DEPARTMENT OF THE INTERIOR

U.S. GEOLOGICAL SURVEY

Conductive Cooling of Dikes with Temperature-dependent Thermal Properties and Heat of Crystallization

by

Paul T. Delaney [1]

Open-File Report 86-444

[1] U.S. Geological Survey
2255 North Gemini Dr.
Flagstaff, Arizona, 86001

1986

CONTENTS Page

Abstract

Temperature histories obtained from transient heat-conduction theory are applicable to most dikes despite potential complicating effects related to magma flow during emplacement, groundwater circulation, and metamorphic reaction during cooling. Here, machine-independent FORTRAN77 programs are presented to calculate temperatures in and around dikes as they cool conductively. Analytical solutions can treat thermal-property contrasts between the dike and host rocks, but cannot address the release of magmatic heat of crystallization after the early stages of cooling or the appreciable temperature dependence of thermal conductivity and diffusivity displayed by most rock types. Numerical solutions can incorporate these additional factors. The heat of crystallization can raise the initial temperature at the dike contact, Θ_{ci}, about $100^{\circ}C$ above that which would be estimated if it were neglected, and can decrease the rate at which the front of solidified magma moves to the dike center by a factor of as much as three. Thermal conductivity and diffusivity of rocks increase with decreasing temperature and, at low temperatures, these properties increase still more if the rocks are saturated with water. Models that treat these temperature dependencies yield estimates of Θ_{ci} that are as much as $75^{\circ}C$ beneath those which would be predicted if they were neglected.

Key words: Numerical analysis, finite-difference method, dikes, heat conduction, igneous intrusion, contact metamorphism.

Introduction

Recent advances, particularily in the field of paleomagnetics, have provided measurement techniques that allow maximum temperatures attained in wallrocks adjacent to cooling dikes to be estimated with an accuracy of about $\pm 25^oC$ (*e.g.*, Buchan *et al.* 1980; McClelland-Brown, 1981). By combining these results with estimates of the initial magma temperature and the thermal properties of the magma and host rocks, it is possible to calculate the ambient temperature of the host rocks at the time of dike emplacement. This ambient temperature, in turn, can be combined with an estimate of the geothermal gradient to determine the approximate depth of emplacement, as well as the subsequent rate of uplift and denudation (see Buchan and Schwarz, 1986, for a review of the method). The method uses heat-conduction theory to determine cooling history; this is an excellent approximation for most dikes because they cool too quickly for buoyant hydrothermal circulation to become established. However, analytical solutions for conductive cooling are based upon assumptions that are too restrictive to permit temperatures to be calculated with an accuracy comparable to that of the measurements cited above (see Delaney, 1986). It is the purpose of this paper to provide FORTRAN77 programs that allow magmatic heat of crystallization and temperature-dependent thermal properties to be included in conductive cooling models of dikes.

Thermal properties

The accuracy of theoretical calculations for temperatures depends not only upon validity of the assumptions used to derive solutions, but also upon the accuracy of the thermal property data, particularily conductivity, $\underline{k}$, diffusivity, κ, and magmatic heat of crystallization, $\underline{h}$ (all mathematical symbols are summarized in Table 1). Although $\underline{k}$ and κ vary with lithology, these properties vary even more with temperature (Fig. 1). For most rocks at temperatures below 200^oC, $\underline{k}$ lies between 1.5 and 3.0 W/m^oC (quartzite is a notable exception), and κ typically ranges from 0.5 to 1 $x10^{-6}$ m^2/s. Data collected at temperatures above 500^oC are sparse for most rock types. Fortunately, thermal properties of dense basalt have been measured at temperatures well in excess of 500^oC, and extrapolation to supersolidus conditions should not introduce large errors in thermal calculations. For example, *k* values from 1 to 1.5 W/m^{o}C and κ values from 0.25- to 0.75-x10^{-6} m^2/s appear to be typical for basalt above 500^{o}C. High-temperature values of *k* and κ tend to be about 1/2 to 2/3 that which would be measured at low temperature.

Data for *k* and κ can be described by a variety of empirical equations (*e.g*, Ramey *et al*., 1974); we chose

$$k = a_1 + b_1/(273.15+\Theta) \quad (1a)$$

$$\kappa = a_2 + b_2/(273.15+\Theta) \quad (1b)$$

where temperature Θ is in degrees Centigrade and the coefficients are obtained by least-squares fitting. Although equations of this form are used in the examples cited below, any equation can be used by altering the functions CONDM, CONDH, DIFFM, and DIFFH (Appendix 1), which return conductivity and diffusivity of the magma and host-rocks regions, respectively, given an input temperature. The conductivity modulus, γ, defined by

$$\gamma = (dk/d\Theta)/k \quad (2)$$

is returned by the functions CONDMM and CONDMH (Appendix 1), given an input temperature.

The thermal properties summarized in Figure 1 apply to "dry" rocks. The presence of pore water, however, can have a profound effect upon thermal conductivity. For instance, rocks with porosities of only 1% typically have water-saturated *k* about 10% greater than that for the dry sample (Touloukian *et al*., 1981, p. 428). Thermal conductivity of water-saturated rock can be estimated from data for the dry rock using

$$k_{wet} = k_{dry}(k_w/k_{air})^{\phi} \quad (3)$$

where ϕ is porosity and the subscript *w* refers to water (Sass *et al*., 1971). At modest pressures and 50^{o}C, $k_w/k_{air} \simeq 20$, so that a rock with 10% porosity may exhibit an increase in conductivity of ~40% if saturated with water. At 20 MPa and 550^{o}C, a characteristic pressure and temperature near a dike contact at 2 km depth, the conductivity of water is small and $k_w/k_{air} \simeq 2$, so that the increase in conductivity due to water saturation is only about 7%. The influence of water upon $\underline{k}$ is therefore greatly temperature dependent.

Although measurements of conductivity and diffusivity can, in principle, be used to estimate heat capacity per unit volume $\rho\underline{C}$ ($= \underline{k}/\kappa$) such estimates rarely provide good fits to heat-capacity data. In this report, data for $\underline{k}$ and κ are used rather than data for $\rho\underline{C}$. It is nevertheless instructive to note that heat capacity of a water-saturated rock closely follows the relation $(\rho\underline{C})_{wet} = (1-\phi)(\rho\underline{C})_r + \phi(\rho\underline{C})_w$, where the subscript r denotes the rock matrix. For temperatures typical of the upper crust, $(\rho\underline{C})_r \simeq$ 2-3 MJ/m$^3\cdot{}^oC$ and $(\rho\underline{C})_w \simeq$ 3-4 MJ/m$^3\cdot{}^oC$. Because, in general, $\phi << 1$, most heat resides in the solid matrix and not in pore water, so that porosity and pore fluid do not substantially influence the heat capacity of most rocks.

The heat released by a magma as it cools from its initial temperature Θ_{mi} to its solidus temperature Θ_s varies with chemistry and cooling rate. For a quickly quenched, glassy magma, the heat released is $(\rho\underline{C})_m(\Theta_{mi}-\Theta_s)$; if cooled slowly to form a holocrystalline rock, then a heat of crystallization $(\rho\underline{h})_m$ is released also. For unvesiculated magmas, the heat of crystallization per unit volume is typically 700-1000 MJ/m^3. It is assumed to be released evenly through the temperature interval from the intrusion temperature Θ_{mi} to the solidus temperature. The heat of crystallization can then be treated by defining a modified heat capacity and diffusivity:

$$(\rho\underline{C})'_m = (\rho\underline{C})_m + (\rho\underline{h})_m/(\Theta_{mi}-\Theta_s) \quad (4a)$$

$$\kappa'_m = \underline{k}_m/(\rho\underline{C})'_m \quad (4b)$$

where the subscript $\underline{m}$ denotes the magma and the subscript $\underline{i}$ denotes its initial (uncooled) value. These modified values apply only above the solidus temperature.

Problem formulation

Dikes are commonly idealized as tabular bodies of thickness $\underline{T}$. At all positions not near the dike tip, heat conducts only in directions normal to the contact. We position an $\underline{X}$ coordinate so that it originates at one of the contacts and points away from the dike; the center of the dike is at $\underline{X} = -\underline{T}/2$.

We wish to obtain solutions to the heat-conduction equation for temperature-dependent thermal properties:

$$(\rho \underline{C})\ \partial\Theta/\partial\underline{t} = \partial\underline{k}\partial\Theta/\partial\underline{X}^2 \qquad (5)$$

(Carslaw and Jaeger, 1959, Ch. 1). Heat flux is given by Fourier's law, $\underline{Q} = -\underline{k}\partial\Theta/\partial\underline{X}$, and the gradient in this flux (right-hand side, eqs. 5) is exactly balanced with the accumulation of heat (left-hand side, eqs. 5). All heats of reaction--other than the crystallization of the magma, which will be incorporated through use of eqs. 4--and all heat transfer arising from fluid motion are assumed to be negligible.

From symmetry considerations, it is only necessary to consider the half-space $\underline{X} > -\underline{T}/2$. Thermal diffusion equations are distinguished for the host-rock region ($\underline{X} > 0$), the solidified-magma region ($\chi_s < \underline{X} < 0$), and the liquid-magma region ($-\underline{T}/2 < \underline{X} < \chi_s$), where χ_s is the position of the solidus isotherm Θ_s. Thermal properties are denoted by the subscripts $\underline{h}$ and $\underline{m}$ in the host-rock and magma (dike) regions, respectively. Properties of that portion of the dike occuppied by magma are denoted with a prime (eqs. 4); this region need not be distinguished from the rest of the dike if no magmatic heat of crystallization is released. The magma region diminishes in size during cooling and disappears after the center of the dike cools beneath the solidus temperature. After expanding the right-hand side of eq. 5, the three energy equations are:

$$\partial\Theta/\partial\underline{t} = \kappa_h[\partial^2\Theta/\partial\underline{X}^2 + \underline{k}_h^{-1}(\partial\underline{k}_h/\partial\underline{X})(\partial\Theta/\partial\underline{X})] \qquad (0 < \underline{X} < \infty) \qquad (6a)$$

$$\partial\Theta/\partial\underline{t} = \kappa_m[\partial^2\Theta/\partial\underline{X}^2 + \underline{k}_m^{-1}(\partial\underline{k}_m/\partial\underline{X})(\partial\Theta/\partial\underline{X})] \qquad (\chi_s < \underline{X} < 0) \qquad (6b)$$

$$\partial\Theta/\partial\underline{t} = \kappa_m'[\partial^2\Theta/\partial\underline{X}^2 + \underline{k}_m^{-1}(\partial\underline{k}_m/\partial\underline{X})(\partial\Theta/\partial\underline{X})] \qquad (-\underline{T}/2 < \underline{X} < \chi_s;\ \Theta > \Theta_s) \qquad (6c)$$

These equations have not been solved by analytical methods for cases where $\underline{k}$ and κ have temperature dependencies similar to those exhibited by rocks; they have been treated by numerical methods (e.g., Giberti et al., 1984). To obtain solutions, three initial and six boundary conditions must be specified:

$$\Theta(\underline{X}>0,\underline{t}=0) = \Theta_{hi}, \quad \Theta(\chi_s<\underline{X}<0,\underline{t}=0) = \Theta_{mi}, \quad \Theta(-\underline{T}/2<\underline{X}<\chi_s,\underline{t}=0) = \Theta_{mi} \tag{7a}$$

$$\begin{aligned} &\Theta(\infty,\underline{t}) = \Theta_{hi}, \quad \Theta(0^-,\underline{t}) = \Theta(0^+,\underline{t}) \equiv \Theta_c, \quad \Theta(\chi_s^-,\underline{t}) = \Theta(\chi_s^+,\underline{t}) \equiv \Theta_s \\ &\underline{Q}(0^-,\underline{t}) = \underline{Q}(0^+,\underline{t}), \quad \underline{Q}(\chi_s^-,\underline{t}) = \underline{Q}(\chi_s^+,\underline{t}), \quad \underline{Q}(\underline{X}=-\underline{T}/2,\underline{t}) = 0 \end{aligned} \tag{7b}$$

Equations 7a give the initial temperatures of the host rocks and magma. The first condition of eq. 7b assures that temperatures at great distance from the dike contact remain unchanged throughout the duration of cooling; the next two conditions assure that temperatures are continuous at the dike contact ($\underline{X} = 0$) and at the position of the solidus isotherm ($\underline{X} = \chi_s$). The latter three conditions assure that heat fluxes are continuous at the dike contact and at the position of the solidus isotherm, and that no heat crosses the mid-plane of the dike.

