Thermal Analysis of LANL Ion Exchange Column

by
J. E. Laurinat
Westinghouse Savannah River Company
Savannah River Site
Aiken, South Carolina 29808

DOE Contract No. **DE-AC09-96SR18500**

Strategic Materials Technology Department
Chemical & Hydrogen Technology Section

WSRC-TR-99-00027
Revision 0

[redacted]

Thermal Analysis of LANL Ion Exchange Column (U)

May 20, 1999

[redacted]

Westinghouse Savannah River Company
Aiken, SC 29802

Prepared by the U. S. Department of Energy under Contract DE-AC09-96SR18500

DISCLAIMER

WSRC-TR-99-00027
Revision 0

Thermal Analysis of LANL Ion Exchange Column (U)

By J. E. Laurinat and M. E. Pansoy-Hjelvik, LANL

Issued: May 20, 1999

Approvals

Table of Contents

1.0 Introduction

This document reports results from an ion exchange column heat transfer analysis requested by Los Alamos National Laboratory (LANL). LANL plans to operate a laboratory-scale anion exchange column to extract Pu-238 from nitric acid solutions. The analysis is needed for a process hazards analysis.

The object of the analysis is to demonstrate that the decay heat from the Pu-238 will not cause resin bed temperatures to increase to a level where the resin significantly degrades. At low temperatures, resin bed temperatures increase primarily due to decay heat. At higher temperatures, an exothermic reaction between nitric acid and the resin predominates. This reaction degrades the resin and can lead to a potentially violent thermal excursion.

The analysis is based on a previous study of the SRS Frames Waste Recovery (FWR) column, performed in support of the Pu-238 production campaign for NASA's Cassini mission. In that study, temperature transients following an interruption of flow to the column were calculated. The transient calculations were terminated after the maximum resin bed temperature reached the Technical Standard of 60°C, which was set to prevent significant resin degradation.

LANL has requested calculations of temperature transients both during normal operation, including loading, washing, and elution, and following an interruption of flow. The LANL column differs from the FWR column in that it has a significantly smaller radius, 3.81 cm nominal versus approximately 28 cm. It follows that natural convection removes heat much more effectively from the LANL column, so that the column may reach thermal equilibrium before the maximum resin bed temperature reaches 60°C. Consequently, the calculations for a flow interruption were extended until an approach to thermal equilibrium was observed.

The LANL ion exchange process also uses a different resin than was used in the FWR column. The LANL column uses Reillex HPQ™ resin, which is more resistant to attack by nitric acid than the Ionac 641™ resin used in the FWR column. Therefore, heat generation and resin consumption by the nitric acid reaction, which was included in the previous study, is neglected in the LANL analysis.

Four parameters are varied in the LANL study: the plutonium feed concentration, the flow rate for loading, washing, and elution, the heat transfer conditions (either cooling by ambient air at 25°C or by a water jacket at 15°C), and the amount of plutonium loaded (either one or two times bed capacity). Temperature transients are calculated for a two-level matrix of sixteen different conditions. Calculations were performed using a finite difference computer code, which incorporates models for absorption and elution of plutonium and for forced and natural convection within the resin bed.

Calculations were performed to determine the maximum bed temperature during loading, washing, or elution for normal column operation and to determine the equilibrium

temperature for no flow to the column. Calculations for normal column operation were performed using an initial temperature and a feed temperature equal to either the ambient air temperature or the cooling jacket temperature. The model for the normal flow calculations did not include natural convection. The no flow calculations were started with the temperature and concentration profiles at the end of the loading stage, when there would be a maximum amount of plutonium either adsorbed on the resin or in the feed solution in the column. The amount of plutonium that was loaded was varied by changing the time for loading (e.g., twice the normal loading time for two times the bed capacity).

2.0 Summary of Results

Calculated maximum bed temperatures range from close to the ambient air or cooling water temperature for normal operation at the high feed flow rate of 20 mL/sec to 58.1°C for operation at the low feed flow rate of 0.5 mL/sec with air cooling, a feed amount equal to twice the bed plutonium capacity, and a high feed concentration of 0.1304 gm/mL Pu-238. Use of a 15°C cooling water jacket would lower this maximum temperature to 44.8°C for the same conditions. For the same feed conditions, the maximum temperatures for natural convection during a flow stoppage are 84.3°C for air cooling and 60.6°C for the water jacket. The maximum temperatures during normal operation occurred during washing. Following stoppages of flow, the maximum bed temperatures either peaked and then dropped slightly to an equilibrium temperature or reached their highest values at thermal equilibrium. Thermal equilibrium was achieved about 2 hours after flow was stopped.

The maximum calculated bed temperature exceeds 60°C, where significant resin degradation commences,[2] but is below 110°C, the approximate temperature at which resin-nitric acid reactions become thermally excursive.[3] Thus, the column can be operated safely at the specified conditions with either air or water-jacket cooling. Use of a water jacket will minimize resin degradation but might also slow rates of adsorption and elution, particularly at high feed concentrations and flow rates.

3.0 Input Conditions for Heat Transfer Analysis

Table 1 lists the input conditions for the LANL ion exchange column. As this table indicates, fixed conditions include the column dimensions and the resin bed physical characteristics. Variable conditions include the feed flow rate, the feed concentration, the amount of plutonium fed to the column, and the provisions for cooling the column, either a water cooling jacket or exposure to ambient air. The low and high feed concentrations apply to the solution that is fed to the column with and without pretreatment, respectively. Other variables have been selected to bound anticipated operating conditions.

4.0 Models for Ion Exchange Chemistry

In the ion exchange process, plutonium is selectively adsorbed onto the resin from concentrated nitric acid solution during loading and washing and is eluted from the resin with dilute nitric acid. In the model, the adsorption and elution processes occur irreversibly during normal operation. During a flow interruption, however, the model assumes that plutonium adsorbs onto the resin when the acid solution between the resin beads is saturated with plutonium and is eluted when the acid solution is not saturated. Saturation conditions are calculated using measured equilibrium relationships for nitrated resins.

The following reaction occurs on the resin surfaces,[2] which are assumed to be evenly distributed throughout the resin beads.

$$R_2Pu(NO_3)_6(\text{solid}) \Leftrightarrow PuNO_3^{3+} + 3NO_3^- + 2RNO_3(\text{solid})$$

where R represents a chemically active site on the resin substrate.

Equilibrium for this reversible reaction is governed by the solubility product[2]

$$K_1 = [PuNO_3^{3+}][NO_3^-]^3 \qquad (1)$$

where K_1 is an equilibrium constant that depends on the resin type and the plutonium loading on the resin. The model uses a value of 0.26, which has been measured for fully loaded Dowex-1 X-4 resin.[2]

In solution, the plutonium ions complex with nitrate ions according to[2]:

$$Pu^{4+} + NO_3^- \Leftrightarrow PuNO_3^{3+}$$

The equilibrium constant for this reaction is given by[2]

$$K_2 = \frac{[PuNO_3^{3+}]}{[Pu^{4+}][NO_3^-]} = 2.9 \pm 0.6 \qquad (2)$$

These equilibrium relations can be combined with the following stoichiometric equations to calculate the plutonium and nitrate concentrations in a saturated solution.

$$[NO_3]_{total} = [NO_3^-] + [PuNO_3^{3+}] \qquad (3)$$

and

$$[Pu]_{total} = [Pu^{4+}] + [PuNO_3^{3+}] \qquad (4)$$

For natural convection during a flow stoppage, the model solves these equations iteratively at each calculation node to calculate the saturation plutonium concentration based on the actual nitrate concentration and the saturation nitrate concentration based on the actual plutonium concentration. The calculated saturation values are then used to compute a rate of adsorption if the solution is supersaturated or desorption if the solution is unsaturated.

5.0 Models for Adsorption and Elution

The model assumes that the rates of adsorption and elution are controlled by diffusion of plutonium to and from the resin bead surfaces. Because the plutonium ionic species react rapidly at the resin surface, adsorption takes place at the outer surfaces of the resin beads before the plutonium can penetrate to the interior of the beads. Consequently, solid-phase, or particle, diffusion controls the rate of adsorption.[2] The particle diffusion model is analogous to a molecular diffusion model, with a particle diffusivity and the bulk adsorbed concentration replacing the ordinary diffusivity and the solution concentration.

A shrinking core diffusion approximation describes desorption of plutonium from the resin beads.[2] For elution during normal operation, the rate of desorption is modified by a factor that accounts for mixing of the dilute elution acid with the concentrated acid used to load the resin.[1] In addition, a pore diffusion model is used to calculate the rate of interchange of plutonium and nitrate between the interparticle and intraparticle, or pore, solutions. This takes the same form as the particle diffusion model for plutonium adsorption, with a pore diffusivity replacing the particle diffusivity. This interchange rate is calculated for normal operation but not for flow stoppages. To simplify the calculations during a flow stoppage, the model assumes that the acid is well mixed both in the spaces between resin beads and in the resin bead pores. The concentrations of plutonium between the beads and in the bead pores are combined on a volume average basis at the start of the flow stoppage. Thereafter, both solutions are treated as if they are located outside the resin beads.

Reference 1 describes details of the adsorption and elution models, so the adsorption and elution rate equations are not listed in this report. It must be noted, however, that there is an error in the equations for the rate of plutonium desorption, G_a (Equations 33, A-34, A-36, and A-37 in Reference 1). Equation A-34 should read

$$1-\frac{\bar{y}}{x_{max}}=\frac{\sqrt{c-\frac{x}{x_{max}}}}{\sqrt{\frac{x}{x_{max}}}+\sqrt{c-\frac{x}{x_{max}}}} \qquad (5)$$

With this change, Equations 33 and A-36 become

$$G_a\left(=-\frac{df}{dt}\right)=\frac{-cf_0\left(\frac{\sqrt{c-\frac{x}{x_{max}}}}{\sqrt{\frac{x}{x_{max}}}+\sqrt{c-\frac{x}{x_{max}}}}\right)}{\tau_{pore}\left\{1-\frac{4}{5}\sqrt[3]{\frac{f_0}{f}}-\frac{1}{5}\left(\frac{f_0}{f}\right)^2\right\}} \qquad (6)$$

Equation A-37 no longer is valid. Instead, as expected, the concentration gradient factor, $1-\frac{\bar{y}}{x_{max}}$, goes to zero as the saturation level of the acid, c, approaches that of the plutonium in solution, $\frac{x}{x_{max}}$.

Particle and pore diffusion coefficients control the rates of particle diffusion during loading and pore diffusion during elution, respectively. In the calculations for the FWR column, diffusivity values were selected to fit loading profiles and discharge transients for test elutions.[1] The particle diffusivity was set to 2 x 10^{-9} cm^2/sec, and the pore diffusivity was set to 1 x 10^{-6} cm^2/sec.

LANL has stated that their Reillex resin loads much more rapidly than the typical resins used at SRS. This implies that both the particle and pore diffusivities are much greater than these values. A bounding value for both diffusivities is the effective pore diffusivity for plutonium (IV) nitrate in aqueous solutions. This assumes that the controlling resistance for both adsorption and elution is pore diffusion and that the resin bead pores are sufficiently large so that steric hindrance is not a factor. The formula for the pore diffusivity is

$$D_{pore}=\frac{\varepsilon_i}{1-\varepsilon_o}\frac{D_{mol}}{\tau} \qquad (7)$$

where the term $\frac{\varepsilon_i}{1-\varepsilon_o}$ represents a volumetric correction factor relating the pore volume to the total volume of the resin beads and τ is the tortuosity. In Reference 1, the pore volume fraction ε_i was estimated to be 0.132 and the interparticle volume fraction ε_o was estimated to be 0.326. A tortuosity factor of 2 is used; this is typical for diffusion through either pores or unconsolidated solids.[4] The molecular diffusivity for plutonium ions is approximately 1.25 x 10^{-5} cm^2/sec (see Equation 28, Section 7.0). These values give an effective diffusivity of 1.2 x 10^{-6} cm^2/sec, which agrees with the pore diffusivity of 1 x 10^{-6} cm^2/sec for the FWR column analysis. Preliminary calculations demonstrated that the use of this diffusivity for both the particle and pore diffusion coefficients resulted in most of the plutonium being loaded onto the resin at the concentrations and flow rates specified by LANL.

6.0 Models for Bulk Mass and Heat Transfer

The previous section describes the model for mass transfer within the resin beads. Mass and heat transfer in the bulk solution between the resin beads are modeled using standard equations for convection and dispersion. Because flow is restricted to the interparticle solution, the convective mass transfer equations for plutonium and nitrate apply to the interparticle concentrations. The total concentrations are calculated by summing the concentrations in the interparticle solution, in the intraparticle, or pore, solution, and adsorbed on the particles:

$$C_{Pu} = x_{Pu} + y_{Pu} + f \tag{8}$$

$$C_{NO_3} = x_{NO_3} + y_{NO_3} \tag{9}$$

The convective mass transfer equations take the form

$$\frac{\partial x_{Pu}}{\partial t} + v_{zs}\frac{\partial x_{Pu}}{\partial z} + v_{rs}\frac{\partial x_{Pu}}{\partial r} = (D_{mol} + \alpha_z + D_{mix})\frac{\partial^2 x_{Pu}}{\partial z^2} + \frac{\alpha_r}{r}\frac{\partial}{\partial r}\left(r\frac{\partial x_{Pu}}{\partial r}\right) - \frac{dy_{Pu}}{dt} - \frac{df}{dt} \tag{10}$$

and

$$\frac{\partial x_{NO_3}}{\partial t} + v_{zs}\frac{\partial x_{NO_3}}{\partial z} + v_{rs}\frac{\partial x_{NO_3}}{\partial r} = (D_{mol} + \alpha_z + D_{mix})\frac{\partial^2 x_{NO_3}}{\partial z^2} + \frac{\alpha_r}{r}\frac{\partial}{\partial r}\left(r\frac{\partial x_{NO_3}}{\partial r}\right) - \frac{dy_{NO_3}}{dt} \tag{11}$$

The rates of change of the pore concentrations, $\frac{dy_{Pu}}{dt}$ and $\frac{dy_{NO_3}}{dt}$, are modeled using a pore diffusion model, and the rate of change of the adsorbed plutonium, $\frac{df}{dt}$, is modeled using a particle diffusion model. The pore and particle diffusion models are described in the previous section and in more detail in Reference 1.

The convective heat transfer equation includes a term for the decay heat, Q_{decay}. This equation is

$$\begin{aligned}&\rho_{bed}c_{p,bed}\frac{\partial T}{\partial t} + \rho_f c_{p,f} v_{zs}\frac{\partial T}{\partial z} + \rho_f c_{p,f} v_{rs}\frac{\partial T}{\partial r} \\ &= (k_{bed} + \rho_f c_{p,f}\alpha_z + \rho_f c_{p,f} D_{mix})\frac{\partial^2 T}{\partial z^2} + (k_{bed} + \rho_f c_{p,f}\alpha_r)\frac{1}{r}\frac{\partial}{\partial r}\left(r\frac{\partial T}{\partial r}\right) + Q_{decay}C_{Pu}\end{aligned} \tag{12}$$

The axial dispersion coefficients in the mass and heat transfer equations is given by[5]

$$\alpha_z = 2d_p v_{zs} \tag{13}$$

The radial dispersion coefficient is about one-fifth as large.[6] Thus,

$$\alpha_r = 0.4d_p v_{zs} \tag{14}$$

For normal operating conditions (loading, washing, and elution), the superficial axial velocity, v_{zs}, is specified, and the superficial radial velocity, v_{rs}, is assumed to be zero. For flow stoppages, these velocities result from natural convection. Natural convection velocities are ignored during loading washing, and elution. At high feed flow rates, the omission of natural convection does not have a significant effect. At lower feed flow rates, this omission is conservative in that it results in higher resin bed temperatures than would be predicted if natural convection were included.

Velocity components for natural convection flow are calculated by combining the Ergun equation with axial and radial component momentum equations, as suggested by Stewart and Dona.[7] The axial velocity for natural convection is modeled using a form of the Ergun equation for laminar flow, which is[8,9]

$$v_{zs} = \frac{d_p^2 \varepsilon_o^4}{200(1-\varepsilon_o)^2 \mu_L}\left[-\frac{\Delta P}{\Delta z}\right] \tag{15}$$

It may be noted that in Equation 15, $v_{zs} \propto \frac{\varepsilon_o^4}{(1-\varepsilon_o)^2}$, while in the Ergun equation, $v \propto \frac{\varepsilon_o^3}{(1-\varepsilon_o)^2}$. The extra void fraction multiplier is added because the buoyant force applies only to the fraction of the total volume occupied by the bulk solution and not to the resin beads.

The Ergun equation was originally derived for unidirectional flow. To model the two-dimensional flow in the resin bed, separate equations are used for the axial and radial velocity components. These equations take the form

$$v_{zs} = \frac{d_p^2 \varepsilon_o^4}{200(1-\varepsilon_o)^2 \mu}\left[-\frac{\partial P}{\partial z} + (\rho - \bar{\rho})g\right] \tag{16}$$

and

$$v_{rs} = \frac{d_p^2 \varepsilon_o^4}{200(1-\varepsilon_o)^2 \mu}\left[-\frac{\partial P}{\partial r}\right] \tag{17}$$

The buoyancy term in the axial velocity equation contains a reference density, $\overline{\rho}$. At equilibrium, the reference density must be set so that there is no net flow upward or downward at any bed level. This implies that the reference density must equal the average fluid density at a given bed level. In other words,

$$\overline{\rho} = \frac{2}{R^2}\int_{r=0}^{R} r\rho dr \tag{18}$$

The pressure gradient terms must be eliminated from Equations 17 and 18 to solve for the velocity components. This is accomplished by taking cross derivatives. The resulting equation is

$$\frac{\partial v_{rs}}{\partial z} - \frac{\partial v_{zs}}{\partial r} + \frac{d_p^2 \varepsilon_o^4 g}{200(1-\varepsilon_o)^2 \mu}\frac{\partial \rho}{\partial r} = 0 \tag{19}$$

The radial velocity is calculated by combining the continuity relation,

$$\frac{1}{r}\frac{\partial r v_{rs}}{\partial r} + \frac{\partial v_{zs}}{\partial z} = 0 \tag{20}$$

with Equation 19.

The vorticity equation is derived by eliminating the pressure gradient terms from the axial and radial component equations of motion.

To simplify the solution of these equations, the axial and radial velocities are expressed in terms of a stream function, defined by

$$v_{zs} = -\frac{1}{r}\frac{\partial r\psi}{\partial r} \tag{21}$$

and

$$v_{rs} = \frac{\partial \psi}{\partial z} \tag{22}$$

In terms of this stream function, the vorticity equation becomes:

$$\frac{\partial^2 \psi}{\partial z^2} + \frac{\partial}{\partial r}\left(\frac{1}{r}\frac{\partial r\psi}{\partial r}\right) + \frac{d_p^2 \varepsilon_o^4 g}{200(1-\varepsilon_o)^2 \mu}\frac{\partial \rho}{\partial r} = 0 \tag{23}$$

This equation is solved with the boundary conditions

$$\psi = 0 \text{ at } r = 0 \text{ and } r = \frac{d}{2} \tag{24}$$

and

$$\frac{\partial^2 \psi}{\partial z^2} = 0 \text{ at } z = 0 \text{ and } z = L \tag{25}$$

The first set of conditions specifies that there is no flow across the column wall or the column centerline. The second set of conditions states that the radial velocity gradient is zero at the top and bottom of the resin bed.

