                              KRSIB=K0IB+KDB*(IPP-1)
                              KRSJB=K0JB+KDB*(IPP-1)

                              step=(IPP-stepS)*JD(IDB)
                              pos=J0B+step
                              !Save pointer information in vectors along the edge
                              boundWU(1,pos,IMB)=valgA
                              boundWU(2,pos,IMB)=KRSJA*IRS(IDA)+KRSIA*JRS(IDA)
                              boundWU(3,pos,IMB)=valgB
                              boundWU(4,pos,IMB)=KRSIB
                              boundWU(5,pos,IMB)=KCA
                              boundW(1,pos,IMB)=AJ(KRSJA)*IRS(IDA)+AI(KRSIA)*JRS(IDA)
                         END DO
                END IF !End J-direction

        END IF


    1 CONTINUE
  100 FORMAT(I4,6F10.3)
  101 FORMAT(10I4)

      END SUBROUTINE !End INIT_IMPLICIT


      SUBROUTINE FLUX
      USE GlobaleVariable
      implicit none

      CALL FLUX1
      IF(NIF .NE.0)CALL FLUX2
      IF(NBND.NE.0)CALL FLUX3
      CALL FLUX5
      RETURN
      END

      SUBROUTINE FLUX1                !    INTERNAL
      USE GlobaleVariable
      implicit none

! Lokale variable
      INTEGER       :: IMESH,I,J,K1,K2,K3
      DOUBLE PRECISION :: QI,QJ


      DO 1 IMESH=1,NMESH
        DO 2 J=1,NJ(IMESH)-1
        DO 2 I=1,NI(IMESH)-1
          K1=I +NI(IMESH)*(J-1)+NMP(IMESH)
          DQ(K1)=0.
   2    CONTINUE
   1  CONTINUE

      DO 3 IMESH=1,NMESH
        DO 4 J=1,NJ(IMESH)-1
        DO 4 I=1,NI(IMESH)-2
          K1=I +NI(IMESH)*(J-1)+NMP(IMESH)
          K2=K1+1
!     I- DIRECTION FLUXES
          QI=AI(K2)/RI(K2)*(TEMP(K1)-TEMP(K2))
!     UPDATE FLUX INTO CELL K2
          DQ(K2)=DQ(K2)+QI
```

```
!     UPDATE FLUX OUT OF CELL K1
          DQ(K1)=DQ(K1)-QI

   4    CONTINUE
        DO 5 J=1,NJ(IMESH)-2
        DO 5 I=1,NI(IMESH)-1
          K1=I +NI(IMESH)*(J-1)+NMP(IMESH)
          K3=K1+NI(IMESH)
!     J- DIRECTION FLUXES
          QJ=AJ(K3)/RJ(K3)*(TEMP(K1)-TEMP(K3))
!     UPDATE FLUX INTO CELL K3
          DQ(K3)=DQ(K3)+QJ
!     UPDATE FLUX OUT OF CELL K1
          DQ(K1)=DQ(K1)-QJ
   5    CONTINUE
   3  CONTINUE

      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE FLUX2                !    INTERFACES
      USE GlobaleVariable
      implicit none

! Lokale variable
      INTEGER          :: IIF,IMA,I0A,J0A,&
                          IDA,NPA,IMB,I0B,J0B,IDB,K0A,K0B,K0IA,K0JA,    &
                          K0IB,K0JB,KDA,KDB,KCA,KCB,KRSIA,KRSJA,KRSIB,  &
                          KRSJB,IPP !,k1,npb,K0AP
      DOUBLE PRECISION :: RCONDA,RCONDB,RCOND,ABND,QIF
      INTEGER, DIMENSION(4) :: ID,JD,ICA,JCA,ICB,JCB,IRS,JRS


      DATA ID/ 1, 0,-1, 0/
      DATA JD/ 0, 1, 0,-1/
      DATA ICA/ 0,-1,-1, 0/
      DATA JCA/ 0, 0,-1,-1/
      DATA ICB/ 0, 0,-1,-1/
      DATA JCB/-1, 0, 0,-1/
      DATA IRS/ 1, 0, 1, 0/
      DATA JRS/ 0, 1, 0, 1/

!     START OUTERMOST LOOP OVER ALL INTERFACES

      DO 1 IIF=1,NIF
        IMA=IMAIF(IIF)
        I0A=I0AIF(IIF)
        J0A=J0AIF(IIF)
        IDA=IDAIF(IIF)
        NPA=NPAIF(IIF)

        IMB=IMBIF(IIF)
        I0B=I0BIF(IIF)
        J0B=J0BIF(IIF)
        IDB=IDBIF(IIF)
!      NPB=NPBIF(IIF)

!     COMPUTE NECESSARY POINTER INFORMATION

!      K0AP=I0A          +NI(IMA)*(J0A          -1)+NMP(IMA)
        K0A =I0A+ICA(IDA)+NI(IMA)*(J0A+JCA(IDA)-1)+NMP(IMA)
        K0B =I0B+ICB(IDB)+NI(IMB)*(J0B+JCB(IDB)-1)+NMP(IMB)
        K0IA=I0A          +NI(IMA)*(J0A+JCA(IDA)-1)+NMP(IMA)
        K0JA=I0A+ICA(IDA)+NI(IMA)*(J0A          -1)+NMP(IMA)
        K0IB=I0B          +NI(IMB)*(J0B+JCB(IDB)-1)+NMP(IMB)
        K0JB=I0B+ICB(IDB)+NI(IMB)*(J0B          -1)+NMP(IMB)

        KDA =ID(IDA)+NI(IMA)*JD(IDA)
```

```
        KDB =ID(IDB)+NI(IMB)*JD(IDB)

        DO 2 IPP=1,NPA-1
!          K1   =K0AP+KDA*(IPP-1)
          KCA  =K0A +KDA*(IPP-1)
          KCB  =K0B +KDB*(IPP-1)
          KRSIA=K0IA+KDA*(IPP-1)
          KRSJA=K0JA+KDA*(IPP-1)
          KRSIB=K0IB+KDB*(IPP-1)
          KRSJB=K0JB+KDB*(IPP-1)

          RCONDA=RJ(KRSJA)*IRS(IDA)+RI(KRSIA)*JRS(IDA)
          RCONDB=RJ(KRSJB)*IRS(IDB)+RI(KRSIB)*JRS(IDB)
          RCOND=RCONDA+RCONDB
          ABND =AJ(KRSJA)*IRS(IDA)+AI(KRSIA)*JRS(IDA)

          QIF=ABND/RCOND*(TEMP(KCA)-TEMP(KCB))
!     UPDATE FLUX INTO CELL KCB
          DQ(KCB)=DQ(KCB)+QIF
!     UPDATE FLUX OUT OF CELL KCA
          DQ(KCA)=DQ(KCA)-QIF
    2   CONTINUE
    1 CONTINUE
  100 FORMAT(I4,6F10.3)
  101 FORMAT(10I4)
      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE FLUX3                 !    BOUNDARIES
      USE GlobaleVariable
      implicit none