Before cooling commences at the dike center, the problem posed by eqs. 6 and 7 can be written in terms of the similarity variable:

$$\eta = \underline{X}/\sqrt{4\kappa_{hi}\underline{t}} \tag{8}$$

where the subscript $\underline{hi}$ denotes the host rock at its initial temperature. Distance and time are combined into a single independent variable, and the position of the solidus isotherm is transformed to a constant

$$\lambda_s = \chi_s/\sqrt{4\kappa_{hi}\underline{t}} \tag{9}$$

so that $\chi_s(\underline{t}) \propto 1/\sqrt{\underline{t}}$. Incorporating the definition of γ, eq. 2, the transformed thermal diffusion equations can be written as:

$$-2\eta\, \underline{d}\Theta/\underline{d}\eta = (\kappa_h/\kappa_{hi})[\underline{d}^2\Theta/\underline{d}\eta^2 + \gamma_h(\underline{d}\Theta/\underline{d}\eta)^2] \qquad ({}^0 < \eta < \infty) \tag{10a}$$

$$-2\eta\, \underline{d}\Theta/\underline{d}\eta = (\kappa_m/\kappa_{hi})[\underline{d}^2\Theta/\underline{d}\eta^2 + \gamma_m(\underline{d}\Theta/\underline{d}\eta)^2] \qquad (\lambda_s < \eta < 0) \tag{10b}$$

$$-2\eta\, \underline{d}\Theta/\underline{d}\eta = (\kappa_m'/\kappa_{hi})[\underline{d}^2\Theta/dx^2 + \gamma_m(\underline{d}\Theta/\underline{d}\eta)^2] \qquad (-\infty < \eta < \chi_s) \tag{10c}$$

Thus, the partial differential equations 6 are transformed into nonlinear ordinary differential equations. Equations 10 are to be solved in conjunction with six boundary conditions:

$$\Theta(\infty) = \Theta_{hi}, \qquad \Theta(0^-) = \Theta(0^+) \equiv \Theta_{ci}, \qquad \Theta(\lambda_s^-) = \Theta(\lambda_s^+) \equiv \Theta_s$$
$$\underline{Q}(0^-) = \underline{Q}(0^+), \qquad \underline{Q}(\lambda^-) = \underline{Q}(\lambda^+), \qquad \Theta(-\infty) = \Theta_{mi} \tag{11}$$

The problem posed by eqs. 10 and 11 is known as a Stephan problem (Carslaw and Jaeger, 1959, Ch. 11). The properties of the transformation, eq. 8, show that, within the assumptions of the continuum theory, the temperature at the dike contact instantaneously rises to a value Θ_{ci} and remains unchanged until some time after the transformation becomes invalid. This occurs when the boundary condition $\Theta(-\infty) = \Theta(-\underline{T}/2,\underline{t}) = \Theta_{mi}$, becomes invalid and must be replaced by $\underline{Q}(-\underline{T}/2,\underline{t}) = 0$; this is when cooling commences at the dike center.

Analytical solutions

For brevity and to emphasize scaling relations, analytical solutions are often written in terms of nondimensional variables. Nondimensional distance, time, and temperature are given by:

$$\underline{x} = \underline{X}/(\underline{T}/2) \tag{12a}$$
$$\tau = \underline{t}\cdot\kappa_{hi}/(\underline{T}/2)^2 \tag{12b}$$
$$\theta = (\Theta-\Theta_{hi})/(\Theta_{mi}-\Theta_{hi}) \tag{12c}$$

Distance is normalized by dike half-thickness, time by the diffusivity of the host rocks at ambient temperature and the square of dike half-thickness, and temperature by the initial temperature difference between the magma and host rocks. All analytical solutions require that $\underline{k}_m$, $\underline{k}_h$, κ_m, and κ_h ($\equiv\kappa_{hi}$) be constants.

<u>Early-time solutions</u>. The solution to the problem posed by eqs. 10 and 11, is:

$$\theta = 1 - (1-\theta_s)[2-\mathrm{erfc}(\eta\sqrt{\kappa_h/\kappa_m})]/[2-\mathrm{erfc}(\lambda_s\sqrt{\kappa_h/\kappa_m})] \qquad (\eta<\lambda_s) \tag{13a}$$

$$\theta = \theta_{ci} + (\theta_s-\theta_{ci})\cdot\mathrm{erf}(\eta\sqrt{\kappa_h/\kappa_m})/\mathrm{erf}(\lambda_s\sqrt{\kappa_h/\kappa_m}) \qquad (\lambda_s<\eta<0) \tag{13b}$$

$$\theta = \theta_{ci}\mathrm{erfc}(\eta) \qquad (\eta>0) \tag{13c}$$

(see Carslaw and Jaeger, 1959, Ch. 11) where erf is the error function and erfc its compliment. Differentiating eqs. 13 to obtain heat fluxes, and requiring that heat flux into and out of the interfaces between adjoining regions be the same, we obtain two equations to determine the unknowns λ_s and θ_{ci}:

$$\frac{\sqrt{\kappa_h/\kappa'_m}(1-\theta_s)\cdot e^{-\lambda_s^2\kappa_h/\kappa'_m}}{2-\mathrm{erfc}\left(\lambda_s\sqrt{\kappa_h/\kappa'_m}\right)}+\frac{\sqrt{\kappa_h/\kappa_m}(\theta_s-\theta_{ci})\cdot e^{-\lambda_s^2\kappa_h/\kappa_m}}{\mathrm{erf}\left(\lambda_s\sqrt{\kappa_h/\kappa_m}\right)}=0 \qquad (14a)$$

$$\frac{(\underline{k}_m/\underline{k}_h)\sqrt{\kappa_h/\kappa_m}(\theta_s-\theta_{ci})}{\mathrm{erf}\left(-\lambda_s\sqrt{\kappa_h/\kappa_m}\right)}-\theta_{ci}=0 \qquad (14b)$$

If the magma is quenched quickly such that $(\rho\underline{h})_m = 0$, then $\Theta_s = \Theta_{mi}$, $\lambda_s \to -\infty$, and temperatures are given by eqs. 13b, 13c, and 14b only. Equations 13 and 14 are used in the program EARLYA (Appendix 1) to provide temperatures as $\Theta(\eta)$ and $\Theta(\underline{X},\underline{t})$.

Whole-time solution. The most general analytical solution to the problem posed by eqs. 6 and 7 is:

$$\theta = \frac{1+\underline{c}}{2}\left\{\sum_{\underline{n}=1}^{\infty}(-\underline{c})^{\underline{n}-1}\left[\mathrm{erf}\left(\frac{2\underline{n}-\underline{x}}{\sqrt{4\underline{d}\tau}}\right)-\underline{c}\cdot\mathrm{erf}\left(\frac{2\underline{n}+\underline{x}}{\sqrt{4\underline{d}\tau}}\right)\right]-\mathrm{erf}\left(\frac{\underline{x}}{\sqrt{4\underline{d}\tau}}\right)\right\} \qquad (-2<\underline{x}<0) \qquad (15a)$$

$$\theta = \frac{1-\underline{c}}{2}\left\{(1+\underline{c})\sum_{n=1}^{\infty}(-\underline{c})^{\underline{n}-1}\cdot\mathrm{erf}\left(\frac{\underline{x}+2\underline{n}/\underline{d}}{\sqrt{4\tau}}\right)-\mathrm{erf}\left(\frac{\underline{x}}{\sqrt{4\tau}}\right)\right\} \qquad (\underline{x}>0) \qquad (15b)$$

where

$$\underline{c} = \frac{\underline{d}-\underline{k}_m/\underline{k}_h}{\underline{d}+\underline{k}_m/\underline{k}_h} \qquad (16a)$$

$$\underline{d} = \sqrt{\kappa_m/\kappa_h} \qquad (16b)$$

(Lovering, 1936). The initial temperature at the dike contact is given by:

$$\theta_{ci} = \frac{\underline{k}_m/\underline{k}_h}{\underline{k}_m/\underline{k}_h+\sqrt{\kappa_m/\kappa_h}} \qquad (17)$$

Equations 15-17 show that temperatures are affected by thermal property contrasts, rather than the absolute values of those properties. If $\underline{k}_m/\underline{k}_h = \kappa_m/\kappa_h = 1$, then

$$\theta = \frac{1}{2}\{\mathrm{erf}[(2-\underline{x})/\sqrt{4\tau}] - \mathrm{erf}[\underline{x}/\sqrt{4\tau}]\} \qquad (18)$$

so that the initial contact temperature is $\theta_{ci} = 0.5$. Equations 15 and 16, or 18, can be used to estimate the influence of the magmatic heat of crystallization on temperatures in the wallrocks by defining an equivalent intrusion temperature

$$\Theta'_{mi} = \Theta_{mi} + \underline{h}/\underline{C} \qquad (19)$$

This approximation is inaccurate at distances less than about $\underline{T}/4$ from the dike contact, but is acceptable at greater distances if accuracies not better than about 40°C are acceptable (Jaeger, 1964).

The program WHOLEA (Appendix 1) provides temperatures as $\Theta(\underline{X},\underline{t})$, as well as the maximum temperature $\Theta_{max}(\underline{X})$.

Numerical solutions

At times less than that required for cooling to commence at the dike center, numerical solutions to the problem posed by eqs. 6 and 7 are not accurate unless a great number of finite-difference nodes are used. Rather, at early times, eqs. 10 and 11 are solved, and temperatures are then converted from $\Theta(\eta)$ to $\Theta(\underline{X},\underline{t})$. This temperature distribution is used as an initial condition for subsequent solution of eqs. 6 and 7.

Early-time solutions. Integration of eqs. 10 are executed by a Runge-Kutta method (Thompson, 1970) for first-order differential equations. Each eq. 10 is converted to two first-order equations by the substitution

$$\psi = \underline{d}\Theta/\underline{d}\eta \qquad (20)$$

which must be solved in conjunction with

$$-2\eta\psi = (\kappa_h/\kappa_{hi})[d\psi/d\eta + \gamma_h\psi^2] \qquad (0 < \eta < \infty) \qquad (21a)$$

$$-2\eta\psi = (\kappa_m/\kappa_{hi})[d\psi/d\eta + \gamma_m\psi^2] \qquad (\lambda_s < \eta < 0) \qquad (21b)$$

$$-2\eta\psi = (\kappa_m'/\kappa_{hi})[d\psi/d\eta + \gamma_m\psi^2] \qquad (-\infty < \eta < \lambda_s) \qquad (21c)$$

Because χ_s occurs at a known temperature Θ_s, eqs. 21b and 21c can be treated as the same equation with a temperature-dependent diffusivity function that employs eq. 4b above the solidus temperature.

Equations 20 and 21 are solved for the boundary conditions by use of the shooting method (Hornbeck, 1975, p. 205), whereby the coupled two-point boundary-value problems are treated as coupled initial-value problems. Successive quesses of Θ_{ci} and $d\Theta/d\eta$ evaluated at $\eta = 0$ are integrated to $\eta = +\infty$ and $-\infty$. If the guesses are correct, the temperature at $\eta = +\infty$ is Θ_{hi}, and that at $\eta = -\infty$ is Θ_{mi}. Newton-Raphson iteration is used to find the unknown temperature and temperature gradient at the dike wall.

The program EARLYN (Appendix 1) returns temperatures as $\Theta(\eta)$ and $\Theta(X,t)$.

Late-time solution. Using results from either the analytical or numerical early-time solution as an initial condition, temperatures are subsequently integrated using the Crank-Nicolson method (Hornbeck, 1975, p. 275), which is unconditionally stable for time steps of any size and retains second-order accuracy in the finite-difference approximation of the both the space and time derivatives. Denoting X-nodes and t-nodes with i subscripts and j superscripts, respectively, the finite-difference forms of each term in eqs. 6 are:

$$\kappa^{-1}\partial\Theta/\partial t \simeq [2/(\kappa_i^{j+1}+\kappa_i^j)]\cdot(\Theta_i^{j+1}-\Theta_i^j)/(\Delta t) \qquad (22a)$$

$$\partial^2\Theta/\partial X^2 \simeq \tfrac{1}{2}(\Theta_{i+1}^j-2\Theta_i^j+\Theta_{i-1}^j)/(\Delta X)^2 + \tfrac{1}{2}(\Theta_{i+1}^{j+1}-2\Theta_i^{j+1}+\Theta_{i-1}^{j+1})/(\Delta X)^2 \qquad (22b)$$

$$k^{-1}(\partial k/\partial X)(\partial\Theta/\partial X) \simeq [2/(k_i^{j+1}+k_i^j)]\cdot\tfrac{1}{2}(k_{i+1}^j+k_{i-1}^j)(\Theta_{i+1}^j-\Theta_{i-1}^j)/(2\Delta X)^2 \qquad (22c)$$

$$+ [2/(k_i^{j+1}+k_i^j)]\cdot\tfrac{1}{2}(k_{i+1}^{j+1}-k_{i-1}^{j+1})(\Theta_{i+1}^{j+1}-\Theta_{i-1}^{j+1})/(2\Delta X)^2$$

where all Θ^{j+1} are unknown temperatures at the t^{j+1} time. Equations 22 assure conservation of energy between each node, so that all boundary conditions, except those at $X = -T/2$ and ∞, are automatically satisfied. If there are n

$\underline{X}$-nodes, then there is a system of $\underline{n}$-2 equations of the form of eqs. 22, the remaining two nodes being subject to the remaining boundary conditions. Equation 22c is nonlinear because $\underline{k}^{j+1}$ is a function of τ^{j+1}. To linearize this term, $\underline{k}^{j+1}$ is initially set equal to $\underline{k}^{j}$; iteration is then used to recompute $\underline{k}^{j+1}$ until it has an arbitrarily small effect upon the calculated temperature Θ^{j+1}. When written in matrix form to solve for the $\underline{n}$ unknown temperatures, the coefficients form a tridiagonal matrix; this system is easily solved using the Thomas algorithm (von Rosenberg, 1975). A similar solution method was devised by Sanford (1982) for binary diffusion.

The program WHOLEN (Appendix 1) provides temperatures as $\Theta(\underline{X},\underline{t})$, as well as the maximum temperature $\Theta_{max}(\underline{X})$.

Use of Programs

The four programs make use of 26 subroutines and functions (Table 2; linking procedures are shown in Table 3). Before running the programs the user must construct a file containing up to 13 times when temperatures are to be saved and written to an output file. The number of times when temperatures are to be saved to file must be on the first line, with the times on the remaining lines; the first time must be zero. The user has a choice of using nondimensional times τ (eq. 12a) stored in a file named TAU.DAT or of dimensional times in units of seconds stored in a file named DTAU.DAT (see Table 4 for examples). The remaining input parameters are read interactively from the default input device (keyboard, which is assumed to open automatically upon invocation as unit 5); examples are shown in Table 4. Some data are written to the standard output device (screen, which is assumed to open automatically upon invocation as unit 6) during execution; a complete listing of input parameters and results is written to one of the files EARLYA.DAT, WHOLEA.DAT, EARLYN.DAT or WHOLEN.DAT upon completion of all calculations. Error conditions are written to the standard error output (screen, assumed to open automatically as unit 0). Data files are opened and closed with calls to subroutines FOPEN and FCLOSE; it is user´s responsibility to assure that opening and closing of files conform to the conventions documented in those subroutines (see Appendix).

