TERMO - the code for the solution of nonstationary heat equations.

Roman Kohut, 8.12.2008, version 1a
Institute og Geonics, Academy of Sciences of the Czech Republic
kohut@ugn.cas.cz

1 Introduction

The code **TERMO** serves for the solution of the discrete form of nonstationary heat equations in 3D. The initial-boundary value problem is discretized by finite elements in space and finite differences in time. Corresponding linear system of algebraic equations in each time step are solved by the preconditioned conjugate gradient method with the preconditioning based on the additive overlapping Schwarz methods.

The specific feature of the code **TERMO** is the using of regular structured grids which, simply speaking, are the grids arising as a result of deformation of some regular rectangular grid (the pattern grig). The discretization is given by the decomposition of the domain into eight-node bricks and consequently into tetrahedral finite elements. Hence the corresponding matrices have all nonzero entries within a 27 node regular stencil. Supposing the symmetry of the matrix, we can additionally store only the upper triangular part of the matrix. It can be done row-by-row by using regular 14 element stencil for the storage of the nonzero matrix entries.

We assume that material is anisotropic and heterogeneous and is not dependent on temperature changes. The heat radiation is not included. The heat source is assumed to descent exponentially with time (special case is a constant heat source). The time is measured in years. The details see next parts.

The whole program is written using Fortran 95, for parallelization OpenMP paradigm is used. The code was tested on symmetric multiprocessor IBM e-server xSeries 455, 8 processors Intel Itanium 2/1300, shared memory 16 GB, RAID controller with disk capacity 2 x 72 GB under UNIX operating system.

2 Code distribution

Code **TERMO** is available from the web site

http//www.ugn.cas.cz/,

questions can be sent to *roman.kohut@ugn.cas.*

The code **Termo** is free software distributed under the terms of the GNU General Public License as published by the Free Software Foundation.

3 Numerical methods and algorithms

The nonstationary heat conduction problem is concerned with finding the temperature $\tau = \tau(x,t)$,

$$\tau : \Omega \times (0,T) \rightarrow R,$$

that fulfills the following equation

$$\kappa\rho\frac{\partial\tau}{\partial t} \quad = \quad k\sum_i \frac{\partial^2\tau}{\partial {x_i}^2} + q(t) \qquad in \quad \Omega \times (0,T) \tag{1}$$

together with the corresponding boundary and initial conditions.

The initial-boundary value problem (1) is discretized by the finite elements in space and the finite differences in time. Using the linear finite elements and the time discretization, it leads to the computation of a vector $\underline{\tau}^j$ of nodal temperatures at the time levels $t_j, j = 1, N,$ with the time steps $\Delta t_j = t_j - t_{j-1}$. It gives the following time stepping algorithm:

find $\underline{\tau}^0 : \; M_h\,\underline{\tau}^0 \; = \; \underline{\tau}_0$,
for j=1,...,N:
find $\underline{\tau}^j : \; B_h^{(j)}\,\underline{\tau}^j \; = \; [M_h \; + \; \theta\Delta t_j\,K_h]\,\underline{\tau}^j \; = \; c^j$.
enddo

Above, M_h is the capacitance matrix, K_h is conductivity matrix, $c_j \; = \; [M_h \; - \; (1-\theta)\Delta t_j K_h]\underline{\tau}_{j-1} + \; \Delta t_j\phi_j$, $\phi_j \; = \theta q(t_j) + (1-\theta)q(t_{j-1})$ and $\underline{\tau}_0$ is from the initial condition. Here parameter $\theta \in < 0,1 >$. In means that in each time level we have to solve the system of linear equations

$$[M_h + \theta\Delta t_j K_h]\underline{\tau}^j = [M_h \; - \; (1-\theta)\Delta t_j K_h]\underline{\tau}^{j-1} + \Delta t_j\phi_j. \tag{2}$$

For $\theta = 0$ we obtain the explicit Euler scheme, for $\theta = 1$ we obtain the backward Euler (BE) scheme, $\theta = 0.5$ gives Crank-Nicholson (CN) scheme. In our case we will use the BE scheme. If we substitute $\underline{\tau}^j = \underline{\tau}^{j-1} + \Delta\underline{\tau}^j$ into (2), we obtain

$$[M_h + \Delta t_j K_h]\Delta\underline{\tau}^j = \Delta t_j(q_h^j - K_h\underline{\tau}^{j-1}). \tag{3}$$