! Lokale variable
      INTEGER          :: ID,JD,IC,JC,IRS,JRS,IM,I0,J0,IDR,NP,K0,K0C,    &
                          K0I,K0J,KD,K1,KC,KRSI,KRSJ,IBND,IPP,IT
      DOUBLE PRECISION :: SIGMA,RCOND,ABND,T,HC,TU,RCONV,RADIUS,RE,ST,   &
                          RBND,RA !,TWALL,AA
      DOUBLE PRECISION :: TMIN,TMAX,T1,T2,XSI
      DOUBLE PRECISION :: TEMPJ,KONDJ,RHOJ,CPJ,ALFAJ,KINJ,PRJ
      INTEGER          :: IIMAT,N
      DIMENSION ID(4),JD(4),IC(4),JC(4),IRS(4),JRS(4)
      INTEGER :: done

      DATA ID / 1, 0,-1, 0/
      DATA JD / 0, 1, 0,-1/
      DATA IC / 0,-1,-1, 0/
      DATA JC / 0, 0,-1,-1/
      DATA IRS/ 1, 0, 1, 0/
      DATA JRS/ 0, 1, 0, 1/

      SIGMA=5.67*1.E-08

      DO 1 IBND=1,NBND
        IM =IMBND(IBND)
        I0 =I0BND(IBND)
        J0 =J0BND(IBND)
        IDR=IDBND(IBND)
        NP =NPBND(IBND)
        K0 =I0         +NI(IM)*(J0         -1)+NMP(IM)
        K0C=I0+IC(IDR)+NI(IM)*(J0+JC(IDR)-1)+NMP(IM)
        K0I=I0         +NI(IM)*(J0+JC(IDR)-1)+NMP(IM)
        K0J=I0+IC(IDR)+NI(IM)*(J0         -1)+NMP(IM)
        KD= ID(IDR)+NI(IM)* JD(IDR)
        DO 2 IPP=1,NP-1
          K1  =K0 +KD*(IPP-1)
          KC  =K0C+KD*(IPP-1)
          KRSI=K0I+KD*(IPP-1)
          KRSJ=K0J+KD*(IPP-1)
```

```
          RCOND=RJ(KRSJ)*IRS(IDR)+RI(KRSI)*JRS(IDR)
          ABND =AJ(KRSJ)*IRS(IDR)+AI(KRSI)*JRS(IDR)


          IF(ITBND(IBND).EQ.1 .OR. ITBND(IBND).EQ.5)THEN
            IF(ITBND(IBND).EQ.5)THEN
              !Material hente rutine
                TEMPJ=TEMP(KC)
                    IIMAT= 6
             N=NINT(AM_DATA(IIMAT))
             TMIN=AM_TEMP(IIMAT,1)
             TMAX=AM_TEMP(IIMAT,N)

       IF(TEMPJ.LE.TMIN)THEN
              T1=TMIN
              T2=TMIN
              RHOJ=AM_RO(IIMAT,1)
              KONDJ=AM_TCJ(IIMAT,1)
                 KINJ=AM_TCI(IIMAT,1)
              CPJ=AM_CP(IIMAT,1)
       ENDIF
       IF(TEMPJ.GE.TMAX)THEN
              T1=TMAX
              T2=TMAX
              RHOJ=AM_RO(IIMAT,N)
              KONDJ=AM_TCJ(IIMAT,N)
                 KINJ=AM_TCI(IIMAT,N)
              CPJ=AM_CP(IIMAT,N)
       ENDIF
       IF(TMIN.LT.TEMPJ .AND. TEMPJ.LT.TMAX)THEN
        done=0
        DO 3 IT=1,N-1
              T1=AM_TEMP(IIMAT,IT)
              T2=AM_TEMP(IIMAT,IT+1)
              XSI=(TEMPJ-T1)/(T2-T1)
        IF(T2.GT.TEMPJ .AND. done.EQ.0)THEN
              RHOJ=(1.-XSI)*AM_RO(IIMAT,IT) +XSI*AM_RO(IIMAT,IT+1)
              KONDJ=(1.-XSI)*AM_TCJ(IIMAT,IT)+XSI*AM_TCJ(IIMAT,IT+1)
                 KINJ=(1.-XSI)*AM_TCI(IIMAT,IT)+XSI*AM_TCI(IIMAT,IT+1)
              CPJ=(1.-XSI)*AM_CP(IIMAT,IT) +XSI*AM_CP(IIMAT,IT+1)
                 done=1

       ENDIF
   3     CONTINUE
         ENDIF



              ! Slutt hente gassdata


              ALFAJ=KONDJ/(RHOJ*CPJ)
              PRJ=KINJ/ALFAJ
              T=(TIME-TSTART)/(TSTOP-TSTART)
              TU=TUBND(IBND)+T*DTTUBND(IBND)
              RA=(9.81*(HCBND(IBND)**3)/(TU*KINJ*ALFAJ))*(TU-TEMP(KC))

HC=((06+((0.387*RA**(1/6))/((1+((0.559/PRJ)**(9/16)))**(8/27))))**2)*(KONDJ/HCBND(I
BND))


            !WRITE(6,*)RHOJ, CPJ, KINJ, KONDJ, ALFAJ, PRJ
             ! HC=HCBND(IBND)+T*DTHCBND(IBND)

            ELSE
              HC=HCBND(IBND)
```

```
              TU=TUBND(IBND)
            ENDIF

            RCONV=1./HC
            RBND=RCONV+RCOND

            DQ(KC)=DQ(KC)+ABND/RBND*(TU-TEMP(KC))
          ENDIF

          IF(ITBND(IBND).EQ.2)THEN
            DQ(KC)=DQ(KC)+QBND(IBND)*ABND
          ENDIF

          IF(ITBND(IBND).EQ.3)THEN     !internal heat transfer
            RADIUS=YM(K1)
            RE=UBND(IBND,IPP)*RADIUS/VISCBND(IBND,IPP)
            ST=0.0791/RE**0.25*0.5/                                          &
               (1.+1.99*RE**(-0.125)*(PRBND(IBND,IPP)-1))
            HC=ROBND(IBND,IPP)*UBND(IBND,IPP)*CPBND(IBND,IPP)*ST
            RCONV=1./HC
            RBND=RCONV+RCOND
            DQ(KC)=DQ(KC)+ABND/RBND*(TUBND(IBND)-TEMP(KC))
          ENDIF
          IF(ITBND(IBND).EQ.4.OR.ITBND(IBND).EQ.6)THEN
            RCONV=1./HCEBND(IBND,IPP)
            RBND=RCONV+RCOND
            DQ(KC)=DQ(KC)+ABND/RBND*(TRBND(IBND,IPP)-TEMP(KC))
          ENDIF

          IF(IRADBND(IBND).EQ.1)THEN   !add radiation heat transfer
!            TWALL=TEMP(KC)    !0 ORDENS TILNARMING BOR FORBEDRES
            DQ(KC)=DQ(KC)+ABND*SIGMA*EMSBND(IBND)*                           &
                          ((TRADBND(IBND))**4-(TEMP(KC))**4)
!            aa=ABND*SIGMA*EMSBND(IBND)*(0-(TEMP(KC))**4)
          ENDIF

          !Konstant temperatur på grense
          IF(ITBND(IBND).EQ.7) THEN
            RBND=RCOND
            DQ(KC)=DQ(KC)+ABND/RBND*(TUBND(IBND)-TEMP(KC))

          END IF


    2   CONTINUE
    1 CONTINUE

  100 FORMAT(2I4,6F10.3)
      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE FLUX5
      USE GlobaleVariable
      implicit none