Example

To illustrate the application of the programs and isolate the influence of the temperature dependence of thermal properties, consider the intrusion of a basaltic magma at an initial temperature of 1150°C to form a 2-m-thick dike in a sequence of basaltic lava flows at an ambient temperature of 50°C. If restricted to results of analytical solutions, one would be tempted to use eq. 18 ($\underline{k}_m/\underline{k}_h = \kappa_m/\kappa_h = 1$). Temperature as a function of distance and time for $\underline{k}_m/\underline{k}_h = \kappa_m/\kappa_h = 1$ is shown in Figure 2, where results are presented both in nondimensional form and in dimensional form for this particular case (see second column, Table 5, for input). Numerical solutions use the best fit for the temperature dependence of $\underline{k}$ and κ (Fig. 1a): $\underline{k}_m = \underline{k}_h = 0.689 + 522/(273.15+\Theta)$ and $\kappa_m = \kappa_h = 3.05\text{x}10^{-7} + 1.25\text{x}10^{-4}/(273.15+\Theta)$. If the host rock had 10% porosity and were saturated with water, then the best fit is: $\underline{k}_h = 0.250 + 944/(273.15+\Theta)$ and $\kappa_h = 1.80\text{x}10^{-6} + 2.50\text{x}10^{-4}/(273.15+\Theta)$. A magmatic heat of crystallization $(\rho\underline{h})_m = 900$ MJ/m^3 is used and contrasted with the case in which $(\rho\underline{h})_m = 0$ MJ/m^3. Temperatures are presented as a function of position and time for the temperature-dependent properties of "wet" basalt in Figure 3 (see third and fourth columns, Table 5, for input).

The analytical solution (eq. 18) indicates that $\Theta_{ci} = 600°C$; the numerical solution for "dry" basalt reveals that $\Theta_{ci} = 545°C$, 55°C less than the analytical solution. The numerical solution for host rocks with the properties of "wet" basalt reveals that $\Theta_{ci} = 525°C$, 75°C less than would be indicated by the analytical solution. If the heat of crystallization is released evenly in the temperature interval between 1150°C and 950°C, then the analytical solution (eqs. 13 and 14) indicates that $\Theta_{ci} = 690°C$; the numerical solution reveals that $\Theta_{ci} = 640°C$, 50°C less than the analytical solution. If the heat of crystallization is released and the host rocks have the properties of "wet" basalt, then $\Theta_{ci} = 618°C$, 72°C less than would be indicated by the analytical solution. The combined effect of treating water-saturation and the temperature dependence of thermal properties reduces estimates of maximum temperatures by almost as much as the latent heat of crystallization raises them. The difference among maximum temperatures in the host rocks is greatest at the dike contact (Fig. 4). Maximum temperatures attained in the wallrocks are everywhere less than would be estimated by any existing analytical method.

Conclusions

The theory of transient heat conduction remains the principal guide for analysis of cooling of most, but by no means all, dikes. Analytical solutions are useful for estimating temperatures and cooling rates where accuracies not greater than about 75°C are required (Delaney, 1986), but numerical solutions are important if temperature estimates in the vicinity of the dike contact are reguired to be more accurate. The reason for this is two-fold. First, analytically obtained estimates of maximum temperatures attained in the host rocks can only approximately account for the heat of crystallization associated with cooling, which can raise estimates of temperatures at the dike contact by 100°C or more. Second, analytical solutions do not address the temperature-dependence of thermal conductivity and diffusivity, which can lower estimates of temperatures at the dike contact by 50-70°C or more. These limitations are overcome by numerical methods of solution. However, the accurate prediction of rock properties _in situ_ and of the relation between cooling rate and release of magmatic heat of crystallization limit the accuracy that even numerical solutions can acheive.

Acknowledgements. I thank Stephen Huebner and Steven Ingebritsen of the U.S. Geological Survey for suggestions which improved the manuscript.

REFERENCES

Buchan, K.L., Schwarz, E.J., Symons, D.T.A., and Stupavsky, M., 1980, Remanent magnetization in the contact zone between Columbia Plateau flows and feeder dikes: evidence for groundwater layer at time of intrusion: Jour. Geophys. Res., v. 85, p. 1888-1898.

Buchan, K.L., and Schwarz, E.J., 1986, Calibration and applications of the maximum temperature profile across dike contact zones using remnant magnetization, _in_ H.C. Halls and W.F. Fahrig, eds., Mafic dyke swarms: Geol. Assoc. Canada Sp. Paper, _in press_.

Carslaw, H.S., and Jaeger, J.C., 1959, Conduction of Heat in Solids: 2nd. ed., Clarendon Press, Oxford, 510 p.

Delaney, P.T., 1986, Heat-transfer during emplacement and cooling of mafic dykes, _in_ H.C. Halls and W.F. Fahrig, eds., Mafic dyke swarms: Geol. Assoc. Canada Sp. Paper, _in press_.

Giberti, G., Moreno, S., and Sartoris, G., 1984, Evaluation of approximations in modelling the cooling of magmatic bodies: Jour. Volcanology & Geothermal Res., v. 20, p. 297-310.

Hanley, E.J., Dewitt, D.P., and Roy, R.F., 1978, The thermal diffusivity of eight well-characterized rocks for the temperature range 300-1000 K: Engineering Geol., v. 12, p. 31-47.

Hornbeck, R.W., 1975, Numerical Methods: Quantum, New York, 310 p.

Jaeger, J.C., 1957, Temperature in the neighborhood of a cooling intrusive sheet: Am. Jour. Sci., v. 255, p. 306-318.

Jaeger, J.C., 1964, Thermal effects of intrusions: Rev. Geophys., v. 2, p. 433-466.

Jaeger, J.C., 1968, Cooling and solidification of igneous rocks, _in_ H.H. Hess and A. Poldervaardt, eds., Basalts: Poldervaardt Treatise on Rocks of Basaltic Composition, 2nd. vol.: John Wiley & Sons, New York, p. 503-536.

Lovering, T.S., 1936, Heat conduction in dissimilar rocks and the use of thermal models: Bull. Geol. Soc. Am., v. 47, p. 87-100.

McClelland-Brown, E.A., 1981, Paleomagnetic estimates of temeratures reached in contact metamorphism: Geol., v. 9, p. 112-116.

Ramey, H.J., Jr., Brigham, W.E., Chen, H.K., Atkinson, P.G., and Arihara, N., 1974, Thermodynamic and hydrodynamic properties of hydrothermal systems: Stanford Geothermal Program SGP-TR-6, Stanford, Calif., 75 p.

Rosenberg, D.U., von, 1975, Methods for the Numerical Solution of Partial Differential Equattions: Elsevier, New York, 128 p.

Sanford, R.F., 1982, Three FORTRAN programs for finite-difference solutions to binary diffusion in one and two phases with composition- and time-dependent diffusion coefficients: Computers & Geosciences, v. 8, pp. 235-263.

Sass, J.H., Lachenbruch, A.H., and Munroe, R.J., 1971, Thermal conductivity of rocks from measurements on fragments and its application to heat-flow measurements: Jour. Geophys. Res., v. 76, p. 3391-3401.

Schwarz, E.J., 1977, Depth of burial from remanent magnetization: the Sudbury irruptive at the time of diabase intrusion (1250 Ma): Can. Jour. Earth Sci., v. 14, p. 82-88.

Thompson, R.J., 1970, Improving roundoff in Runge-Kutta computations with Gill['s] method: Comm. Acm., v. 13, p. 739-741.