The natural convection velocities defined by Equations 16, 17, and 18 do not account for convection due to cross-sectional average axial density gradients. At the start of a flow stoppage, the solution in the resin bed is unstably stratified, with the higher plutonium concentrations and therefore the higher densities at the top of the resin bed. Preliminary calculations with these velocities showed that the solution remained unstably stratified even as equilibrium approached. Actually, significant unstable axial density variations would cause heavier fluid from above to mix with lighter fluid from below in regularly spaced regions called Bénard cells. A detailed model of Bénard cell convection was not attempted. Instead, mixing due to axial density gradients is calculated using an analogy to bubble columns. A dispersion model is used to describe mixing in bubble columns. The dispersion coefficient for bubble columns is defined in terms of the circulation velocity induced by the bubble motion, v_c, and the column diameter:[10]

$$D_{mix} = 0.24 d v_c \tag{26}$$

The same correlation is used to calculate the dispersion due to buoyancy-induced mixing in the resin bed. The circulation velocity is set equal to half that given by the Ergun equation and the pressure gradient based on the difference between the average densities for different levels:

$$v_c = \frac{d_p^2 \varepsilon_o^4 \left(\overline{\rho}\big|_z - \overline{\rho}\big|_{z+\Delta z} \right) g}{400 (1 - \varepsilon_o)^2 \mu} \tag{27}$$

The one-half factor appears because mixing occurs in both directions and can therefore occupy only half of the cross-sectional flow area in either direction. The use of cross-sectional average densities accounts for the axial pressure gradients neglected by Equation 16. That equation already incorporates local density variations in the radial direction, so local density gradients are not included.

In the model, the dispersion coefficient is calculated using a weighted distribution of circulation velocities for density differences at different levels. The distribution is based on the mixing height for the dispersion coefficient, which is 0.8 times the column

diameter.[10] If it is assumed that the mixing intensity decreases exponentially with axial distance, these values for the mixing height and velocity give the following expression for the local mixing velocity:

$$v_{eff} = v_c \exp\left(-2\frac{|z - z_0|}{0.8d} \right) \quad (28)$$

where z_0 is the axial location at which the dispersion coefficient is evaluated. It may be confirmed that an integration of this equation with respect to axial displacement gives the mixing height.

7.0 Physical Property Correlations

Correlations for physical properties, including densities, heat capacities, thermal conductivities, and viscosities, are listed in Reference 1. An additional correlation for molecular diffusivity has been added for this analysis.

The only change from the correlations in Reference 1 is to the density correlation. The terms that account for density changes due to the presence of plutonium and nitrate in solution have been changed. No density data for plutonium nitrate solutions could be found, so the new density correlation uses a correlation based on apparent molar densities. This correlation takes the form[11]

$$\rho = \rho_{H_2O} + \frac{\left(M_{Pu(NO_3)_4} - \phi_{Pu(NO_3)_4}\rho_{H_2O}\right)}{1000}[Pu(NO_3)_4] + \frac{\left(M_{HNO_3} - \phi_{HNO_3}\rho_{H_2O}\right)}{1000}[HNO_3] \quad (29)$$

The terms $\phi_{Pu(NO_3)_4}$ and ϕ_{HNO_3} represent partial molar volume for deviations from ideal solution densities. The partial molar volume for HNO_3 is not calculated because the effect of nitric acid concentration on the density has been measured. The partial molar volume for $Pu(NO_3)_4$ is given by the sum of the ionic partial molar densities:[11]

$$\phi_{Pu(NO_3)_4} = \phi_{Pu^{4+}} + \phi_{NO_3^-} \quad (30)$$

The partial molar density for NO_3^-, in turn, has been determined to be[12]

$$\phi_{NO_3^-} = 29.5 \quad (31)$$

The partial molar density for Pu^{4+} is given by the formula[11]

$$\phi_{Pu^{4+}} = 16 + 4.9r_x^3 - 20n_+ \quad (32)$$

where r_x, the ionic crystal radius for Pu^{4+}, is 0.96 Angstroms.[13]

Substitution of terms for the partial molar volume of $Pu(NO_3)_4$ and of a measured density coefficient for HNO_3[14] results in the following approximate equation for the overall density.

$$\rho = \rho_{H_2O} + 0.4277[Pu(NO_3)_4] + 0.031[HNO_3] \quad (33)$$

The HNO_3 concentration in this equation refers to the excess nitrate that is not ionically coupled with plutonium.

The molecular diffusivity is computed using the Nernst-Haskell equation, which takes the form[15]

$$D_{mol} = \frac{R_g T}{F^2} \frac{\frac{1}{n_+} + \frac{1}{n_-}}{\frac{1}{\lambda_+^0} + \frac{1}{\lambda_-^0}} \quad (34)$$

The cation and anion valences, n_+ and n_-, are 3 and 1, respectively. The value for λ_-^0 for nitrate is 71.4 $A/cm^2(V/cm)(g\text{-}equiv/cm^3)$. For plutonium the value of λ_+^0 for lanthanum, which is also a trivalent ion, 69.5 $A/cm^2(V/cm)(g\text{-}equiv/cm^3)$, was used.

8.0 Initial and Boundary Conditions

Solution of the mass and heat transfer equations requires initial conditions for concentrations, temperatures, and velocities and boundary conditions for velocity and mass and heat transfer rates at the column walls and at the top and bottom of the column. At the start of loading, the plutonium concentrations in the resin bed are set at zero, and the temperature is set equal to either the ambient air temperature of 25°C or the cooling water temperature of 15°C. The starting concentrations and temperatures for flow stoppage analyses are taken from the end of the loading cycle. This is deemed to be conservative in that it results in the maximum amount of plutonium in the resin bed.

Boundary conditions at the ends of the column are fulfilled by having the flow mix with a head of feed solution at the top of the resin bed and discharge into a heel of solution at the bottom of the bed. The rates of heat and mass transfer in the liquid volumes above and below the resin bed are assumed to be much greater than the transfer rates within the bed. In the calculation, this assumption is implemented by setting temperatures and concentrations at the top and bottom bed surfaces equal to those in the head and heel volumes. The rates of heat and mass transfer into the head and the heel are then computed using one-sided gradients that include the temperatures and concentrations at the top two calculation nodes within the resin bed.

The calculation includes a continuity check to ensure that the area-averaged superficial head and heel velocities are equal to the feed velocity. At the side walls of the column, the radial superficial velocity is set equal to zero. There is no restriction on the axial velocity. In cases for which it is assumed that the column is cooled by ambient air, heat losses out the sides, top, and bottom of the column are calculated using relations for thermal radiation and natural and forced convection. When a water cooling jacket is specified, the surface temperature of the resin bed is set equal to the cooling water temperature, and it is assumed that the top and bottom are cooled by ambient air. The following section describes the heat transfer correlations for convection and radiation.

9.0 Heat Transfer Correlations

Heat transfer from the surface of the column is calculated by adding the thermal radiation heat flux to the maximum of the heat fluxes for natural convection and forced convection:

$$q_{tot} = q_{rad} + \max(q_{nc}, q_{fc}) \tag{35}$$

The thermal radiation heat flux is calculated using the equation:

$$q_{rad} = \sigma\varepsilon\left(T_{surf}^4 - T_{amb}^4\right) \tag{36}$$

A value of 0.6 is used for the surface emissivity ε . This emissivity is characteristic for steel surfaces and is a conservatively low estimate for glass surfaces.[16]

The natural convection heat flux is calculated from correlations for vertical surfaces and for horizontal surfaces facing upward. These correlations take the form

$$q_{nc} = h_{nc}\left(T_{surf} - T_{amb}\right) \tag{37}$$

The heat transfer coefficient is defined in terms of a Nusselt number, Nu:

$$h_{nc} = Nu\frac{k}{L} \tag{38}$$

For the side walls of the column, the Nusselt number is given by the correlation[17]

$$Nu = 0.59Ra_L^{0.25} \tag{39}$$

where Ra_L is the Rayleigh number based on the length of the column, defined by

$$Ra_L = \frac{g\rho_{amb}\left(\rho_{amb} - \rho_{surf}\right)L^3}{\mu^2}\frac{c_p\mu}{k} \tag{40}$$

For the top and bottom column surfaces, the Nusselt numbers are given by correlations for heat transfer from a warm surface facing upward and a warm surface facing downward, respectively:[17]

$$Nu = 0.54 Ra_D^{0.25} \tag{41}$$

and

$$Nu = 0.27 Ra_D^{0.25} \tag{42}$$

where Ra_D is the Rayleigh number based on the column diameter, defined by

$$Ra_D = \frac{g\rho_{amb}(\rho_{amb} - \rho_{surf})d^3}{\mu^2}\frac{c_p\mu}{k} \tag{43}$$

The natural convection heat transfer coefficient is compared to the heat transfer coefficient for forced convection, with an assumed air velocity of 30 cm/sec, or 1 ft/sec. If the natural convection coefficient is smaller than the forced convection coefficient, the overall convective coefficient is set equal to that for forced convection. The correlation for forced convection takes the form:

$$h_{fc} = Nu_{fc}\frac{k}{d} \tag{44}$$

The Nusselt number for forced convection, Nu_{fc}, is defined in terms of a Reynolds number, Re_{fc}:[18]

$$Nu_{fc} = 0.683 Re_{fc}^{0.466} \tag{45}$$

where

$$Re_{fc} = \frac{dv_{fc}\rho_{amb}}{\mu} \tag{46}$$

10.0 Calculation Method

The finite difference heat and mass transfer calculations were performed on a VAX computer using Fortran codes. Separate codes were written to calculate transients during normal operation of the column (loading, washing, and elution) and following a stoppage of flow. Each code generates an output of maximum resin bed temperatures as a function of time as well as temperature, concentration, velocity, and modified stream function profiles at specified times. The output files containing temperature and concentration

profiles from the transient calculation for normal column operation are used to initialize the conditions for the flow stoppage calculations. The modified stream function profiles were used to make contour plots that show the natural convection flow streamlines. These plots were used to check the validity of the natural convection velocity calculations. The modified stream function, $r\psi$, is calculated from Equations 23 through 25.

The calculations for normal conditions were performed using a donor cell algorithm that was semi-implicit in the dispersion terms and explicit in all other terms. An explicit donor cell algorithm was used to perform the calculations following a flow stoppage.

Code listings and sample input and output files appear in the Appendix.

11.0 Results of Calculations

Prior to calculating maximum temperature transients for the matrix of test conditions specified by LANL, preliminary analyses were performed to determine an optimum time step size and cell size for the finite difference method. Results were found to be insensitive to the time step size, provided it was small enough to guarantee numerical stability. Time step sizes of 0.02 sec for the high feed flow rate of 20 mL/sec, 1.0 sec for the low feed flow rate of 0.5 mL/sec, and 0.5 sec for natural convection during flow stoppages were used.

Cell sizes were decreased until approximate convergence of temperature profiles and flow streamlines was obtained. The test case with the low feed flow rate of 0.5 mL/sec, the high feed concentration of 0.1304 gm/cm^3 Pu-238, cooling by ambient air, and twice the bed adsorption capacity fed was selected because it should result in the highest maximum bed temperature, at least during normal operation. It was determined that a grid of 48 axial cells by 24 radial cells gave an adequate approximation. Figures 1 through 9 show temperature profiles and flow streamlines for this grid, as well grids of 96 axial cells by 24 radial cells and 48 axial cells by 48 radial cells.

Figures 1 through 3 depict temperature profiles at the end of loading. These figures give a cross-sectional view of the column, split lengthwise along its axis; negative values of the radius are plotted to show both halves of the cross-section. It may be noted that at this low feed flow rate, the maximum bed temperature is located only slightly below the column center, toward the discharge end. A comparison of these figures shows that the radial cell length has little effect on the temperature profile. Decreasing the axial cell length results in a maximum temperature which is about 1°C higher. The increase in the maximum temperature can be attributed to better resolution of the plutonium peak at finer axial discretization.

Figures 4 through 6 show temperature profiles as thermal equilibrium approaches 10 hours after flow is interrupted. Again, a comparison of results for difference cell sizes demonstrates that there are only slight differences in the temperature profiles. The peak

temperature varies between 84.3°C and 84.6°C. For all three calculations, the peak temperature is located near the column center, slightly more than halfway down from the top of the resin bed.

Figures 7 through 9 depict flow streamlines for the same conditions. The numbers in these figures are values of the stream function multiplied by 10,000. Two toroidal recirculation cells appear, one in the top half of the resin bed and another in the bottom half. These cells are driven by mixing with the liquid layers above and below the bed, respectively. At these equilibrium conditions, the solution within the resin bed is well-mixed. Consequently, the recirculation is primarily due to thermal density gradients. Warmer solution flows upward in the center of the resin bed, and cooler solution flow downward along the column walls. Similar recirculation patterns have been calculated for two-dimensional rectangular fluid containers with uniform volumetric heat sources.[19] Both the shape and intensity of the recirculation, as indicated by the values of the stream function, remain about the same for the different cell sizes.

Because the column's length is much greater than its diameter, the calculated recirculation cells have a high axial/radial aspect ratio. This suggests that there might be more than one series of cells stacked one above another. The model does not test for the hydrodynamic stability of the recirculation pattern, so it does not account for this possibility. Nevertheless, the model should give a reasonably accurate estimate of heat transfer. The reasoning is that the magnitude of both the solution density gradients and the recirculation velocities, which are proportional to those gradients, would remain about the same. In fact, measured heat transfer rates for natural convection in volumes filled with porous glass exceed theoretical heat transfer rates at high aspect ratios.[20] These differences diminish as the convection velocities and hence the heat transfer rates increase. This seems to indicate that the assumption of a single vertical layer of recirculation cells is conservative in that it underestimates the actual rate of heat transfer.

Figures 10 and 11 show the effect of cell size on maximum temperature transients during normal operation and after a flow stoppage. As Figure 10 shows, a change from 24 to 48 radial cells does not make any appreciable difference in the calculated maximum temperatures, and an increase from 48 to 96 axial cells raises the maximum bed temperature by less than 1°C. Figure 11 shows that, as thermal equilibrium approaches following a flow stoppage, there is no significant change in the maximum temperature going from 24 to 48 radial cells or from 48 to 96 axial cells.

The preceding comparisons demonstrate that the 48 by 24 cell discretization adequately models natural convection and heat transfer within the column, so this grid was used to analyze the sixteen operating conditions specified by LANL and listed in Table 1.

Table 2 lists maximum temperatures from those calculations, and Figures 12 through 27 portray maximum temperature transients. For normal operation , the maximum calculated temperatures range from just over either the ambient air or cooling water temperature at high feed flow rates to 58.1°C for the high feed concentration of 0.1304 gm/mL, the low feed flow rate of 0.5 mL/sec, cooling by ambient air at 25°C, and

feeding of an amount of plutonium equal to twice the bed capacity. When flow is stopped at the end of the load cycle, the maximum temperatures vary from 42.6°C to 84.3°C.

As Figures 12 through 19 show, during normal operation, temperatures remain low at the high feed flow rate because of the short duration of the load and wash cycles. As may be seen in Figures 20 through 27, at the low feed flow rate, bed temperatures increase during loading and the first stages of washing. Subsequently, after excess plutonium in solution is washed out of the column, the maximum temperature either decreases or remains stable. Maximum temperatures following a flow stoppage increase at a slightly more rapid rate than during the initial phases of washing. Eventually, these temperatures reach equilibrium levels which exceed the maximum temperatures during washing.

A comparison of results in Table 2 shows that the water jacket is effective in lowering temperatures within the resin bed. The maximum temperature is between 10°C and 14°C cooler with the water jacket at the low feed flow rate, and ranges from 16°C to 24°C cooler for natural convection during a flow stoppage. These temperature differences slightly exceed the temperature difference of 10°C between the ambient air and the cooling water due to improved heat transfer to the cooling jacket. (Natural convection correlations were used for cooling by ambient air, and no heat transfer resistance was assumed for cooling by the water jacket.)

Finally, increasing the amount of plutonium fed to the column raised the maximum bed temperatures only in certain cases. Significant temperature increases occurred for normal operation at the low feed flow rate and the high feed concentration and for a flow stoppage at the high feed concentration. The amount of plutonium fed to the column does not produce significantly higher temperatures at low feed concentrations because the plutonium concentration in solution is small and the amount of plutonium adsorbed on the column is limited by the resin capacity. Instead, at low concentrations, most of the excess plutonium that has been fed to the column at any particular time has passed through the column. According to the model, doubling the amount of plutonium fed to the column has a smaller effect on equilibrium temperatures during natural convection than at the low feed flow rate. The reason is that natural convection more effectively mixes the solution and transfers heat than the feed flow itself. This apparent anomaly is due to the lack of natural convection in the normal flow model.

To ensure that the maximum temperatures calculated for flow stoppages are truly bounding cases, additional calculations were performed. These calculations were restricted to the cases with the low feed flow rate and high feed concentration, where the maximum temperatures increased significantly during washing. In the additional cases, it was assumed that the flow stoppages occur when the maximum bed temperature peaks during washing. Figures 28 through 31 illustrate the results of these calculations, and Table 3 compares equilibrium temperatures for flow stoppages at peak resin bed temperatures with those for flow stoppages at maximum loadings. In each case, the equilibrium temperature for flow stoppages at the peak bed temperatures are lower. This confirms that flow stoppages at the end of the load cycle represent bounding conditions.

12.0 Conclusions

Calculated maximum bed temperatures range from close to the ambient air or cooling water temperature for normal operation at the high feed flow rate of 20 mL/sec to 58.1°C for operation at the low feed flow rate of 0.5 mL/sec with air cooling, a feed amount equal to twice the bed plutonium capacity, and a high feed concentration of 0.1304 gm/mL Pu-238. Use of a 15°C cooling water jacket would lower this maximum temperature to 44.8°C for the same conditions. For the same feed conditions, the maximum temperatures for natural convection during a flow stoppage are 84.3°C for air cooling and 60.6°C for the water jacket. The maximum temperatures during normal operation occurred during washing. Following stoppages of flow, the maximum bed temperatures either peaked and then dropped slightly to an equilibrium temperature or reached their highest values at thermal equilibrium. Thermal equilibrium was achieved about 2 hours after flow was stopped.

The maximum calculated bed temperature exceeds 60°C, where significant resin degradation commences,[2] but is below 110°C, the approximate temperature at which resin-nitric acid reactions become thermally excursive.[3] Thus, the column can be operated safely at the specified conditions with either air or water-jacket cooling. Use of a water jacket will minimize resin degradation but might also slow rates of adsorption and elution, particularly at high feed concentrations and flow rates.

13.0 References

1. J. E. Laurinat, "Self-Heating of Pu-238 Anion Exchange Column (U)," WSRC-TR-94-0166, Rev. 0, March 23, 1994.

2. J. L. Ryan and E. J. Wheelwright, "The Recovery, Purification, and Concentration of Plutonium by Anion Exchange in Nitric Acid (Declassified)," General Electric-Hanford Atomic Products Operation Report HW-55893, January 2, 1959.

3. W. J. Van Slyke, G. Jansen, Jr., and W. H. Swift, "Thermal Effects in Anion Exchange Resin – Nitric Acid Systems", Pacific Northwest Laboratory Report BNWL-114, August, 1965.

4. C. N. Satterfield, Mass Transfer in Heterogeneous Catalysis, Colonial Press (1975), 35-36.

5. R. H. Perry and D. W. Green, eds., Perry's Chemical Engineers' Handbook, 6th ed., McGraw Hill, New York (1984), 16-27.

6. E. A. Ebach and R. R. White, "Mixing of Fluids Flowing through Beds of Packed Solids," A.I.Ch.E. J., 4(2), 161-169.

7. W. E. Stewart, Jr., and C. L. G. Dona, "Free Convection in a Heat Generating Porous Medium in a Finite Vertical Cylinder," J. Heat Transfer, 110(5), 1988, 517-520.

8. R. H. Perry and D. W. Green, eds., Perry's Chemical Engineers' Handbook, 6th ed., McGraw Hill, New York (1984), 5-54.

9. M. Leva, Fluidization, McGraw-Hill, New York (1959), 47.

10. J. B. Joshi, "Axial Mixing in Multiphase Contactors – A Unified Correlation," Trans. I. Chem. E., 58, 1980, 155-165.

11. A. M. Couture and K. J. Laidler, "The Partial Molar Volumes of Ions in Aqueous Solution: 1. Dependence on Charge and Radius," Can. J. Chem., 34, 1956, 1209-1216.