! Lokale variable
      INTEGER          :: IMESH,I,J,K


      DO 1 IMESH=1,NMESH
        DO 2 J=1,NJ(IMESH)-1
        DO 2 I=1,NI(IMESH)-1
          K=I +NI(IMESH)*(J-1)+NMP(IMESH)
          DTEMP(K)=DQ(K)/(RO(K)*C(K)*VOL(K))*TSF
   2    CONTINUE
   1  CONTINUE
      RETURN
      END
```

```
      SUBROUTINE STEP(ISTEP)
      USE GlobaleVariable
      implicit none
      INTEGER          :: ISTEP

! Lokale variable
      INTEGER          :: IMESH,I,J,K,IPR



      IPR=MOD(ISTEP,25)
      IF(ISTEP.EQ.1)IPR=0
      IF(IPR.EQ.0)THEN
        CALL PICKMDATA
        CALL RESMAT
        CALL LTSP
      ENDIF

      CALL FLUX

      DO 3 IMESH=1,NMESH
        DO 4 J=1,NJ(IMESH)-1
        DO 4 I=1,NI(IMESH)-1
          K=I+NI(IMESH)*(J-1)+NMP(IMESH)
         TEMP(K)=TEMP(K)+CFL*DTEMP(K)
    4   CONTINUE
    3 CONTINUE
      RETURN
      END

!-------------------------------------------------------------------------------
    SUBROUTINE MATRSAVE
    USE GlobaleVariable
    IMPLICIT NONE

    ! LOKALE VARIABLE
    INTEGER          :: IMesh, I,J,K,L

    L=1
    OPEN(50,FILE=MATRFIL,FORM='FORMATTED',STATUS='REPLACE')
    WRITE (50,*) 'TITLE = "MATERIAL"'
    WRITE (50,*) 'VARIABLES = "X","Y","Matr"'

    DO IMESH=1,NMESH
      WRITE(50,*) 'ZONE F=POINT, I=',NI(IMESH),',J=',NJ(IMESH)
      DO J=1,NJ(IMESH)
        DO I=1,NI(IMESH)
         K=I+NI(IMESH)*(J-1)+NMP(IMESH)
         WRITE(50,*) XM(L),YM(L),Mat(K)
         L=L+1
        ENDDO
      ENDDO
    ENDDO

    RETURN
    END SUBROUTINE

      SUBROUTINE MATERIALS
      USE GlobaleVariable
      IMPLICIT NONE


! Lokale variable
      INTEGER          :: IREG,IM,NMPL,NIL,I,J,K!,NJL
```

```
      DO 3 IREG=1,NREG
      IM=IMR(IREG)
      NMPL=NMP(IM)
      NIL = NI(IM)
!      NJL = NJ(IM)
      DO 4 J=JR1(IREG),JR2(IREG)-1
      DO 4 I=IR1(IREG),IR2(IREG)-1
      K=I+NIL*(J-1)+NMPL
      MAT(K)=IMAT(IREG)
    4 CONTINUE
    3 CONTINUE
      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE RESMAT
      USE GlobaleVariable
      IMPLICIT NONE

! Lokale variable
      INTEGER          :: IMESH,I,J,K1,K2,K3
      DOUBLE PRECISION :: R1,R2


      DO IMESH=1,NMESH
       DO J=1,NJ(IMESH)-1
        DO I=1,NI(IMESH)-1
         K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
         K2=K1+1
         K3=K1+NI(IMESH)
!        I- DIRECTION RESISTANCE
         IF (I.EQ.1) THEN
           RI(K1)=.5*DI(K1)/TCI(K1)
         END IF
         IF (I.EQ.(NI(IMESH)-1)) THEN
           RI(K2)=.5*DI(K1)/TCI(K1)
         ELSE
           R1=.5*DI(K1)/TCI(K1)
           R2=.5*DI(K2)/TCI(K2)
           RI(K2)=R1+R2
         END IF
!        J- DIRECTION RESISTANCE
         IF (J.EQ.1) THEN
           RJ(K1)=.5*DJ(K1)/TCJ(K1)
         END IF
         IF (J.EQ.(NJ(IMESH)-1)) THEN
           RJ(K3)=.5*DJ(K1)/TCJ(K1)
         ELSE
           R1=.5*DJ(K1)/TCJ(K1)
           R2=.5*DJ(K3)/TCJ(K3)
           RJ(K3)=R1+R2
         END IF
        END DO
       END DO
      END DO

      RETURN
      END
!-----------------------------------------------------------------------
      SUBROUTINE PICKMDATA
      USE GlobaleVariable
      IMPLICIT NONE
! Lokale variable
      INTEGER          :: I,J,IMESH,K1,IIMAT,N,IT
      DOUBLE PRECISION :: T,TMIN,TMAX,T1,T2,XSI



      DO IMESH=1,NMESH
```

```
        DO J=1,NJ(IMESH)-1
         DO I=1,NI(IMESH)-1
          K1=I+NI(IMESH)*(J-1)+NMP(IMESH)
          T=TEMP(K1)
          IIMAT=MAT(K1)
          N=NINT(AM_DATA(IIMAT))
          TMIN=AM_TEMP(IIMAT,1)
          TMAX=AM_TEMP(IIMAT,N)
          IF (T.LE.TMIN) THEN
            T1=TMIN
            T2=TMIN
            RO(K1) =AM_RO(IIMAT,1)
            TCI(K1)=AM_TCI(IIMAT,1)
            TCJ(K1)=AM_TCJ(IIMAT,1)
            C(K1)  =AM_CP(IIMAT,1)
          ELSE IF (T.GE.TMAX) THEN
            T1=TMAX
            T2=TMAX
            RO(K1) =AM_RO(IIMAT,N)
            TCI(K1)=AM_TCI(IIMAT,N)
            TCJ(K1)=AM_TCJ(IIMAT,N)
            C(K1)  =AM_CP(IIMAT,N)
          ELSE
            DO IT=1,N-1
              T1=AM_TEMP(IIMAT,IT)
              T2=AM_TEMP(IIMAT,IT+1)
              XSI=(T-T1)/(T2-T1)
              IF(T2.GT.T)EXIT
            END DO
            RO(K1) =(1.-XSI)*AM_RO(IIMAT,IT) +XSI*AM_RO(IIMAT,IT+1)
            TCI(K1)=(1.-XSI)*AM_TCI(IIMAT,IT)+XSI*AM_TCI(IIMAT,IT+1)
            TCJ(K1)=(1.-XSI)*AM_TCJ(IIMAT,IT)+XSI*AM_TCJ(IIMAT,IT+1)
            C(K1)  =(1.-XSI)*AM_CP(IIMAT,IT) +XSI*AM_CP(IIMAT,IT+1)
          END IF
         END DO !End I
        END DO !End J
       END DO !End MESH

       RETURN
       END
!-----------------------------------------------------------------------
       SUBROUTINE DEFMATDATA
       USE GlobaleVariable
       IMPLICIT NONE

! aluminium (Typiske data for legert Al)
       AM_DATA(1)=5
       AM_TEMP(1,1)=  -100.+273.15
       AM_TEMP(1,2)=  273.15+ 20.
       AM_TEMP(1,3)=  273.15+ 100.
       AM_TEMP(1,4)=  273.15+ 200.
       AM_TEMP(1,5)=  273.15+ 300.