Touloukian, Y.S., Judd, W.R., and Roy, R.F., eds., 1981, Physical Properties of Rocks and Minerals: McGraw-Hill, New York, 548 p.

```
c-345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      program earlya
c
c conductive cooling of an instantaneously emplaced dike; early-time,
c analytic solution for the cooling history.  magma and host rocks can
c have different but constant thermal conductivity and diffusivity; the
c magma can release a heat of crystallization in the interval between
c its intrusion temperature and some specified lower temperature.
c solution applies until cooling commences at dyke center; before this
c time temperatures can be expressed in terms of a single variable that
c combines distance and time.  temperature data is calculated at ii
c times, the first of which is 0.  these times must exist in the file
c TAU.DAT or DTAU.DAT before running this program.  the former are
c nondimensional times; the latter are dimensional in seconds.
c
c all variables are in units of m-k-s & degrees centigrade.
c coordinates are x and tau, and eta = x/sqrt(4*xkaph*tau).
c x = 0 at the contact; x = -dike_1/2_thickness at the dike center.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      parameter (ii=13, jj=601)
      dimension tau(ii), x(jj), t(jj,ii), eta(jj), tt(jj)
      external start, solve2, tmpae1, tmpae2, tmpae3, finish
      character ans*1
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
      common /propa/ rkd, rkm, sqkapd, sqkapm, p
c
c (1) get dike half-thickness (dthck2), initial temperatures of dike
c rock and host rock (tdi, thi), and conductivity and diffusivity of
c dike rock (xkd, xkapd) and host rocks (xkh, xkaph).  if a seperate
c magma region is specified (ians = 3), get latent heat of
c crystallization per unit volume (xl), solidus temperature (ts),
c as well as conductivity and diffusivity of magma (xkm, xkapm).
c provide initial quesses of initial temperature at dike contact
c (tci) and transformed position of solidification surface (xlam).
c latent heat is released evenly in the interval from tdi to ts.
c
      write(6,9)
 9    format(/' Include heat of crystallization ? [y/n] ?? '$)
      read(5,'(1a)') ans
      write(6,10)
 10   format(/'  dike half-thickness [m]:                       = '$)
      read(5,*) dthck2
      write(6,11)
 11   format(/'    initial temp. [deg.c]: dike rock & host rock = '$)
      read(5,*) tdi, thi
      write(6,12)
 12   format(/' conductivity [w/m deg.c]: dike rock & host rock = '$)
      read(5,*) xkd, xkh
      write(6,13)
 13    format(/'      diffusivity [m*m/s]: dike rock & host rock = '$)
       read(5,*) xkapd, xkaph
       xkm = xkd
       xkapm = xkapd
       if (ans .eq. 'n') then
        xl = 0.0e0
        xlam = -4.0e0
        ts = tdi
       else
        write(6,16)
 16     format(/'     latent heat [j/m**3]:  magma                = '$)
        read(5,*) xl
        write(6,17)
 17     format(/'  solidus temp. [deg. c]:  magma                 = '$)
        read(5,*) ts
        write(6,18)
 18     format(/'  quess: contact temp. [deg.c]:  thi < tci < ts = '$)
        read(5,*) tci
        write(6,19)
 19     format(/'          solidification surface: -2 < lambda < 0 = '$)
        read(5,*) xlam
       endif
c
c (2) prepare distance, time & eta vectors, and associated indeces.
c
       call start(x,tau,t,eta,tt,ii,jj)
c
c (3) calculate tci if there is no heat of crystallization,
c and tci and xlam by newton-raphson iteration if there is.
c
       if (xl .eq. 0.0e0) then
        rkd = xkd/xkh
        sqkapd = sqrt(xkapd/xkaph)
        tci = thi + (tdi-thi)*rkd/(rkd+sqkapd)
       else
        rkd = xkd/xkh
        rkm = xkm/xkh
        sqkapd = sqrt(xkapd/xkaph)
        if ((abs((tdi-ts)/(tdi-thi)) .gt. 1.0e-4) .and.
     +     (abs((xl/(tdi-ts))/(xkm/xkapm)) .gt. 1.0e-4)) then
         xkapmp = xkm/((xkm/xkapm)+xl/(tdi-ts))
         sqkapm = sqrt(xkapmp/xkaph)
        else
         sqkapm = sqrt(xkapm/xkaph)
        endif
        itmax = 25
        err = 1.0e-4*(tdi-thi)
        call solve2(tmpae3,xlam,tci,err,itmax)
        if (itmax .ge. 25) stop
       endif
c
       write(6,90) tci
 90    format(/'         contact temperature = ',1pg11.4,' deg. c')
       if (xl .ne. 0.0e0) write(6,91) xlam
 91    format(' solidification surface, xlam = 'lpg11.4/)
c
c (4) find latest possible valid early-time solution.
c
       do 30 i = ie-1, imax
        etax = -dthck2/sqrt(4.0e0*xkaph*tau(i))
        if (xl .eq. 0.0e0) then
         call tmpae1(etax,t(1,i))
        else
         call tmpae2(etax,t(1,i))
        endif
        if ((t(1,i)-thi)/(tdi-thi) .lt. 0.9995e0) goto 31
 30    continue
 31    ie = i - 1
c
c (5) calculate temperatures as a function of eta.
c
       do 41 j = 1, jmax
        if (xl .eq. 0.0e0) then
         call tmpae1(eta(j),tt(j))
        else
         call tmpae2(eta(j),tt(j))
        endif
 41    continue
c
c (6) calculate temperatures in terms of x and tau
c
       do 42 j = 1, jc-1
 42     t(j,1) = tdi
       t(jc,1) = tci
       do 43 j = jc+1, jmax
 43     t(j,1) = thi
       do 44 i = 2, ie
        sqrtt = sqrt(4.0e0*xkaph*tau(i))
        do 45 j = 1, jmax
         etax = x(j)/sqrtt
         if (xl .eq. 0.0e0) then
          call tmpae1(etax,t(j,i))
         else
          call tmpae2(etax,t(j,i))
         endif
 45     continue
 44    continue
c
       write(6,92)
 92    format(/' analytical early-time solution is finished')
c
c (7) finish up: write data to file "earlya.dat"; "tm" is a dummy.
c
       call finish(x,tau,t,eta,tt,tm,ii,jj,'ea')
c
       stop
       end
```

```
4-345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      program earlyn
c
c conductive cooling of an instantaneously emplaced dike.  early-time
c numerical solution for the situation where magma and host rocks have
c different, temperature-dependent thermal conductivities and
c diffusivities, as defined by the functions condd, condmd, diffd,
c condh, condmh, & diffh.  cond_ is the  conductivity function for dike
c or host rocks; condm_ is the function for conductivity modulus,
c (1/k)(dk/dt); diff_ is the function for diffusivity.  the magma can
c release a heat of crystallization in the interval between its
c intrusion temperature and some specified lower temperature.  solution
c applies before cooling commences at the dyke center; before this time
c temperatures can be expressed in terms of a single varieable that
c combines distance and time.  thus, the partial differential
c equations are converted to ordinary differential equations, and
c the transformed boundary conditions constitute a boundary-value
c problem.  the method of solution, "the shooting method," treats the
c problem as an initial-value problem.  the equations are then
c integrated to the "far" boundary.  successive quesses (found by
c newton-raphson integration) are used to find the initial values that,
c when integrated, match the desired far boundary values.  although the
c odes are second-order equations, they are converted to pairs of
c first-order equations for the numerical integration by a modified
c runge-kutta method.  temperature data is calculated at ii times, the
c first of which is 0.  these times must exist in the file TAU.DAT or
c DTAU.DAT before running this program.  the former are nondimensional
c times; the latter are dimensional in seconds.
c
c all variables are in units of m-k-s & degrees centigrade
c coordinates are x (distance), tau (time), eta = x/sqrt(4*kap_h*tau).
c x = 0 at dike contact; x = -dike_1/2_thickness at the dike center.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      real*8 xx, dxx, yy
      character*1 ans
      parameter (ii=13, jj=601)
      dimension tau(ii), x(jj), t(jj,ii), eta(jj), tt(jj), yy(2)
      external condd, condh, diffd, diffh, start, tmpnel, tmpne2,
     +      tmpne3, oderk2, solve2, finish
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (1) get dike half-thickness (dthck2), initial temperatures of dike
c rock and host rock (tdi, thi), and conductivity and diffusivity
c functions of dike rock (akd, bkd, akapd, bkapd) and host rocks
c (akh, bkh, akaph, bkaph).  get the latent heat of crystallization
c per unit volume (xl), and solidus temperature (ts).  get initial
c quesses of initial temperature at dike contact (tci) and trans-
c formed position of solidification surface (xlam).  latent heat is
c released evenly in the temperature interval from tdi to ts.
```

```
c
      write(6,9)
 9    format(/' Include heat of crystallization ? [y/n] ?? '$)
      read(5,'(1a)') ans
      write(6,10)
 10   format(/'   dike half-thickness [m]:                         = '$)
      read(5,*) dthck2
      write(6,11)
 11   format(/'     initial temp. [deg.c]: dike rock & host rock = '$)
      read(5,*) tdi, thi
      write(6,12)
 12   format(/'  conductivity: [w/m deg.c] = a + b/(273.15+t[deg.c])'
     +       //' dike rock -- a, b = '$)
      read(5,*) akd, bkd
      write(6,13)
 13   format(' host rock -- a, b = '$)
      read(5,*) akh, bkh
      write(6,14)
 14   format(/'   diffusivity: [m**2/s] = a + b/(273.15+t[deg.c])'
     +       //' dike rock -- a, b = '$)
      read(5,*) akapd, bkapd
      write(6,13)
      read(5,*) akaph, bkaph
      if (ans .le. 'n') then
       ts = tdi
       xlam = -4.0e0
      else
       write(6,16)
 16    format(/'      latent heat [j/m**3]:  magma                   = '$)
       read(5,*) xl
       write(6,17)
 17    format(/'    solidus temp. [deg. c]:  magma                   = '$)
       read(5,*) ts
      endif
      write(6,18)
 18   format(/' quess: temp. at contact [deg.c]:  thi < tci < ts = '$)
      read(5,*) tci
      write(6,19)
 19   format(/'              error tolerance [generally <1.0e-3] = '$)
      read(5,*) err
c
c (2) compute special values for the conductivity and diffusivity.
c
      xkd = condd(tdi)
      xkapd = diffd(tdi)
      xkh = condh(thi)
      xkaph = diffh(thi)
c
c (3) prepare distance, time & eta vectors, and associated indeces.
c
      call start(x,tau,t,eta,tt,ii,jj)
c
c (4) initial quess of tci is provided before calling this subroutine;
c qch is a quess of the heat flux into the host-rock half-space.  these
```

```
c two parameters are sufficient to integrate the heat to x = +infinity.
c to perform the integration to x = -infinity, we set qcd = qch*(xkd/xkh)
c where xkd and xkh are evaluated at the contact temperature.  the
c integrater, looks to see when temperatures have exceeded ts, and
c positions xlam in that fashion.
c
      itmax = 25
      err = err*(tdi-thi)
      qch = -2.0e0*(tci-thi)/sqrt(3.142e0)
      call solve2(tmpnel,tci,qch,err,itmax)
      err = err/(tdi-thi)
      if (itmax .ge. 25) stop
c
c (5) integrate equations stopping to tabulate tempertures.
c iflag = 0:  integrate equations for magma.
c iflag = 1:  integrate equations for host rock
c
      dxx = deta/(1.0e0*nk)
      rkci = condd(tci)/condh(tci)
      tt(kc) = tci
c
      iflag = 0
      iiflag = 0
      xx = 0.0e0
      yy(1) = tci
      yy(2) = -qch/rkci
      istart = 1
      do 31 j = kc-1, 1, -1
       do 32 k = 1, nk
        call oderk2(dxx,xx,yy,tmpne2,istart)
        if ((sngl(yy(1)) .gt. ts) .and. (iiflag .eq. 0)) then
         xlam = -(xx+(xx-dxx))/2.0e0
         iiflag = 1
        endif
 32    continue
 31    tt(j) = yy(1)
c
      iflag = 1
      xx = 0.0d0
      yy(1) = tci
      yy(2) = qch
      istart = 1
      do 30 j = kc+1, jmax
       do 33 k = 1, nk
        call oderk2(dxx,xx,yy,tmpne2,istart)
 33    continue
 30    tt(j) = yy(1)
c
      write(6,90) tci
 90   format(/'  analytical early-time solution is finished'
     +      //'          contact temperature = ',1pg11.4,' deg. c')
      if (xl .ne. 0.0e0) write(6,91) xlam
 91   format(/'  solidification surface, eta = ',1pg11.4,/)
c
```

```
c (5) convert from tt(eta) to t(x,tau)
c
      call tmpne3(x,tau,t,eta,tt,ii,jj)
c
c (6) write data to file "earlyn.dat";  "tm" is a dummy.
c
      call finish(x,tau,t,eta,tt,tm,ii,jj,´en´)
c
      stop
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      program wholea
c
c conductive cooling of an instantaneously emplaced dike; whole-time,
c analytic solution for the cooling history.  magma and host rocks may
c have different but constant thermal conductivity and diffusivity.
c no latent heat of crystallization can be included in this solution,
c unless by the method of an equivalent intrusion temperature.
c temperature data is calculated at ii times, the first of which is 0.
c these times must exist in the file TAU.DAT or DTAU.DAT before running
c this program.  the former are nondimensional times; the latter are
c dimensional in seconds.
c
c all variables are in units of m-k-s & degrees centigrade.
c coordinates are x and tau.
c x = 0 at the contact; x = -dike_1/2_thickness at the dike center.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      parameter (ii=13, jj=601)
      dimension tau(ii), x(jj), t(jj,ii), eta(jj), tt(jj), tm(jj)
      external start, tmpael, tmpal1, tmpal2, tmpal3, finish
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
      common /propa/ rkd, rkm, sqkapd, sqkapm, p
c
c (1) get dike half-thickness (dthck2), initial temperatures of dike
c rock and host rock (tdi, thi), and conductivity and diffusivity