12. A. L. Horvath, Handbook of Aqueous Electrolyte Solutions: Physical Properties, Estimation, and Corelation Methods, Ellis Horwood Ltd., Chichester, England (1985), 131-133.

13. J. J. Katz, G. T. Seaborg, and L. R. Morss, The Chemistry of the Actinide Elements, 2nd ed., Volume 2, Chapman and Hall, London (1986), 1448.

14. C. M. Slansky, "Appendix I: Physical Properties of the Uranyl Nitrate, Aluminum Nitrate, Nitric Acid and Water System," in F. R. Bruce, J. M. Fletcher, and H. H. Hyman, Progress in Nuclear Energy, Series III: Process Chemistry, Vol. 2, Pergamon Press (New York (1958).

15. R. C. Reid, J. M. Prausnitz, and T. K. Sherwood, The Properties of Gases and Liquids, McGraw-Hill, New York (1977), 591.

16. R. H. Perry and D. W. Green, eds., Perry's Chemical Engineers' Handbook, 6th ed., McGraw Hill, New York (1984), 10-51.

17. R. H. Perry and D. W. Green, eds., Perry's Chemical Engineers' Handbook, 6th ed., McGraw Hill, New York (1984), 10-13.

18. F. Kreith and W. Z. Black, Basic Heat Transfer, Harper and Row, New York (1980), 251.

19. A. A. Emara and F. A. Kulacki, "A Numerical Investigation of Thermal Convection in a Heat-Generating Fluid Layer," J. Heat Transfer, 102(8), 1980, 531-537.

20. N. Seki, S. Fukusako, and H.Inaba, "Heat Transfer in a Cavity Packed with Fibrous Glass," J. Heat Transfer, 100(11), 1978, 748-750.

14.0 Nomenclature

c = relative saturation of nitric acid in the solution between the resin beads, dimensionless
c_p = heat capacity of air, cal/gm/K
$c_{p,bed}$ = heat capacity of the resin bed, cal/gm/K
$c_{p,f}$ = heat capacity of the feed solution, cal/gm/K
C_{NO_3} = total bulk nitrate concentration, gm/cm^3
C_{Pu} = total bulk plutonium concentration, gm/cm^3
d = column diameter, cm
d_p = resin bead diameter, cm
D_{mix} = axial dispersion coefficient for buoyancy-induced or bubble-induced mixing, cm^2/sec
D_{mol} = molecular diffusivity of plutonium nitrate in aqueous solution, cm^2/sec
D_{pore} = pore diffusivity for diffusion of plutonium nitrate in resin beads, cm^2/sec
f = amount of plutonium adsorbed on resin, gm/cm^3
f_0 = amount of plutonium adsorbed at beginning of elution, gm/cm^3
F = Faraday's constant, 96,500 K/gmole
g = gravitational acceleration, 980 cm/sec^2
G_a = mass transfer rate for adsorption of plutonium, gm/cm^3/sec
h_{nc} = natural convection heat transfer coefficient, cal/cm^2/sec/K
k = thermal conductivity of air, cal/cm/sec/K
k_{bed} = thermal conductivity of the resin bed, cal/cm/sec/K
L = column length, cm
M_i = molecular weight of ith compound, gm/gmole
n_+ = cation valence for Nernst-Haskell diffusivity equation
n_- = anion valence for Nernst-Haskell diffusivity equation
Nu = Nusselt number
P = pressure, 10^{-1} Pascal
q_{nc} = natural convection heat flux, cal/cm^2/sec
q_{rad} = radiation heat flux, cal/cm^2/sec
q_{tot} = total heat flux, cal/cm^2/sec
Q_{decay} = volumetric rate of heat generation due to alpha decay, cal/cm^3/sec
r = radial distance from center of resin bed, cm
r_x = ionic radius, Angstroms
R = column radius, cm
R_g = gas law constant, 8.314 J/gmole/K
Ra_D = the Rayleigh number based on the column diameter
Ra_L = the Rayleigh number based on the length of the column
T = temperature, K
T_{amb} = ambient temperature, K

T_{surf} = surface temperature, K

v_c = mixing velocity for interchange between adjacent solution layers, also, circulation velocity for bubble mixing, cm/sec

v_{eff} = effective velocity for mixing between solution layers at different levels within the resin bed, cm/sec

v_{fc} = circulation velocity of ambient air, cm/sec

v_{rs} = radial component of the superficial velocity of solution in the resin bed, cm/sec

v_{zs} = axial component of the superficial velocity of solution in the resin bed, cm/sec

x = plutonium concentration in bulk solution between resin beads, gm/cm^3

x_{max} = maximum plutonium concentration in bulk solution between resin beads, gm/cm^3

x_{NO_3} = nitrate concentration in the bulk solution between resin beads, gm/cm^3

x_{Pu} = plutonium concentration in the bulk solution between resin beads, gm/cm^3

$\bar{y}$ = time-averaged plutonium concentration in the resin bead pores at the bead surface, used in the calculation of elution rates through the bead, gm/cm^3

y_{NO_3} = nitrate concentration in the solution in the resin bead pores, gm/cm^3

y_{Pu} = plutonium concentration in the solution in the resin bead pores, gm/cm^3

z = axial distance from bed entrance, cm

α_r = radial dispersion coefficient for mixing by the feed flow, cm^2/sec

α_z = axial dispersion coefficient for mixing by the feed flow, cm^2/sec

ε = surface emissivity, assumed to be 0.6 (for machined steel)[2]

ε_i = fraction of bulk volume occupied by liquid inside resin beads

ε_o = fraction of bulk volume occupied by liquid in spaces between resin beads

λ_+^0 = limiting cationic conductance for Nernst-Haskell diffusivity equation, A/cm^2 (V/cm)(g-equiv/cm^3)

λ_-^0 = limiting anionic conductance for Nernst-Haskell diffusivity equation, A/cm^2 (V/cm)(g-equiv/cm^3)

μ = viscosity of air, gm/cm/sec

μ_L = viscosity of resin bed solution, gm/cm/sec

ρ = solution mixture density, gm/cm^3

$\bar{\rho}$ = average solution density at a given axial level, gm/ cm^3

ρ_{amb} = ambient air density, gm/cm^3

ρ_{H_2O} = water density, gm/ cm^3

ρ_{surf} = air density at column surface, gm/cm^3

σ = Boltzmann's constant, 1.356 x 10^{-12} cal/cm^2/sec/K^4

τ = tortuosity factor for pore diffusion

τ_{pore} = charcteristic time for pore diffusion, sec

ϕ_i = excess partial molar volume for ith anion, cation, or salt, cm^3/gmole

ψ = stream function, cm^2/sec

Table 1. Conditions for LANL Anion Exchange Column

Constant Condition	Value
Column Radius	3.81 cm nominal 3.73 cm measured by displacement
Length of Resin Bed	34.1 cm
Height of Feed Solution above Bed	2.54 cm
Depth of Discharge Solution below Bed	2.54 cm
Resin Bead Diameter	0.04 cm

Variable Condition	High Value	Low Value
Feed Flow Rate	20 cm^3/sec	0.5 cm^3/sec
Feed Concentration	0.1304 gm/cm^3 Pu-238 0.163 gm/cm^3 total Pu	0.0122 gm/cm^3 Pu-238 0.01525 gm/ cm^3 total Pu
Amount of Pu Fed*	2 times bed capacity for Pu adsorption	equal to bed capacity for Pu adsorption
Heat Transfer Conditions	Exposure to 25°C ambient air	Cooling by 15°C water jacket

*The amount of Pu fed was controlled by varying the times for loading the bed with plutonium nitrate solution. A summary of loading, washing, and elution times follows.

Flow Rate	Concentration	Amount Fed	Load Time	Wash Time	Elution Time
High	High	1 x capacity	29 sec	149 sec	187 sec
High	High	2 x capacity	58 sec	149 sec	187 sec
High	Low	1 x capacity	307 sec	149 sec	187 sec
High	Low	2 x capacity	614 sec	149 sec	187 sec
Low	High	1 x capacity	1150 sec	5974 sec	7468 sec
Low	High	2 x capacity	2300 sec	5974 sec	7468 sec
Low	Low	1 x capacity	12264 sec	5974 sec	7468 sec
Low	Low	2 x capacity	24528 sec	5974 sec	7468 sec

Table 2. Maximum Temperatures for Operation of LANL Anion Exchange Column

Pu Feed Conc. (gm/mL)	Flow Rate (mL/sec)	Cooling Jacket (Yes/No)	Feed Quantity (# times bed capacity)	Max. T during Operation (°C)	Equil. T for no flow (°C)
0.1304	20.0	No	1	25.5	58.7
0.1304	20.0	No	2	25.9	77.6
0.1304	20.0	Yes	1	15.6	42.6
0.1304	20.0	Yes	2	15.9	57.1
0.0122	20.0	No	1	25.7	62.3
0.0122	20.0	No	2	25.7	63.1
0.0122	20.0	Yes	1	15.7	45.2
0.0122	20.0	Yes	2	15.7	45.2
0.1304	0.5	No	1	46.7	66.3
0.1304	0.5	No	2	58.1	84.3
0.1304	0.5	Yes	1	34.6	47.0
0.1304	0.5	Yes	2	44.8	60.6
0.0122	0.5	No	1	47.3	62.5
0.0122	0.5	No	2	47.6	63.1
0.0122	0.5	Yes	1	34.9	45.2
0.0122	0.5	Yes	2	35.2	45.2

Table 3. Comparison of Maximum Equilibrium Temperatures for Flow Stoppages at the End of Loading and at the Maximum Transient Temperature during Washing

Pu Feed Conc. (gm/mL)	Flow Rate (mL/sec)	Cooling Jacket (Yes/No)	Feed Quantity (# times bed capacity)	Equil. T for Flow Stoppage at End of Loading (°C)	Equil. T for Flow Stoppage at Max. T during Wash (°C)
0.1304	0.5	No	1	66.3	60.3
0.1304	0.5	No	2	84.3	60.5
0.1304	0.5	Yes	1	47.0	42.9
0.1304	0.5	Yes	2	60.6	52.0

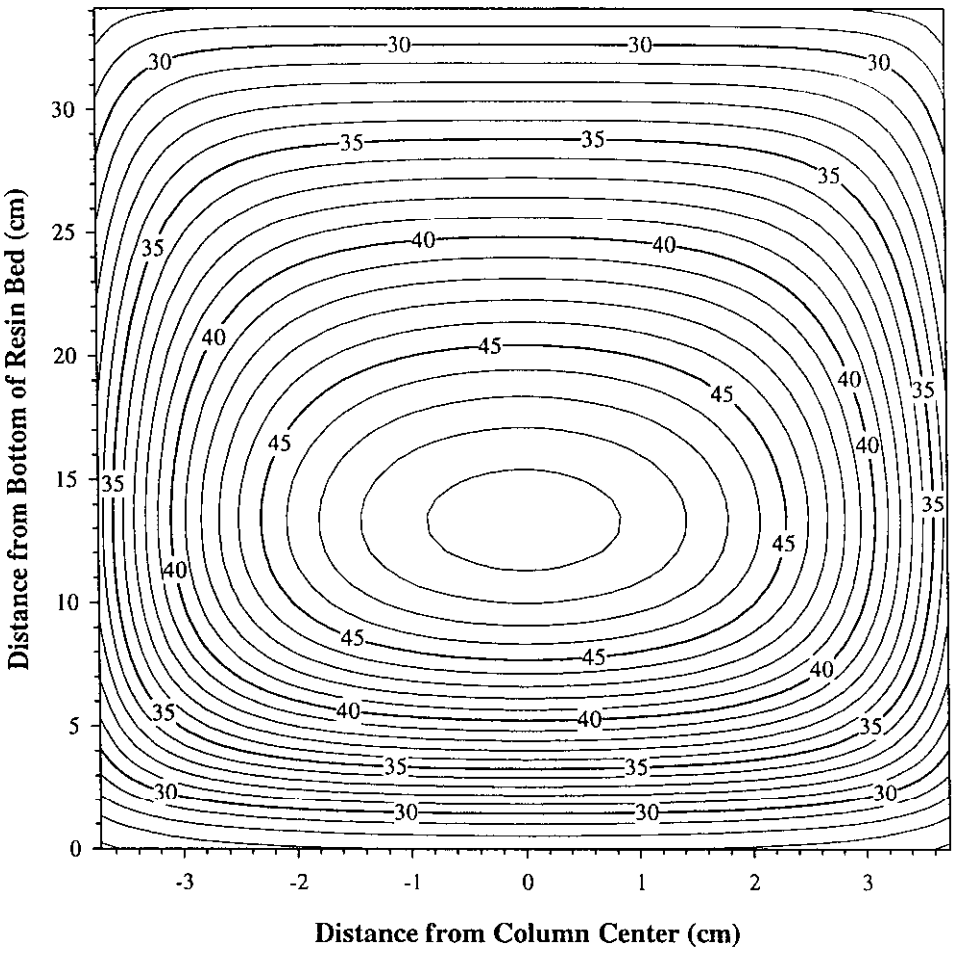

Figure 1. Temperature Distribution at the End of the Load Cycle for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 48 Axial and 24 Radial Cells.

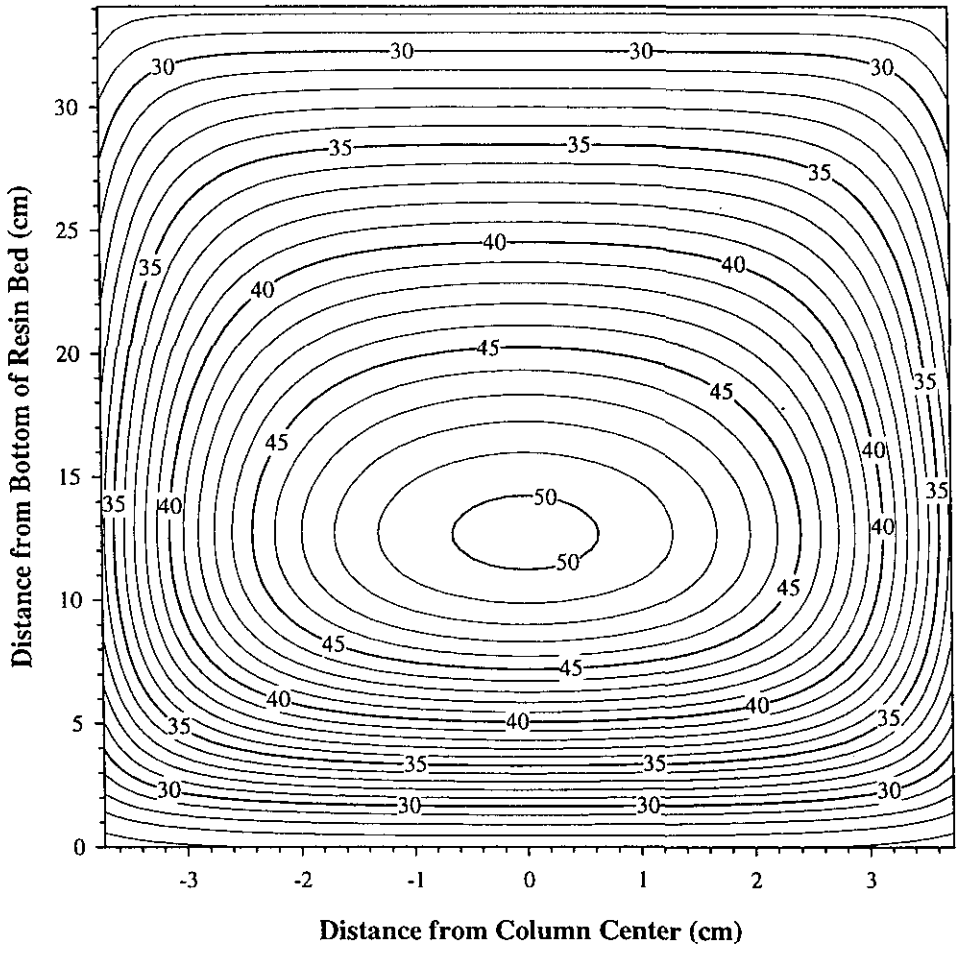

Figure 2. Temperature Distribution at the End of the Load Cycle for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 96 Axial and 24 Radial Cells.

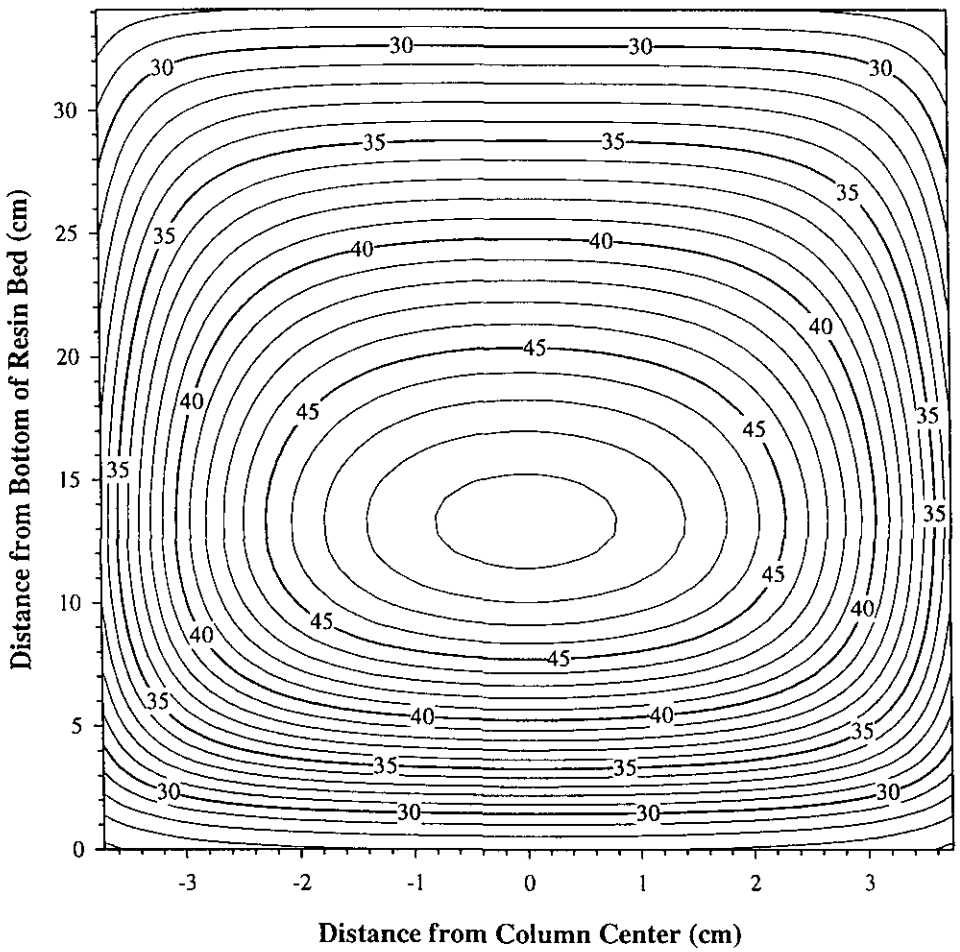

Figure 3. Temperature Distribution at the End of the Load Cycle for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 48 Axial and 48 Radial Cells.