       AM_CP  (1,1)=   955.
       AM_CP  (1,2)=   960.
       AM_CP  (1,3)=   962.
       AM_CP  (1,4)=   967.
       AM_CP  (1,5)=   983.

       AM_TCI (1,1)=   110.
       AM_TCI (1,2)=   130.
       AM_TCI (1,3)=   148.
       AM_TCI (1,4)=   168.
       AM_TCI (1,5)=   188.

       AM_TCJ (1,1)=   110.
       AM_TCJ (1,2)=   130.
       AM_TCJ (1,3)=   148.
```

```
      AM_TCJ (1,4)=   168.
      AM_TCJ (1,5)=   188.

      AM_RO  (1,1)=  1758.
      AM_RO  (1,2)=  1758.
      AM_RO  (1,3)=  1758.
      AM_RO  (1,4)=  1758.
      AM_RO  (1,5)=  1758.
! titan (Typiske data for Ti-legering)
      AM_DATA(9)=6
      AM_TEMP(9,1)=  -173.+273.15
      AM_TEMP(9,2)=  273.15+  25.
      AM_TEMP(9,3)=  273.15+ 100.
      AM_TEMP(9,4)=  273.15+ 204.
      AM_TEMP(9,5)=  273.15+ 426.
      AM_TEMP(9,6)=  273.15+ 648.

      AM_CP  (9,1)=   298.2
      AM_CP  (9,2)=   523.3
      AM_CP  (9,3)=   546.2
      AM_CP  (9,4)=   569.
      AM_CP  (9,5)=   609.8
      AM_CP  (9,6)=   660.4

      AM_TCI (9,1)=    31.2
      AM_TCI (9,2)=    21.9
      AM_TCI (9,3)=    20.7
      AM_TCI (9,4)=    20.1
      AM_TCI (9,5)=    19.5
      AM_TCI (9,6)=    16.3

      AM_TCJ (9,1)=    31.2
      AM_TCJ (9,2)=    21.9
      AM_TCJ (9,3)=    20.7
      AM_TCJ (9,4)=    20.1
      AM_TCJ (9,5)=    19.5
      AM_TCJ (9,6)=    16.3

      AM_RO  (9,1)=  4500.
      AM_RO  (9,2)=  4500.
      AM_RO  (9,3)=  4500.
      AM_RO  (9,4)=  4500.
      AM_RO  (9,5)=  4500.
      AM_RO  (9,6)=  4500.
! steel (Typiske data .. ser ut som karbonstål)
      AM_DATA(2)=5
      AM_TEMP(2,1)=   -56.+273.15
      AM_TEMP(2,2)=   204. +273.15
      AM_TEMP(2,3)=   426. +273.15
      AM_TEMP(2,4)=   648. +273.15
      AM_TEMP(2,5)=  1500. +273.15

      AM_CP  (2,1)=   419.
      AM_CP  (2,2)=   519.
      AM_CP  (2,3)=   620.
      AM_CP  (2,4)=   754.
      AM_CP  (2,5)=   800.

      AM_TCI (2,1)=    43.1
      AM_TCI (2,2)=    42.2
      AM_TCI (2,3)=    38.6
      AM_TCI (2,4)=    32.2
      AM_TCI (2,5)=    25.0

      AM_TCJ (2,1)=    43.1
      AM_TCJ (2,2)=    42.2
      AM_TCJ (2,3)=    38.6
      AM_TCJ (2,4)=    32.2
```

```
      AM_TCJ (2,5)=    25.0

      AM_RO  (2,1)=  7745.
      AM_RO  (2,2)=  7745.
      AM_RO  (2,3)=  7745.
      AM_RO  (2,4)=  7745.
      AM_RO  (2,5)=  7745.

! SiPh (Inkludererer effekt av forgassing/forkulling)
      AM_DATA(3)=10
      AM_TEMP(3, 1)= 21. +273.15
      AM_TEMP(3, 2)= 149.+273.15
      AM_TEMP(3, 3)=   371.   +273.15
      AM_TEMP(3, 4)=   560.   +273.15
      AM_TEMP(3, 5)=   838.   +273.15
      AM_TEMP(3, 6)=  1115.   +273.15
      AM_TEMP(3, 7)=  1393.   +273.15
      AM_TEMP(3, 8)=  1949.   +273.15
      AM_TEMP(3, 9)=  2504.   +273.15
      AM_TEMP(3,10)=  3060.   +273.15

      AM_CP  (3, 1)=   754.
      AM_CP  (3, 2)=  1005.
      AM_CP  (3, 3)=  1193.
      AM_CP  (3, 4)=  2190.
      AM_CP  (3, 5)=  2400.
      AM_CP  (3, 6)=  1298.
      AM_CP  (3, 7)=  1486.
      AM_CP  (3, 8)=  1486.
      AM_CP  (3, 9)=  1486.
      AM_CP  (3,10)=  1486.

      AM_TCI (3, 1)=     0.459
      AM_TCI (3, 2)=     0.513
      AM_TCI (3, 3)=     0.600
      AM_TCI (3, 4)=     0.627
      AM_TCI (3, 5)=     0.750
      AM_TCI (3, 6)=     0.850
      AM_TCI (3, 7)=     1.000
      AM_TCI (3, 8)=     1.812
      AM_TCI (3, 9)=     3.744

      AM_TCI (3,10)=     6.678
      AM_TCJ (3, 1)=     0.303
      AM_TCJ (3, 2)=     0.337
      AM_TCJ (3, 3)=     0.396
      AM_TCJ (3, 4)=     0.400
      AM_TCJ (3, 5)=     0.439
      AM_TCJ (3, 6)=     0.461
      AM_TCJ (3, 7)=     0.492
      AM_TCJ (3, 8)=     1.022
      AM_TCJ (3, 9)=     2.819
      AM_TCJ (3,10)=     3.112

      AM_RO  (3, 1)=  1758.
      AM_RO  (3, 2)=  1758.
      AM_RO  (3, 3)=  1758.
      AM_RO  (3, 4)=  1672.
      AM_RO  (3, 5)=  1548.
      AM_RO  (3, 6)=  1548.
      AM_RO  (3, 7)=  1548.
      AM_RO  (3, 8)=  1548.
      AM_RO  (3, 9)=  1548.
      AM_RO  (3,10)=  1548.

! SiPh    COOLING PHASE
      AM_DATA(11)=10
      AM_TEMP(11, 1)=   21.+273.15
```

```
      AM_TEMP(11, 2)=  149.+273.15
      AM_TEMP(11, 3)=  371.+273.15
      AM_TEMP(11, 4)=  560.+273.15
      AM_TEMP(11, 5)=  838.+273.15
      AM_TEMP(11, 6)= 1115.+273.15
      AM_TEMP(11, 7)= 1393.+273.15
      AM_TEMP(11, 8)= 1949.+273.15
      AM_TEMP(11, 9)= 2504.+273.15
      AM_TEMP(11,10)= 3060.+273.15

      AM_CP  (11, 1)=  754.
      AM_CP  (11, 2)= 1005.
      AM_CP  (11, 3)= 1193.
      AM_CP  (11, 4)= 1200.
      AM_CP  (11, 5)= 1250.
      AM_CP  (11, 6)= 1298.
      AM_CP  (11, 7)= 1486.
      AM_CP  (11, 8)= 1486.
      AM_CP  (11, 9)= 1486.
      AM_CP  (11,10)= 1486.