To ensure accuracy and not waste the computational effort, it is important to adapt the time steps to the behaviour of the solution. We use the procedure based on local comparison of the backward Euler (BE) and Crank-Nicholson (CN) scheme ([1]). We solve the system (2) only using BE scheme. If this solution $\underline{\tau}^j = \underline{\tau}^{j-1} + \Delta\underline{\tau}^j$ is considered as the initial approximation for the solution of system (2) for $\theta = 0.5$ (CN scheme), then the first iteration of the Richardson's method presents an approximation of the solution of the system (2) for $\theta = 0.5$. Thus $\underline{\tau}_{CN}^j \cong \underline{\tau}^j - r^j$, where

$$r^j = (M_h + 0.5\Delta t_j K_h)\underline{\tau}^j - (M_h - 0.5\Delta t_j K_h)\underline{\tau}^{j-1} - 0.5\underline{q}_h(t_j) - 0.5\underline{q}_h(t_{j-1}). \tag{4}$$

The time steps can be controlled with the aid of the ratio $\eta = \frac{\|r^j\|}{\|\underline{\tau}^j\|}$. If $\eta < \varepsilon_{min}$ then we continue with time step $\Delta t = 2 \times \Delta t$, if $\eta > \varepsilon_{max}$ then we continue with time step $\Delta t = 0.5 \times \Delta t$, where ε_{min}, ε_{max} are given values.

For the solution of the linear system $B_h\Delta\underline{\tau}^j = (M_h + \Delta_j K_h)\Delta\underline{\tau}^j = f_j$ (3) we shall use the preconditioned CG method where the preconditioning is given by the additive overlapping Schwarz method. In this case the domain is divided into m subdomains Ω_k (in our case the domain is divided only in z direction), nonoverlaping subdomains Ω_k are then extended to domains Ω'_k in such a way that overlaping between the subdomains are given by two or more layers of elements. If B'_{kk} are the FE matrices corresponding to problems on Ω'_k, I'_k and $R'_k = (I'_k)^T$ are the interpolation and restriction matrices, respectively, then introduced matrices $B'_{kk} = R'_k B I'_k$ allow to define one-level additive Schwarz preconditioner G,

$$g = Gr = \sum_{k=1}^{m} I'_k {B'_{kk}}^{-1} R'_k r.$$

The system matrix B_h is composed from two matrices, the matrix M_h which is not M-matrix (all matrix elements are positive for linear FE basis functions) and the matrix $\Delta t_j K_h$ which is M-matrix in many practical situations, e.g. if the heat flow is isotropic and the inner angles of tetrahedra do not exceed $\pi/2$. For small values of Δ_j the matrix $M_h + \Delta t_j K_h$ is not M-matrix

and the incomplete factorization fails for preconditioning. But it is possible to apply the incomplete factorization to the matrix $M_h^L + \Delta t_j K_h$, where M_h^L is the lumped matrix to the matrix M_h, which means that its diagonal has diagonal elements equal to the sum of the elements on the corresponding row.

4 The storage of the files

As was written in the first part, a specific feature of our code is the using of regular structured grids. If nx, ny, nz represent the the numbers of nodes in corresponding directions, the nodes can be easily indexed by the triples (i,j,k),

$$1 \le i \le Nx, \quad 1 \le j \le Ny, \quad 1 \le k \le Nz$$

or enumerated by 1D numbers

$$Indn = i + (j-1) * Nx + (k-1) * Nx * Ny.$$

It means that enumeration is done first in the direction x, then in direction y and finally in direction z.

a) matrix storage

Supposing the symmetry of the matrix, we can store only the upper triangular part of the matrix. It can be done row-by-row by using regular 14 element stencil. The row numbers are given by the order of the records, column numbers have not to be stored due to the regular stencil. More exactly, the column numbers for the regular stencil and for the row index i will be the following:

If i is the row number, then the column numbers used in the 14 element stencil are subsequently:

$$\begin{aligned}
j &= i, i+1 \\
j &= i+nx-1, i+nx, i+nx+1 \\
j &= i+nx*ny-nx-1, i+nx*ny-nx, i+nx*ny-nx+1 \\
j &= i+nx*ny-1, i+nx*ny, i+nx*ny+1 \\
j &= i+nx*ny+nx-1, i+nx*ny+nx, i+nx*ny+nx+1
\end{aligned}$$

For opening and reading files with stored matrices we use folowing instructions:

open(1,file=filename, access='direct', recl=56),

if the matrix is stored in $real * 4$ or

open(1,file=filename, access='direct', recl=112),

if the matrix is stored in $real * 8$.

