Pergamon

*Computers & Geosciences* Vol. 22, No. 10, pp. 1147–1158, 1996
Copyright © 1996 Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0098-3004/96 $15.00 + 0.00

PII: S0098-3004(96)00052-0

# NUMERICAL SOLUTIONS FOR THE ONE-DIMENSIONAL HEAT-CONDUCTION EQUATION USING A SPREADSHEET

ZOHAR GVIRTZMAN and ZVI GARFUNKEL

Institute of Earth Sciences, Hebrew University, Givat Ram 91904, Jerusalem, Israel
(*e-mail*: haimg.vms.huji.ac.il)

*(Received 30 September 1995; revised 21 March 1996)*

**Abstract**—We show how to use a spreadsheet to calculate numerical solutions of the one-dimensional time-dependent heat-conduction equation. We find the spreadsheet to be a practical tool for numerical calculations, because the algorithms can be implemented simply and quickly without complicated programming, and the spreadsheet utilities can be used not only for graphics, printing, and file management, but also for advanced mathematical operations.

We implement the explicit and the Crank–Nicholson forms of the finite-difference approximations and discuss the geological applications of both methods. We also show how to adjust these two algorithms to a nonhomogeneous lithosphere in which the thermal properties (thermal conductivity, density, and radioactive heat generation) change from the upper crust to the lower crust and to the mantle.

The solution is presented in a way that can fit any spreadsheet (Lotus-123, Quattro-Pro, Excel). In addition, a Quattro-Pro program with macros that calculate and display the thermal evolution of the lithosphere after a thermal perturbation is enclosed in an appendix. Copyright © 1996 Elsevier Science Ltd

*Key Words*: Heat conduction, Finite-difference methods, Explicit, Crank–Nicholson, Spreadsheet, Thermal modeling, Basin analysis.

## INTRODUCTION

The thermal regime of the lithosphere is one of the main factors which govern its geodynamic development and influence magmatic processes, tectonic movements, and vertical oscillations which control sedimentation and erosion. In the last few years thermal modeling has become an essential tool for quantitative basin analysis (e.g. Beaumont, Keen, and Boutillier, 1982; Royden, Sclater, and Von Herzen, 1980; England and Thompson, 1984; Day, 1987; Peacock, 1987; Sandiford, 1989; De Yoreo, Lux, and Guidotti, 1991). On the other hand, to geologists who are not familiar with partial differential equations, the thermal calculations might look mathematically complicated, especially because most natural problems can be solved only by numerical methods which require computer programming. In this paper we present an implementation of two numerical algorithms in spreadsheet programs. Spreadsheets are used widely by the scientific community. We will show that in addition to the advantages associated with the spreadsheet utilities (advanced mathematical functions, graphics, printing, simultaneous use of a few files...), the spreadsheet is especially suitable for this purpose. The spreadsheet approach is also a good one for teaching purposes.

The numerical algorithms which we implement solve the finite-difference approximation to the one-dimensional time-dependent heat-conduction equation:

$$\frac{\partial T}{\partial t} = \frac{k}{\rho c_p}\frac{\partial^2 T}{\partial z^2} + \frac{H(z)}{\rho c_p} \tag{1}$$

where $T$ is temperature, $k$ is thermal conductivity, $t$ is time, $H(z)$ is internal heat production rate per unit volume, $z$ is depth, $c_p$ is specific heat per unit mass (at constant $p$), and $\rho$ is density.

This equation is subject to the initial condition $T(t = 0,\ z) = f(z)$, and boundary conditions at $z = 0$ and $z = L$ $(0 \leq z \leq L)$. It is based on the assumptions that conduction is the only heat transfer mechanism, that conduction is vertical only, and that $k$ is constant. These assumptions are made frequently for many geological applications, though there are geological circumstances in which advection also should be considered, and there are situations in which two- or three-dimensional solutions are essential. In addition, it should be noted that Equation (1) takes into account the radioactive heat generation which is important for a continental crust, and thus its analytical solution often is relatively complicated. The numerical methods, however, have several advantages over the analytical solution, which make them a powerful tool for geological applications. They are computationally simple, and do not become complicated when the initial conditions have a complex form. This allows the user to perturb the temperature distribution at any time, and then calculate how the disturbance decays with time. It is possible also to deal with depth changes of the thermal conductivity, density, and radioactive heat generation.

At first, we present the algorithms for a homogeneous lithosphere with constant thermal properties, and then we show how to adjust the algorithms to a more realistic situation in which the thermal conductivity, density, and heat generation change from upper crust to the lower crust and to the mantle.

## FINITE-DIFFERENCE SOLUTIONS

There are several schemes available to express the time-dependent heat-conduction equation in finite-difference form. We present the explicit method and the Crank–Nicholson algorithm which is a modification of the so-called fully implicit method. As it would neither be possible nor appropriate to present, within the context of this paper, a thorough review of these numerical methods, we briefly present only the main principles. For more details and background the reader is referred to Carslaw and Jaeger (1959), Ozisik (1980), Smith (1985) and many other textbooks about heat-conduction and about numerical solutions for partial differential equations.

### *The explicit method*

Let us construct a finite-difference net of space and time with intervals of $\delta z$ and $\delta t$ as illustrated in Figure 1. In this net the space and time coordinates are denoted by:

$$z = j\delta z \quad j = 0,1,2,\ldots N$$
$$t = n\delta t \quad n = 0,1,2,\ldots$$

and the temperature $T(z,t)$ is represented by: $T(j\delta z, n\delta t) \equiv T^n_j$.

The explicit method provides a relatively straightforward expression for the calculation of the unknowns $T^{n+1}_0, T^{n+1}_1, T^{n+1}_2, \ldots T^{n+1}_N$ (the geotherm at the n + 1 time-step) from the values $T^n_0, T^n_1, T^n_2, \ldots T^n_N$ of the previous time-step $n$.