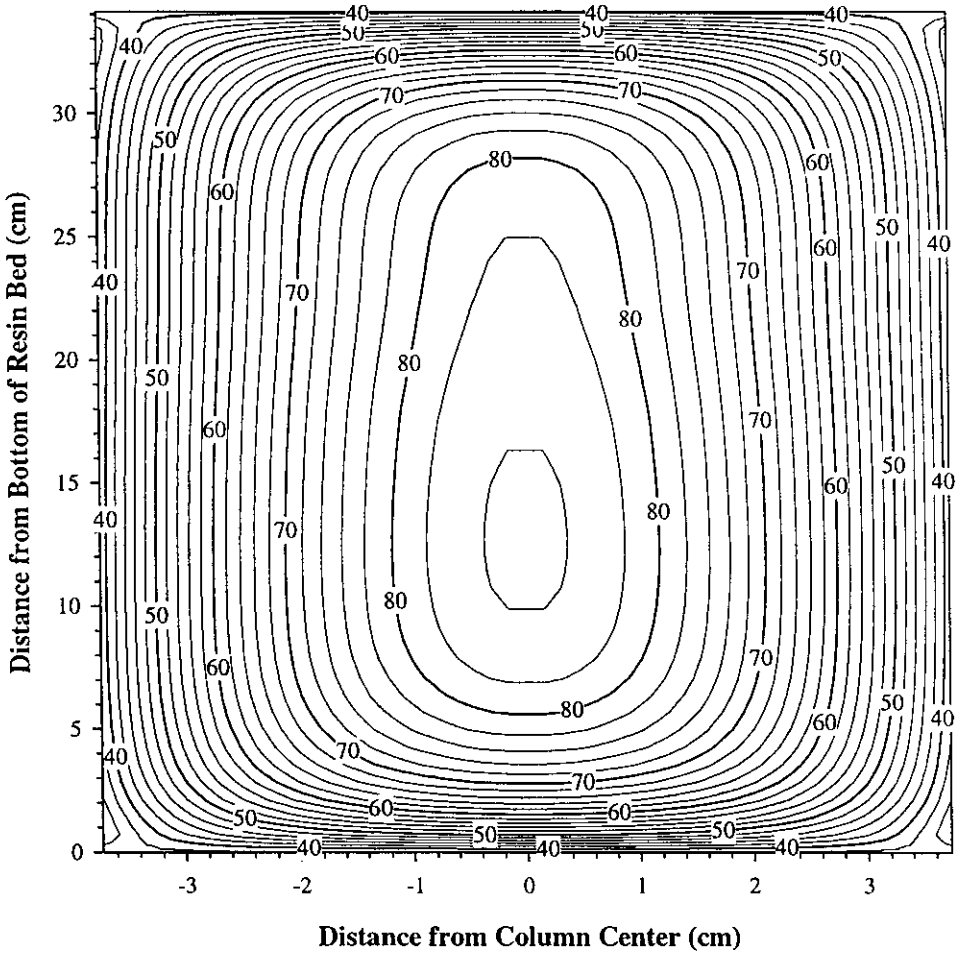

Figure 4. Temperature Distribution at Thermal Equilibrium after a Flow Stoppage for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 48 Axial and 24 Radial Cells.

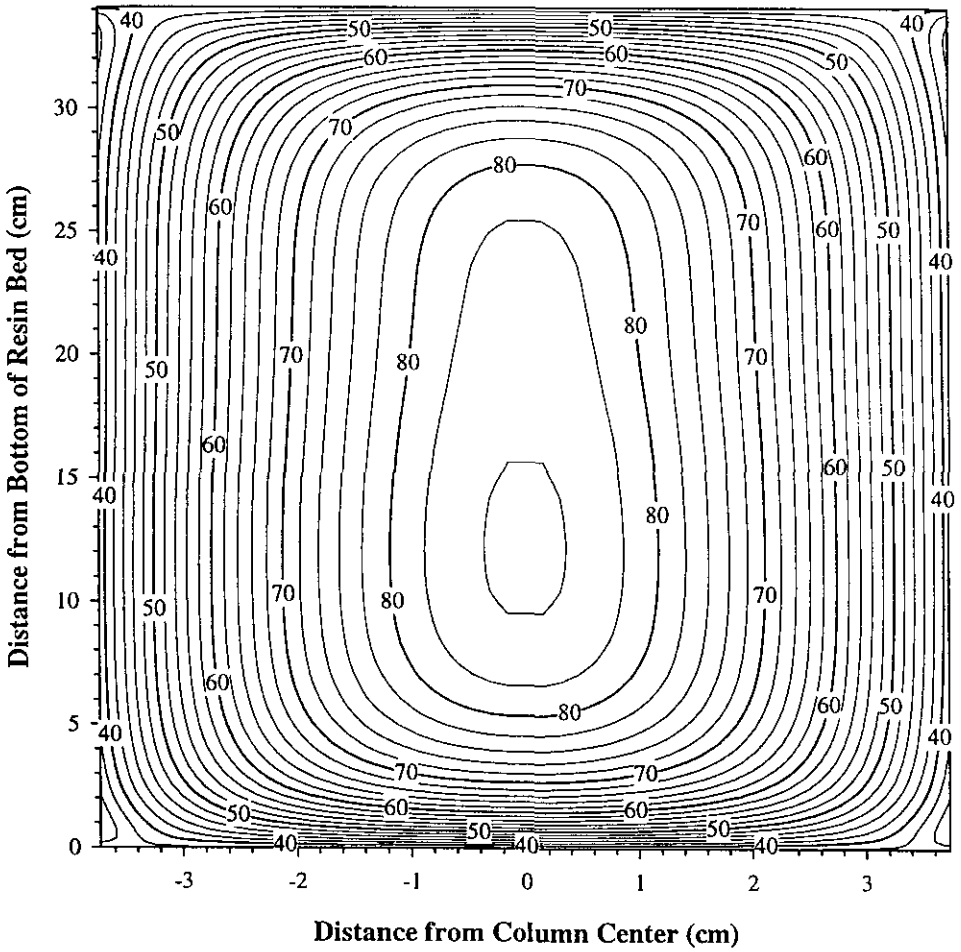

Figure 5. Temperature Distribution at Thermal Equilibrium after a Flow Stoppage for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 96 Axial and 24 Radial Cells.

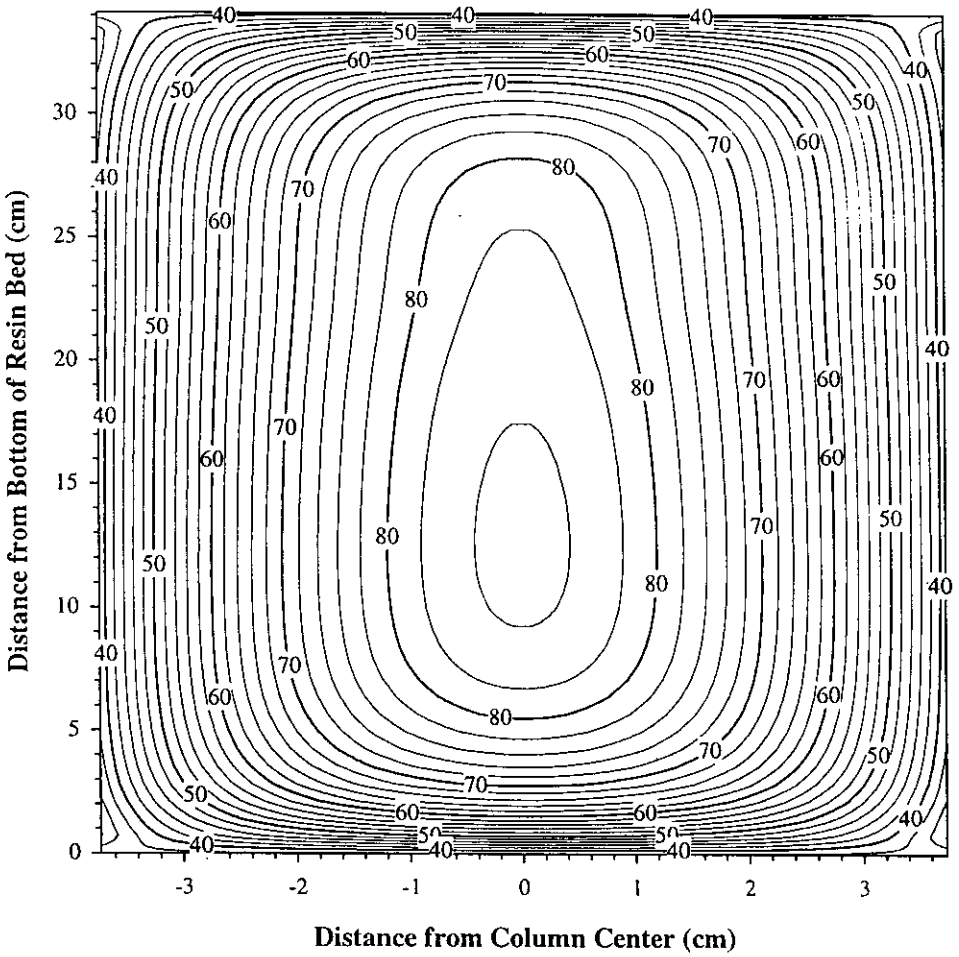

Figure 6. Temperature Distribution at Thermal Equilibrium after a Flow Stoppage for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 48 Axial and 48 Radial Cells.

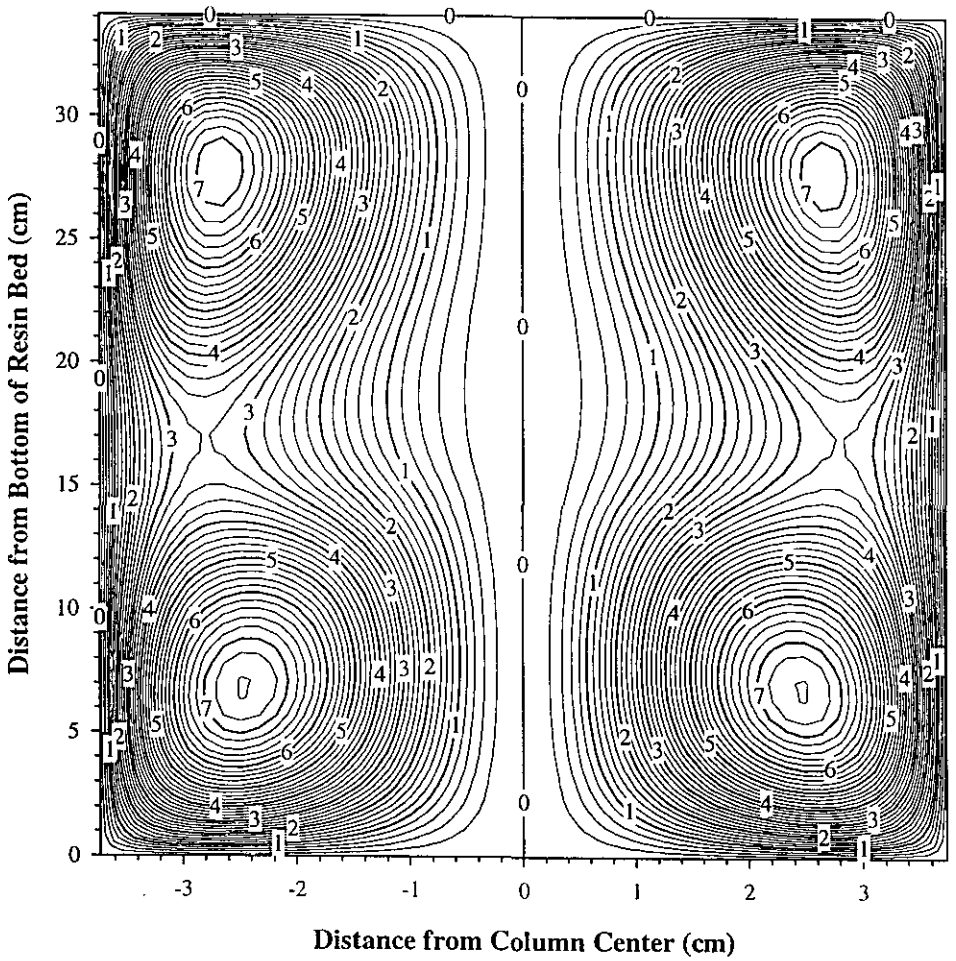

Figure 7. Flow Streamlines at Thermal Equilibrium after a Flow Stoppage for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 48 Axial and 24 Radial Cells. (Numbers on chart refer to values of stream function.)

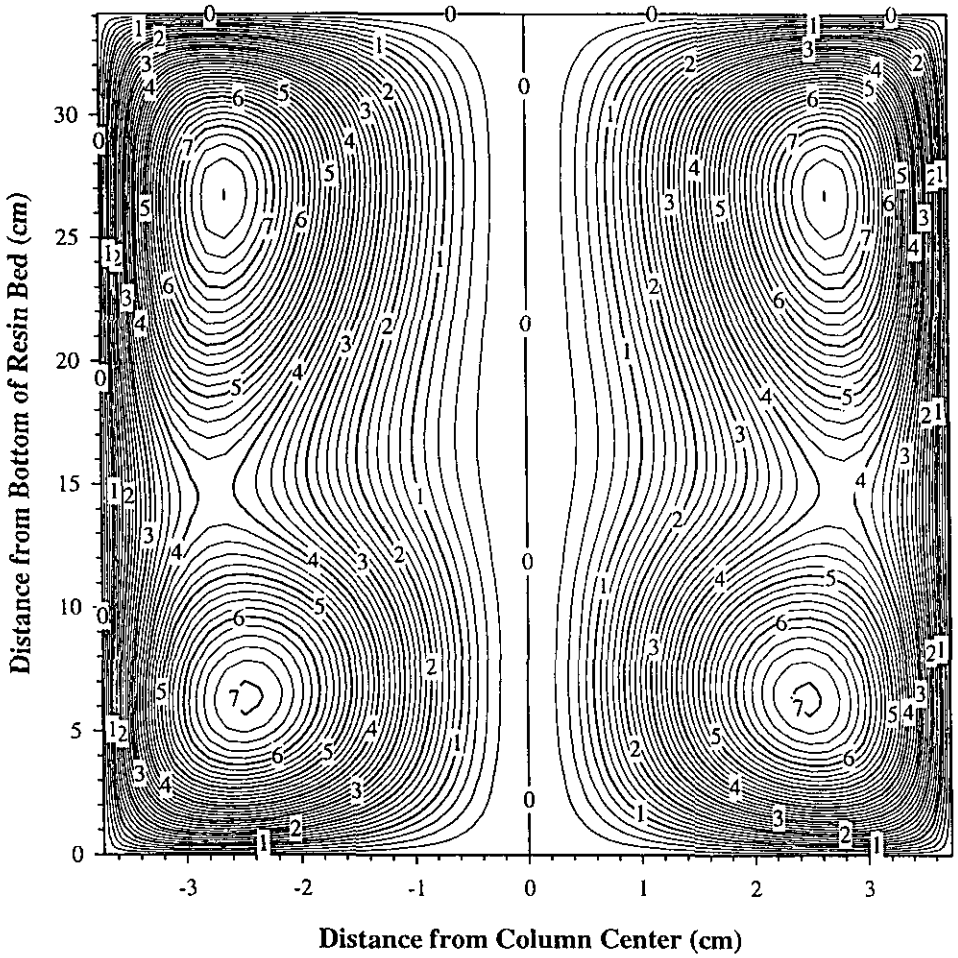

Figure 8. Flow Streamlines at Thermal Equilibrium after a Flow Stoppage for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 96 Axial and 24 Radial Cells. (Numbers on chart refer to values of stream function.)

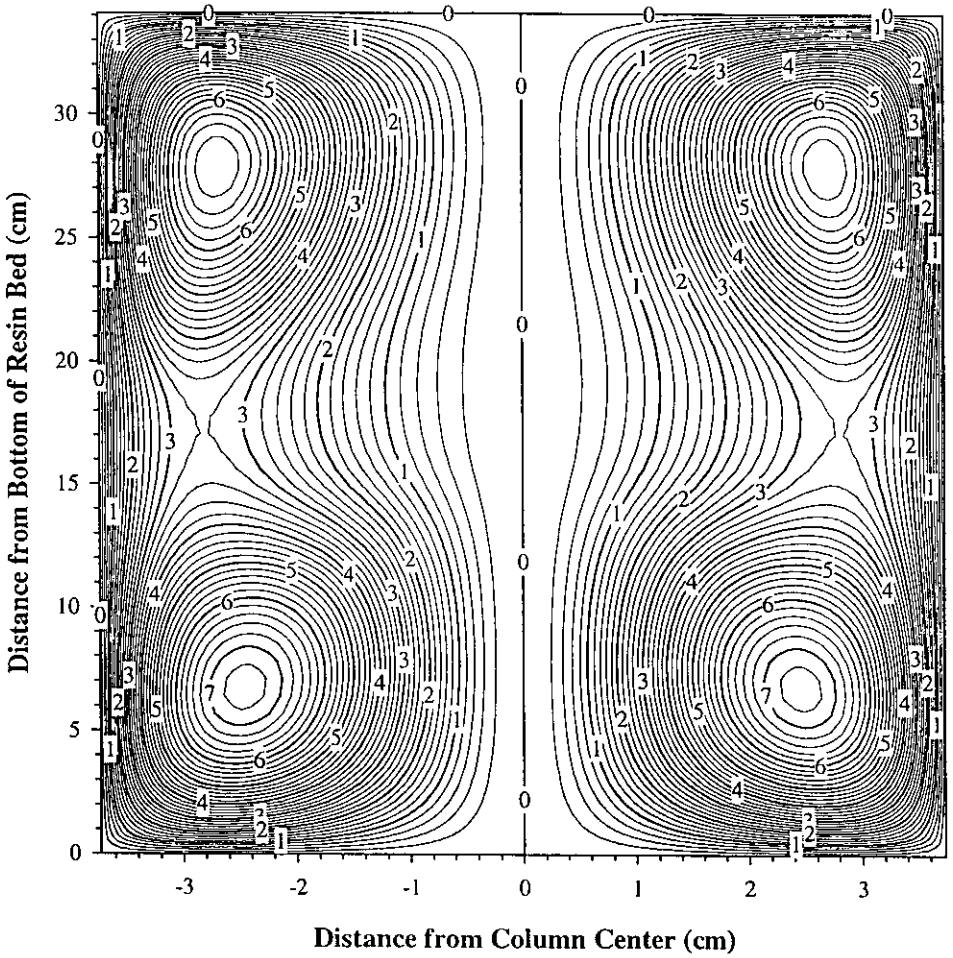

Figure 9. Flow Streamlines at Thermal Equilibrium after a Flow Stoppage for Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, for a Grid Resolution of 48 Axial and 48 Radial Cells. (Numbers on chart refer to values of stream function.)