The files are read as follows:

```
do i=1,nn
    read(1,rec=i) a
    do j=1,14
       mh(j+(i-1)*14)=a(j)
    enddo
enddo
```

Vector storage

Vectors contain some nodal values. A value corresponding to the triples of indeces (i,j,k) is in position $v(Indn)$, $Indn = i+(j-1)*Nx+(k-1)*Nx*Ny$. For opening and reading files with stored vectors we use following instructios:

open(1,file=filename, form='unformatted'),

read(1) (v(i)=1,nn),

where nn is the number of nodes.

Input data storage

Files containing input data are text files with a free format.

5 Routines used in code TERMO

Routines used in code **TERMO** are inserted in folowing files:

term_IS.for	- the main routine for controling the runing of the code.
mxv_t_IS.for	- the subroutine for the multiplication of matrix A by vector v.
mxv_ns_IS.for	- the subroutine for the multiplication of matrix $(a * A + b * B)$ by vector v.
pcond_IS.for	- two subroutines for the realization of preconditioning based on the additive overlapping Schwarz methods.
. **rmat_t_IS.for**	- subroutine for reading of matrix K_h (see 3).
rmatmh_t_IS.for	- subroutine for reading of matrix M_h (see 3).
georens_IS	- subroutine for modification of matrices and rhs according to boundary conditions.
nstcin_IS.for	- subroutine for reading of input parameters for given task.
pcg_IS.for	- subroutine for realization of preconditioned conjugate gradients.
load_IS.for	- subroutine for the determination of a nodal vector given by heat source in time point t.
rtime_IS	- subroutine for measuring of CPU time. Time is measured by Fortran inner subroutine **itime**.
lin_IS.mki	- information for compiler
termo_IS.mk	- Makefile

6 Makefile for compilation

The compilation is done in two steps:

1.step: **make -f termo_IS.mk clean**
The files $*.f, *.fo, *.o$ are deleted.

2. step **make -f termo_IS.mk**
The compilation is done.

7 Input files needed for running of the code TERMO

The code uses two matrices as input(see 3):
the matrix M_h (capacitance matrix) is stored in the file $mh.g32$
the matrix K_h (conductivity matrix) is stored in the file $fkbct.g32$.

The structure of these files was desrcibed in Section 4. Note that rows corresponding to nodes in emty area (nodes inside holes, tunnels) have all elements equal to zero. The code replaces in matrix M_h corresponding zero element in position of diagonal element with value one.

The code uses four vectors as input. The structure of these files was desribed in Section 4. The part of rhs in j time step (see 3) presents vector q_j^h. In our code this vector is distributed to two vectors, $q_j^h = r^h + rq_j^h$. The nodal vector r^h is generated from boundary conditions (the given temperature on part of boundary, heat transfer to emty area on part of inner boundary) and is constant during the computation. This vector is saved in file $frt.g32$. The second part forms the vector rq_j^h corresponding to a heat source which is changing with time. We assume that this part of the vector q_j^h descents exponentially with time and has in the time point t_j value

$$q_j^h = q_0^h(a * e^{-a1*t_j} + b * e^{-b1*t_j} + c * e^{-c1*t_j}), \tag{5}$$

where $a, a1, b, b1, c, c1$ are input parameters that must fullfil equality $a + b + c = 1$. The initial value in time $t = 0$ (the vector q_0^h) is saved in file *frq.g32*.

The initial value conditions are saved in file *ft0.g32*, the Dirichlet boundary conditions are saved in file *fbct.g32*. The file *fbct.g32* contains the nodal vector *bc* that has following nodal values:

bc($node_i$) - v_i for $node_i$ with prescribed value v_i
bc($node_i$) - $1e9$ for free $node_i$