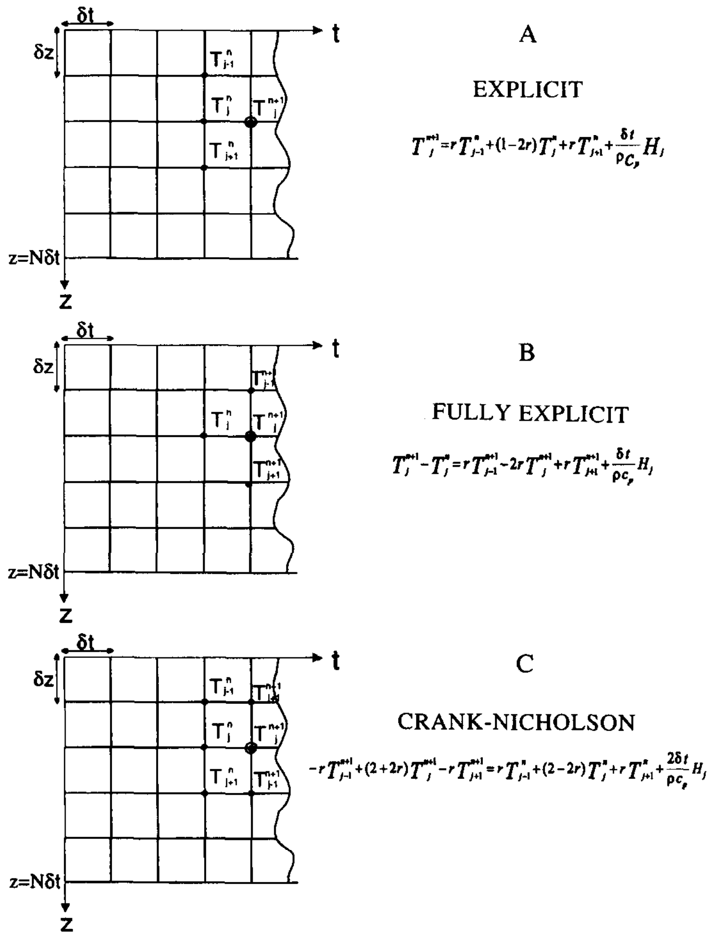

δt
δz
t
z=Nδt
z
$T^n_{j-1}$
$T^n_j$
$T^{n+1}_j$
$T^n_{j+1}$
A
EXPLICIT
$T^{n+1}_j = rT^n_{j-1} + (1-2r)T^n_j + rT^n_{j+1} + \frac{\delta t}{\rho c_p}H_j$
δt
δz
t
z=Nδt
z
$T^{n+1}_{j-1}$
$T^n_j$
$T^{n+1}_j$
$T^{n+1}_{j+1}$
B
FULLY EXPLICIT
$T^{n+1}_j - T^n_j = rT^{n+1}_{j-1} - 2rT^{n+1}_j + rT^{n+1}_{j+1} + \frac{\delta t}{\rho c_p}H_j$
δt
δz
t
z=Nδt
z
$T^n_{j-1}$
$T^{n+1}_{j-1}$
$T^n_j$
$T^{n+1}_j$
$T^n_{j+1}$
$T^{n+1}_{j+1}$
C
CRANK-NICHOLSON
$-rT^{n+1}_{j-1} + (2+2r)T^{n+1}_j - rT^{n+1}_{j+1} = rT^n_{j-1} + (2-2r)T^n_j + rT^n_{j+1} + \frac{2\delta t}{\rho c_p}H_j$

Figure 1. Discretisation of time–space continuum for finite-difference approximation scheme. A, Explicit method—approximation of temperature at any point depends on temperature at three previously calculated points. B, Fully implicit method—new calculated point depends on three points, two of them still unknown. C, Crank–Nicholson method—each calculated point depends on five neighboring points, two of them still unknown.

$$T_j^{n+1} = rT_{j-1}^n + (1-2r)T_j^n + rT_{j+1}^n + \frac{\delta t}{\rho C_p} H_j \quad (2)$$

where $r = \frac{k\delta t}{\rho C_p \delta z^2}$ and $H_j = \frac{1}{\delta z}\int_{j-\frac{1}{2}}^{j+\frac{1}{2}} H(z)dz$.

In the graphical presentation of our time–space net it is easy to see that each point is calculated by using its three previous neighbors (left neighbors in Fig. 1A), whereas the leftmost column ($n = 0$, i.e. $t = 0$) includes the initial values. In this way the entire net can be calculated from left to right, except for the uppermost and lowermost rows which have no neighbors on the top or bottom. These rows, however, actually are given when the boundary conditions are specified by constant temperatures (Fig. 2A), and are calculated according to Equations (3) and (4), respectively, when the boundary conditions are given as heat flows (Fig. 2B).

$$T_0^{n+1} = 2r(T_1^n - \frac{\delta z}{k} q_0) + (1-2r)T_0^n + \frac{\delta t}{\rho C_p} H_0 \quad (3)$$

$$T_N^{n+1} = 2r(T_{N-1}^n + \frac{\delta z}{k} q_N) + (1-2r)T_N^n + \frac{\delta t}{\rho C_p} H_N \quad (4)$$

where $q_0$ = heat flow at $z = 0$, and $q_N$ = heat flow at $z = N\delta z$ (both positive upward).

For geological problems, in which the upper boundary ($z = 0$) is the surface, Equation (3) is not applicable because there is no reason to assume a constant surface heat flow during the transient stage of the problem, and usually a constant temperature is assumed. However, we use Equation (3) to develop the solution for a nonhomogeneous lithosphere as will be shown in a later section.

The explicit method is easy to calculate, but its serious disadvantage is that $r$ is restricted to the range [0,0.5]. Otherwise, the numerical calculation becomes unstable resulting from an amplification of errors (Ozisik, 1980; Smith, 1985). As a result, the time-step $\delta t$ is necessarily small because the calculation is valid only for the condition

$$\delta t < \frac{\rho c_p}{2k} \delta z^2$$

and $z$ must be small enough to attain reasonable resolution of the space (depth). For example, in geological problems we often use $\rho = 3300\,\mathrm{kg{\cdot}m^{-3}}$, $c_p = 1200\,\mathrm{W{\cdot}s{\cdot}kg^{-1}{\cdot}{}^0C^{-1}}$, $k = 3\,\mathrm{W{\cdot}m^{-1}{\cdot}{}^0C^{-1}}$, and thus obtain $\delta t < 0.02\,\mathrm{year{\cdot}m^{-2}{\cdot}}\delta z^2$. Now, if the depth resolution is 1 km ($\delta z = 1000$ m) we get $\delta t < 0.02$ Ma, implying that for a long time period such as 100 Ma we need at least 5000 time-steps. In practice, however, this can be achieved readily with modern computers, and the numerous steps introduce small errors. This makes the explicit algorithm a practical tool for geological applications as will be shown later.