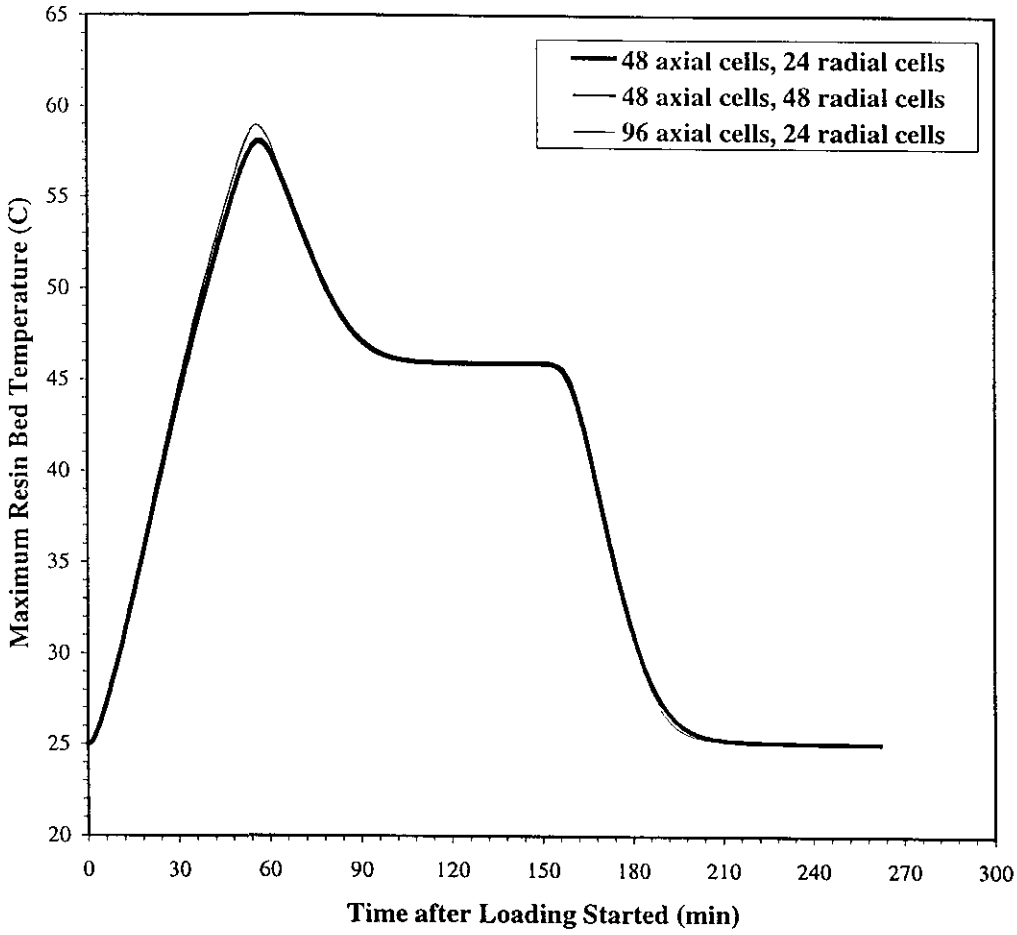

Figure 10. Effect of Finite Difference Cell Size on Maximum Resin Bed Temperatures during Normal Operation of the Column. (Conditions are low feed flow rate, high feed concentration, exposure to 25°C air, and twice the resin capacity fed.)

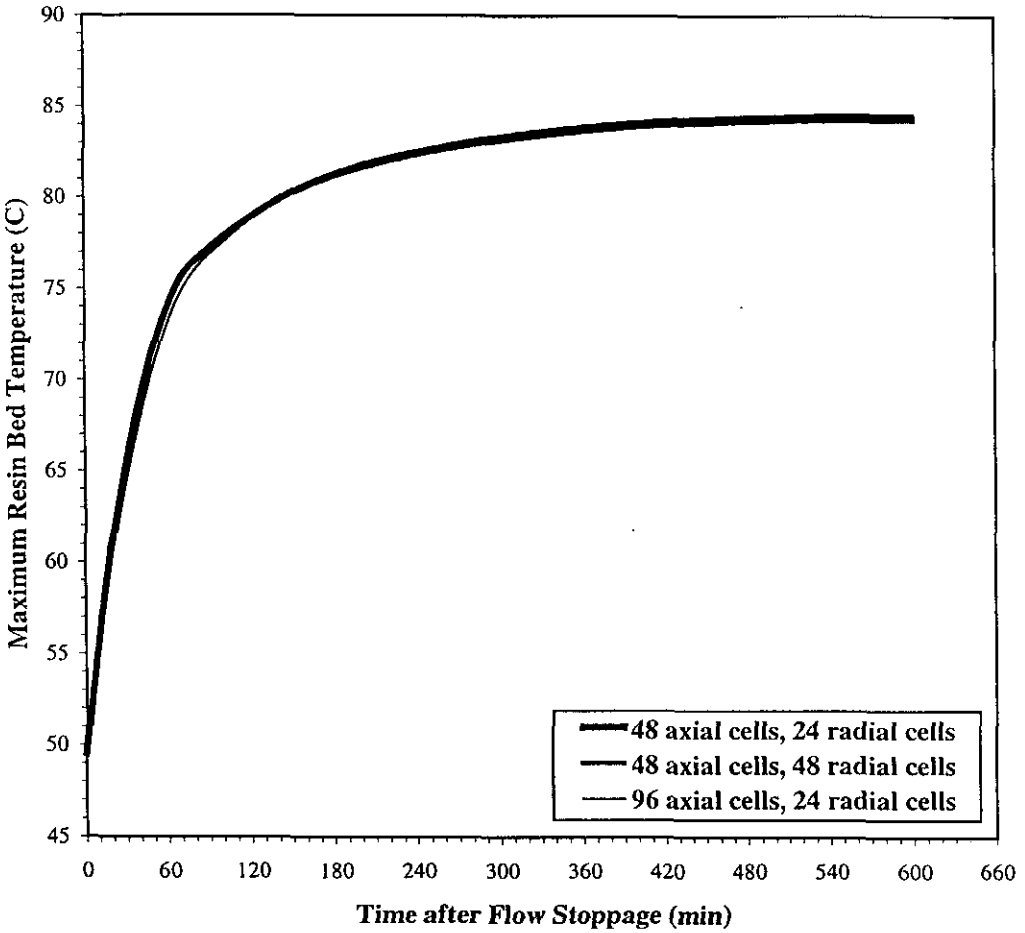

Figure 11. Effect of Finite Difference Cell Size on Maximum Resin Bed Temperatures during a Flow Stoppage. (Conditions are low feed flow rate, high feed concentration, exposure to 25°C air, and twice the resin capacity fed.)

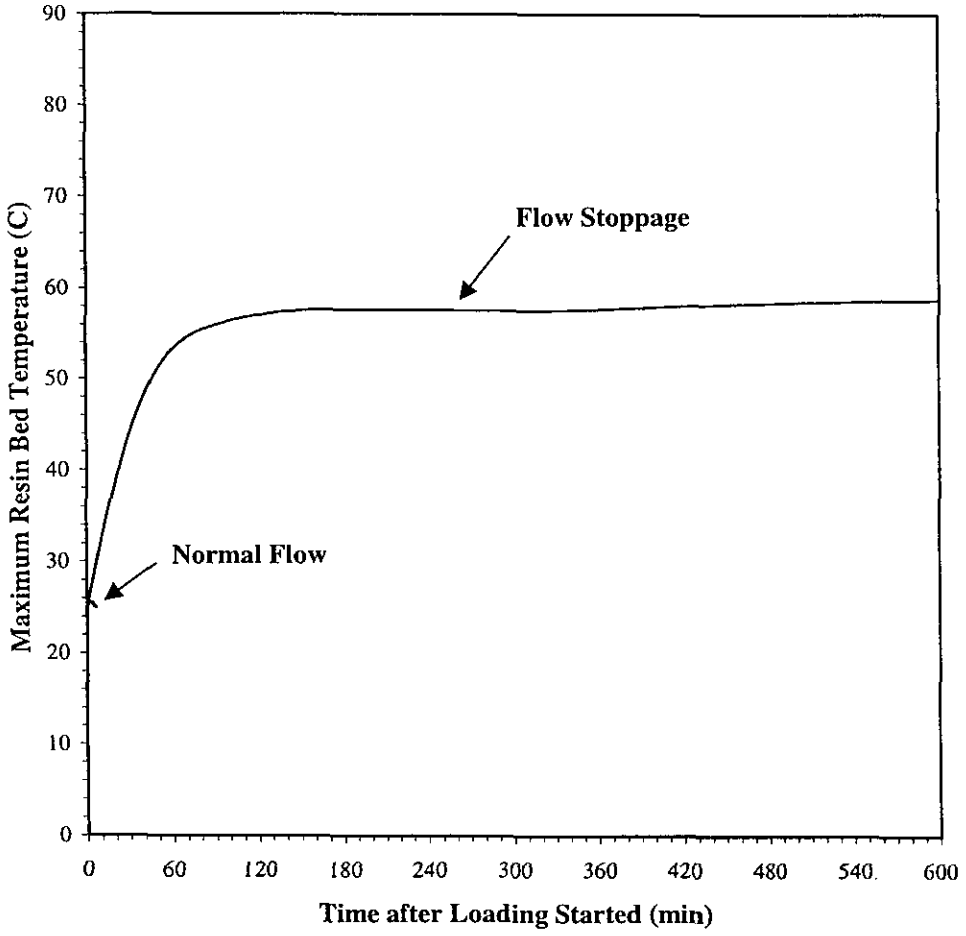

Figure 12. Maximum Resin Bed Temperatures for a High Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Stoichiometric Resin Capacity Fed.

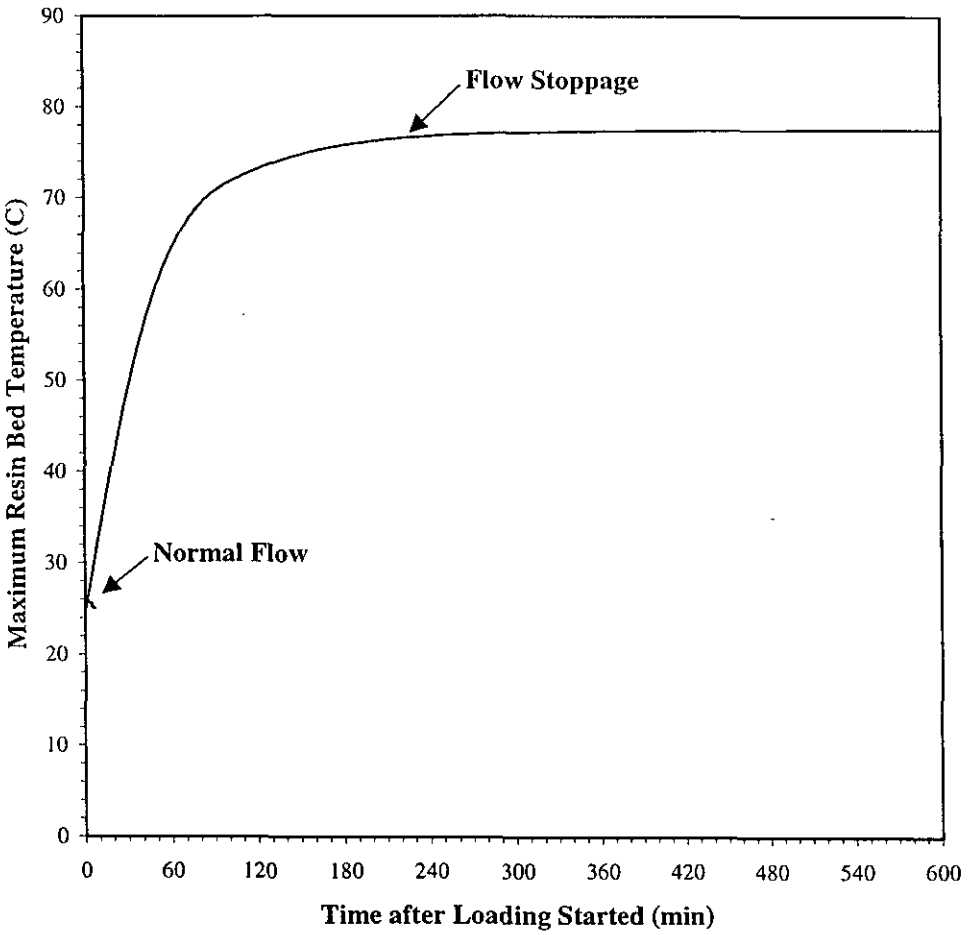

Figure 13. Maximum Resin Bed Temperatures for a High Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed.

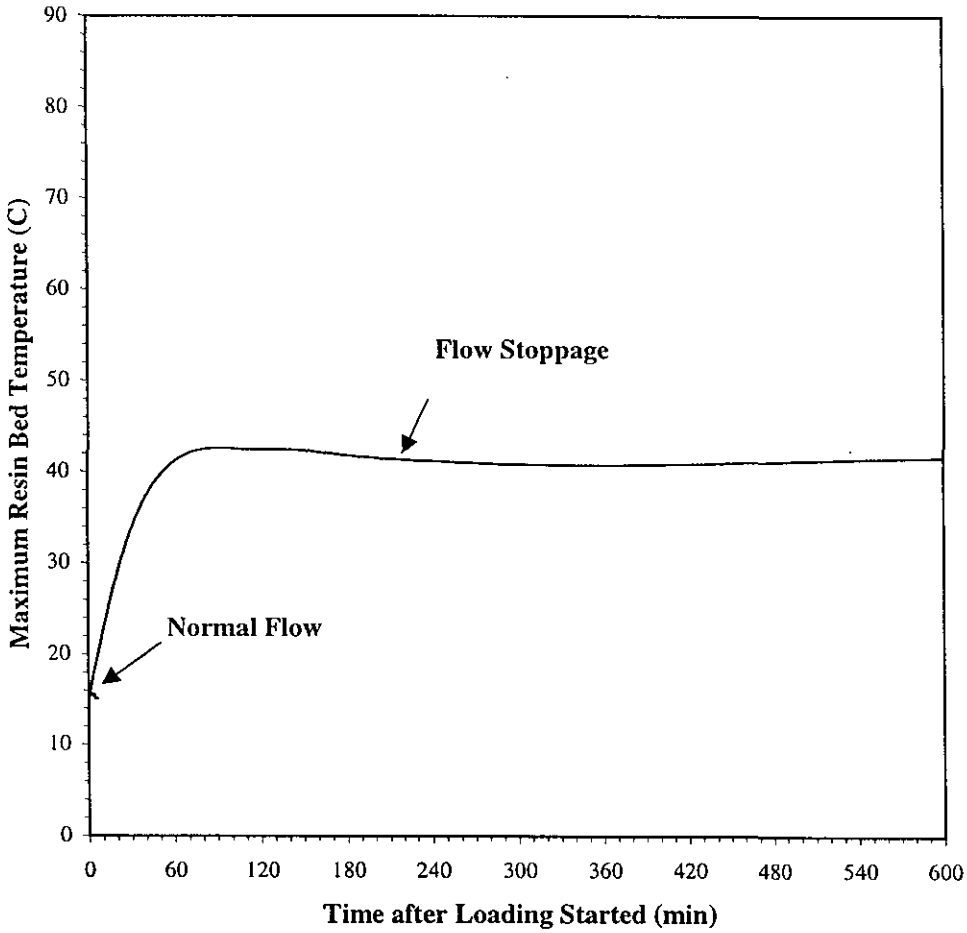

Figure 14. Maximum Resin Bed Temperatures for a High Feed Flow Rate, High Feed Concentration, Cooling by 15°C Water Jacket, and Stoichiometric Resin Capacity Fed.

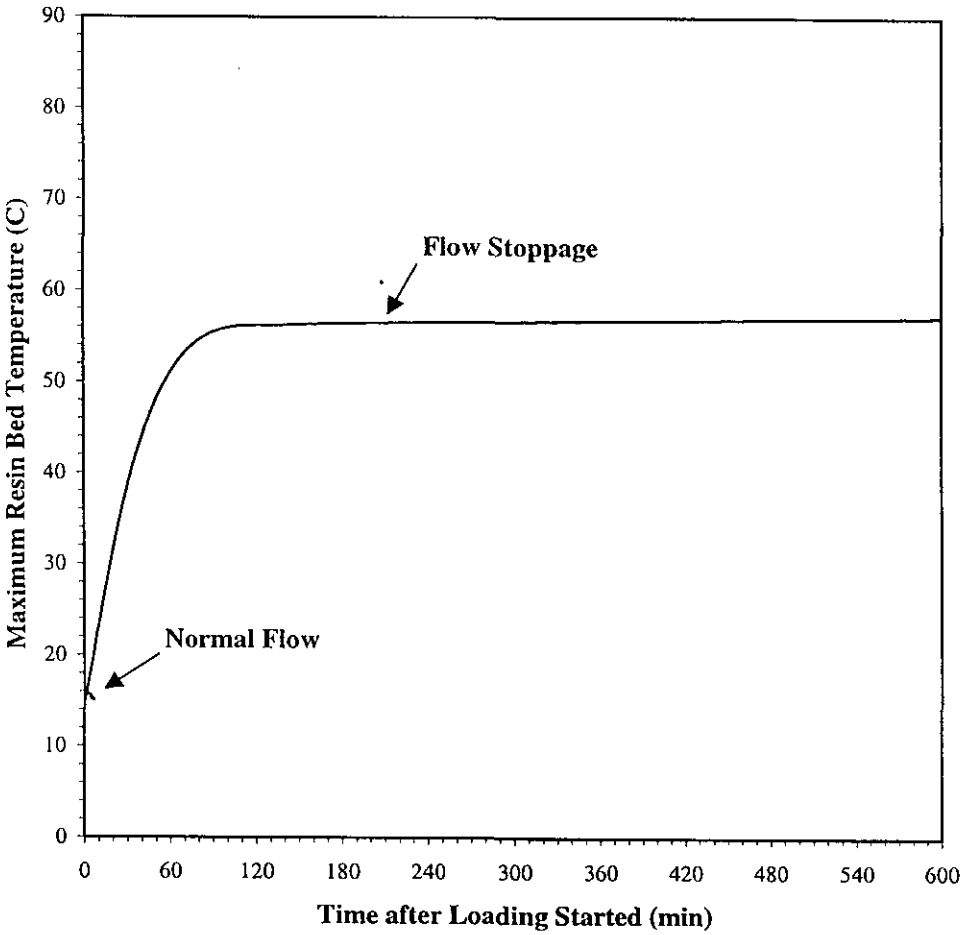

Figure 15. Maximum Resin Bed Temperatures for a High Feed Flow Rate, High Feed Concentration, Cooling by 15°C Water Jacket, and Twice the Resin Capacity Fed.

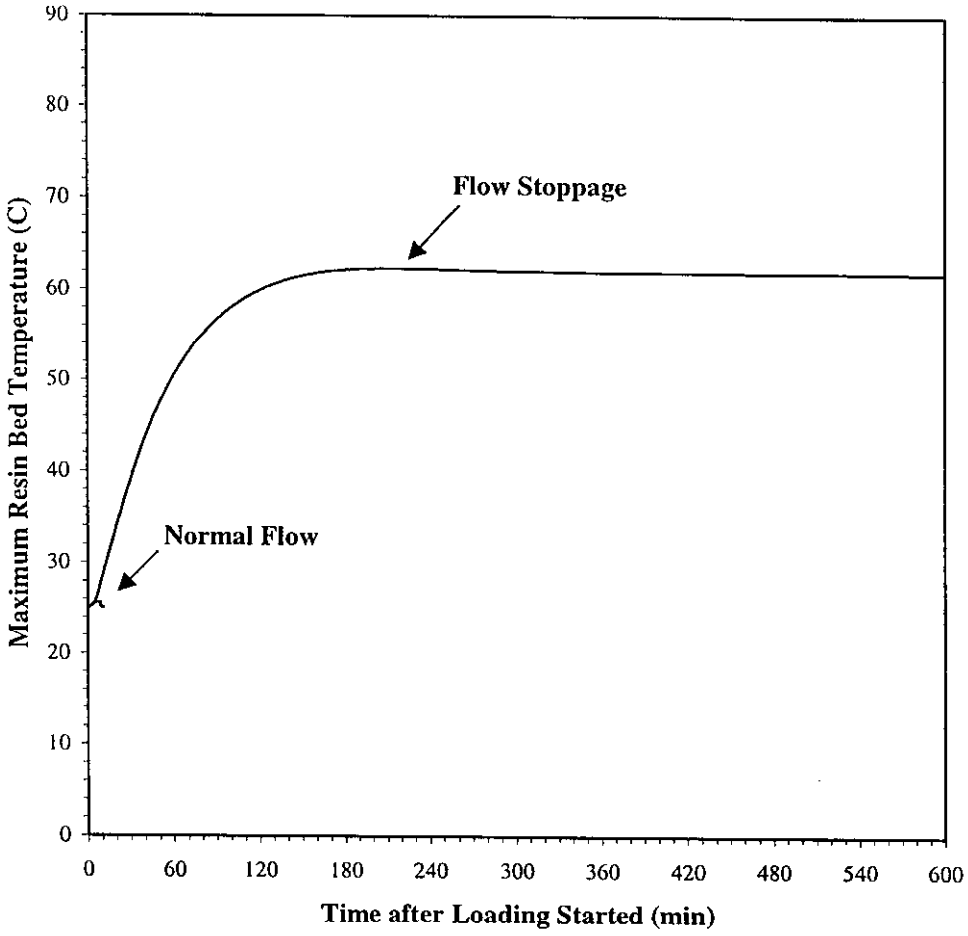

Figure 16. Maximum Resin Bed Temperatures for a High Feed Flow Rate, Low Feed Concentration, Exposure to 25°C Air, and Stoichiometric Resin Capacity Fed.