The input file *nstc_IS.in* contains input data for code **TERMO**. The file is text file with data in free format. The data are stored in following order:

npro npr	npro - the number of processors (subdomains) npr - the number of overlapping layers
nx, ny, nz	the number of nodes in directions x,y,z
a, a1	parameters for heat source descent($a * e^{-a1*t}+$
b, b1	parameters for heat sourse descent $b * e^{-b1*t}+$
c, c1	parameters for heat sourse descent $b * e^{-c1*t}$), see (5)
ir	ir = 0 for input files in real*4, ir=1 for real*8
eps	the accuracy for the solution of linear system in each time step
imax	max. number of pcg iterations for each time step
dt, tt	dt - initial size of time step, tt - the whole time period
ad	ad=0 - constant time step dt, ad=n - adaptive change of time step, the test, if to change, is after each n iterations, ad=-1 - time steps are prescribed (saved in file *t_point.in* (see below))
emin, emax	parameters for adaptivity, can differ for various task, usual values emin=0.0001, emax=0.001.
n	we can save solution vectors in various time points (fut_1,...,fut_n). n presents the number of such time points. For n=0 the list of time points is missing. The final solution is in file *fut.g32*
tmu(1) . **tmu(n)**	list of time points
m	after each time step we can save values of the solution vectors in prescribed nodes. m is the number of such nodes. The values are saved in file *temp_points.rep*. The first row contains the value of time steps, next rows contain the temperatures in given points. For nn=0 the list of nodes is missing.
i(1),j(1),k(1) . **i(m), j(m),k(m)**	the list of triplets of nodes.

8 Output files

The vectors of nodal temperatures in given time points $t_1, t_2, ...$ (see input file *nstc_IS.in*) are saved in files *fut_1.g32, fut_2.g32, ...*, the final solution in time *ct* is saved in file *fut.g32*. The structure of these files is the same as the structure of input files of nodal vectors (see Section 4). The behaviour of temperatures in given points (see input file *nstc_IS.in*) is stored into text file *temp_points.rep*. The structure of this file is following:

t_0	initial time $t = 0$
τ_0^1	
τ_0^2	the temperatures in time $t = 0$ in given nodes
.	
.	
τ_1^1	
τ_1^2	the temperatures in time t_1 in given nodes
.	
.	

The time points $t_1, t_2, ...$ correspond to the ends of the used time steps.

The informations about the running of the code are written to text file *termoel.rep*. For each time step the following informations are saved:

n_it , tol	the number of pcg it. for given time step, the reached accuracy $\frac{\|r^j\|}{\|q^j\|}$
it , dt, the global time	the serial number of time step, the size of the time step, the reached time after *n_it* time steps
max. dTau, max. temp.	max. incr. of the temp. in this time step, the max. temperature in this time step
computing time	the CPU time

9 Test example and getting started

The code was tested on 3D benchmark probmlem BMT3 (Figure 1) suggested within the frame of DECOVALEX project (see [2]). In this test case, a repository tunnel is located at a depth of 500m and nuclear waste, which is a source of heat, is disposed in a borehole below this tunnel. We suppose that modelled area is the cube $50m \times 50m \times 50m$, upper side has z-coordinate

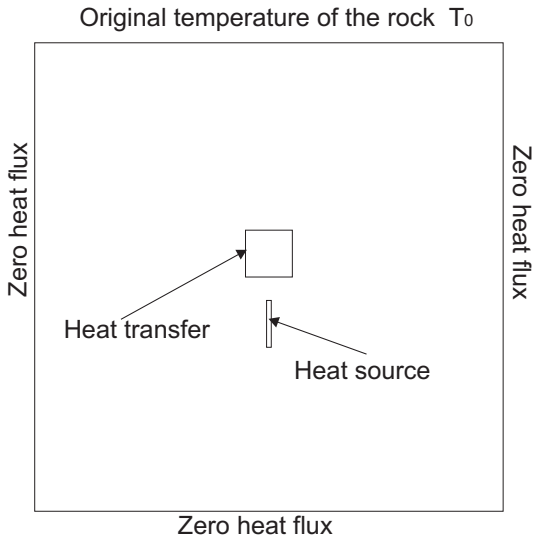

Figure 1: The BMT3 benchmark task.

z=0, bottom side has z-coorinate z=50, the tunnel is oriented in y direction, the profile is squared, 5×5 metres, the length is 50m (throw the whole area), the footwall is situated on the plane with z-coordinate z=25m. To compare the results with results of DECOVALEX teams which solved the problem as 2D, we suppose that "borehole" with heat source is block with coordinates $< 24.75, 25.25 >< 0, 50 >< 27.5, 32.5 >$.