c of dike rock (xkd, xkapd) and host rocks (xkh, xkaph).
c
      write(6,10)
 10   format(/' dike half-thickness [m]:                        = '$)
      read(5,*) dthck2
      write(6,11)
 11   format(/'   initial temp. [deg.c]: dike rock & host rock = '$)
      read(5,*) tdi, thi
      write(6,12)
 12   format(/' conductivity [w/m deg.c]: dike rock & host rock = '$)
      read(5,*) xkd, xkh
      write(6,13)
 13   format(/'     diffusivity [m*m/s]: dike rock & host rock = '$)
      read(5,*) xkapd, xkaph
c
c (2) prepare distance & time vectors, and associated indeces.
c
      call start(x,tau,t,eta,tt,ii,jj)
c
c (3) calculate initial temperatures.
c
      sqkapd = sqrt(xkapd/xkaph)
      rkd = xkd/xkh
      p = (sqkapd-rkd)/(sqkapd+rkd)
      tci = thi + (tdi-thi)*rkd/(rkd+sqkapd)
c
      do 21 j = 1, jc-1
 21    t(j,1) = tdi
      t(jc,1) = tci
      do 22 j = jc+1, jmax
 22    t(j,1) = thi
c
c (4) calculate temperatures.
c
      if ((abs(p).lt.1.0e-5) .and. (abs(sqkapd-1.0e0).lt.1.0e-5)) then
       do 40 i = 2, imax
        call tmpall(tau(i),x(1),t(1,i))
        do 45 j = 2, jmax
         if ((t(j-1,i)-thi)/(tdi-thi) .gt. 1.0e-4) then
          call tmpall(tau(i),x(j),t(j,i))
         else
          t(j,i) = thi
         endif
 45     continue
        write(6,90) tau(i), t(1,i), t(jc,i)
 40    continue
      else
       do 41 i = 2, ie
        sqrtt = sqrt(4.0e0*xkaph*tau(i))
        do 46 j = 1, jmax
         etal = x(j)/sqrtt
 46      call tmpael(etal,t(j,i))
        write(6,90) tau(i), t(1,i), t(jc,i)
 41    continue
       do 42 i = ie+1, imax
        call tmpal2(tau(i),x(1),t(1,i))
        do 47 j = 2, jmax
         if ((t(j-1,i)-thi)/(tdi-thi) .gt. 1.0e-4) then
          call tmpal2(tau(i),x(j),t(j,i))
         else
          t(j,i) = thi
         endif
 47     continue
        write(6,90) tau(i), t(1,i), t(jc,i)
 42    continue
      endif
c
 90   format(/' time = ',1pg11.4,' sec.:  ',
     +        ' temperature: dike center = ',1pg11.4,' deg. c',
     +       /'                            ',
     +        '                 contact = ',1pg11.4,' deg. c')
c
c (5) find maximum temperature attained at each position.
c
      do 60 j = 1, jc-1
 60    tm(j) = tdi
      tm(jc) = tci
c
```

```
      if ((abs(p).lt.1.0e-5) .and. (abs(sqkapd-1.0e0).lt.1.0e-5)) then
       do 61 j = jc+1, jmax
        xlog = alog(x(j)/(2.0e0*dthck2+x(j)))
        taum = -(2.0e0*dthck2+x(j))*dthck2/(2.0e0*xkaph*xlog)
 61     call tmpall(taum,x(j),tm(j))
      else
       taum = tau(ie+1)
       do 62 j = jc+1, jmax
 62     call tmpal3(x(j),tm(j),taum)
      endif
c
      write(6,91)
 91   format(/' analytical late-time solution is finished')
c
c (4) finish up: write data to file "wholea.dat".
c
      call finish(x,tau,t,eta,tt,tm,ii,jj,'wa')
c
 990  stop
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      program wholen
c
c conductive cooling of an instantaneously emplaced dike.  whole-time
c numerical solution for the situation where magma and host rocks have
c different, temperature-dependent thermal conductivities and
c diffusivities, as defined by the functions condd, condmd, diffd,
c condh, condmh, & diffh.  cond_ is the  conductivity function for dike
c or host rocks; condm_ is the function for conductivity modulus,
c (1/k)(dk/dt); diff_ is the function for diffusivity.  the magma can
c release a heat of crystallization in the interval between its
c intrusion temperature and some specified lower temperature.  the
c initial temperature distribution is given by an early-time solution
c computed from either of the programs EARLYA or EARLYN, which must,
c therefore be run first.  the method of solution is crank-nicolson
c with iterative improvement.  temperature data is calculated at ii
c times, the first of which is 0.  these times must exist in the file
c TAUD.DAT before running this program; these are dimensional in seconds.
c
c all variables are in units of m-k-s & degrees centigrade.
c coordinates are x and tau.
c x = 0 at the contact; x = -dike_1/2_thickness at the dike center.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      parameter (ii=13, jj=601)
      dimension tau(ii), x(jj), t(jj,ii), tm(jj)
      character meth*2, str*50
      external tmpnl, finish
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (1) get times when temperatures will be stored from file TAUD.DAT
c
      call fopen(2,'TAUD.DAT','old','exclusive',ios)
      if (ios .ne. 0) then
       write(0,995)
 995   format('error trying to open file TAUD.DAT')
       stop
      endif
      read(2,'(1x,i4)') imax
      read(2,'(1x,1pg14.5)') (tau(i), i=1,imax)
      call fclose(2,'keep',ios)
c
c (2) get data on temperatures, thermal properties, etc., from
c file EARLYN.DAT or EARLYA.DAT.  (this means program earlyn or
c earlya must be run before wholen.)
c
      write(6,1)
 1    format(/' Is input file EARLYA.DAT or EARLYN.DAT ?? [a/n] ?? '$)
      read(5,'(1a)') meth(2:2)
      write(6,3)
 3     format(/' number of iterations between output-time steps'
    +        /'                                   (generally >5) = '$)
       read(5,*) itnum
       write(6,'(/)')
c
       if ((meth(2:2) .eq. 'a') .or. (meth(2:2) .eq. 'A')) then
        call fopen(1,'EARLYA.DAT','old','exclusive',ios)
       else
        call fopen(1,'EARLYN.DAT','old','exclusive',ios)
       endif
       if (ios .ne. 0) then
        write(0,996)
 996    format('error trying to open EARLY_.DAT')
        stop
       endif
c
       read(1,10) ie, jmax, tdi, thi, tci, dthck2, dx, jc, meth(2:2)
       write(6,10) ie, jmax, tdi, thi, tci, dthck2, dx, jc, meth(2:2)
 10    format(1x,i4/1x,i4//1x,1pe10.3/1x,1pe10.3/1x,1pe10.3//
    +        1x,1pe10.3/1x,1pe10.3/1x,i4//1x,a1/)
       read(1,20) xl, ts, xlam
       write(6,20) xl, ts, xlam
 20    format(1x,1pe10.3/1x,1pe10.3/1x,1pe10.3/)
       read(1,30) xkd, xkapd, xkh, xkaph
       write(6,30) xkd, xkapd, xkh, xkaph
 30    format(1x,1pe10.3/1x,1pe10.3/1x,1pe10.3/1x,1pe10.3/)
       if (meth(2:2) .eq. 'a') then
        akd = xkd
        bkd = 0.0e0
        akapd = xkapd
        bkapd = 0.0e0
        akh = xkh
        bkh = 0.0e0
        akaph = xkaph
        bkaph = 0.0e0
       else
        read(1,34) akd, bkd, akapd, bkapd, akh, bkh, akaph, bkaph
        write(6,34) akd, bkd, akapd, bkapd, akh, bkh, akaph, bkaph
        read(1,36) err
        write(6,36) err
 34     format(1x,4(1pe10.3,1x)//1x,4(1pe10.3,1x)/)
 36     format(1x,1pe10.3/)
       endif
c
       read(1,'(1x,/1x,/1x,/1x,/)')
c
       str(1:20) = '(1x,18x,  (f8.2,1x))'
       write(str(9:10),'(i2)') ie+1
       do 70 j = 1, jmax
 70     read(1,str(1:20)) x(j), (t(j,i),i=1,ie)
c
       call fclose(1,'keep',ios)
c
c (3) temperatures: late-time solution, t[x,tau].
```

```
c
      call tmpnl(tau,x,t,tm,ii,jj)
c
c (4) finish up: write data to file "wholen.dat";  "eta" & "tt" are dummies.
c
      call finish(x,tau,t,eta,tt,tm,ii,jj,'wn')
c
 990  stop
      end
```

```
c2345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpael(eta,t)
c
c temperature as a function of x/sqrt(4*tau).  analytic and early-time
c history for cooling of a hot half-space brought instantaneously
c into contact with a cool half-space.  thermal properties of the
c two bodies can differ, but are constants.  the problem is
c a modification of the newmann problem discussed in chapter 11 of
c carslaw and jaeger, 1959, p. 288.  sqkapd and tci must be set before
c calling this subroutine.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external erfunc
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
      common /propa/ rkd, rkm, sqkapd, sqkapm, p
c
      if (eta .lt. 0.0e0) then
       erfcx = 1.0e0 - erfunc(eta/sqkapd)
       t = tdi - (tdi-tci)*(2.0e0-erfcx)
      elseif (eta .eq. 0.0e0) then
       t = tci
      else
       erfcx = 1.0e0 - erfunc(eta)
       t = thi + (tci-thi)*erfcx
      endif
c
      return
      end
```

```
c-345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpae2(eta,t)
c
c temperature as a function of x/sqrt(4*tau).  analytic and early-time
c history for cooling of a hot half-space brought instantaneously
c into contact with a cool half-space.  the hot body consists of a cool
c part and a hot part that is releasing additional heat due to
c crystallization.  thermal properties in these three regions can
c differ, but are constant within each.  the problem is a modification
c of the newmann problem discussed in chapter 11 of carslaw and jaeger,
c 1959, p. 288.  tci, ts, xlam, sqkapm, and sqkapd must be set prior
c to calling this subroutine.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external erfunc
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
      common /propa/ rkd, rkm, sqkapd, sqkapm, p
c
      if (eta .lt. xlam) then
       erfcx = 1.0e0 - erfunc(eta/sqkapm)
       erfcl = 1.0e0 - erfunc(xlam/sqkapm)
       t = tdi - (tdi-ts)*(2.0e0-erfcx)/(2.0e0-erfcl)
      elseif (eta .eq. xlam) then
       t = ts
      elseif (eta .lt. 0.0e0) then
       erfx = erfunc(eta/sqkapd)
       erfl = erfunc(xlam/sqkapd)
       t = tci + (ts-tci)*erfx/erfl
      elseif (eta .eq. 0.0e0) then
       t = tci
      else
       erfcx = 1.0e0 - erfunc(eta)
       t = thi + (tci-thi)*erfcx
      endif
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpae3(xl,x2,yl,y2)
c
c interface equations for the generalized newmann problem of cooling
c with latent heat xl released over the temperature interval between
c tdi to ts.  these equations are derived by requiring that the heat
c lost from the zone of crystallizing magma be equal to that entering
c the zone of solidified magma (for yl), and that lost from the zone
c of solidified magma be equal to that entering the host rock (for y2).
c the first equation has a known temperature (ts) and an unknown
c position (xl); the second has a known position (0) and an unknown
c temperature (x2).  called by solve2.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external erfunc
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
      common /propa/ rkd, rkm, sqkapd, sqkapm, p
c
      xld = xl/sqkapd
      erfld = erfunc(xld)
      tci = thi + rkd*(ts-thi)/(rkd-sqkapd*erfld)
      xlm = xl/sqkapm
      erfclm = 1.0e0 - erfunc(xlm)
      yl = rkm*(tdi-ts)*exp(-xlm*xlm)*(sqkapd*erfld) +
     +     rkd*(ts-x2)*exp(-xld*xld)*(sqkapm*(2.0e0-erfclm))
      y2 = tci - x2
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpall(tau,x,t)
c
c temperature as a function of time and position.  analytic, whole
c time history for cooling of a heated slab embedded in an infinite
c body.  slab and body have identical, constant thermal properties.
c carslaw and jaeger, 1959, p. 54, has an equivalent solution for a
c coordinate system with an origin at the slab center.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external erfunc
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      sqrtt = sqrt(4.0e0*xkaph*tau)
      x1 = (2.0e0*dthck2+x)/sqrtt
      x2 = x/sqrtt
      erfx1 = erfunc(x1)
      erfx2 = erfunc(x2)
      t = thi + (tdi-thi)*0.5e0*(erfx1-erfx2)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpal2(tau,x,t)
c
c temperature as a function of time and position.  analytic, whole-time
c history for cooling of a heated slab embedded in an infinite body.
c although slab and body can have different thermal properties, they
c are constants and not functions of temperature.  solution given by
c lovering, 1936, geol. soc. am. bull. 47:87-100.  sqkapd and p must be
c set prior to calling this subroutine.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external erfunc
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
      common /propa/ rkd, rkm, sqkapd, sqkapm, p
c
      if (x/dthck2 .lt. 0.0e0) then
       sqrm = sqrt(4.0e0*xkapd*tau)
       x1 = (2.0e0*dthck2+x)/sqrm
       x2 = (2.0e0*dthck2-x)/sqrm
       t = erfunc(x1) - p*erfunc(x2)
       do 20 l = 2, 50
        x1 = (2.0e0*l*dthck2+x)/sqrm
        x2 = (2.0e0*l*dthck2-x)/sqrm
        tt = (-p)**(l-1)*(erfunc(x1)-p*erfunc(x2))
        t = t + tt
        if (abs(tt) .lt. 1.0e-6) goto 30
 20    continue
 30    x1 = x/sqrm
       t = thi + (tdi-thi)*0.5e0*(1.0e0+p)*(t-erfunc(x1))
      elseif (x/dthck2 .eq. 0.0e0) then
       sqrh = sqrt(4.0e0*xkaph*tau)
       x1 = (2.0e0*dthck2/sqkapd)/sqrh
       t = erfunc(x1)
       do 22 l = 2, 50
        x1 = (2.0e0*l*dthck2/sqkapd)/sqrh
        tt = (-p)**(l-1)*erfunc(x1)
        t = t + tt
        if (abs(tt) .lt. 1.0e-6) goto 32
 22    continue
 32    t = thi + (tdi-thi)*0.5e0*(1.0e0-p*p)*t
      else
       sqrh = sqrt(4.0e0*xkaph*tau)
       x1 = ((2.0e0*dthck2/sqkapd)+x)/sqrh
       t = erfunc(x1)
       do 21 l = 2, 50
        x1 = ((2.0e0*l*dthck2/sqkapd)+x)/sqrh
        tt = (-p)**(l-1)*erfunc(x1)
        t = t + tt
        if (abs(tt) .lt. 1.0e-6) goto 31
 21    continue
 31      xl = x/sqrh
         t = thi +  (tdi-thi)*0.5e0*(1.0e0-p)*((1.0e0+p)*t-erfunc(xl))
        endif
c
        return
        end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpal3(x,tm,taum)
c
c maximum temperature as a function of position.  analytic, whole
c history for cooling of a heated slab embedded in an infinite body.
c although slab and body can have different thermal properties, they
c are constants and not functions of temperature.  solution given by