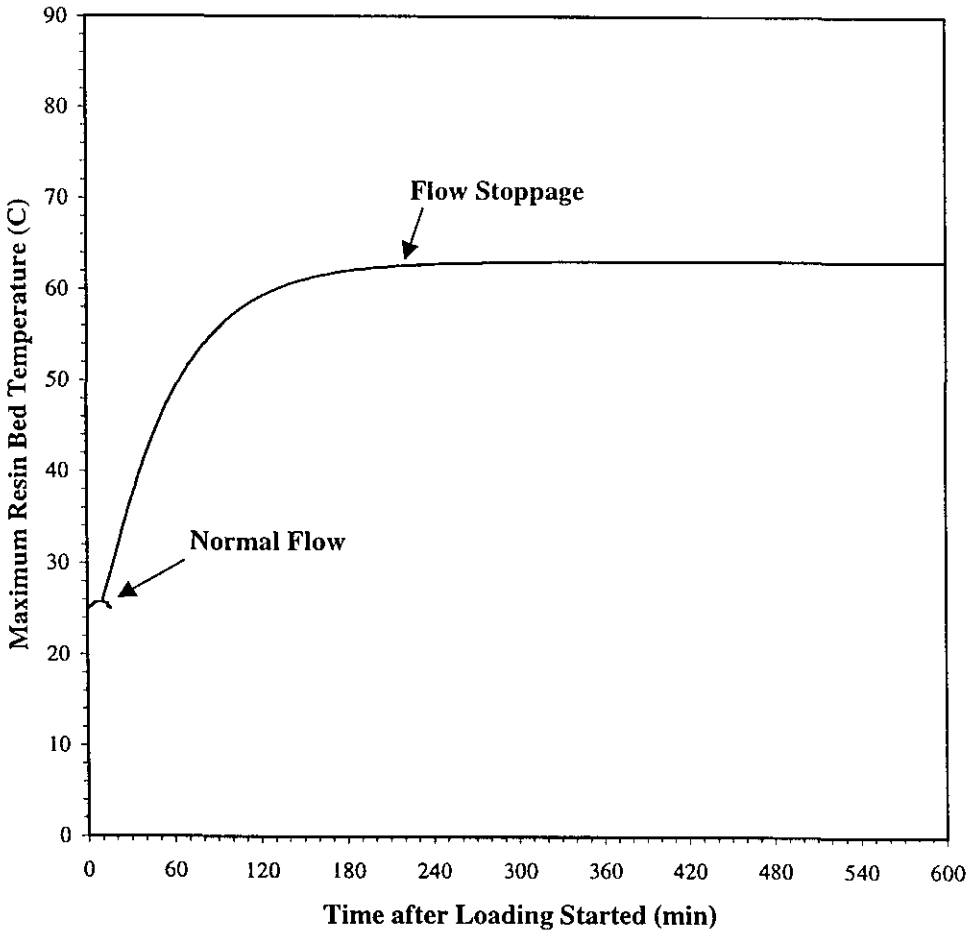

Figure 17. Maximum Resin Bed Temperatures for a High Feed Flow Rate, Low Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed.

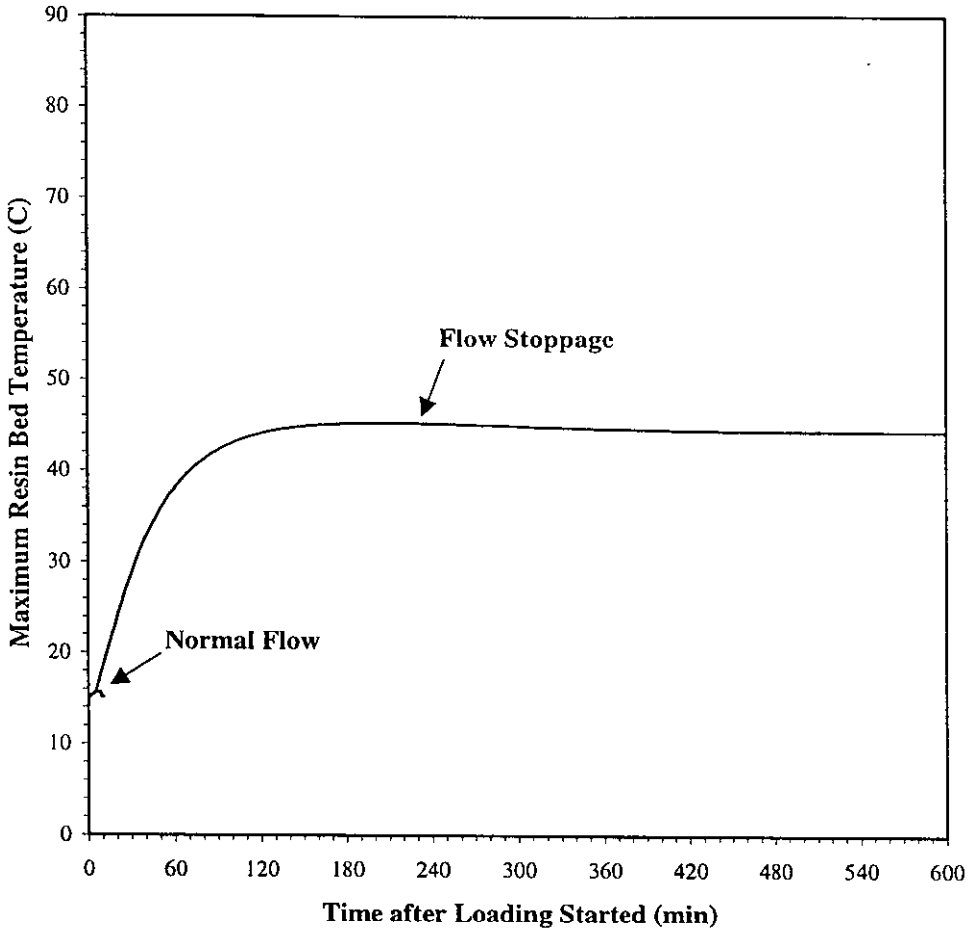

Figure 18. Maximum Resin Bed Temperatures for a High Feed Flow Rate, Low Feed Concentration, Cooling by 15°C Water Jacket, and Stoichiometric Resin Capacity Fed.

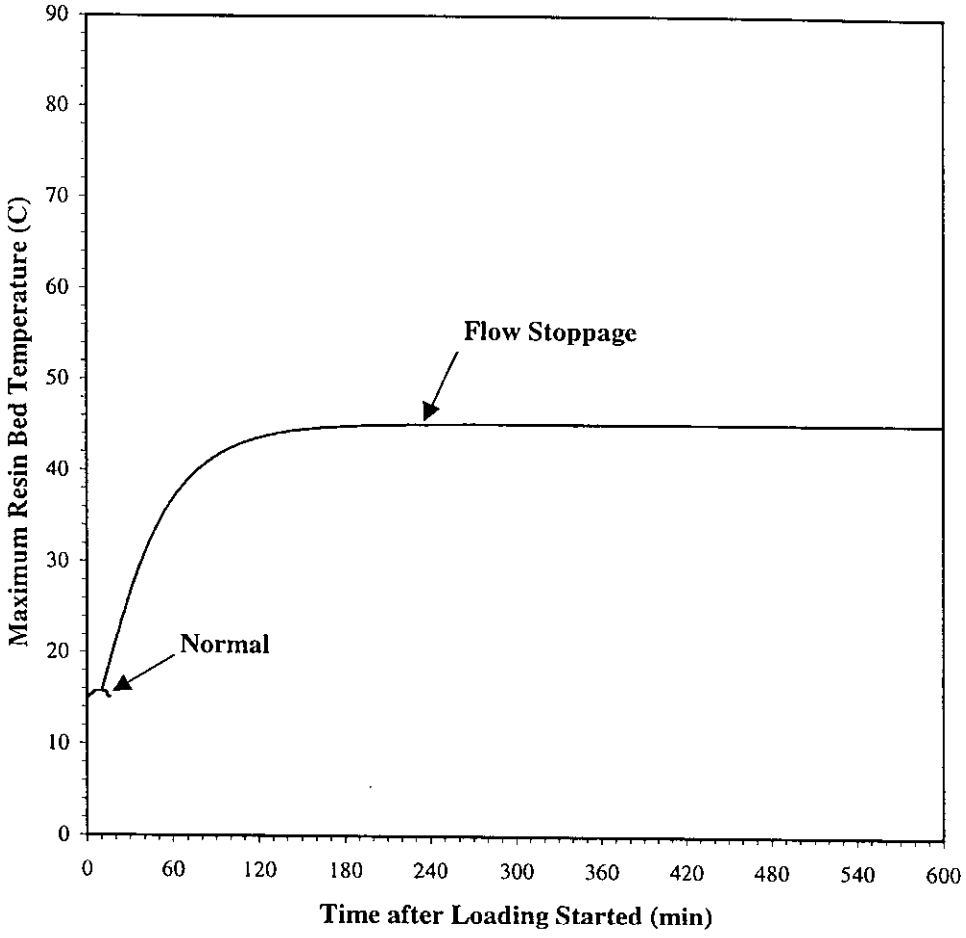

Figure 19. Maximum Resin Bed Temperatures for a High Feed Flow Rate, Low Feed Concentration, Cooling by 15°C Water Jacket, and Twice the Resin Capacity Fed.

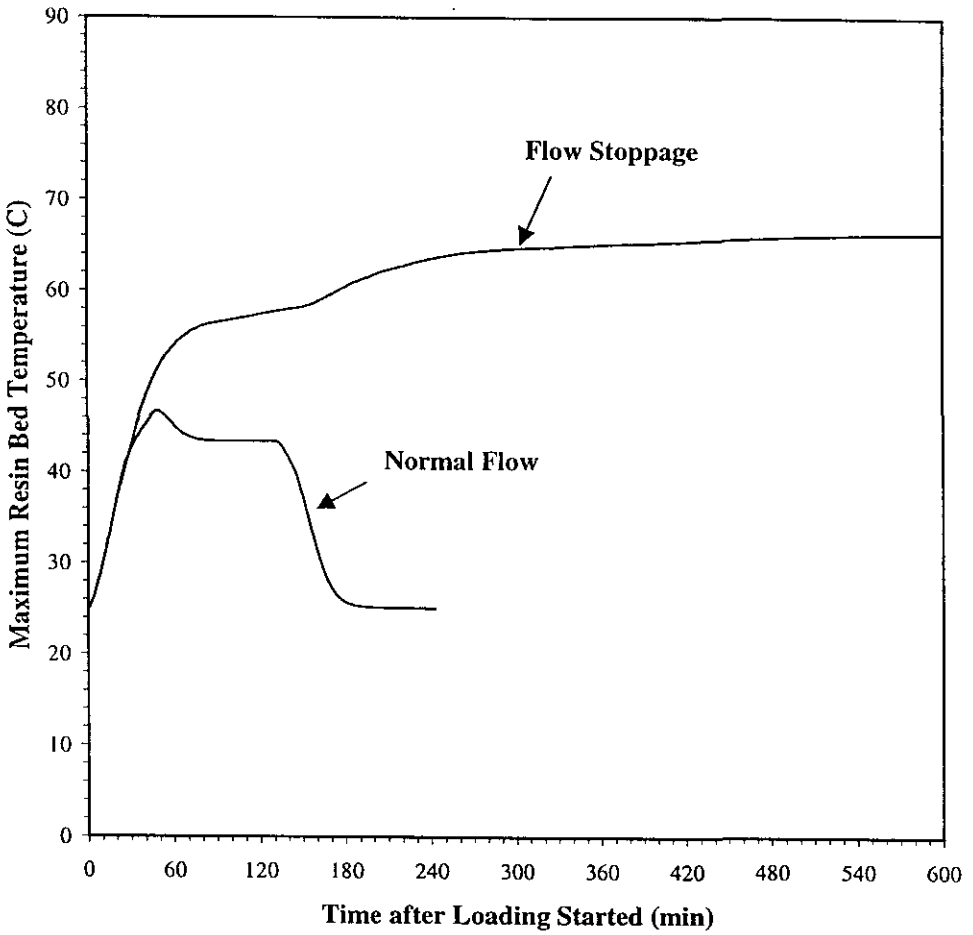

Figure 20. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Stoichiometric Resin Capacity Fed.

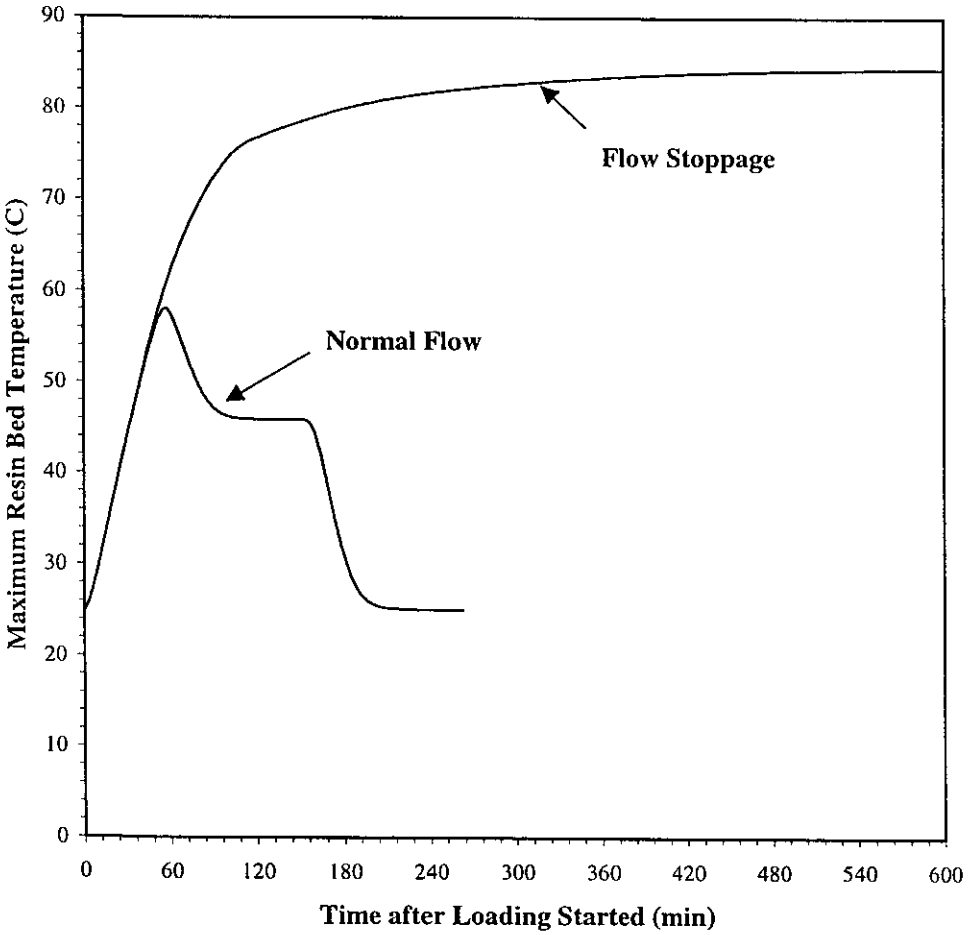

Figure 21. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed.

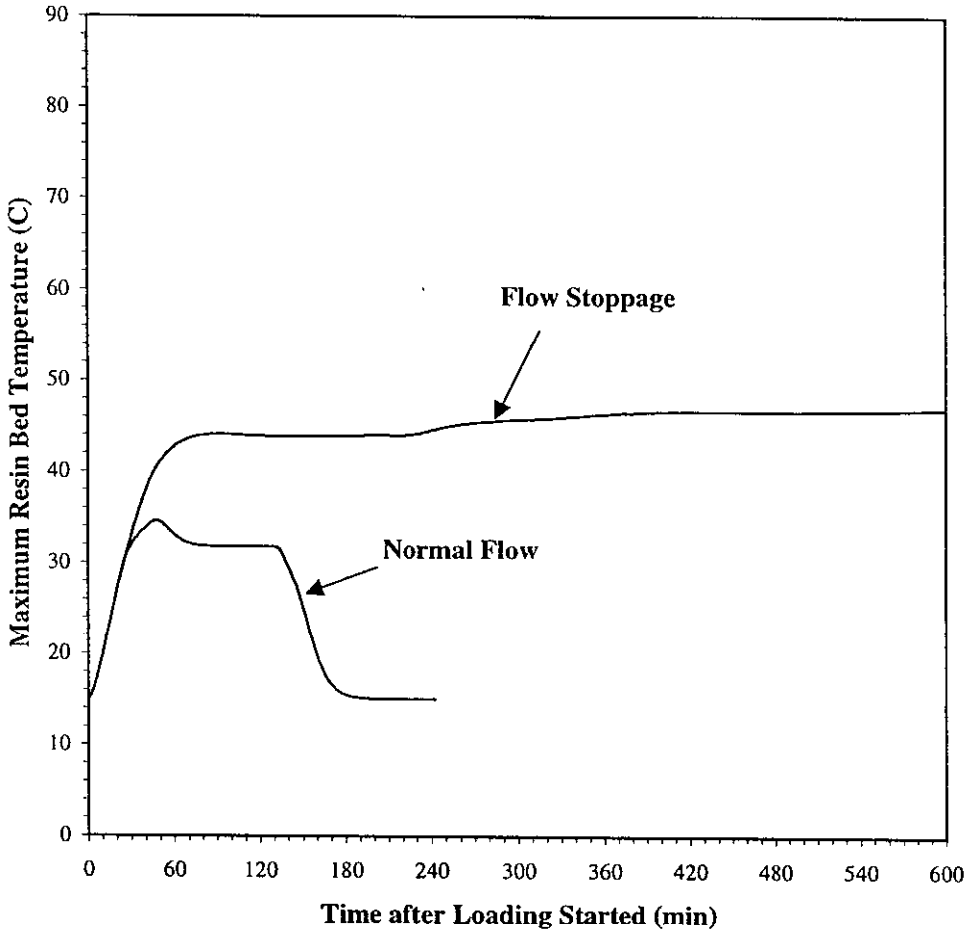

Figure 22. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Cooling by 15°C Water Jacket, and Stoichiometric Resin Capacity Fed.