### *The implicit method and the Crank–Nicholson algorithm*

The fully implicit method expresses the relations between $T_j^{n+1}$ and its neighbors from the left $T_j^n$, bottom $T_{j+1}^{n+1}$, and top $T_{j-1}^{n+1}$ (Fig. 1B)

$$T_j^{n+1} - T_j^n = rT_{j-1}^{n+1} - 2rT_j^{n+1} + rT_{j+1}^{n+1} + \frac{\delta t}{\rho c_p} H_j. \quad (5)$$

Obviously, $T_j^{n+1}$ cannot be calculated directly from Equation (5) because at the $n + 1$ time-step

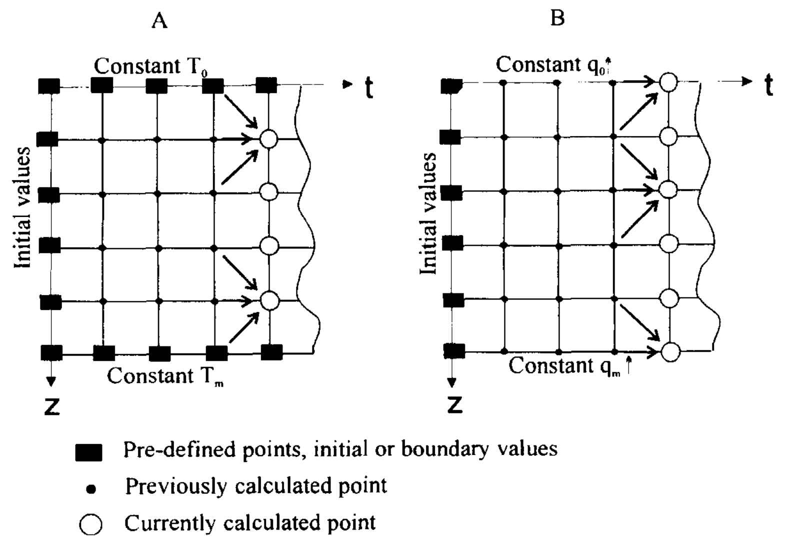

A
Constant T₀
t
Initial values
Constant Tₘ
Z
B
Constant q₀
t
Initial values
Constant qₘ
Z
Pre-defined points, initial or boundary values
Previously calculated point
Currently calculated point

Figure 2. Explicit calculation—table is calculated successively from left to right. A, Boundary conditions are given as constant temperatures. B, Boundary conditions are given as constant heat flows.

$T_{j-1}^{n+1}$ and $T_{j+1}^{n+1}$are still unknown. The only solution is to solve the whole column simultaneously; that is, $N + 1$ algebraic equations in each time-step. This method is more complicated than the explicit method, but it has the advantage of being valid for all values of $r$. Nevertheless, $\delta t$ is still restricted because of the truncation error, which is of the order of $\delta t + \delta z^2$ in both the explicit and the fully implicit methods. The Crank–Nicholson algorithm improves upon the fully implicit method by reducing the truncation error to $O(\delta t^2) + O(\delta z^2)$ and thus allows an enlargement of $\delta t$. This involves some additional computations as expressed in Equation (6) and Figure 1C

$$-rT_{j-1}^{n+1} + (2+2r)T_j^{n+1} - rT_{j+1}^{n+1} = rT_{j-1}^{n} + (2-2r)T_j^{n} + rT_{j+1}^{n} + \frac{2\delta t}{\rho c_p}H_j. \tag{6}$$

When the boundary conditions are given as heat flows, the temperature of the uppermost and lowermost points are calculated using Equations (7) and (8) respectively

$$(2+2r)T_0^{n+1} - 2rT_1^{n+1} = (2-2r)T_0^n + 2rT_1^n - 4r\frac{\delta z}{k}q_0 + \frac{2\delta t}{\rho c_p}H_0 \tag{7}$$

$$(2+2r)T_N^{n+1} - 2rT_{N-1}^{n+1} = (2-2r)T_N^n + 2rT_{N-1}^n + 4r\frac{\delta z}{k}q_N + \frac{2\delta t}{\rho c_p}H_N. \tag{8}$$

Two frequently used methods for solving simultaneous algebraic equations include the direct methods, such as the Gauss elimination or Gauss–Jordan elimination algorithms, and iterative methods such as the Jacobi or Gauss–Seidel algorithms (Ozisik, 1980; Smith, 1985; Cheny and Kincaid, 1980; Press and others, 1986). The banded structure of our system of equations makes the Gaussian elimination procedure efficient and easy to implement in a spreadsheet as will be discussed next.

## IMPLEMENTING THE ALGORITHMS IN A SPREADSHEET

### *The explicit method*

The graphic formulation of the algorithm (Fig. 2) immediately shows how it can be implemented in a spreadsheet. The net is represented physically by a table whose first (leftmost) column is the initial geotherm at $t = 0$, and its $n$th column is the $n$th time-step, that is, the geotherm at $t = n\delta t$. Each table cell contains the formula given by Equation (2), and the table is calculated successively from left to right. The boundaries, the top and bottom rows, contain the constants $T_0$ (surface temperature) and $T_N$ (asthenosphere temperature) respectively. Alternatively, in situations of heat flow boundaries they contain the formulae given by Equations (3) and (4).

The main problem of this implementation is the size of the required table. For instance, in the previously mentioned example at least 5000 columns are needed. However, considering the fact that each time-step is based only on one previous step, the whole net can be represented by two spreadsheet columns only, if each time-step includes copying of the new column to the position of the last column. On the other hand, copying is a relatively time-consuming operation and therefore, we suggest keeping a table with 20–30 columns according to the available memory, and in this way reduce the number of the copy operations to once every 20–30 steps. In addition, all columns which are saved for the final presentation of the solution must be saved as values and not as formulae, which also has the advantage of reducing memory and time resources.

A fully programmed example in a Quattro-Pro file is enclosed in Appendix. A macro called "EVOLUTION" evaluates the thermal evolution of the lithosphere after a thermal perturbation. A sequence of geotherms is calculated and displayed for $t_0$, $t_1$, $t_2$,... $t_n$ according to a time list, presented by the user.

### *The Crank–Nicholson algorithm*

The matrix form of Equation (6) is:

$$\underset{\mathbf{A}}{\begin{bmatrix} 1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 \\ -r & (2+2r) & -r & 0 & \dots & 0 & 0 & 0 & 0 \\ 0 & -r & (2+2r) & -r & \dots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 & \dots & -r & (2+2r) & -r & 0 \\ 0 & 0 & 0 & 0 & \dots & 0 & -r & (2+2r) & -r \\ 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 \end{bmatrix}} \cdot \underset{\mathbf{T}_j^{n+1}}{\begin{bmatrix} T_0^{n+1} \\ T_1^{n+1} \\ T_2^{n+1} \\ \vdots \\ T_{N-2}^{n+1} \\ T_{N-1}^{n+1} \\ T_N^{n+1} \end{bmatrix}} = \underset{\mathbf{D}}{\begin{bmatrix} T_0 \\ rT_0^n + (2-2r)T_1^n + rT_2^n + \frac{2\delta t}{\rho c_p}H_1 \\ rT_1^n + (2-2r)T_2^n + rT2_3^n + \frac{2\delta t}{\rho c_p}H_2 \\ \vdots \\ rT_{N-3}^n + (2-2r)T_{N-2}^n + rT2_{N-1}^n + \frac{2\delta t}{\rho c_p}H_{N-2} \\ rT_{N-2}^n + (2-2r)T_{N-1}^n + rT_N^n + \frac{2\delta t}{\rho c_p}H_{N-1} \\ T_N \end{bmatrix}} \tag{9}$$

The first and bottom rows of **A** and **D** include constant boundary temperatures $T_0$ and $T_N$, in all time-steps, that is:

$$T_0^{n+1} = T_0^n = T_0^{n-1} = .... = T_0^0 = T_0$$

and

$$T_N^{n+1} = T_N^n = T_N^{n-1} = .... = T_N^0 = T_N.$$

Alternatively, when boundary conditions are given by constant heat flow the top and bottom cells should be changed according to Equations (7) and (8) respectively.