c lovering, 1936, geol. soc. am. bull. 47:87-100.  sqkapd and p must be
c set prior to calling this subroutine.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external tmpal2
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (1) search to find time when dtemp/dtime = 0, for
c a given x, then calculate temperature at that time.
c
      chart = dthck2*dthck2/xkaph
      do 10 j = 1, 25
       taum0 = 0.99e0*taum
       tauml = 1.01e0*taum
       dtaum = tauml - taum0
       dtaum0 = taum - taum0
       dtauml = tauml - taum
       call tmpal2(taum,x,tm)
       call tmpal2(taum0,x,tm0)
       call tmpal2(tauml,x,tml)
       f = (tml-tm0)/dtaum
       f0 = (tm-tm0)/dtaum0
       fl = (tml-tm)/dtauml
       fp = (fl-f0)/((dtaum0+dtauml)/2.0e0)
       er = -f/fp
       taum = taum + er
       if (abs(er/chart) .lt. 1.0e-3) then
        call tmpal2(taum,x,tm)
        return
       endif
 10   continue
c
      if (j .ge. 25) then
       write(0,25)
 25    format(/' tmpal4: loop not converging!'/)
       stop
      endif
c
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpnl(tau,x,t,tm,ii,jj)
c
c temperature as a function of time and position.  numerical & late-
c time history for cooling of a heated slab embedded in an infinite
c body.  slab and body can have different temperature-dependent
c thermal properties.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      parameter (jjj=600)
      dimension tau(ii), x(jj), t(jj,ii), tt1(jjj), tt2(jjj),
     +     xk1(jjj), xk2(jjj), xkap1(jjj), xkap2(jjj), tm(jj)
      external tmpnl1, tmpnl2
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (1) initialize tt2(j) to t(j,ie), xk2(j) and rhc2(j).  tt2(j) is the
c best guess of the temperature of the current time; t(j,i) is the
c temperature at the last time, tau(i); xk2(j) is the thermal
c conductivity at the best quess of the temperature of the current
c time, as is heat capacity rhc2(j).  tt1, xk1 & rhc1 are temperature,
c conductivity & heat capacity at last successful time step.
c
      jn = jmax - 1
      do 1 j = 1, jc-1
       tt2(j) = t(j,ie)
 1     tm(j) = tdi
      tt2(jc) = t(jc,ie)
      tm(jc) = tci
      do 2 j = jc+1, jn
       tt2(j) = t(j,ie)
 2     tm(j) = thi
      tm(jmax) = thi
c
      call tmpnl1(tt2,xk2,xkap2,jjj)
c
c (2) step through successive temperatures.  this gets a bit confusing.
c outside loop increments to the next time when temperatures will be
c saved in the matrix t(j,i), tau(i).  the next loop increments itnum
c times from tau(i-1) to tau(i).  the inside loop is only used if
c thermal properties are nonconstant, and is the iterative improvement.
c
c primary loop point; loops are for i = ie, ie+1, ie+2, ... imax.
c define time step for 2nd loop, and normalize it by dyke_thick**2.
c
      do 10 i = ie + 1, imax
       dxtau = dx*dx/((tau(i)-tau(i-1))/itnum)
c
c secondary loop point; itnum steps between successive values of i.
c at top of loop, a successful time step has just been completed, so
c that tt2, rhc2 & xk2 are assigned to tt1, rhc1 & xk1; temperatures
```

```
c at the next time are then computed.
c
       do 20 ii = 1, itnum
        do 29 j = 1, jn
         tt1(j) = tt2(j)
         xkap1(j) = xkap2(j)
 29      xk1(j) = xk2(j)
c
        call tmpnl2(dxtau,tt1,tt2,xk1,xk2,xkap1,xkap2,jjj)
c
c tertiary loop point; iterative improvement, if necessary.
c tt2 is used to calculate temperature-dependent thermal props.
c t(j,i) is used only to save storage, and compare previous guess
c with the newest guess.  if they are very nearly the same, then
c solution has converged and is ready to leave this loop and go
c to the next time step.
c
        if (bkd+bkh+bkapd+bkaph .ne. 0.0e0) then
         do 30 iii = 1, 10
          do 31 j = 1, jn
 31        t(j,i) = tt2(j)
          call tmpnl1(tt2,xk2,xkap2,jjj)
          call tmpnl2(dxtau,tt1,tt2,xk1,xk2,xkap1,xkap2,jjj)
          terr = abs(t(1,i)-tt2(1))
          do 32 j = 2, jn
 32        terr = amax1(terr,abs(t(j,i)-tt2(j)))
          if (terr/(tdi-thi) .lt. err) goto 22
 30      continue
 22      if (iii .ge. 10) then
          write(0,21)
 21       format(/' problem in tempnl: not converging')
          stop
         endif
        endif
c
c end of 3rd loop.
c
c at each time step, see if temperatures at each x-node have peaked
c out.  if so, this is the maximum temperature attained at that
c position in the host rock
c
       do 28 j = jc, jmax-1
 28     tm(j) = amax1(tm(j),tt2(j))
 20    continue
c
c end of 2nd loop.
c
       do 11 j = 1, jn
 11     t(j,i) = tt2(j)
       t(jmax,i) = thi
       write(6,12) tau(i), t(1,i), t(jc,i)
 12    format(/' time = ',1pg11.4,' sec.:  ',
    +         ' temperature: dike center = ',1pg11.4,' deg. c',
    +        /'                            ',
     +          '                         contact = ',1pg11.4,' deg. c')
 10   continue
c
c end of 1st loop.
c
c now find x-nodes where temperatures have not yet peaked out.
c
      do 9 j = jn, jc, -1
       if (tm(j) .gt. tt2(j)) goto 8
 9     tm(j) = thi
c
 8    write(6,90)
 90   format(/'   numerical late-time solution is finished')
      terr = abs((t(jn,imax)-thi))
      if (terr/(tdi-thi) .gt. err/10.0e0) write(0,91)
 91   format(/' beware!  temperatures near the node furthest from the'/
     +        ' dike are no longer equal to thi.  perhaps the input'/
     +        ' parameter ndt, in program earlya or earlyn,'
     +        ' should be increased.'/)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpnll(tt2,xk2,xkap2,jjj)
c
c initialize tt2, xk2 and rhc2.  these are vectors of temperature,
c conductivity and heat capacity to be used as the "current guess" at
c each time step.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      dimension tt2(jjj), xk2(jjj), xkap2(jjj)
      external condd, condh, diffd, diffh
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c heat capacity is conductivity/diffusivity, and is calculated at each
c x-node.  at contact it is averaged between dike and host-rock.
c
      do 1 j = 1, jc-1
       xk2(j) = condd(tt2(j))
 1     xkap2(j) = diffd(tt2(j))
c
      xk2(jc) = (condd(tt2(jc))+condh(tt2(jc)))/2.0e0
      xkap2(jc) = (diffd(tt2(jc))+diffh(tt2(jc)))/2.0e0
c
      do 2 j = jc+1, jn
       xk2(j) = condh(tt2(j))
 2     xkap2(j) = diffh(tt2(j))
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine tmpnl2(dxtau,tt1,tt2,xk1,xk2,xkap1,xkap2,jjj)
c
c finite difference equations for the cooling of a dyke (-dthck2 <= x
c <= 0) adjacent to host rocks (x >= 0), where both magma and host
c rocks have temperature-dependent thermal conductivity and heat
c capacity.  the difference equations use the crank-nicolson approach
c which is unconditionally stable and has second-order accuracy.
c this routine is called iteratively:  a succesion of guesses of the
c new temperature, t(j,i), is used to estimate the values of
c conductivity, xk2, and heat capacity, rhc, so that the sytem of
c equations can be solved.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      parameter (j4=600)
      dimension a(j4), b(j4), c(j4), yy(j4), tt1(jjj),
     +   tt2(jjj), xk1(jjj), xk2(jjj), xkap1(jjj), xkap2(jjj)
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (2) prepare coefficients for first eqn., x = -dthck2, the center
c of the dike.  at that node, x(1), the temp gradient is zero.  from
c symmetry, the temp. at the "imaginary" node on the far side of the
c dike center is the same as at the first node in from the center.
c thus, t(x(0)) = t(x(2)), where t(x(1)) is the temperature at the
c dike center.
c
      xkap = (xkap1(1)+xkap2(1))/2.0e0
      xtxkap = dxtau/xkap
      a(1)  = 0.0e0
      b(1)  = -2.0e0*(1.0e0+xtxkap)
      c(1)  = 2.0e0
      yy(1) = 2.0e0*(1.0e0-xtxkap)*tt1(1) - 2.0e0*tt1(2)
c
c (3) prepare values for -dthck2 < x < infinity -- actually, just
c short of infinity, the third to last node.
c
      do 20 j = 2, jn-1
       xkap = (xkap1(j)+xkap2(j))/2.0e0
       xtxkap = dxtau/xkap
       xk = (xk1(j)+xk2(j))/2.0e0
       dxk1 = (xk1(j+1)-xk1(j-1))/(xk*4.0e0)
       dxk2 = (xk2(j+1)-xk2(j-1))/(xk*4.0e0)
       a(j)  = 1.0e0 - dxk2
       b(j)  = -2.0e0*(1.0e0+xtxkap)
       c(j)  = 1.0e0 + dxk2
       yy(j) = 2.0e0*(1.0e0-xtxkap)*tt1(j)
     +         - (1.0e0-dxk1)*tt1(j-1) - (1.0e0+dxk1)*tt1(j+1)
 20   continue
c
c (4) prepare values for x = infinity.  here, the next value of
```

```
c temperature is known as a boundary condition, and can therefore
c be put on the "right-hand-side" of the system of eqns.
c thus, t(x(jn+1)) = thi, which is known for all time steps.
c
      xkap = (xkap1(jn)+xkap2(jn))/2.0e0
      xtxkap = dxtau/xkap
      xk = (xk1(jn)+xk2(jn))/2.0e0
      dxk1 = (xkh-xk1(jn-1))/(xk*4.0e0)
      dxk2 = (xkh-xk2(jn-1))/(xk*4.0e0)
      a(jn)  = 1.0e0 - dxk2
      b(jn)  = -2.0e0*(1.0e0+xtxkap)
      c(jn)  = 0.0e0
      yy(jn) = 2.0e0*(1.0e0-xtxkap)*tt1(jn)
     +         - (1.0e0-dxk1)*tt1(jn-1) - 2.0e0*(1.0e0+dxk1)*thi
c
c (5) solve for tt2
c
      call thomas(jn,a,b,c,tt2,yy,jjj)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function condd(t)
c
c thermal conductivity of dike rock from temperature:  k = a + b/t[k]
c          for basalt: a = 0.6893, b = 522.3
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      condd = akd + bkd/(273.15e0+t)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function condmd(t)
c
c thermal conductivity modulus of magma from temperature, (dk/dT)/k.
c thermal conductivity: k = a + b/t[k]
c          for basalt: a = 0.6893, b = 522.3
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      tt = 273.15e0 + t
      xk = akd + bkd/tt
      condmd = -(1.0e0/xk)*bkd/(tt*tt)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function diffd(t)
c
c diffusivity of dike rock, kap[m*2/s] = a + b/t[k]
c          for basalt: a = 0.3052e-6,  b = 0.1247e-3
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      external condd
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      diffd = akapd + bkapd/(t+273.15e0)
c
c take heat of crystallization into account, if necessary.
c
      if ((t.gt.ts) .and. (abs((tdi-ts)/(tdi-thi)).gt.1.0e-4)) then
       xk = condd(t)
       diffd = xk/((xk/diffd)+xl/(tdi-ts))
      endif
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function condh(t)
c
c thermal conductivity of host rock from temperature: k = a + b/t[k]
c         for basalt: a = 0.6893, b = 522.3
c          sandstone:   = 0.4049,   = 732.9
c          limestone:   = 0.2101,   = 630.3
c            granite:   = 0.5543,   = 567.4
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      condh = akh + bkh/(273.15e0+t)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function condmh(t)
c
c thermal conductivity modulus of host rock from temperature, (dk/dT)/k.
c thermal conductivity: k = a + b/t[k]
c         for basalt: a = 0.6893, b = 522.3
c          sandstone:   = 0.4049,   = 732.9
c          limestone:   = 0.2101,   = 630.3
c            granite:   = 0.5543,   = 567.4
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      tt = 273.15e0 + t
      xk = akh + bkh/tt
      condmh = -(1.0e0/xk)*bkh/(tt*tt)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function diffh(t)
c
c diffusivity of host rock: kap[m*2/s] = a + b/t[k]
c           for basalt: a = 0.3052e-6,  b =  0.1247e-3
c            sandstone:   = 0.5773e-7     =  0.3032e-3
c            limestone:   = 0.1464e-6     =  0.1805e-3
c              granite:   = 0.3147e-6     =  0.3172e-3
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
      diffh = akaph + bkaph/(273.15e0+t)
c
      return
      end
```

```
c-345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine start(x,tau,t,eta,tt,ii,jj)
c
c initialize various vectors and quantities used in calculations.  this
c subroutine is called by earlya, earlyn, wholea, where storage is set
c for the variables initialized here.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      dimension tau(ii), x(jj), t(jj,ii), eta(jj), tt(jj)
      character ans*1
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (1) make vector x.
c
c ndt : number of dike thickness solution will be taken from contact.
c nndt: number of nodes per dike thickness.
c jmax: total number of nodes. (jmax-1)/2 must be an even number,
c        and jmax must be less than jj.
c jc  : index where x(jc) = 0 = dike contact.
c dx  : distance between x-points.
c
      write(6,1)
 1    format(/' ndt: # of dike half-thicknesses away from contact'
     +      /'      that are to be included in solution'
     +      //'     $ EARLYA   if output will be used by $'
     +      /'      $ EARLYN     WHOLEN, then ndt > 15   $'
     +      //' nndt: # of nodes per dike half-thickness.'
     +      //' (ndt+1)*nndt <= 600,                 ndt, nndt = '$)
      read(5,*) ndt, nndt
c
      do 7 j = 0, 10
       nndt = nndt + j
       jmax = nndt*(ndt+1) + 1
       jmax2 = (jmax-1)/2
       if (jmax-1 .eq. jmax2*2) goto 8
 7    continue
 8    if (jmax .gt. jj) then
       write(6,2)
 2     format(/' Too many nodes!!!'/)
       stop
      endif
      jc = nndt + 1
      dx = dthck2/nndt
      do 3 j = 1, jmax
 3     x(j) = -dthck2 + (j-1)*dx
c
c (2) make vector eta = x/sqrt(4*kappa_h*tau) and associated indeces
c
c kc: index where eta(kc) = 0 = dike contact.