The initial thermal conditions consist of a constant initial temperature T_0 (file $ft0.g32$) of 27^oC throughout the model. The initial temperature is fixed along the top surface of the model (the value 27 in corresponding nodes in file $fbct.g32$), while both vertical surfaces and the lower boundary surface are assumed adiabatic (zero heat flux). The tunnel surface is assumed to behave as a convective boundary (heat transfer) in which the flux condtition is given as

$$q = H(T_{wall} - T_g), \tag{6}$$

where q is thermal flux across the tunel surface (W/m^2), $H = 7W/m^{2.o}C$ is the coefficient of surface heat transfer, T_{wall} is the wall temperature and $T_g = 27^oC$ is the constant tunnel temperature.

The heat source simulating the waste canister is assumed to decay exponen-

tially with time according to the following relation

$$Q(t) = q_0 exp(-\beta t) \tag{7}$$

where $Q(t)$ is the heat flux at time t (W/m^3)), $Q_0 = 470 W/m^3$ is the initial heat flux (file *frq.g32* and $\beta = 0.02\,1/year$ is the heat decay coefficient.

We suppose the material is homogeneous and isotropic (the rock and the heat source has the same properties), the specific heat capacity $\kappa = 900 J/kg.^oC$, the thermal conductivity $k = 3 W/m.^oC$.

The structured rectangular FE grid consists of $45 \times 51 \times 45$ nodes (105570 equations), the coordinates of nodes where the size in correponding direction is changed is presented in following Table. The division of nodes between these nodes is uniform.

direction x		direction y		direction z	
nodes x dir. x	x-coord.	nodes dir. y	y coord.	nodes dir. z	z coord. z
1	0.00	1	0.00	1	0.00
6	10.00	17	22.00	7	12.00
11	17.50	21	24.00	12	12.75
16	22.50	24	24.75	32	27.50
21	24.75	28	25.25	38	33.50
25	25.25	31	26.00	46	50.00
30	27.50	35	28.00		
35	32.50	51	50.00		
40	40.00				
45	50.00				

Table 1: The coordinates of grid nodes

After grid generation we assemble matrices and vectors described in Section 7. The input file *nstc_IS.in* for this task has following form:

8 2
45 51 46
1.0 0.02
0.0 0.0
0.0 0.0
0
0.001
1000
0.00001 100
5
0.00001 0.0001
0
1
11 26 37

The behaviour of temperature during 100 years in the point A (indices 11,26,37, coordinates 17.5m, 25.0m, 32.5m) is presented in Figure 2. If

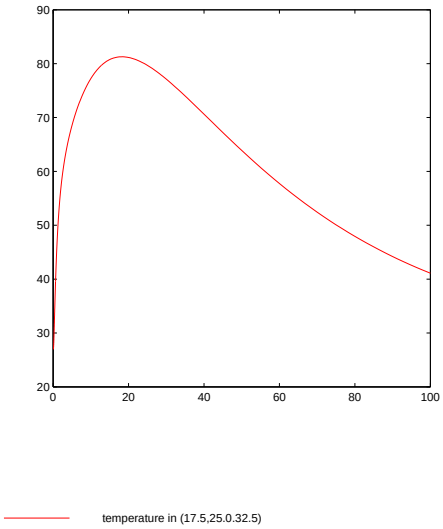

Figure 2: The behaviour of temperature in pont A.

the time steps are prescribed ($ad = -1$) the code reads these values from text file *t_point.in*. The first number in the list is the number of the time steps followed by the values of corresponding time steps. Each row contains one number.

This test example was computed with the code developed in COMSOL, with code developed in MATLAB. The results obtained by these codes are very close to resuls obtained by code **TERMO**. Very similar results were obtained also by various codes used by teams in DECOVALEX project (see ([2]).

References

[1] Blaheta R., Byczanski P., Kohut R., Starý J.: Algorithm for parallel FEM modelling of thermo–mechanical phenomena arising from the disposal of the spent nuclear fuel. In: O. Stephansson, J.B. Hudson, Lanru Jing eds., Coupled Thermo-Hydro-Mechanical-Chemical processes in Geo-systems, Elsevier 2004

[2] Stephansson, O., Jing, L., Tsang, Ch.F.: Coupled Thermo-Hydro-Mechanical Processes of Fractured Media, Elsevier 1996, pp. 311-340.