      AM_TCI (11, 1)=    0.459
      AM_TCI (11, 2)=    0.513
      AM_TCI (11, 3)=    0.600
      AM_TCI (11, 4)=    0.627
      AM_TCI (11, 5)=    0.750
      AM_TCI (11, 6)=    0.850
      AM_TCI (11, 7)=    1.000
      AM_TCI (11, 8)=    1.812
      AM_TCI (11, 9)=    3.744
      AM_TCI (11,10)=    6.678

      AM_TCJ (11, 1)=    0.303
      AM_TCJ (11, 2)=    0.337
      AM_TCJ (11, 3)=    0.396
      AM_TCJ (11, 4)=    0.400
      AM_TCJ (11, 5)=    0.439
      AM_TCJ (11, 6)=    0.461
      AM_TCJ (11, 7)=    0.492
      AM_TCJ (11, 8)=    1.022
      AM_TCJ (11, 9)=    2.819
      AM_TCJ (11,10)=    3.112

      AM_RO  (11, 1)= 1758.
      AM_RO  (11, 2)= 1758.
      AM_RO  (11, 3)= 1758.
      AM_RO  (11, 4)= 1672.
      AM_RO  (11, 5)= 1548.
      AM_RO  (11, 6)= 1548.
      AM_RO  (11, 7)= 1548.
      AM_RO  (11, 8)= 1548.
      AM_RO  (11, 9)= 1548.
      AM_RO  (11,10)= 1548.

! Molybden
      AM_DATA(4)=10
      AM_TEMP(4, 1)=  -100.+273.15
      AM_TEMP(4, 2)=     0.+273.15
      AM_TEMP(4, 3)=   100.+273.15
      AM_TEMP(4, 4)=   200.+273.15
      AM_TEMP(4, 5)=   300.+273.15
      AM_TEMP(4, 6)=   400.+273.15
      AM_TEMP(4, 7)=   600.+273.15
      AM_TEMP(4, 8)=   800.+273.15
      AM_TEMP(4, 9)=  1000.+273.15
      AM_TEMP(4,10)=  2500.+273.15

      AM_CP  (4, 1)=   250.
```

```
      AM_CP  (4, 2)=   255.
      AM_CP  (4, 3)=   260.
      AM_CP  (4, 4)=   265.
      AM_CP  (4, 5)=   270.
      AM_CP  (4, 6)=   275.
      AM_CP  (4, 7)=   280.
      AM_CP  (4, 8)=   285.
      AM_CP  (4, 9)=   290.
      AM_CP  (4,10)=   350.

      AM_TCI (4, 1)=   138.
      AM_TCI (4, 2)=   125.
      AM_TCI (4, 3)=   118.
      AM_TCI (4, 4)=   114.
      AM_TCI (4, 5)=   111.
      AM_TCI (4, 6)=   109.
      AM_TCI (4, 7)=   106.
      AM_TCI (4, 8)=   102.
      AM_TCI (4, 9)=    99.
      AM_TCI (4,10)=    75.

      AM_TCJ (4, 1)=   138.
      AM_TCJ (4, 2)=   125.
      AM_TCJ (4, 3)=   118.
      AM_TCJ (4, 4)=   114.
      AM_TCJ (4, 5)=   111.
      AM_TCJ (4, 6)=   109.
      AM_TCJ (4, 7)=   106.
      AM_TCJ (4, 8)=   102.
      AM_TCJ (4, 9)=    99.
      AM_TCJ (4,10)=    75.

      AM_RO  (4, 1)= 10220.
      AM_RO  (4, 2)= 10220.
      AM_RO  (4, 3)= 10220.
      AM_RO  (4, 4)= 10220.
      AM_RO  (4, 5)= 10220.
      AM_RO  (4, 6)= 10220.
      AM_RO  (4, 7)= 10220.
      AM_RO  (4, 8)= 10220.
      AM_RO  (4, 9)= 10220.
      AM_RO  (4,10)= 10220.

! EPDM (kun "rene" termiske egenskaper; ikke kompensert for forgassing/forkulling)
      AM_DATA(5)=10
      AM_TEMP(5, 1)=  -73.+273.15
      AM_TEMP(5, 2)=  -18.+273.15
      AM_TEMP(5, 3)=   10.+273.15
      AM_TEMP(5, 4)=   94.+273.15
      AM_TEMP(5, 5)=  200.+273.15
      AM_TEMP(5, 6)=  315.+273.15
      AM_TEMP(5, 7)=  455.+273.15
      AM_TEMP(5, 8)=  594.+273.15
      AM_TEMP(5, 9)= 1093.+273.15
      AM_TEMP(5,10)= 3316.+273.15

      AM_CP  (5, 1)= 1000.
      AM_CP  (5, 2)= 1330.
      AM_CP  (5, 3)= 1280.
      AM_CP  (5, 4)= 1520.
      AM_CP  (5, 5)= 1890.
      AM_CP  (5, 6)= 2020.
      AM_CP  (5, 7)= 2902.
      AM_CP  (5, 8)= 2300.
      AM_CP  (5, 9)= 2050.
      AM_CP  (5,10)= 2050.

      AM_TCI (5, 1)=     .230
```

```
      AM_TCI (5, 2)=     .213
      AM_TCI (5, 3)=     .204
      AM_TCI (5, 4)=     .171
      AM_TCI (5, 5)=     .147
      AM_TCI (5, 6)=     .121
      AM_TCI (5, 7)=     .113
      AM_TCI (5, 8)=     .104
      AM_TCI (5, 9)=     .100
      AM_TCI (5,10)=     .100

      AM_TCJ (5, 1)=     .230
      AM_TCJ (5, 2)=     .213
      AM_TCJ (5, 3)=     .204
      AM_TCJ (5, 4)=     .171
      AM_TCJ (5, 5)=     .147
      AM_TCJ (5, 6)=     .121
      AM_TCJ (5, 7)=     .113
      AM_TCJ (5, 8)=     .104
      AM_TCJ (5, 9)=     .100
      AM_TCJ (5,10)=     .100

      AM_RO  (5, 1)= 1100.
      AM_RO  (5, 2)= 1100.
      AM_RO  (5, 3)= 1100.
      AM_RO  (5, 4)= 1100.
      AM_RO  (5, 5)= 1100.
      AM_RO  (5, 6)= 1100.
      AM_RO  (5, 7)=  715.
      AM_RO  (5, 8)=  325.
      AM_RO  (5, 9)=  325.
      AM_RO  (5,10)=  325.

! Luft
      AM_DATA(6)=11
      AM_TEMP(6,1)= 300
      AM_TEMP(6,2)=     350
      AM_TEMP(6,3)=     400
      AM_TEMP(6,4)=     550
      AM_TEMP(6,5)=     600
      AM_TEMP(6,6)=     650
      AM_TEMP(6,7)=     700
      AM_TEMP(6,8)=     800
      AM_TEMP(6,9)=    1000
      AM_TEMP(6,10)=   1200
      AM_TEMP(6,11)=   1600

      AM_CP  (6,1)=   1007.
      AM_CP  (6,2)=   1009.
      AM_CP  (6,3)=   1014.
      AM_CP  (6,4)=   1040.
      AM_CP  (6,5)=   1051.
      AM_CP  (6,6)=   1063.
      AM_CP  (6,7)=   1075.
      AM_CP  (6,8)=   1099.
      AM_CP  (6,9)=   1142.
      AM_CP  (6,10)=  1172.
      AM_CP  (6,11)=  1248.