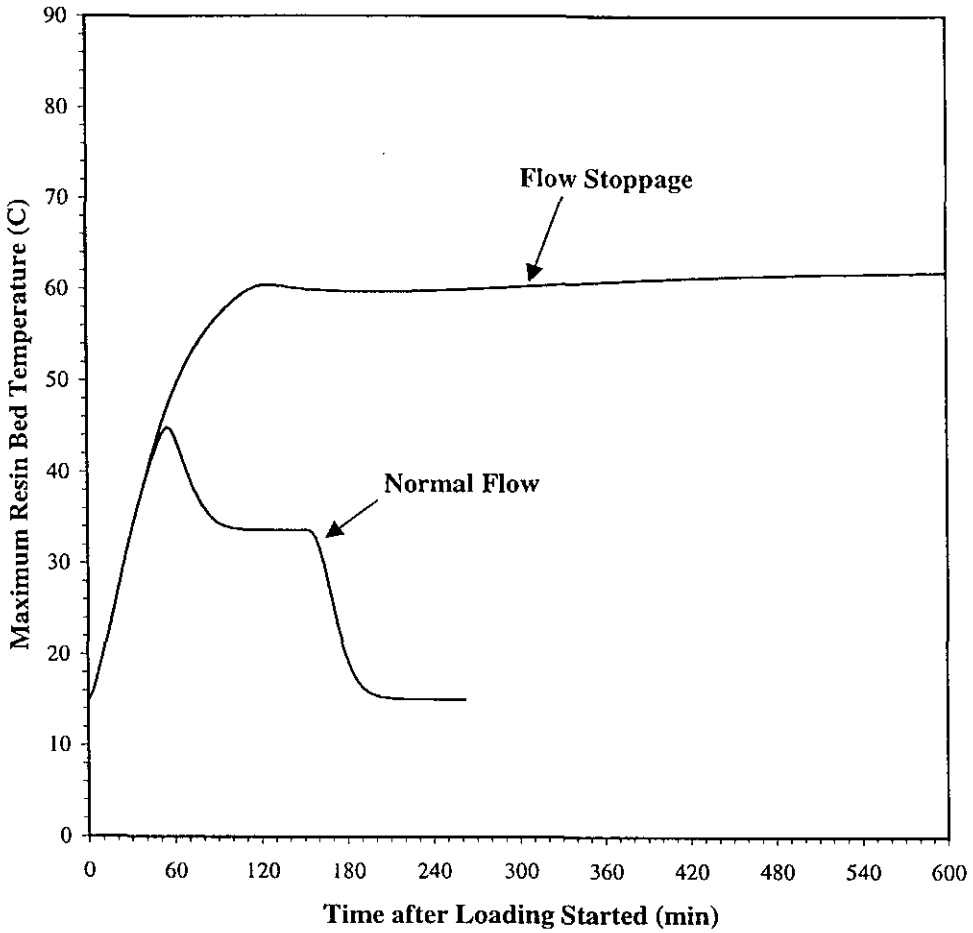

Figure 23. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Cooling by 15°C Water Jacket, and Twice the Resin Capacity Fed.

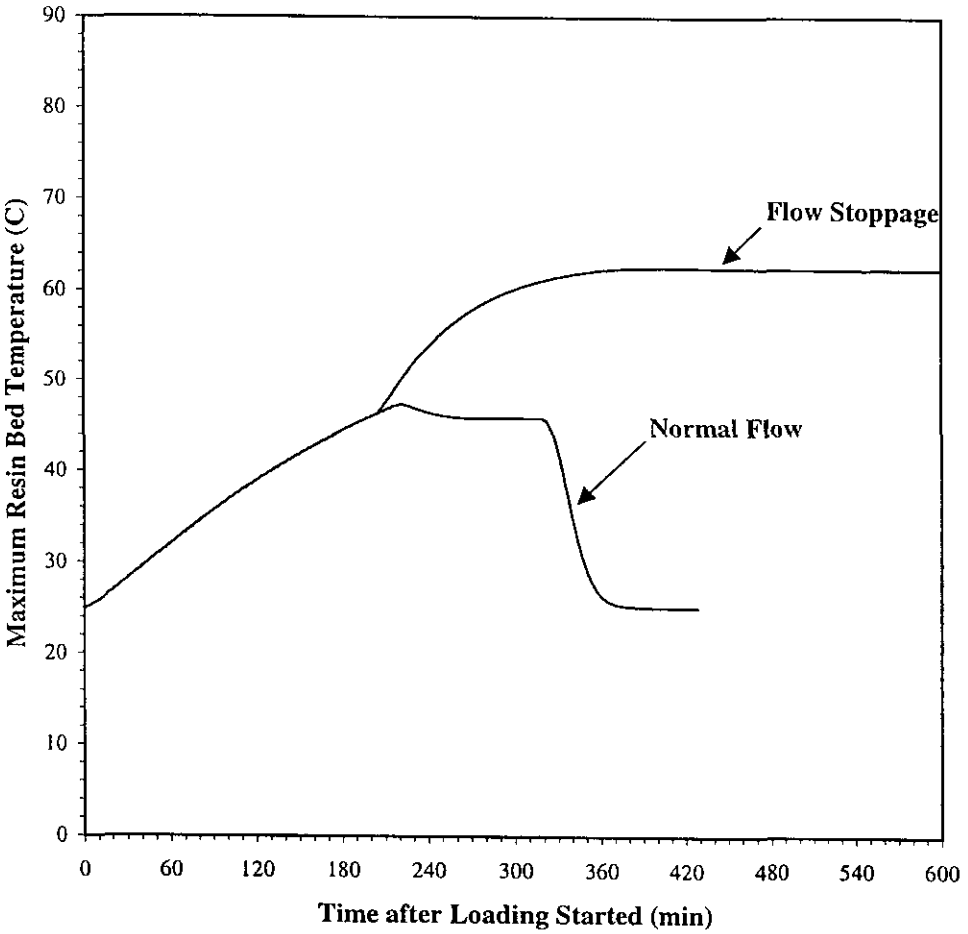

Figure 24. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, Low Feed Concentration, Exposure to 25°C Air, and Stoichiometric Resin Capacity Fed.

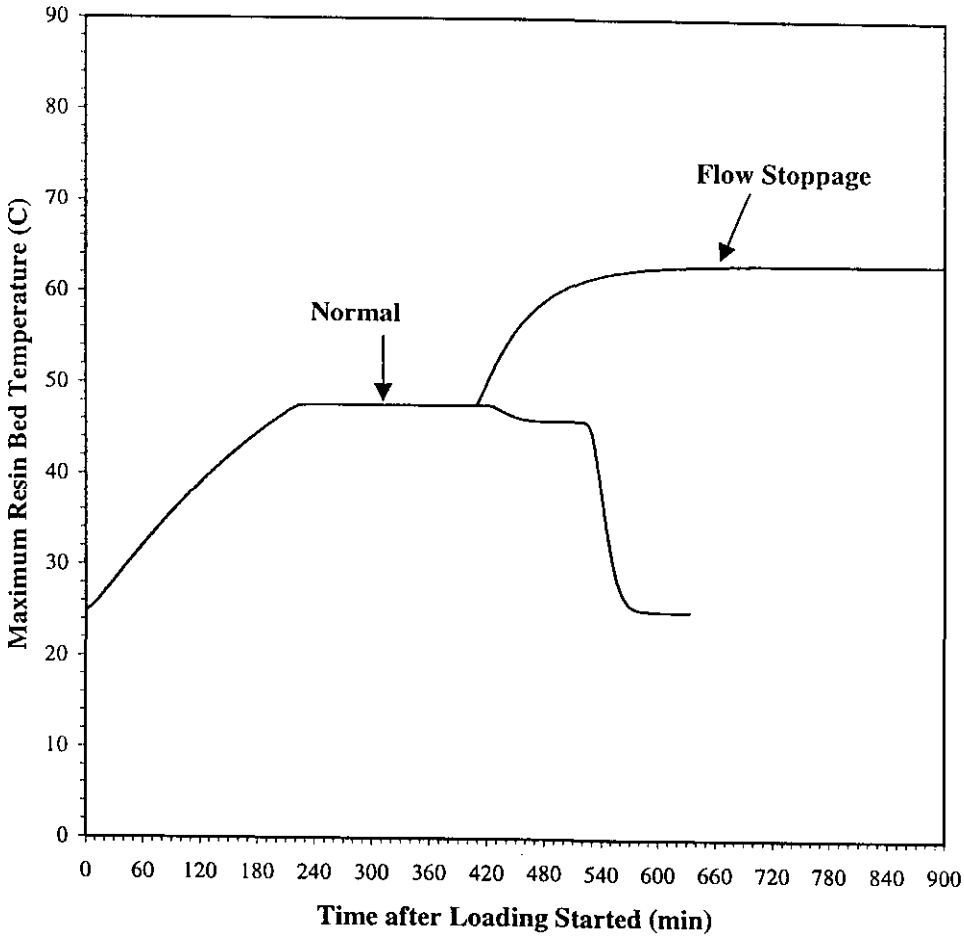

Figure 25. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, Low Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed.

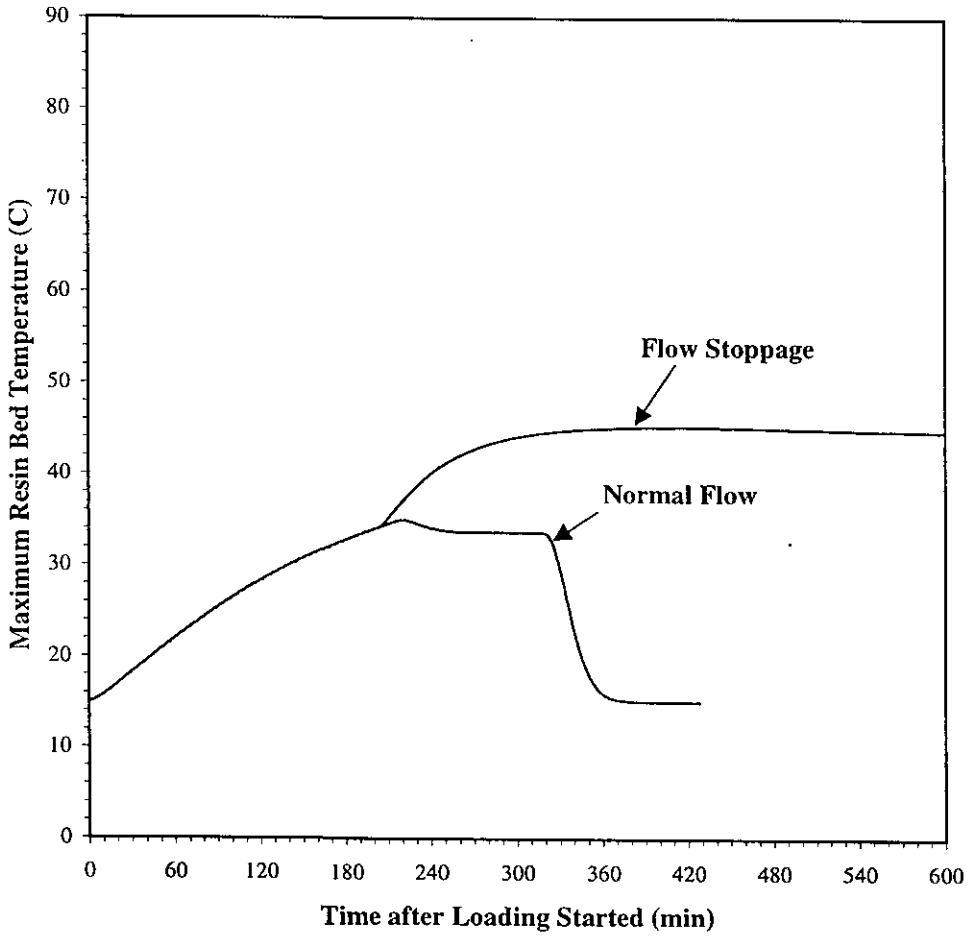

Figure 26. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, Low Feed Concentration, Cooling by 15°C Water Jacket, and Stoichiometric Resin Capacity Fed.

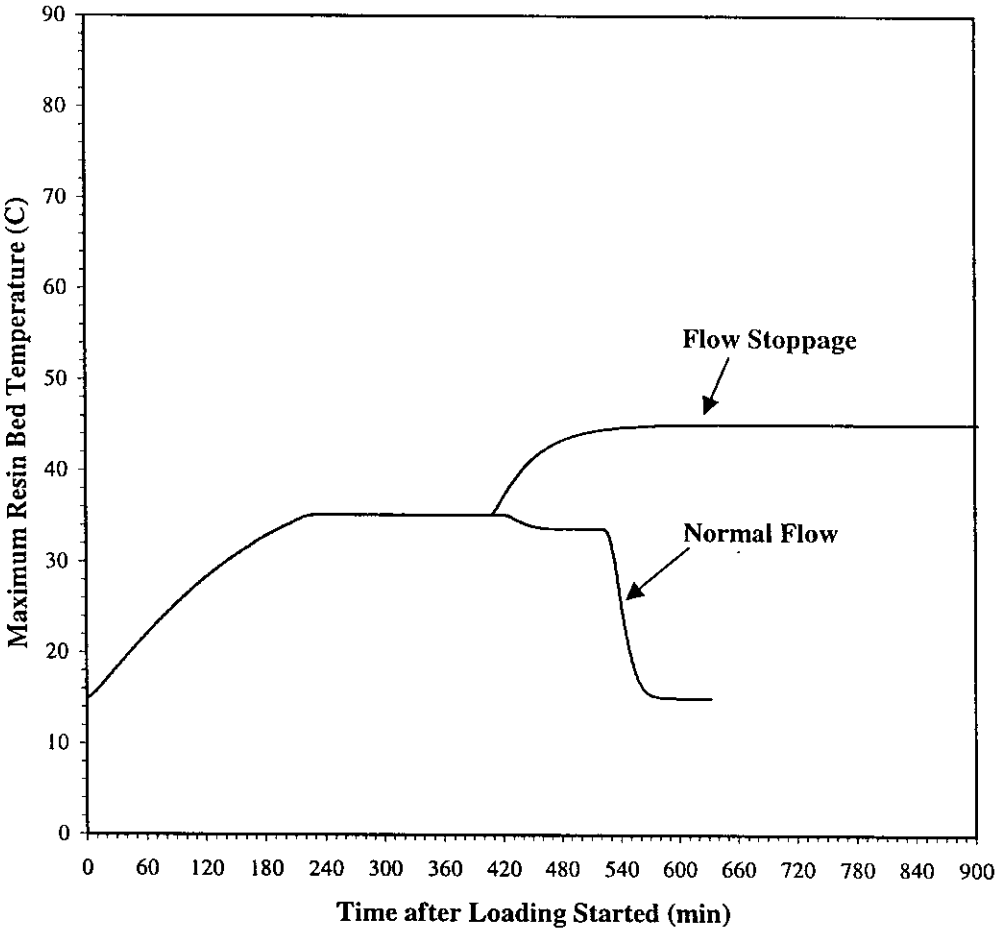

Figure 27. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, Low Feed Concentration, Cooling by 15°C Water Jacket, and Twice the Resin Capacity Fed.

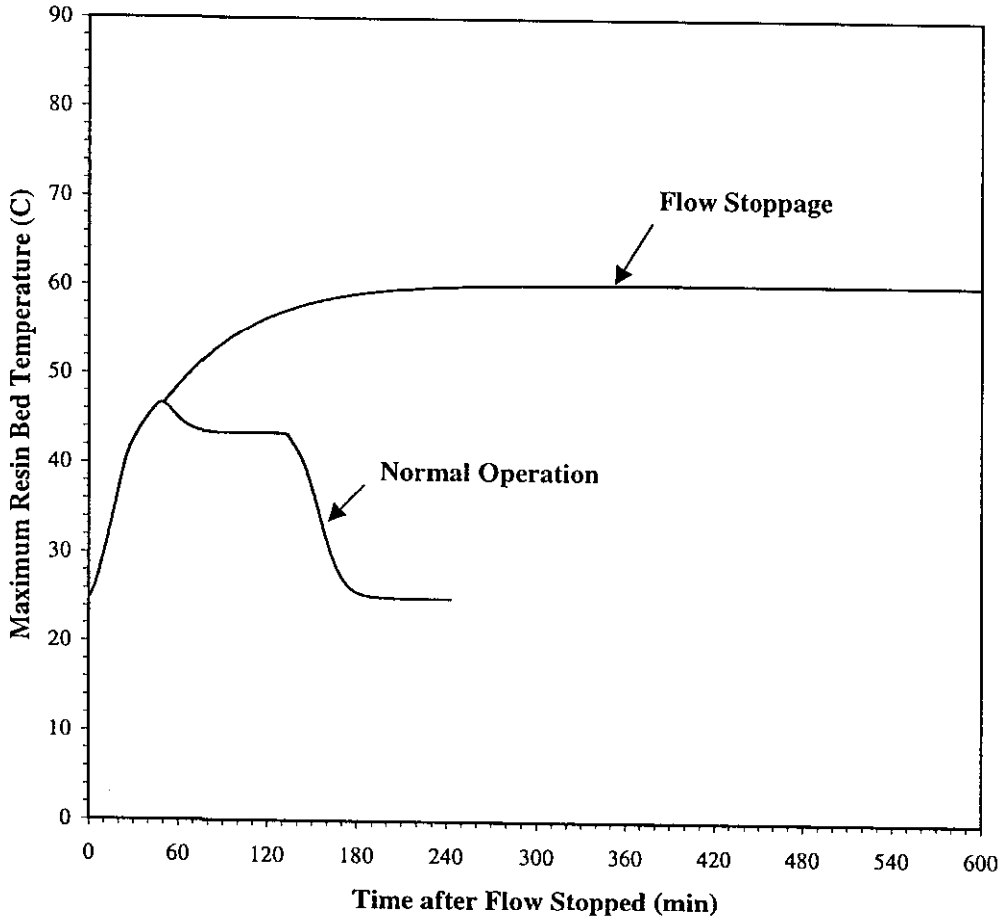

Figure 28. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Stoichiometric Resin Capacity Fed, with Stoppage of Flow at the Maximum Bed Temperature during Washing.

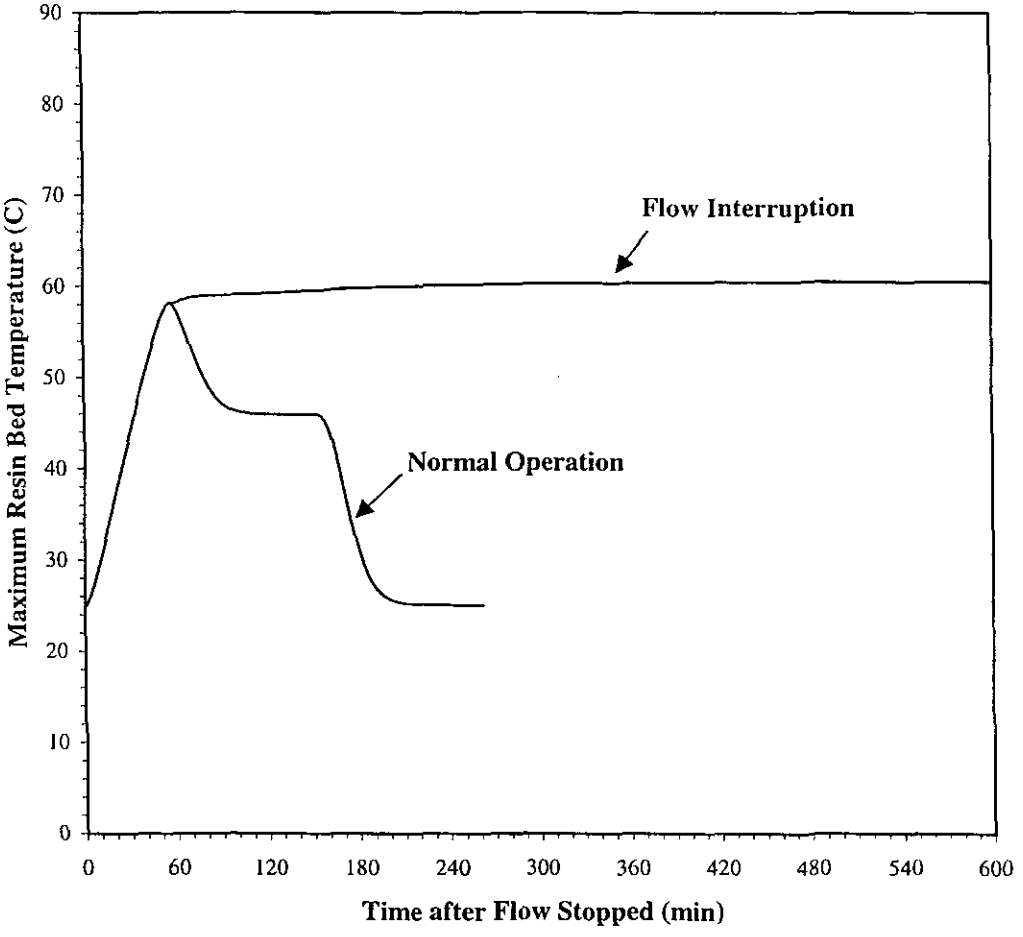

Figure 29. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Exposure to 25°C Air, and Twice the Resin Capacity Fed, with Stoppage of Flow at the Maximum Bed Temperature during Washing.