c nk: number of nodes between successive values of eta such that
```

```
c      at least 100 nodes are used in each numerical integration
c      of the early-time equations.
c
       deta = 5.0e0/(jmax-1)
       do 6 k = 1, jmax
 6      eta(k) = (-2.5e0) + (k-1)*deta
       kc = 2.5e0/deta + 1
       do 4 nk = 1, 200
        if (nk*kc .ge. 100) goto 5
 4     continue
 5     continue
c
c (3) get times when temperatures are to be calculated.  there are
c ii times, but the first one must be 0.  the times are stored in the
c files TAUN.DAT or TAUD.DAT, one time per line.  TAUD.DAT must have
c time values in units of seconds; TAUN.DAT has nondimensional times,
c which are converted to dimensional values using dyke half-thickness
c and the diffusivity of the host rocks at ambient temperature.
c
c imax : number of times when solutions are calculated (<=ii).
c ie   : number of times when solutions are calculated
c         using early-time approximations.
c penet: penetration time for temperatures to first begin
c         dropping at dike center.  approximate and
c         conservatively estimated.
c chart: characteristic cooling time, used to convert from
c         nondimensional to dimensional times.  most cooling
c         is finished when tau = chart.
c
       chart = dthck2*dthck2/xkaph
       penet = 0.5e0*(dthck2*dthck2/(16.0e0*xkapd))
       ie = 0
c
       write(6,11)
 11    format(/' input times are in files TAUD.DAT [Dimensional]'
     +        /'                           or TAUN.DAT [Nondimensional]'
     +        /' which one?? [d/n] ? '$)
       read(5,'(1a)') ans
       if ((ans .eq. 'n') .or. (ans .eq. 'N')) then
        call fopen(1,'TAUN.DAT','old','exclusive',ios0)
        if (ios0 .ne. 0) stop
        read(1,*) imax
        read(1,*) (tau(i), i=1,imax)
        do 10 i = 2, imax
         tau(i) = tau(i)*chart
         if ((tau(i) .ge. penet) .and. (ie .eq. 0)) ie = i - 1
 10      if ((i .eq. imax) .and. (ie .eq. 0)) ie = imax
        rewind(1)
        write(1,'(1x,i4)') imax
        write(1,'(1x,1pg14.5)') (tau(i)/chart, i=1,imax)
        call fopen(2,'TAUD.DAT','unknown','exclusive',ios1)
        if (ios1 .ne. 0) stop
        write(2,'(1x,i4)') imax
        write(2,'(1x,1pg14.5)') (tau(i), i=1,imax)
```

```
      else
       call fopen(1,'TAUD.DAT','old','exclusive',ios0)
       if (ios0 .ne. 0) stop
       read(1,*) imax
       read(1,*) (tau(i), i=1,imax)
       do 12 i = 2, imax
        if ((tau(i) .ge. penet) .and. (ie .eq. 0)) ie = i - 1
 12     if ((i .eq. imax) .and. (ie .eq. 0)) ie = imax
       rewind(1)
       write(1,'(1x,i4)') imax
       write(1,'(1x,1pg14.5)') (tau(i), i=1,imax)
       call fopen(2,'TAUN.DAT','unknown','exclusive',ios1)
       if (ios1 .ne. 0) stop
       write(2,'(1x,i4)') imax
       write(2,'(1x,1pg14.5)') (tau(i)/chart, i=1,imax)
      endif
      call fclose(1,'keep',ios0)
      call fclose(2,'keep',ios1)
c
      write(6,90) jmax, dx
 90   format(/' total number of nodes, jmax = ',i4
     +       /' spacing between nodes,   dx = ',1pg11.4,' meters'/)
c
      return
      end
```

```
c-345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine finish(x,tau,t,eta,tt,tm,ii,jj,meth)
c
c write data to file.  output units of time can be hours, days, years.
c starting with a summary of the input parameters, the remainder of the
c file is a matrix.  the first column is distance, starting at the dike
c center (x=-dthck2), and going past the contact (x=0) to some
c specified distance.  the remaining columns are temperatures at times
c that are laballed above each column.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      dimension tau(ii), x(jj), t(jj,ii), eta(jj), tt(jj), tm(jj)
      character meth*2, str*50, time*5
      common /prop/ tdi, thi, dthck2, xkaph, xkapd, xkapm, xkh, xkd,
     +              xkm, xl, ts, xlam, tci, dx, deta, imax, ie, jmax,
     +              jc, nk, kc, akd, akh, akapd, akaph, bkd, bkh,
     +              bkapd, bkaph, iflag, itnum, err, jn
c
c (1) figure out best units for output times
c
      chart = dthck2*dthck2/xkaph
      if (chart .lt. 1.0e6) then
       time = 'hours'
       tunit = 3600.0e0
      elseif (chart .lt. 8.6e7) then
       time = 'days '
       tunit = 3600.0e0*24.0e0
      else
       time = 'years'
       tunit = 3600.0e0*24.0e0*365.25e0
      endif
c
c (2) open output file
c
      if (meth .eq. 'ea') then
       call fopen(1,'EARLYA.DAT','unknown','exclusive',ios)
      elseif (meth .eq. 'en') then
       call fopen(1,'EARLYN.DAT','unknown','exclusive',ios)
      elseif (meth .eq. 'wa') then
       call fopen(1,'WHOLEA.DAT','unknown','exclusive',ios)
      else
       call fopen(1,'WHOLEN.DAT','unknown','exclusive',ios)
      endif
      if (ios .ne. 0) stop
c
c (3) put out general conditions
c
      if (meth(1:1) .eq. 'e') then
       write(1,10) ie, jmax, tdi, thi, tci, dthck2, dx, jc, meth(2:2)
       write(1,20) xl, ts, xlam
       write(1,30) xkd, xkapd, xkh, xkaph
       if (meth(2:2) .eq. 'n') then
        write(1,34) akd, bkd, akapd, bkapd
```

```
         write(1,35) akh, bkh, akaph, bkaph
         write(1,36) err
        endif
        write(1,60) time
        str(1:14) = '(1x,  (f11.3))'
        write(str(5:6),'(i2)') ie
        write(1,str(1:14)) (tau(i)/tunit,i=1,ie)
        write(1,'(/)')
        str(1:27) = '(1x,3(f8.2,1x),  (f8.2,1x))'
        write(str(16:17),'(i2)') ie
        do 70 j = 1, jmax
 70      write(1,str(1:27)) eta(j), tt(j), x(j), (t(j,i),i=1,ie)
       else
        write(1,10) imax, jmax, tdi, thi, tci, dthck2, dx, jc, meth(2:2)
        write(1,30) xkd, xkapd, xkh, xkaph
        if (meth(2:2) .eq. 'n') then
         write(1,34) akd, bkd, akapd, bkapd
         write(1,35) akh, bkh, akaph, bkaph
         write(1,36) err
        endif
        write(1,61) time
        str(1:14) = '(1x,  (f10.2))'
        write(str(5:6),'(i2)') imax
        write(1,str(1:14)) (tau(i)/tunit,i=1,imax)
        write(1,'(/)')
        str(1:32) = '(1x,f9.3,1x,f6.1,2x,  (f6.1,1x))'
        write(str(21:22),'(i2)') imax
        do 71 j = 1, jmax
 71      write(1,str(1:32)) x(j), tm(j), (t(j,i),i=1,imax)
       endif
c
       call fclose(1,'keep',ios)
c
 990   return
c
c (4) formats
c
 10    format(1x,i4,' = number of temperature-distance profiles'/
    +         1x,i4,' = number of distance data points'//
    +         1x,1pe10.3,' deg c = initial temperature of dike rocks'/
    +         1x,1pe10.3,' deg c = initial temperature of host rocks'/
    +         1x,1pe10.3,' deg c = initial temperature at contact'//
    +         1x,1pe10.3,' meters = dike half-thickness'/
    +         1x,1pe10.3,' m = distance between distance data points'/
    +         1x,i4,' = index number for x(jc) = 0 = dyke wall'//
    +         1x,a1,' = solution method (a = analytic, n = numeric)'/)
 20    format(1x,1pe10.3,' j/m*m*m = total heat of crystallization'/
    +         1x,1pe10.3,' deg c = solidus temperature'/
    +         1x,1pe10.3,' = x(solidus)/sqrt(4*kappa_h*tau)'/)
 30    format(1x,1pe10.3,' w/m deg.c = k_m, conductivity, dike rocks'/
    +         1x,1pe10.3,' m*m/s = kappa_m, diffusivity, dike rocks'/
    +         1x,1pe10.3,' w/m deg.c = k_h, conductivity, host rocks'/
    +         1x,1pe10.3,' m*m/s = kappa_h, diffusivity, host rocks'/)
 34    format(1x,4(1pe10.3,1x),'a_k, b_k, a_kap, b_kap -- dike rocks'/)
 35   format(1x,4(1pe10.3,1x),'a_k, b_k, a_kap, b_kap -- host rocks'/)
 36   format(1x,1pe10.3,' = fractional maximum relative error'/)
 60   format(1x,'       eta   t(eta)       x      t(x) with ',a5,
     +        1x,'  = time units; values on next line.'/)
 61   format(1x,'         x     t_max   t(x) with ',a5,
     +        1x,'  = time units; values on next line.'/)
c
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      function erfunc(z)
c
c the error function; numerical recipes, p. 164
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
c
      x = abs(z)
      t = 1.0e0/(1.0e0+0.5e0*x)
      arg = x*x + 1.26551223e0 - t*(1.00002368e0+t*(0.37409196e0+
     +          t*(0.09678418e0+t*(-0.18628806e0+t*(0.27886807e0+
     +         t*(-1.13520398e0+t*(1.48851587e0+t*(-0.82215223e0+
     +         t*0.17087277e0))))))))
c
      if (z .ge. 0.0e0) then
        if (arg .gt. 10e0) then
          erfunc = 1.0e0
        else
          erfunc = 1.0e0 - t*exp(-arg)
        endif
      else
        if (arg .gt. 10.0e0) then
          erfunc = -1.0e0
        else
          erfunc = -1.0e0 + t*exp(-arg)
        endif
      endif
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine oderk2(h,t,y,deriv,istart)
c
c solve the system of 2 first-order ordinary differential equations:
c
c dy(i)/dt = f(i,t,y(1),y(2)), for i = 1, 2
c
c this routine advances the solution by a distance h employing a
c fourth-order runge_kutta method.  preceeding the first call, istart
c must be set to 1.  upon input, t is the current distance, y is the
c current solution of the system of ordinary differential equations
c as given in subroutine deriv, wk is a work space that must be saved
c between calls to oderk.  upon output, t is updated, to t+h, solution
c y is updated by dy(i)/dt = f(i,t+h,y(1),y(2)), and istart = 2 to
c signal a successful integration.
c
      implicit real*8 (a-h,o-z), integer*4 (i-n)
      dimension a(4), b(4), c(4), y(2), wk(2,2)
      external deriv
      data a /0.5e0, 0.292893219e0, 1.707106781e0, 0.166666667e0/
      data b /2.0e0, 1.0e0,         1.0e0,         2.0e0/
      data c /0.5e0, 0.292893219e0, 1.707106781e0, 0.5e0/
c
      if (istart .ne. 2) then
       wk(1,2) = 0.0e0
       wk(2,2) = 0.0e0
       qt = 0.0e0
       istart = 2
       call deriv(t,y,wk)
      endif
c
      do 105 j = 1, 4
       do 104 i = 1, 2
        temp = a(j)*(wk(i,1)-b(j)*wk(i,2))
        w = y(i)
        y(i) = y(i) + h*temp
        temp = (y(i)-w)/h
 104    wk(i,2) = wk(i,2) + 3.0e0*temp - c(j)*wk(i,1)
       temp = a(j)*(1.0e0-b(j)*qt)
       w = t
       t = t + h*temp
       temp = (t-w)/h
       qt = qt + 3.0e0*temp - c(j)
       call deriv(t,y,wk)
 105  continue
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine thomas(n,a,b,c,x,y)
c
c the thomas algorithim is an efficient method for solving a tri-
c diagonal system of equations.  the 3 central diagonals of the
c matrix are given as the 3 vectors a, b, c, each of length n.  b is
c the central vector.  the first element of a and the last of c are
c never referenced.  note that no element of b can equal 0, and that
c b and y are altered by this routine.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      dimension a(*), b(*), c(*), x(*), y(*)
c
c forward elimination
c
      do 2 i = 2, n
       w = a(i)/b(i-1)
       b(i) = b(i) - c(i-1)*w
 2     y(i) = y(i) - y(i-1)*w
c
c back substitition.
c
      x(n) = y(n)/b(n)
      do 3 i = n-1, 1, -1
 3    x(i) = (y(i)-c(i)*x(i+1))/b(i)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine solve2(func,a,b,error,itmax)
c
c determination of the roots to a system of 2 equations for 2 unknowns,
c f(x) = 0, where x(1) and x(2) are the unknowns and f(1) and f(2)
c are the equations.  upon input, x, a vector of length 2, is the
c initial quess of the roots; as output, it is the solution.  upon
c input, itmax, is the maximum allowable number of interations, or
c quesses; as output, it is the number of quesses required to find
c the solution.  error is the allowable difference between an exact
c and acceptable solution.
c
      implicit real*4 (a-h,o-z), integer*4 (i-n)
      dimension y(2), x1(2), y1(2), x2(2), y2(2), ddx(2)
      external func
c
c begin by checking to see if present quess of x is a solution.
c
      do 1 j = 1, itmax
       call func(a,b,y(1),y(2))
       if ((abs(y(1)) .lt. error) .and. (abs(y(2)) .lt. error)) then
        itmax = j
        return
       else
        write(6,91) a, b, y(1), y(2)
 91     format('for x(1,2) = ',2(1pg11.4,1x),
    +          ', func(x(1,2)) = ',2(1pg11.4,1x))
        ddx(1) = 1.0e-4*a
        ddx(2) = 1.0e-4*b
        x1(1) = a + ddx(1)
        x1(2) = b
        x2(1) = a
        x2(2) = b + ddx(2)
        call func(x1(1),x1(2),y1(1),y1(2))
        call func(x2(1),x2(2),y2(1),y2(2))
        dy1dx1 = (y1(1)-y(1))/ddx(1)
        dy1dx2 = (y2(1)-y(1))/ddx(2)
        dy2dx1 = (y1(2)-y(2))/ddx(1)
        dy2dx2 = (y2(2)-y(2))/ddx(2)
        det = dy1dx1*dy2dx2 - dy2dx1*dy1dx2
        ddx(1) = (y(2)*dy1dx2-y(1)*dy2dx2)/det
        ddx(2) = (y(1)*dy2dx1-y(2)*dy1dx1)/det
        a = a + ddx(1)
        b = b + ddx(2)
       endif
 1    continue
c
      itmax = j
      write(6,90) itmax
 90   format(/'solve2 failed after ',i4,' iterations'/)
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
    subroutine fopen(unit,name,status,acces,iostat)
c
c  note that this subroutine is highly system dependent; beware.
c
      integer unit,iostat
      character*(*) name,status,access
c
      open(unit,status=status,iostat=iostat)
c
      return
      end
```

```
c 345678-1-2345678-2-2345678-3-2345678-4-2345678-5-2345678-6-2345678-7-2
c
      subroutine fclose(unit,status,iostat)
c
c  note that this subroutine is highly system dependent; beware.
c
      integer unit,iostat
      character*(*) status
      close(unit,status=status,iostat=iostat)
c
      return
      end
```