      AM_TCI (6,1)=       0.00001589
      AM_TCI (6,2)=       0.00002092
      AM_TCI (6,3)=       0.00002641
      AM_TCI (6,4)=       0.00004557
      AM_TCI (6,5)=       0.00005269
      AM_TCI (6,6)=       0.00006021
      AM_TCI (6,7)=       0.00006810
      AM_TCI (6,8)=       0.00008493
      AM_TCI (6,9)=       0.00012190
      AM_TCI (6,10)=      0.00016290
```

```
      AM_TCI (6,11)=      0.00026800

      AM_TCJ (6,1)=       0.0263
      AM_TCJ (6,2)=       0.0300
      AM_TCJ (6,3)=       0.0338
      AM_TCJ (6,4)=       0.0439
      AM_TCJ (6,5)=       0.0469
      AM_TCJ (6,6)=       0.0497
      AM_TCJ (6,7)=       0.0524
      AM_TCJ (6,8)=       0.0573
      AM_TCJ (6,9)=       0.0667
      AM_TCJ (6,10)=      0.0763
      AM_TCJ (6,11)=      0.1060

      AM_RO  (6,1)=      1.1614
      AM_RO  (6,2)=      0.9950
      AM_RO  (6,3)=      0.8711
      AM_RO  (6,4)=      0.6329
      AM_RO  (6,5)=      0.5804
      AM_RO  (6,6)=      0.5356
      AM_RO  (6,7)=      0.4975
      AM_RO  (6,8)=      0.4354
      AM_RO  (6,9)=      0.3482
      AM_RO  (6,10)=     0.2902
      AM_RO  (6,11)=     0.2177

! ARAMIDE-EPOXY
      AM_DATA(7)=9

      AM_TEMP(7, 1)=   -50.+273.15
      AM_TEMP(7, 2)=     0.+273.15
      AM_TEMP(7, 3)=    50.+273.15
      AM_TEMP(7, 4)=   100.+273.15
      AM_TEMP(7, 5)=   150.+273.15
      AM_TEMP(7, 6)=   200.+273.15
      AM_TEMP(7, 7)=   250.+273.15
      AM_TEMP(7, 8)=   300.+273.15
      AM_TEMP(7, 9)=   500.+273.15

      AM_CP  (7, 1)=   858.
      AM_CP  (7, 2)=  1080.
      AM_CP  (7, 3)=  1382.
      AM_CP  (7, 4)=  1696.
      AM_CP  (7, 5)=  1696.
      AM_CP  (7, 6)=  2198.
      AM_CP  (7, 7)=  2303.
      AM_CP  (7, 8)=  2382.
      AM_CP  (7, 9)=  2400.

      AM_TCI (7, 1)=      .198
      AM_TCI (7, 2)=      .210
      AM_TCI (7, 3)=      .214
      AM_TCI (7, 4)=      .217
      AM_TCI (7, 5)=      .217
      AM_TCI (7, 6)=      .206
      AM_TCI (7, 7)=      .154
      AM_TCI (7, 8)=      .100
      AM_TCI (7, 9)=      .070

      AM_TCJ (7, 1)=      .198
      AM_TCJ (7, 2)=      .210
      AM_TCJ (7, 3)=      .214
      AM_TCJ (7, 4)=      .217
      AM_TCJ (7, 5)=      .217
      AM_TCJ (7, 6)=      .206
      AM_TCJ (7, 7)=      .154
      AM_TCJ (7, 8)=      .100
      AM_TCJ (7, 9)=      .070
```

```
      AM_RO  (7, 1)=  1380.
      AM_RO  (7, 2)=  1380.
      AM_RO  (7, 3)=  1380.
      AM_RO  (7, 4)=  1380.
      AM_RO  (7, 5)=  1380.
      AM_RO  (7, 6)=  1380.
      AM_RO  (7, 7)=  1380.
      AM_RO  (7, 8)=  1380.
      AM_RO  (7, 9)=  1380.

! grain
      AM_DATA(8)=6
      AM_TEMP(8,1)=  -100.+273.15
      AM_TEMP(8,2)=    20.+273.15
      AM_TEMP(8,3)=    77.+273.15
      AM_TEMP(8,4)=   149.+273.15
      AM_TEMP(8,5)=   200.+273.15
      AM_TEMP(8,6)=   371.+273.15

      AM_CP  (8,1)=  1047.
      AM_CP  (8,2)=  1047.
      AM_CP  (8,3)=  1336.
      AM_CP  (8,4)=  1336.
      AM_CP  (8,5)=  1532.
      AM_CP  (8,6)=  1570.

      AM_TCI (8,1)=      .36
      AM_TCI (8,2)=      .36
      AM_TCI (8,3)=      .36
      AM_TCI (8,4)=      .36
      AM_TCI (8,5)=      .36
      AM_TCI (8,6)=      .36

      AM_TCJ (8,1)=      .36
      AM_TCJ (8,2)=      .36
      AM_TCJ (8,3)=      .36
      AM_TCJ (8,4)=      .36
      AM_TCJ (8,5)=      .36
      AM_TCJ (8,6)=      .36

      AM_RO  (8,1)=  1722.
      AM_RO  (8,2)=  1722.
      AM_RO  (8,3)=  1722.
      AM_RO  (8,4)=  1722.
      AM_RO  (8,5)=  1722.
      AM_RO  (8,6)=  1722.
! graphite
      AM_DATA(10)=3
      AM_TEMP(10,1)= -100.+273.15
      AM_TEMP(10,2)=   20.+273.15
      AM_TEMP(10,3)= 3000.+273.15

      AM_CP  (10,1)= 2500.
      AM_CP  (10,2)= 2500.
      AM_CP  (10,3)= 2500.

      AM_TCI (10,1)= 1163.
      AM_TCI (10,2)= 1163.
      AM_TCI (10,3)= 1163.

      AM_TCJ (10,1)= 1163.
      AM_TCJ (10,2)= 1163.
      AM_TCJ (10,3)= 1163.

      AM_RO  (10,1)= 1750.
      AM_RO  (10,2)= 1750.
      AM_RO  (10,3)= 1750.
```

```
! Carbon fiber composite, intermediate modulus (IM-7)

      AM_DATA(12)=3
      AM_TEMP(12,1)= -100.+273.15
      AM_TEMP(12,2)=   20.+273.15
      AM_TEMP(12,3)= 3000.+273.15

      AM_CP  (12,1)=  750.
      AM_CP  (12,2)=  750.
      AM_CP  (12,3)=  750.

      AM_TCI (12,1)=    6.
      AM_TCI (12,2)=    6.
      AM_TCI (12,3)=    6.

      AM_TCJ (12,1)=     .6
      AM_TCJ (12,2)=     .6
      AM_TCJ (12,3)=     .6

      AM_RO  (12,1)= 1600.
      AM_RO  (12,2)= 1600.
      AM_RO  (12,3)= 1600.