At first sight, Equation (9) seems to be solved readily by the spreadsheet matrix operations:

$$\begin{bmatrix} column \\ \mathbf{T}_j^{n+1} \end{bmatrix} = \begin{bmatrix} matrix \\ \mathbf{A}^{-1} \end{bmatrix} \cdot \begin{bmatrix} column \\ \mathbf{D} \end{bmatrix}.$$

However, this is inefficient because it does not take advantage of the tridiagonal form of matrix **A** that may be large. In such instances a simple algorithm is available (see for example Cheny and Kincaid, 1980).

Let

$$\mathrm{A} = \begin{bmatrix} a_0 & b_0 & & & & \mathbf{0} & d_0 \\ c_0 & a_1 & b_1 & & & & d_1 \\ & c_1 & a_2 & b_2 & & & d_2 \\ & & \ddots & \ddots & \ddots & & \vdots \\ & & & & a_{N-1} & b_{N-1} & d_{N-1} \\ \mathbf{0} & & & & c_{N-1} & a_N & d_N \end{bmatrix}. \tag{10}$$

For notation we use $a$'s for the main diagonal, $b$'s for the super diagonal, $c$'s for the sub diagonal and $d$'s for the coefficients of the column matrix **D**. Initially, a forward elimination phase is applied to the array. In step 1 we subtract $c_0/a_0$ times row 1 from row 2, thus creating a zero in $c_0$ position. Note that only the entries $d_1$ and $a_1$ are altered while $b_1$ is unchanged. In step 2 the process is repeated, using the new row 2 as the operating row. In general, the $a$'s and $d$'s are altered in the following way:

$$\overset{*}{a}_0 = a_0,\ \overset{*}{a}_1 = a_1 - b_0\left(\frac{c_0}{\overset{*}{a}_0}\right),$$
$$\overset{*}{a}_2 = a_2 - b_1\left(\frac{c_1}{\overset{*}{a}_1}\right), ...\overset{*}{a}_N = a_N - b_{N-1}\left(\frac{c_{N-1}}{\overset{*}{a}_{N-1}}\right)$$

and

$$\overset{*}{d}_0 = d_0,\ \overset{*}{d}_1 = d_1 - d_0\left(\frac{c_0}{\overset{*}{a}_0}\right),$$
$$\overset{*}{d}_2 = d_2 - d_1\left(\frac{c_1}{\overset{*}{a}_1}\right), ...\overset{*}{d}_N = d_N - d_{N-1}\left(\frac{c_{N-1}}{\overset{*}{a}_{N-1}}\right). \tag{11}$$

At the end of the forward elimination phase the form of the array is as follows:

$$\begin{bmatrix} \overset{*}{a}_0 & b_0 & & & \mathbf{0} & \overset{*}{d}_0 \\ & \overset{*}{a}_1 & b_1 & & & \overset{*}{d}_1 \\ & & \overset{*}{a}_2 & b_2 & & \overset{*}{d}_2 \\ & & & \ddots & \ddots & \vdots \\ & & & \overset{*}{a}_{N-1} & b_{N-1} & \overset{*}{d}_{N-1} \\ \mathbf{0} & & & & \overset{*}{a}_N & \overset{*}{d}_N \end{bmatrix}$$

Now, a back substitution phase solves for $T_n$, $T_{n-1}$,... $T_0$ as follows:

$$T_N = \frac{\overset{*}{d}_N}{\overset{*}{a}_N}$$

$$T_{N-1} = \frac{\overset{*}{d}_{N-1} - b_{N-1}T_N}{\overset{*}{a}_{N-1}}$$

$$T_{N-2} = \frac{\overset{*}{d}_{N-2} - b_{N-2}T_N}{\overset{*}{a}_{N-2}}$$

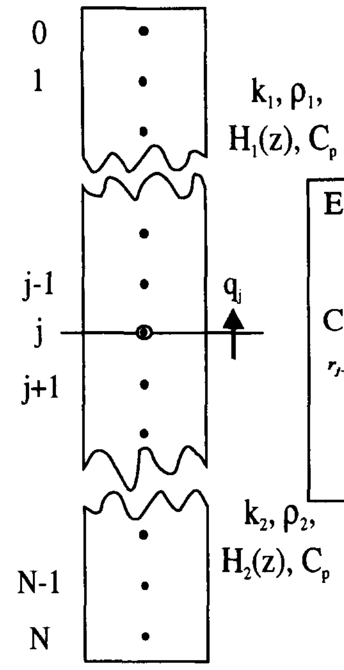
$$\text{etc.} \tag{12}$$

The implementation of this algorithm in a spreadsheet is trivial and requires only seven columns. The original coefficients $a$'s, $b$'s, $c$'s, and $d$'s are stored in four data columns. The new coefficients $a^*$'s and $d^*$'s are calculated by two more columns, containing the formula given by Equation (11). The solution vector $T_j^n$ is calculated by one last column, containing the formula given by Equation (12).

In summary, each time-step of the Crank–Nicholson algorithm requires seven columns, in addition to the previous calculation of the column vector **D**. Although this is more complicated than the single column required for each explicit step, the number of steps is reduced significantly. The efficiency and geological applicability of these two algorithms are discussed next.