Table 1: Symbols Used

$\underline{C}$	heat capacity per kilogram mass (at constant volume)
$\underline{h}$	heat of crystallization
$\underline{k}$	thermal conductivity
$\underline{Q}$	heat flux, $-\underline{k}\cdot\partial\Theta/\partial\underline{X}$
$\underline{T}$	dike thickness
$\underline{t}$	time
τ	time, $t\cdot\kappa_{hi}/(\underline{T}/2)^2$
$\underline{X}$,	coordinate pointed away from dike wall
$\underline{x}$	coordinate pointed away from dike wall, $\underline{X}/(\underline{T}/2)$
η	similarity variable, $\underline{X}/\sqrt{4\kappa_{hi}\underline{t}}$
Θ	temperature
θ	temperature, $(\Theta-\Theta_{hi})/(\Theta_{mi}-\Theta_{hi})$
κ	diffusivity, $\underline{k}/(\rho\underline{C})$
λ	similarity constant, $\chi_s/\sqrt{4\kappa_{hi}\underline{t}}$
ρ	density
χ	interface position

Subscripts

c	dike contact
h	host rock
i	initial
m	magma
s	solidus

TABLE 2: Subroutines and Functions.

START	Initialize distance arrays for $\underline{X}$ and η; read input times in non-dimensional τ or dimensiopnal $\underline{t}$ form; perform initial estimate to find greatest time when early-time solutions apply.
FINISH	Write results to file, including all parameters; times are converted to units of hours, days, or years.
TMPAE1	Analytic early-time, calculate $\Theta(\eta)$ for $(\rho \underline{h})_m = 0$ (eqs. 13b,c).
TMPAE2	Anal. early-time, calculate $\Theta(\eta)$ for $(\rho \underline{h})_m \neq 0$ (eqs. 13a,b,c).
TMPAE3	Anal. early-time, calculate Θ_{ci} and λ_s (eqs. 14a,b), called before TMPAE1 or TMPAE2.
TMPAL1	Anal. whole-time, calculate $\Theta(\underline{X},\underline{t})$ for $\underline{k}_m/\underline{k}_h = \kappa_m/\kappa_h = 1$ (eq. 18).
TMPAL2	Anal. whole-time, calculate $\Theta(\underline{X},\underline{t})$ for $\underline{k}_m/\underline{k}_h \neq 1$, $\kappa_m/\kappa_h \neq 1$ (eqs. 15a,b).
TMPAL3	Anal. whole-time, calculate $\Theta_{max}(\underline{X})$.
TMPNE1	Numeric early-time, driving routine for numerical integrator, called by SOLVE2, calls ODERK2.
TMPNE2	Num. early-time, called by integrator to evaluate eqs. 20, 21, called by ODERK2.
TMPNE3	Num. early-time, convert $\Theta(\eta)$ to $\Theta(\underline{X},\underline{t})$.
TMPNL	Num. late-time, driving routine, calls TMPNL1, TMPNL2
TMPNL1	Num. late-time, set arrays of temperatures and rock properties for current time step.
TMPNL2	Numeric late-time, set up eqs. 22a,b,c, calls THOMAS.
ERFUNC	(function) The error function.
SOLVE2	Newton-Raphson iteration to solve 2 equations with 2 unknowns.
ODERK2	Fourth-order Runge-Kutta solution for 2 1st-order ordinary differential equations.
THOMAS	Thomas algorithm for solving a tridiagonal system of equations.
CONDM	(function) Calculate $\underline{k}_m(\Theta)$ using eq. 1a.
CONDH	(function) Calculate $\underline{k}_h(\Theta)$ using eq. 1a.
CONDMM	(function) Calculate $\gamma_m(\Theta)$ using eq. 2.
CONDMH	(function) Calculate $\gamma_h(\Theta)$ using eq. 2.
DIFFM	(function) Calculate $\kappa_m(\Theta)$ using eq. 1b, or κ_m' using eqs. 1b, 3 if $\Theta > \Theta_s$ and $(\rho \underline{h})_m \neq 0$.
DIFFH	(function) Calculate $\kappa_h(\Theta)$ using eq. 1b.
FOPEN	open a file
FCLOSE	close a file

Table 3: Linking programs to subroutines and functions.

Analytical		Numerical	
EARLYA	WHOLEA	EARLYN	WHOLEN
START	START	START	TMPNL
TMPAE1	TMPAE1	SOLVE2	TMPNL1
TMPAE2	TMPA11	TMPNE3	TMPNL2
FINISH	TMPA12	FINISH	FINISH
SOLVE2	TMPAL3	TMPNE1	THOMAS
TMPAE3	FINISH	ODERK2	CONDM
ERFUNC	ERFUNC	TMPNE2	CONDMM
FOPEN	FOPEN	CONDM	DIFFM
FCLOSE	FCLOSE	CONDMM	CONDH
		DIFFM	CONDMH
		CONDH	DIFFH
		CONDMH	FOPEN
		DIFFH	FCLOSE
		FOPEN	
		FCLOSE	

Table 4: Examples of files TAU.DAT, DTAU.DAT

TAU.DAT	DTAU.DAT (seconds)
($\underline{t} \cdot (\underline{T}/2)^2/\kappa_{hi}$)	(equivalent to TAU.DAT if $\underline{T}/2$ = 1 meter and κ_{hi} = 0.75×10^{-6} m^2/s)
13	13
0.0000	0.0000
2.0000e-3	2.6667e3
5.0000e-3	6.6667e3
1.0000e-3	1.3333e4
2.0000e-2	2.6667e4
5.0000e-2	6.6667e4
1.0000e-2	1.3333e5
2.0000e-1	2.6667e5
5.0000e-1	6.6667e5
1.0000e-1	1.3333e6
2.0000e0	2.6667e6
5.0000e0	6.6667e6
1.0000e1	1.3333e7

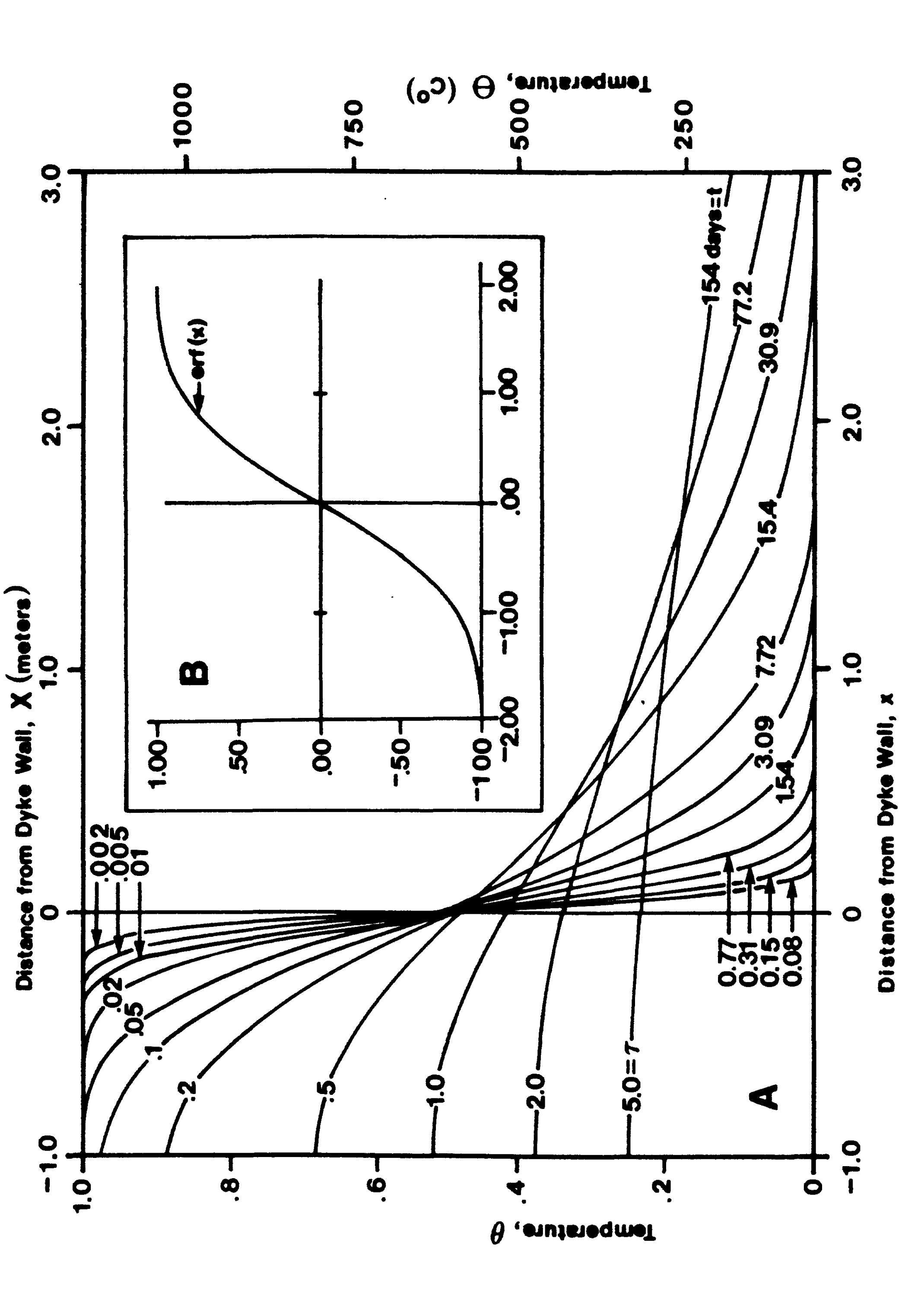
Table 5: Examples for keyboard input

	EARLYA	WHOLEA	EARLYN	WHOLEN
include heat of crystallization ? (y/n)	y		n	
dike thickness (m)	1	1	1	
initial temperature, magma & host rock (^oC)	1150 50	1150 50	1150 50	
conductivity, magma & host rock ($W/m \cdot {}^oC$)	2.25 2.25	2.25 2.25		
conductivity coefficients a_1 & b_1: dike rock			0.689 522	
host rock			0.250 944	
diffusivity, magma host rock (m^2/s)	7.5e-7 7.5e-7	7.5e-7 7.5e-7		
diffusivity coefficients a_2 & b_2: dike rock			3.06e-7 1.25e-4	
host rock			1.80e-7 2.50e-4	
latent heat (MJ/m^3)	900e+6			
solidus temperature (^oC)	950			
quess initial dike-contact temperature (^oC)	700		600	
quess λ_s	-0.4			
relative error			1.0e-4	
number of dike thicknesses from contact, & number of points per dike thickness, where temperatures are to be calculated.	3 30	3 30	19 30	
times when temperatures are to be calclulated are in file TAU.DAT (nondimensional) or DTAU.DAT dimensional) (n/d)	n	n	d	
read input from EARLYA.DAT (analytic early time) or EARLYN.DAT (numerical early time) (a/n)				n
number of time steps between times TAU?				10

FIGURE CAPTIONS

1. Thermal conductivity $\underline{k}$ and diffusivity κ as functions of temperature for basalt (A), granite (B), limestone (C) and sandstone (D). Also shown is heat capacity per unit volume $\rho\underline{C}$ for basalt, $\underline{k}$ for diabase and quartzite, and κ for diabase. Data from Touloukian _et al._ (1981).
2. Temperature as a function of distance from dike contact at various times, analytical solution. Thermal properties of the dike and host rocks are identical and constant; no latent heat. Bottom and left axes are nondimensional distance and time, respectively; top and right axes are dimensional equivalents for example (Table 5, column 1).
3. Temperature as a function of distance from dike contact at various times. Thermal properties of magma are that of "dry" basalt, and those of host rocks are that of "wet" basalt with a porosity of 10%; no heat of crystallization (Table 5, columns 3, 4).
4. Maximum temperature achieved in wallrocks versus distance from dike contact. Solid lines are for constant $\kappa_m/\kappa_h = \underline{k}_m/\underline{k}_h = 1$ with no latent heat (lower line) and latent heat of 900 MJ/m^3 (upper line). Dotted lines are for temperature dependent properties of "dry basalt" with latent heat and without. Dashed lines with dots are for temperature dependent properties of "wet basalt" host rocks with latent heat and without. Dashed line is analytical solution using eqs. 18 and 19 to approximate the influence of latent heat.

FIGURE 2



Distance from Dyke Wall, X (meters)
Temperature, Θ (C°)
Temperature, θ
Distance from Dyke Wall, x
A
B
erf (x)
154 days=t
77.2
30.9
15.4
7.72
3.09
1.54
0.77
0.31
0.15
0.08
.002
.005
.01
.02
.05
.1
.2
.5
1.0
2.0
5.0=τ

FIGURE 3

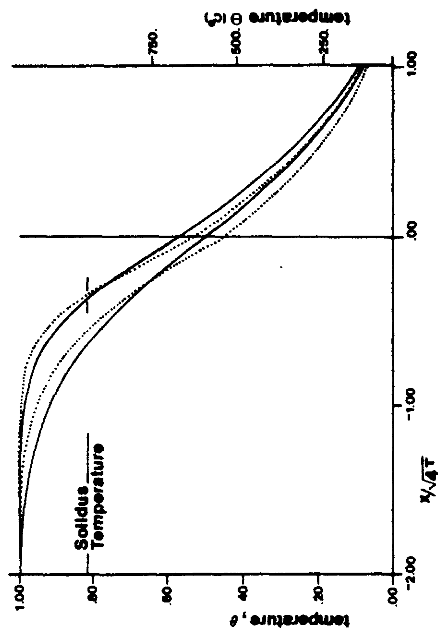
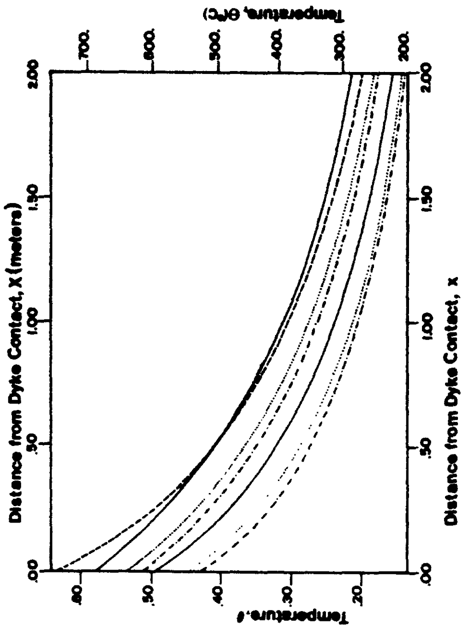

temperature, θ
temperature Θ (C)
Solidus Temperature
x/√4τ
1.00
.80
.60
.40
.20
.00
-2.00
-1.00
.00
1.00
750.
500.
250.

FIGURE 4


Distance from Dyke Contact, X (meters)
Distance from Dyke Contact, x
Temperature, θ
Temperature, Θ(°C)
.00
.50
1.00
1.50
2.00
.60
.50
.40
.30
.20
700.
600.
500.
400.
300.
200.