! TZM (Molybdenlegering: Mo:0.4-0.55, Ti:0.06-1.2, Zr:0.01-0.04) (Plansee)
      AM_DATA(13)=26
      AM_TEMP(13, 1)=    0.+273.15
      AM_TEMP(13, 2)=  100.+273.15
      AM_TEMP(13, 3)=  200.+273.15
      AM_TEMP(13, 4)=  300.+273.15
      AM_TEMP(13, 5)=  400.+273.15
      AM_TEMP(13, 6)=  500.+273.15
      AM_TEMP(13, 7)=  600.+273.15
      AM_TEMP(13, 8)=  700.+273.15
      AM_TEMP(13, 9)=  800.+273.15
      AM_TEMP(13,10)=  900.+273.15
      AM_TEMP(13,11)= 1000.+273.15
      AM_TEMP(13,12)= 1100.+273.15
      AM_TEMP(13,13)= 1200.+273.15
      AM_TEMP(13,14)= 1300.+273.15
      AM_TEMP(13,15)= 1400.+273.15
      AM_TEMP(13,16)= 1500.+273.15
      AM_TEMP(13,17)= 1600.+273.15
      AM_TEMP(13,18)= 1700.+273.15
      AM_TEMP(13,19)= 1800.+273.15
      AM_TEMP(13,20)= 1900.+273.15
      AM_TEMP(13,21)= 2000.+273.15
      AM_TEMP(13,22)= 2100.+273.15
      AM_TEMP(13,23)= 2200.+273.15
      AM_TEMP(13,24)= 2300.+273.15
      AM_TEMP(13,25)= 2400.+273.15
      AM_TEMP(13,26)= 2500.+273.15

      AM_CP  (13, 1)=  229.5
      AM_CP  (13, 2)=  231.3
      AM_CP  (13, 3)=  236.7
      AM_CP  (13, 4)=  242.1
      AM_CP  (13, 5)=  247.5
      AM_CP  (13, 6)=  256.5
      AM_CP  (13, 7)=  263.7
      AM_CP  (13, 8)=  272.7
      AM_CP  (13, 9)=  279.9
      AM_CP  (13,10)=  288.8
      AM_CP  (13,11)=  296.
      AM_CP  (13,12)=  303.2
      AM_CP  (13,13)=  312.2
      AM_CP  (13,14)=  319.4
      AM_CP  (13,15)=  328.4
```

```
AM_CP  (13,16)=  337.4
AM_CP  (13,17)=  348.2
AM_CP  (13,18)=  360.8
AM_CP  (13,19)=  373.4
AM_CP  (13,20)=  389.6
AM_CP  (13,21)=  404.
AM_CP  (13,22)=  421.6
AM_CP  (13,23)=  441.7
AM_CP  (13,24)=  461.8
AM_CP  (13,25)=  481.9
AM_CP  (13,26)=  502.

AM_TCI (13, 1)=  128.49
AM_TCI (13, 2)=  125.69
AM_TCI (13, 3)=  122.9
AM_TCI (13, 4)=  120.11
AM_TCI (13, 5)=  118.01
AM_TCI (13, 6)=  115.57
AM_TCI (13, 7)=  113.12
AM_TCI (13, 8)=  111.03
AM_TCI (13, 9)=  108.24
AM_TCI (13,10)=  106.14
AM_TCI (13,11)=  103.35
AM_TCI (13,12)=  100.21
AM_TCI (13,13)=   97.76
AM_TCI (13,14)=   94.62
AM_TCI (13,15)=   92.18
AM_TCI (13,16)=   89.03
AM_TCI (13,17)=   86.24
AM_TCI (13,18)=   83.45
AM_TCI (13,19)=   81.
AM_TCI (13,20)=   78.21
AM_TCI (13,21)=   75.42
AM_TCI (13,22)=   72.97
AM_TCI (13,23)=   71.23
AM_TCI (13,24)=   69.48
AM_TCI (13,25)=   68.43
AM_TCI (13,26)=   67.04

AM_TCJ (13, 1)=  128.49
AM_TCJ (13, 2)=  125.69
AM_TCJ (13, 3)=  122.9
AM_TCJ (13, 4)=  120.11
AM_TCJ (13, 5)=  118.01
AM_TCJ (13, 6)=  115.57
AM_TCJ (13, 7)=  113.12
AM_TCJ (13, 8)=  111.03
AM_TCJ (13, 9)=  108.24
AM_TCJ (13,10)=  106.14
AM_TCJ (13,11)=  103.35
AM_TCJ (13,12)=  100.21
AM_TCJ (13,13)=   97.76
AM_TCJ (13,14)=   94.62
AM_TCJ (13,15)=   92.18
AM_TCJ (13,16)=   89.03
AM_TCJ (13,17)=   86.24
AM_TCJ (13,18)=   83.45
AM_TCJ (13,19)=   81.
AM_TCJ (13,20)=   78.21
AM_TCJ (13,21)=   75.42
AM_TCJ (13,22)=   72.97
AM_TCJ (13,23)=   71.23
AM_TCJ (13,24)=   69.48
AM_TCJ (13,25)=   68.43
AM_TCJ (13,26)=   67.04

AM_RO  (13, 1)=10220.
AM_RO  (13, 2)=10220.
```

```
      AM_RO  (13, 3)=10220.
      AM_RO  (13, 4)=10220.
      AM_RO  (13, 5)=10220.
      AM_RO  (13, 6)=10220.
      AM_RO  (13, 7)=10220.
      AM_RO  (13, 8)=10220.
      AM_RO  (13, 9)=10220.
      AM_RO  (13,10)=10220.
      AM_RO  (13,11)=10220.
      AM_RO  (13,12)=10220.
      AM_RO  (13,13)=10220.
      AM_RO  (13,14)=10220.
      AM_RO  (13,15)=10220.
      AM_RO  (13,16)=10220.
      AM_RO  (13,17)=10220.
      AM_RO  (13,18)=10220.
      AM_RO  (13,19)=10220.
      AM_RO  (13,20)=10220.
      AM_RO  (13,21)=10220.
      AM_RO  (13,22)=10220.
      AM_RO  (13,23)=10220.
      AM_RO  (13,24)=10220.
      AM_RO  (13,25)=10220.
      AM_RO  (13,26)=10220.

! WL10 (Tungstenlegering) (Tungsten+1% La2O3) (Plansee)
      AM_DATA(14)=17
      AM_TEMP(14, 1)=     0.+273.15
      AM_TEMP(14, 2)=    50.+273.15
      AM_TEMP(14, 3)=   100.+273.15
      AM_TEMP(14, 4)=   200.+273.15
      AM_TEMP(14, 5)=   300.+273.15
      AM_TEMP(14, 6)=   400.+273.15
      AM_TEMP(14, 7)=   500.+273.15
      AM_TEMP(14, 8)=   600.+273.15
      AM_TEMP(14, 9)=   700.+273.15
      AM_TEMP(14,10)=   800.+273.15
      AM_TEMP(14,11)=   900.+273.15
      AM_TEMP(14,12)=  1000.+273.15
      AM_TEMP(14,13)=  1100.+273.15
      AM_TEMP(14,14)=  1200.+273.15
      AM_TEMP(14,15)=  1300.+273.15
      AM_TEMP(14,16)=  1400.+273.15
      AM_TEMP(14,17)=  2500.+273.15
      AM_CP  (14, 1)=   122.3
      AM_CP  (14, 2)=   130.8
      AM_CP  (14, 3)=   139.3
      AM_CP  (14, 4)=   142.6
      AM_CP  (14, 5)=   143.7
      AM_CP  (14, 6)=   145.3
      AM_CP  (14, 7)=   145.9
      AM_CP  (14, 8)=   147.5
      AM_CP  (14, 9)=   148.6
      AM_CP  (14,10)=   150.2
      AM_CP  (14,11)=   151.6
      AM_CP  (14,12)=   152.5
      AM_CP  (14,13)=   153.4
      AM_CP  (14,14)=   154.9
      AM_CP  (14,15)=   155.5
      AM_CP  (14,16)=   154.8
      AM_CP  (14,17)=   147.1
      AM_TCI (14, 1)=   123.
      AM_TCI (14, 2)=   121.7
      AM_TCI (14, 3)=   120.4
      AM_TCI (14, 4)=   117.8
      AM_TCI (14, 5)=   113.6
      AM_TCI (14, 6)=   109.
      AM_TCI (14, 7)=   105.8
```