## SOLUTION FOR A NONHOMOGENEOUS MEDIUM

The finite-difference equations, which we used in both the explicit and Crank–Nicholson methods, solve the one-dimensional heat-conduction equation (Eq. 1) which is only valid in domains that have constant thermal conductivity $k$. However, in reality, $k$ depends on temperature and pressure as well as on rock type, and therefore varies with depth. The solution for a nonhomogeneous medium is based on the division of the medium into several intervals, each with a separate value of the constant $k$. To be more general, we will develop the solution for a m^edium in which not only the thermal conductivity changes, but also the density and the heat



Explicit:

$$T_j^{n+1} = r_{j-1} T_{j-1}^n + (1-2r_j) T_j^n + r_{j+1} T_{j+1}^n + \frac{\delta t}{(\rho_1+\rho_2)c_p}(H_{1_j} + H_{2_j})$$

Crank-Nicholson:

$$r_{j-1} T_{j-1}^{n+1} + (2+2r_j) T_j^{n+1} - r_{j+1} T_{j+1}^{n+1} = r_{j-1} T_{j-1}^n + (2-2r_j) T_j^n + r_{j+1} T_{j+1}^n + \frac{2\delta t}{(\rho_1+\rho_2)c_p}(H_{1_j} + H_2$$

$$r_{j-1} = \frac{k_1 \delta t}{\frac{1}{2}(\rho_1+\rho_2)c_p \delta z^2} \qquad r_j = \frac{(k_1+k_2)\delta t}{(\rho_1+\rho_2)c_p \delta z^2} \qquad r_{j+1} = \frac{k_2 \delta t}{\frac{1}{2}(\rho_1+\rho_2)c_p \delta z^2}$$

Figure 3. Finite-difference formulas for grid point at boundary between two regions with different properties. Each region is uniform; $c_p$ changes little.

generation function. The changes occur at points of the net, as indicated in Figure 3. For simplicity, we assume a constant $c_p$ for the whole lithosphere because $c_p$ does not change greatly.

## *The explicit method*

The idea is to express the transition point $j$ both as the base of the upper interval and as the top of the lower interval. Equation (3) is used for the top of the lower interval by replacing the subscripts 0 and 1 to $j$ and $j+1$ respectively, and Equation (4) is used for the bottom of the upper interval by replacing the subscripts $N$ and $N-1$ to $j$ and $j-1$ respectively. The two equations use different $\rho$, $k$ and $H(z)$ as described in Figure 3, but share the same $q_j$, which is the heat flow from the lower interval into the upper interval. In this way we obtain two Equations (3′) and (4′) with two unknowns $q_j$ and $T_j^{n+1}$:

$$T_j^{n+1} = 2r_2\left(T_{j+1}^n - \frac{\delta z}{k_2} q_j\right) + (1-2r_2)T_j^n + \frac{\delta t}{\rho_2 c_p} H_{2j} \qquad (3')$$

and

$$T_j^{n+1} = 2r_1\left(T_{j-1}^n - \frac{\delta z}{k_1} q_j\right) + (1-2r_1)T_j^n + \frac{\delta t}{\rho_1 c_p} H_{1j}. \qquad (4')$$

Solving these two equations, the explicit solution for the transition point $T_j^{n+1}$ is:

$$T_j^{n+1} = r_{j-1} T_{j-1}^n + (1-2r_j)T_j^n + r_{j+1} T_{j+1}^n + \frac{\delta t}{(\rho_1+\rho_2)c_p}(H_{1_j} + H_{2_j}), \qquad (2')$$

where

$$r_{j-1} = \frac{k_1 \delta t}{\frac{1}{2}(\rho_1+\rho_2)c_p \delta z^2}, \quad r_j = \frac{(k_1+k_2)\delta t}{(\rho_1+\rho_2)c_p \delta z^2},$$

$$r_{j+1} = \frac{k_2 \delta t}{\frac{1}{2}(\rho_1+\rho_2)c_p \delta z^2}.$$

Note that Equation (2′) has the same form as Equation (2) but with different coefficients that express the properties of the two different intervals.

Now, proper implementation simply requires that we use the appropriate coefficients in the correct positions of our space–time table. That is: $r_1$ in rows of the upper interval, $r_2$ in rows of the lower interval, and Equation (2′) with its three different coefficients $r_{j-1}$, $r_j$ and $r_{j+1}$ in the transition row. In addition, we should use two different functions, $H_1$ and $H_2$, for the distribution of the heat generation. Note that when more than two intervals exist the same procedure can be followed for each interval and each transition row.

## *The Crank–Nicholson method*

The Crank–Nicholson solution for the transition points is obtained in a similar way, using the heat flow $q_j$ to transform Equations (7) and (8) into (7′) and (8′):

$$(2+2r_2)T_j^{n+1} - 2r_2T_{j+1}^{n+1} = (2-2r_2)T_j^n + 2r_2T_{j+1}^n - 4r_2\frac{\delta z}{k_2}q_j + \frac{2\delta t}{\rho_2 c_p}H_{2j} \qquad (7')$$

and

$$(2+2r_1)T_j^{n+1} - 2r_1T_{j-1}^{n+1} = (2-2r_1)T_j^n + 2r_1T_{j-1}^n + 4r_1\frac{\delta z}{k_1}q_j + \frac{2\delta t}{\rho_1 c_p}H_{j1}. \qquad (8')$$

From these two equations we can deduce Equation (6′) which has the same form as Equation (6):

$$-r_{j-1}T_{j-1}^{n+1} + (2+2r_j)T_j^{n+1} - r_{j+1}T_{j+1}^{n+1} = r_{j-1}T_{j-1}^n + (2-2r_j)T_j^n + r_{j+1}T_{j+1}^n + \frac{2\delta t}{(\rho_1+\rho_2)c_p}(H_{1_j}+H_{2_j}). \qquad (6')$$

Again, implementation of this solution requires that we use the appropriate coefficients in the correct positions of matrix **A** and column vector **D**, that is $r_1$ and $r_2$ for the rows of the upper and lower intervals, respectively, and $r_{j-1}$, $r_j$ and $r_{j+1}$ for the transitional rows. In addition, densities and heat generation functions in **D** also should be used properly.

## GEOLOGICAL APPLICATIONS

Mathematically, the Crank–Nicholson algorithm has the advantage of yielding stable results for any positive value of $r$. In practice, however, stability problems occur with large values of $r$. The mathematical stability implies that eventually, when $n$ tends to infinity, all errors tend to zero. However, during that interval unwanted finite oscillations may be introduced into the calculations if large values of $r$ are used. Such oscillations die away with increasing $n$, and usually occur in the $z$-neighborhood of points of discontinuity in the initial conditions or between initial conditions and boundary conditions (Smith, 1985). These oscillations limit our ability to use the Crank–Nicholson algorithm, because in many geological applications discontinuous functions are needed to describe thermal perturbations, and sometimes the unwanted oscillations die away too slowly even on a geological time-scale.