```
      AM_TCI (14, 8)=   102.9
      AM_TCI (14, 9)=   102.8
      AM_TCI (14,10)=   100.3
      AM_TCI (14,11)=   100.7
      AM_TCI (14,12)=    98.5
      AM_TCI (14,13)=    96.2
      AM_TCI (14,14)=    94.8
      AM_TCI (14,15)=    94.4
      AM_TCI (14,16)=    94.2
      AM_TCI (14,17)=    92.
      AM_TCJ (14, 1)=   123.
      AM_TCJ (14, 2)=   121.7
      AM_TCJ (14, 3)=   120.4
      AM_TCJ (14, 4)=   117.8
      AM_TCJ (14, 5)=   113.6
      AM_TCJ (14, 6)=   109.
      AM_TCJ (14, 7)=   105.8
      AM_TCJ (14, 8)=   102.9
      AM_TCJ (14, 9)=   102.8
      AM_TCJ (14,10)=   100.3
      AM_TCJ (14,11)=   100.7
      AM_TCJ (14,12)=    98.5
      AM_TCJ (14,13)=    96.2
      AM_TCJ (14,14)=    94.8
      AM_TCJ (14,15)=    94.4
      AM_TCJ (14,16)=    94.2
      AM_TCJ (14,17)=    92.
      AM_RO  (14, 1)= 19300.
      AM_RO  (14, 2)= 19300.
      AM_RO  (14, 3)= 19300.
      AM_RO  (14, 4)= 19300.
      AM_RO  (14, 5)= 19300.
      AM_RO  (14, 6)= 19300.
      AM_RO  (14, 7)= 19300.
      AM_RO  (14, 8)= 19300.
      AM_RO  (14, 9)= 19300.
      AM_RO  (14,10)= 19300.
      AM_RO  (14,11)= 19300.
      AM_RO  (14,12)= 19300.
      AM_RO  (14,13)= 19300.
      AM_RO  (14,14)= 19300.
      AM_RO  (14,15)= 19300.
      AM_RO  (14,16)= 19300.
      AM_RO  (14,17)= 19300.

! graphite Toyo Tanso IG-11
      AM_DATA(15)=13
      AM_TEMP(15,1)=     0.+273.15
      AM_TEMP(15,2)=   100.+273.15
      AM_TEMP(15,3)=   200.+273.15
      AM_TEMP(15,4)=   300.+273.15
      AM_TEMP(15,5)=   400.+273.15
      AM_TEMP(15,6)=   500.+273.15
      AM_TEMP(15,7)=   600.+273.15
      AM_TEMP(15,8)=   700.+273.15
      AM_TEMP(15,9)=   800.+273.15
      AM_TEMP(15,10)=  900.+273.15
      AM_TEMP(15,11)= 1000.+273.15
      AM_TEMP(15,12)= 1100.+273.15
      AM_TEMP(15,13)= 2500.+273.15

      AM_CP  (15,1)=   751.6
      AM_CP  (15,2)=   977.7
      AM_CP  (15,3)=  1203.7
      AM_CP  (15,4)=  1387.8
      AM_CP  (15,5)=  1508.4
      AM_CP  (15,6)=  1628.9
      AM_CP  (15,7)=  1688.7
```

```
      AM_CP  (15,8)=  1711.3
      AM_CP  (15,9)=  1733.9
      AM_CP  (15,10)= 1756.5
      AM_CP  (15,11)= 1779.2
      AM_CP  (15,12)= 1801.1
      AM_CP  (15,13)= 1919.

      AM_TCI (15,1)=   117.9
      AM_TCI (15,2)=   106.4
      AM_TCI (15,3)=    97.
      AM_TCI (15,4)=    87.1
      AM_TCI (15,5)=    79.3
      AM_TCI (15,6)=    72.3
      AM_TCI (15,7)=    66.6
      AM_TCI (15,8)=    61.2
      AM_TCI (15,9)=    57.1
      AM_TCI (15,10)=   53.
      AM_TCI (15,11)=   49.3
      AM_TCI (15,12)=   46.4
      AM_TCI (15,13)=   41.5

      AM_TCJ (15,1)=   117.9
      AM_TCJ (15,2)=   106.4
      AM_TCJ (15,3)=    97.
      AM_TCJ (15,4)=    87.1
      AM_TCJ (15,5)=    79.3
      AM_TCJ (15,6)=    72.3
      AM_TCJ (15,7)=    66.6
      AM_TCJ (15,8)=    61.2
      AM_TCJ (15,9)=    57.1
      AM_TCJ (15,10)=   53.
      AM_TCJ (15,11)=   49.3
      AM_TCJ (15,12)=   46.4
      AM_TCJ (15,13)=   41.5

      AM_RO  (15,1)=  1770.
      AM_RO  (15,2)=  1770.
      AM_RO  (15,3)=  1770.
      AM_RO  (15,4)=  1770.
      AM_RO  (15,5)=  1770.
      AM_RO  (15,6)=  1770.
      AM_RO  (15,7)=  1770.
      AM_RO  (15,8)=  1770.
      AM_RO  (15,9)=  1770.
      AM_RO  (15,10)= 1770.
      AM_RO  (15,11)= 1770.
      AM_RO  (15,12)= 1770.
      AM_RO  (15,13)= 1770.
! C-C/SiC DLR materiale
      AM_DATA(16)=5
      AM_TEMP(16,1)=     0.             !lagt inn kunstig
      AM_TEMP(16,2)=   473.15
      AM_TEMP(16,3)=  1273.15
      AM_TEMP(16,4)=  1923.15
      AM_TEMP(16,5)=  5000.             ! lagt in kunstig

      AM_CP  (16,1)=   748.
      AM_CP  (16,2)=   748.
      AM_CP  (16,3)=  1414.
      AM_CP  (16,4)=  1514.
      AM_CP  (16,5)=  1514.

      AM_TCI (16,1)=   16.8        !langs fiber
      AM_TCI (16,2)=   16.8
      AM_TCI (16,3)=   18.
      AM_TCI (16,4)=   16.8
      AM_TCI (16,5)=   16.8
```

```
AM_TCJ (16,1)=   9.2        !normalt til fiber
AM_TCJ (16,2)=   9.2
AM_TCJ (16,3)=   7.6
AM_TCJ (16,4)=   7.5
AM_TCJ (16,5)=   7.5

AM_RO  (16,1)=  2000.
AM_RO  (16,2)=  2000.
AM_RO  (16,3)=  2000.
AM_RO  (16,4)=  2000.
AM_RO  (16,5)=  2000.

RETURN
END
```