For example, the thermal perturbation caused by a magmatic intrusion is described in Figure 4 by a discontinuous function. Various values of $r$ were tested in order to evaluate the amplitude and duration of the unwanted oscillations. The results show that oscillations are developed near the discontinuity points in all cases of $r$ larger than 1.5. The duration of these oscillations is about 0.5 Ma for $r = 1.5$; 2–3 Ma for $r = 2.5$; about 12 Ma for $r = 5$; and about 45 Ma for $r = 10$. It also should be noted that for $r = 10$, small oscillations (10–15°C) develop not only near the discontinuity points, but also near the zero point where the curve is continuous mathematically. This is explained by the fact that the numerical representation of the curve near the zero point is not "smooth enough", that is, the depth intervals are too large. This can be seen better in Figure 5 where continuous initial functions are used to simulate heating of the lithosphere, and nevertheless, oscillations are developed where the function is too curved. Moreover, Figure 5B indicates that for the same initial function if $r$ is increased to 10, these oscillations are moderately strong (100–150°C). Figure 5C indicates that sometimes the oscillations are not so strong, but decay slowly (more than 150 Ma).

Therefore, we conclude that it is unwise to use large values of $r$ with the Crank–Nicholson algorithm, because they may cause unexpected stability problems. Thus, if an unconditional computer program is wanted, either the explicit algorithm with $r < 0.5$ or the Crank–Nicholson algorithm with $r < 1$ should be used. However, such small $r$ values make the Crank–Nicholson algorithm expensive in computer time. Nevertheless, if only large-scale results are investigated, that is, after the unwanted oscillations die away, or if continuous functions are used as initial values, the Crank–Nicholson algorithm may be used with relatively large $r$ values (5–10) to save computer time. Another practical solution we suggest is to begin with the explicit algorithm in order to smooth the curve, and then to change to the Crank–Nicholson algorithm using relatively large $r$ values (Fig. 6). Changing $r$ in the middle of the computation process, or switching from explicit to Crank–Nicholson files, is achieved easily with a spreadsheet that enables much flexibility and a simultaneous use of several opened files.

## SUMMARY AND CONCLUSIONS

(1) We have shown how to implement the explicit and the Crank–Nicholson algorithms in a spreadsheet program.
(2) The explicit algorithm is implemented exactly as it is described in textbooks as a space–time net, and is straightforward to work with. A fully programmed example of an explicit file is enclosed in Appendix 1.
(3) The Crank–Nicholson algorithm is described in a matrix form and is implemented by the Gaussian elimination procedure, taking advantage of the tridiagonal structure of the matrix.
(4) The mathematical advantage of the Crank–Nicholson algorithm, that is, the stability for

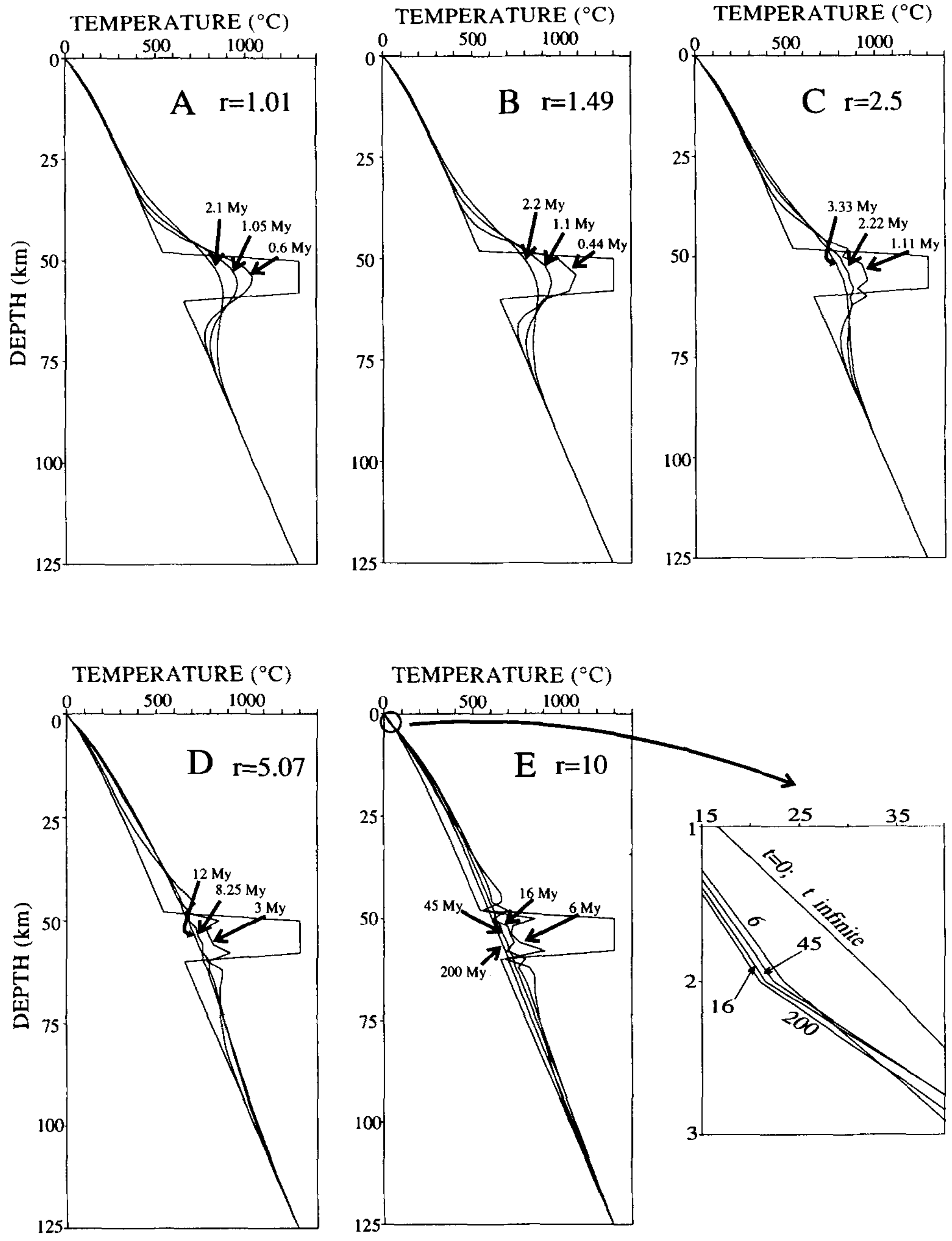
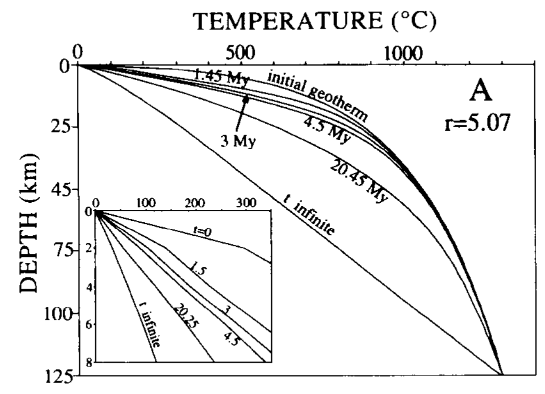
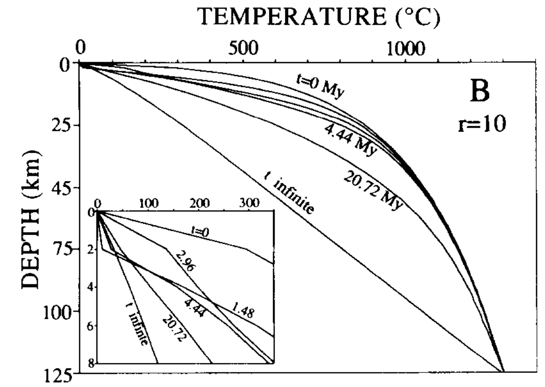


Figure 4. Dissipation of heat, following thermal perturbation caused by magmatic intrusion, calculated by Crank–Nicholson algorithm. Finite oscillations develop near discontinuity points in all situations of $r \geq 1.5$. Amplitudes and duration of unwanted oscillations increase when larger values of $r$ are used. Note that for $r = 10$ (Fig. 4E) small oscillations are developed not only near discontinuous points but also at depth of 2 km, where numerical presentation "breaks" the continuous curve. Phenomenon is emphasized in Figure 5.





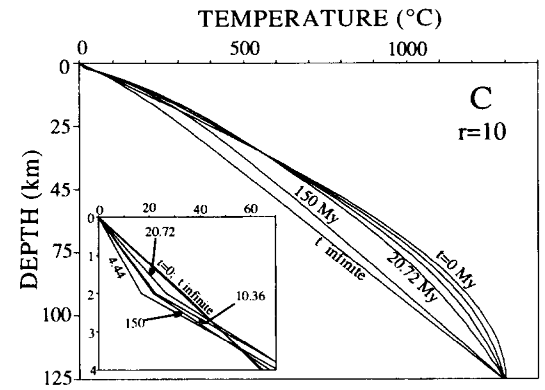


Figure 5. Dissipation of heat, following thermal perturbation modeled by continuous initial functions, calculated by Crank–Nicholson algorithm. Oscillations are developed near surface when r $>5$, because numerical presentation of curves is not "smooth" enough. These oscillations might be strong (100–150°C) as seen in Figure 5B, or long (more than 150 Ma), as seen in Figure 5C.

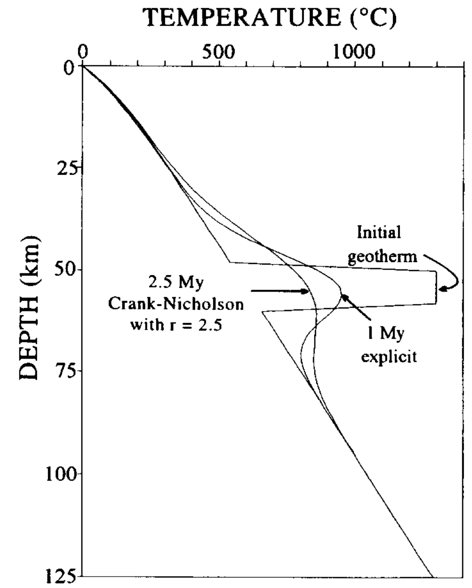


Figure 6. Combination of explicit and Crank–Nicholson algorithms in order to calculate heat dissipation after same initial perturbation as in Figure 4. At first, explicit algorithm was used to smooth the discontinuous initial function. Then, Crank–Nicholson algorithm was used for another 1.5 Ma with $r = 2.5$ and did not produce unwanted oscillations. Note that in Figure 4C when Crank–Nicholson algorithm was used with same $r$ value and initial function, it produced oscillations that lasted more than 3 Ma.

all positive $r$ values, is limited practically only to $r < 10$ and to smooth and continuous initial functions. Otherwise, unwanted oscillations may occur. Therefore, we recommend the routine use of the explicit algorithm which was found to be suitable for these geological applications, and the saving of the Crank–Nicholson algorithm for special situations in which it can be used with large $r$ values.

(5) Another practical approach is to begin with the explicit method until the geotherm is smoothed and then change to the Crank–Nicholson algorithm and proceed with larger time-steps.

The macro code is available by anonymous FTP from the server at IAMG.ORG.

## REFERENCES

Beaumont, C., Keen, C. E., and Boutillier, R., 1982, On the evolution of rifted continental margins: comparison of models and observations for Nova Scotia margin: Geophys. Jour. Roy. Astronomical Soc., v. 70, no. 3, p. 667–715.

Carslaw, H., and Jaeger, J. C. 1959, Conduction of heat in solids: Oxford Univ. Press, New York, 518 p.

Cheny, W., and Kincaid, D., 1980, Numerical mathematics and computing: Brooks/Cole, Monterey, California, 362 p.

Day, H. W., 1987, Controls on the apparent thermal and baric structure of mountain belts: Jour. Geology, v. 95, p. 807–824.

De Yoreo, J. J., Lux, D. R., and Guidotti, C. V., 1991, Thermal modeling in low-pressure/high-temperature metamorphic belts: Tectonophysics, v. 188, no. 3/4, p. 209–238.

England, P. C., and Thompson, A. B., 1984, Pressure–temperature–time paths of regional metamorphism, part I, heat transfer during the evolution of regions of thickened continental crust: Jour. Petrology, v. 25, no. 4, p. 894–928.

Ozisik, N., 1980, Heat conduction: John Wiley & Sons, New York, 687 p.

Peacock, S. M., 1987, Creation and preservation of subduction-related inverted metamorphic gradients: Jour. Geophys. Res., v. B92, no. 12, p. 763–781.

Press, W. P., Flannery, B. P., Teukolsky, S. A., and Vetterling, W. T., 1986, Numerical recipes: the art of scientific computing: Cambridge Univ. Press, Cambridge, 818 p.

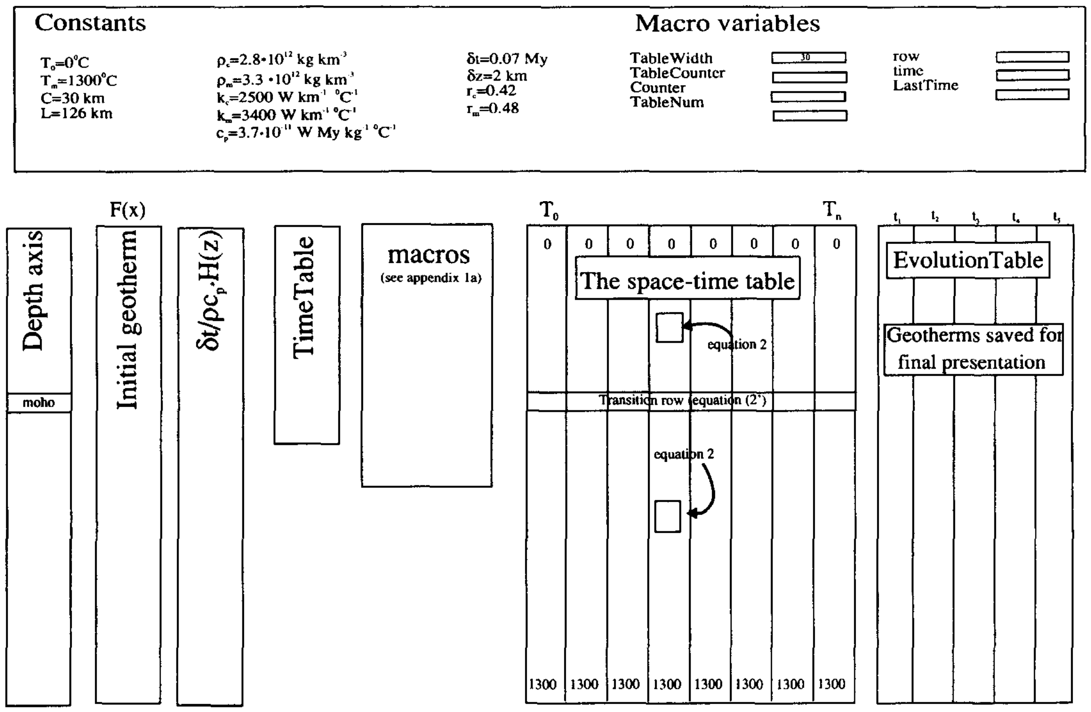
Royden, L., Sclater, J. G., and Von Herzen, R. P., 1980, Continental margin subsidence and heat flow: important parameters in formation of petroleum hydrocarbons: Am. Assoc. Petroleum Geologists Bull., v. 64, no. 2, p. 173–187.

Sandiford, M., 1989, Secular trends in the thermal evolution of metamorphic belts: Earth Planet. Sci. Lett., v. 95, p. 85–96.

Smith, G. D., 1985, Numerical solutions of partial differential equations: finite difference methods (3rd ed.): Oxford Univ. Press, Oxford, 337 p.

## APPENDIX 1

Upper part of spreadsheet contains constants and variables. Note that km and Ma are used for space and time units not m and s. Variables are described in macros listing (Appendix 2). Space–time table is heart of computation and its formulae are described in text. Initial geotherm and heat generation function $H(z)$ should be specified by user. In order to use this spreadsheet, user should fill the TimeTable with time values $t_1,t_2,t_3,...$ as required, and then run main macro "Evolution". Desired geotherms at times $t_1,t_2,t_3,...$ will be presented in Evolution Table.



## APPENDIX 2

*Listing Of Macros For Quattro-Pro*

Description of variables:

| | |
|---|---|
| TableWidth: | The width of the space-time table determined by the user according to the available memory. |
| TableNum: | The number of tables required for the calculation of the current geotherm. |
| TableCounter: | Counts the number of space-time tables, which were calculated until now. |
| Counter: | Counts the number of geotherms that should be saved in the EvolutionTable for final presentation. |
| Time: | The time of the currently calculated geotherm. |
| LastTime: | The time of the last calculated geotherm. |
| Row: | A temporary counter used for some macros for the row number. |

Evolution is the main macro. It goes over the TimeTable, calculates the geotherm for each time value and save it for final presentation.

```
Evolution  {initialize}
           {let counter,0}
           {goto}TimeTable~
loop       {if @cellpointer("type")="b"} {branch end}            / End of TimeTable /
           {let LastTime,Time}
           {let time,@cellpointer("contents")}      / Get the next value in the TimeTable /
           {geotherm}                      / Calculate geotherm  using the space-time table/
           {SaveGeotherm}          / Save the calculated geotherm in the EvolutionTable /
           {goto}TimeTable~
           {let counter, counter+1}
           {down counter}
           {branch loop}

end        {beep 1}{beep 2}{beep 3}{beep 4}{esc}
           {graph}  / End of macro. Present graph. Series are the columns of the EvolutionTable /
```

Initialize copies the initial condition F(x), specified by the user, into the first column $T_0$, of the space-time table.

```
Initialize {/ CompCalc;Manual}              / Do not calculate spreadsheet after each change /
           {let TableCounter,1}
           {let time,0}
           {let @cellindex("address",To,0,0),0}   / First row of T0 represents the time step n;  /
           {let @cellindex("address",To,0,1),0}   / The second  row is nδt. both are initially 0.  /
           {let row,3}
           {goto}F(x)~
LoopInit   {if @cellpointer("type")="b"}{branch EndInit}  / Copy the initial condition F(x) to T0 /*
           {let @cellindex("address",To,0,row),@cellpointer("contents")}
           {down}
           {let row,row+1}
           {branch LoopInit}

EndInit    {calc}{/ CompCalc;Background}
```

Geotherm calculates the current geotherm at t=Time. The number of the required space-time tables is calculated according to the number of the required time-steps. In each iteration of the macro the last column Tn of the table is copied to the first column $T_0$.

```
Geotherm  {let TableNum,@int((time/delta-t/TableWidth)+1)-TableCounter}
  again    {if TableNum<=0}{branch finish}
           {SaveLast}                          / Copy Tn into T0 /
           {let TableCounter,TableCounter+1}
           {let TableNum,TableNum-1}
           {branch again}

  finish
```

SaveLast copies the last column into the first.

```
SaveLast   {/ CompCalc;Manual}
           {goto}To~{right TableWidth}
           {let row,0}
  SL-loop  {if @cellpointer("type")="b"}{branch SL-end}
           {let @cellindex("address",To,0,row),@cellpointer("contents")}
           {down}
           {let row,row+1}
           {branch SL-loop}

  SL-end   {calc}{/ CompCalc;Background}
```

Savegeotherm finds the appropriate column in the space-time table for t=Time and save it in the EvolutionTable.

```
SaveGeotherm  {/ CompCalc;Manual}
              {goto}To~
              {right @mod(time/Delta-t,TableWidth)}
              {let row,0}
    LoopSave  {if @cellpointer("type")="b"} {branch EndSave}
              {let @cellindex("address",EvolutionTable,Counter,row),@cellpointer("contents")}
              {let @cellindex("address",T(t),0,row),@cellpointer("contents")}
              {down}
              {let row,row+1}
              {branch LoopSave}

    EndSave   {calc}{/ CompCalc;Background}{esc}
              {let @cellindex("address",EvolutionTable,Counter,0),uplift}
```