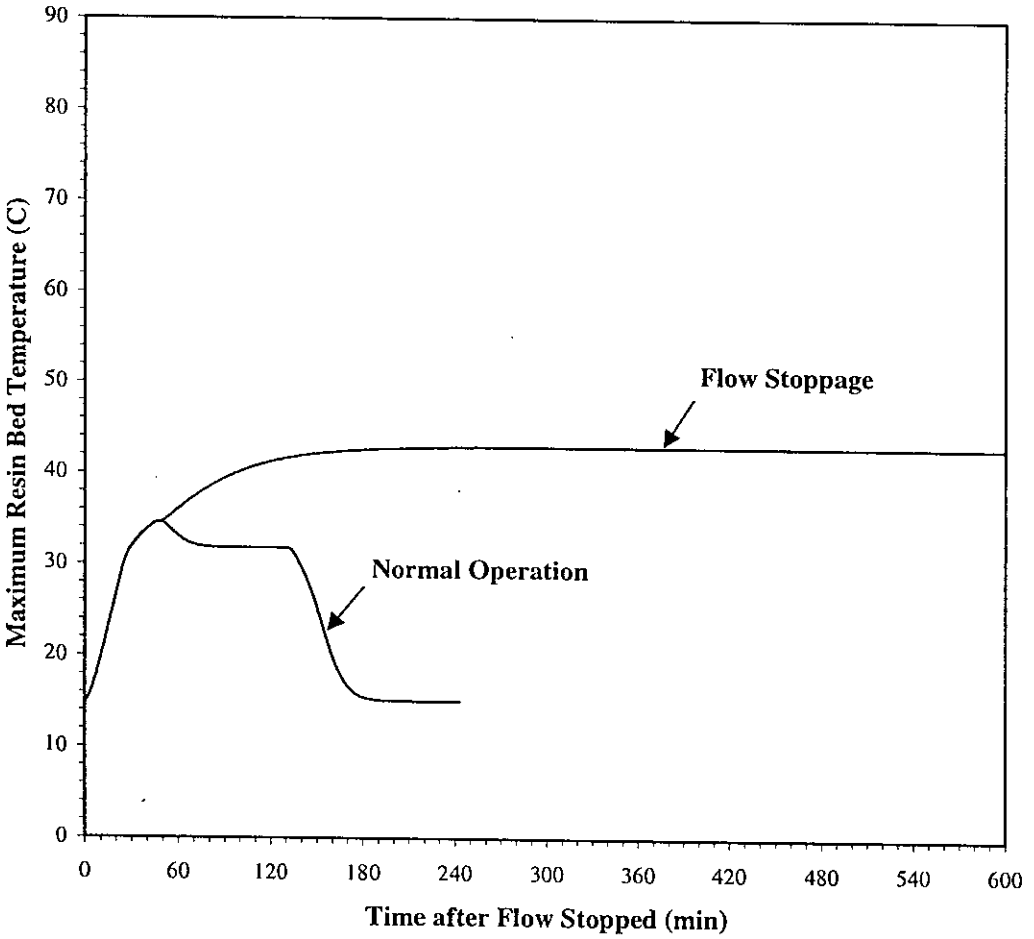

Figure 30. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Cooling by 15°C Water Jacket, and Stoichiometric Resin Capacity Fed, with Stoppage of Flow at the Maximum Bed Temperature during Washing.

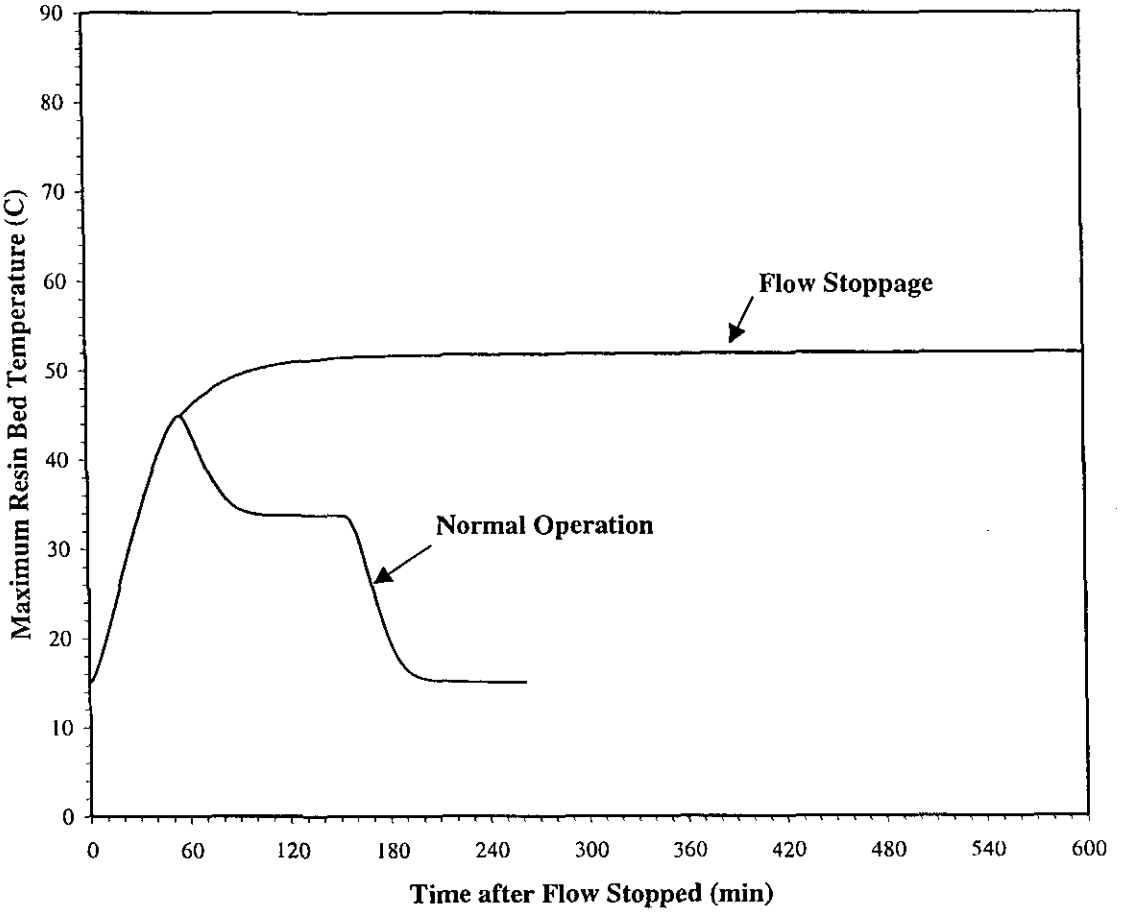

Figure 31. Maximum Resin Bed Temperatures for a Low Feed Flow Rate, High Feed Concentration, Cooling by 15°C Water Jacket, and Twice the Resin Capacity Fed, with Stoppage of Flow at the Maximum Bed Temperature during Washing.

Appendix: Source Code, Sample Input, and Sample Output Listings

This appendix contains the source code, sample input, and sample output listings for the LANL ion exchange column heat transfer analysis. Two sets of listings appear, one for calculations during normal operation, which includes loading, washing, and elution, and another assuming that the feed flow to the column has stopped. The calculations performed by these codes are described in the text. In addition, both codes include comments that give definitions of variables and outline the code structure. The sample input and output listings are for the case with a high flow rate, a high plutonium feed concentration, exposure to ambient air at 25°C, and addition of a stoichiometric amount of plutonium to the column. The output files that contain the maximum temperature transients appear; other output files are not shown.

The source code for normal operation is named FEDFLO.FOR, and the source code for a flow stoppage is called NATCON.FOR. Both are stored in Vax subdirectory [t6930.hcolumn].

Source Code for Thermal Analysis during Normal Operation

```
c This program calculates the temperature transient during elution of an ion
c exchange column.  It includes the effects of thermal conduction to the column
c walls and forced convection heat and mass transfer.  The effect of plutonium
c and nitric acid on the bulk solution density is included.  The program solves
c finite difference energy balance equations with forward time differencing,
c donor cell differencing for convection terms, and centered space differencing
c for conduction terms.  A list of variable definitions follows.
c
c aa = one-dimensional matrix coefficient for heat transfer equation
c aa1 = two-dimensional matrix coefficient for heat transfer equation
c alpha = thermal diffusivity, cm2/s
c an1 = fraction of void volume that lies between resin beads
c area = cross-sectional area of resin bed, cm2
c bb = one-dimensional matrix coefficient for heat transfer equation
c bb1 = two-dimensional matrix coefficient for heat transfer equation
c br = nondimensional radial dispersion coefficient for heat transfer equation
c br1 = nondimensional radial dispersion coefficient for heat transfer
equation,
c       due to area change with radius
c bz = nondimensional axial dispersion coefficient for heat transfer equation
c c = volume fraction acid in solution between resin beads
c caps = heat capacity of liquid, cal/g/C
c capsb = heat capacity of liquid in volume below resin bed, cal/g/C
c capst = heat capacity of liquid in volume above resin bed, cal/g/C
c cbot = volume fraction acid in liquid volume below resin bed
c cc = one-dimensional matrix coefficient for heat transfer equation
c cc1 = two-dimensional matrix coefficient for heat transfer equation
c cfeed = plutonium concentration in feed solution, g/cc
c cf30 = volume fraction occupied by HNO3 in entering feed solution
c cno3 = nitric acid concentration, gmole/lit
c cno3b = nitric acid concentration in liquid volume below resin bed, gmole/lit
c cno3t = nitric acid concentration in liquid volume above resin bed, gmole/lit
c cold = volume fraction acid in solution between resin beads at previous time
c        step
c cp = heat capacity, cal/g/C
c cpair = heat capacity of ambient air, cal/g/C
```

```
c cond = thermal conductivity
c c1feed = plutonium concentration in liquid volume above resin bed, g/cc
c den1 = density of pure HNO3, g/cc
c den1b = density of pure HNO3 in liquid volume below resin bed, g/cc
c den1t = density of pure HNO3 in liquid volume above resin bed, g/cc
c den2 = density of pure H2O, g/cc
c den2b = density of pure H2O in liquid volume below resin bed, g/cc
c den2t = density of pure H2O in liquid volume above resin bed, g/cc
c dens = liquid density, g/cc
c densb = liquid density in volume below resin bed, g/cc
c denst = liquid density in volume above resin bed, g/cc
c dfpor = pore diffusivity, cm2/s
c dfprt = particle diffusivity, cm2/s
c df = change in adsorbed plutonium concentration during feeding and elution,
c      g/cc
c dg = change in plutonium concentration in pore solution due to elution, g/cc
c dpart = resin bead diameter, cm
c dr = radial discretization step size, cm
c dt = time step, s
c dz = axial discretization step size, cm
c dy2 = change in plutonium concentration in solution between resin beads
c       during feeding and elution, g/cc
c emiss = surface emissivity for column
c f = plutonium loading on resin, g/cc
c fcx = weighting factor for elution of plutonium by acid dilution
c ffcx = weighting factor for elution of plutonium by acid dilution
c ffeed = maximum plutonium resin bed concentration, g/cc
c fold = plutonium loading on resin at previous time step, g/cc
c fx = total bulk plutonium concentration, g/cc
c grav = gravitational acceleration, cm/s2
c hconv = coefficient for natural convection heat transfer to sides of column,
c         also overall convective heat transfer coefficient, cal/cm2/s/C
c hconv2 = coefficient for natural convection heat transfer to top and bottom
c          of column, cal/cm2/s/C
c hconv3 = coefficient for forced convection heat transfer to column, assuming
c          an ambient air velocity of 30 cm/s (1 ft/s)
c hrad = radiation heat transfer coefficient, cal/cm2/s/C
c htc = heat transfer coefficient, cal/cm2/s/C
c htc2b = heat transfer coefficient for the liquid volume below the resin bed,
c         cal/cm2/s/C
c htc2t = heat transfer coefficient for the liquid volume above the resin bed,
c         cal/cm2/s/C
c iht = indicator for heat transfer to ambient air (iht=1) or to water-cooled
c       jacket (iht=2) (A heat transfer coefficient is calculated for heat
c       transfer to ambient air, and a surface temperature is specified for
c       cooling by a water jacket.)
c ind2 = indicator for stage of process (1 = loading, 2 = washing, 3 = elution)
c iss = indicator that switches between implicit radial, explicit axial (iss=0)
c       and explicit radial, implicit axial (iss=1) finite differencing
c ldown = depth of liquid in column below resin bed, cm
c lup = height of liquid in column above resin bed, cm
c length = column length, cm
c m = number of axial discretization nodes
c n = number of radial discretization nodes
c nu = Nusselt number for natural convection heat transfer to ambient air from
c      sides of column
c nu2 = Nusselt number for natural convection heat transfer to ambient air from
c       top and bottom of column
c nu3 = Nusselt number for forced convection heat transfer to ambient air,
c       assuming air velocity of 30 cm/s (1 ft/s)
c phi = updated temperature, K
c poszn = distance from top of resin bed, cm
c pr = Prandtl number for ambient air
```

```
c pua = total amount of plutonium adsorbed on the resin beads, g
c puf = total amount of plutonium in the column, g
c put = amount of plutonium fed to the column, g
c radius = column radius, cm
c rayl = Rayleigh number for natural convection heat transfer to ambient air
c        from sides of column
c rayl2 = Rayleigh number for natural convection heat transfer to ambient air
c         from top and bottom of column
c re = Reynolds number for forced convection heat transfer to column, assuming
c      an ambient air velocity of 30 cm/s (1 ft/s)
c rhcp = product of density and heat capacity for bulk resin bed, cal/cc/C
c rhcp2 = product of density and heat capacity for solution between
c         resin beads, cal/cc/C
c rhcp2b = product of density and heat capacity for liquid volume below resin
c          bed, cal/cc/C
c rhcp2t = product of density and heat capacity for liquid volume above resin
c          bed, cal/cc/C
c rho = bulk density, g/cc
c rhoair = density of air at resin column surface, g/cc
c rhoamb = density of ambient air, g/cc
c rr = right-hand side term of one-dimensional matrix heat transfer equation, K
c rr1 = right-hand side term of two-dimensional matrix heat transfer equation,
K
c rra = radial position of calculation node, cm
c rrn = radial position, cm
c rro = radial position, cm
c rrr = radial position, cm
c r1 = radial position, cm
c r2 = radial position, cm
c qdot = heat production rate due to plutonium loading, cal/cc/s
c sigma = Boltzmann's constant, cal/cm2/s/K4
c stem1 = temperature in the liquid volume above the resin bed, K
c stem1a = change in temperature in the liquid volume above the resin bed, K
c stem1c = temperature in the liquid volume above the resin bed, C
c stem2 = temperature in the liquid volume below the resin bed, K
c stem2a = change in temperature in the liquid volume below the resin bed, K
c stem2c = temperature in the liquid volume below the resin bed, C
c stp = total number of calculation time steps
c sumf = total loading of plutonium on resin, gm
c svz2 = integrated volumetric flow rate out column, cc/s
c tamb = ambient temperature, K
c tau = time constant for loading plutonium on resin, s
c tau2 = time constant for eluting plutonium from resin, s
c tcn1 = thermal conductivity for nitric acid, cal/cm/s/C
c tcn2 = thermal conductivity for water, cal/cm/s/C
c tcnm = thermal conductivity for bulk resin bed, cal/cm/s/C
c telut = time for elution, s
c temb = product of volumetric flow rate out of resin bed and temperature at
c        bottom of resin bed, K-cc/s
c temch = maximum temperature in column, C
c temc = temperature, C
c temp = temperature, K
c time = elapsed time after stage of process started, m
c time2 = elapsed time after feeding started, s
c tinte = time interval for printing maximum resin bed temperatures, s
c tinte2 = time for printing bed temperature profiles, s
c tload = time for loading, s
c trpt = time to print out temperature matrix, s
c twash = time for washing, s
c tzero = initial temperature, K
c tzero = temperature of solution entering column, K
c uu = vector containing solution to one-dimensional heat transfer equation, K
c vbot = volume of liquid below resin bed, cc
```

```
c vel2 = velocity of ambient air, cm/s
c vfi = volume fraction in resin bead pores, prior to adsorption of plutonium
c vfi2 = volume fraction in resin bead pores, after adsorption of plutonium
c vfi3 = volume fraction in resin bead pores, adjusted for actual amount of
c        plutonium adsorbed
c vfo = volume fraction occupied by flowing solution
c vis = liquid dynamic viscosity, g/cm/s
c visair = dynamic viscosity of ambient air, g/cm/s
c vmid = total volume of resin bed, cc
c vr = superficial radial velocity, cm/s
c vtop = volume of liquid above resin bed, cc
c vz = superficial axial velocity, cm/s
c x = fractional loading of plutonium on resin
c xg = elution rate constant, 1/s
c xkair = thermal conductivity of ambient air, cal/cm/s/C
c xx1 = mass fraction nitric acid in the solution between resin beads
c xx1b = mass fraction nitric acid in liquid volume below resin bed
c xx1t = mass fraction nitric acid in liquid volume above resin bed
c y = bulk plutonium concentration in pore solution, g/cc
c yold = bulk plutonium concentration in pore solution at previous step, g/cc
c y2 = bulk plutonium concentration in solution between resin beads, g/cc
c y2bot = plutonium concentration in liquid volume below resin bed, g/cc
c y2feed = plutonium concentration in solution entering resin bed, g/cc
c y2max = maximum bulk plutonium concentration in solution between resin beads
c         during elution, g/cc
c y2mx2 = maximum bulk plutonium concentration in solution between resin beads
c         during elapsed time during elution, g/cc
c y2old = bulk plutonium concentration in solution between resin beads at
c         previous time step, g/cc
c x = fractional loading of plutonium on resin
c xarea = cross-sectional area of resin bed, cm2
c xold = fractional loading of plutonium on resin at previous time step
c zden = area-average liquid density at a given level, g/cc
c zvz = area-average superficial axial velocity at a given level, cm/s
      implicit double precision (a-z)
      integer i,j,m,mm1,n,nn1,q,r,s,stp,iskip,isl,iss,jlev,
     1ind2
      character*10 ch1
      dimension temc(200,100),temp(200,100),phi(200,100)
      dimension dens(200,100),zden(200),htc(200)
      dimension vis(200,100),y2mx2(200)
      dimension area(100),den1(200,100),den2(200,100)
      dimension rhcp(200,100),rhcp2(200,100),caps(200,100)
      dimension bz(200,100),br(200,100),br1(200,100),tcnm(200,100)
      dimension aa(200),bb(200),cc(200)
      dimension uu(200),rr(200),rra(100)
      dimension aa1(200,100),bb1(200,100),cc1(200,100),rr1(200,100)
      character tab /9/
      dimension c(200),cold(200),f(200),fold(200),x(200),xold(200)
      dimension y(200),yold(200),y2(200),y2old(200),fx(200)
      common /elut1/c,cold,f,fold,x,xold,y,yold,y2,y2old,cfeed,
     1dt,tau,tau2,ffeed,an1,dpart,vz,dz,xg,vfi2,vfo,
     2y2feed,y2max,y2mx2,c1feed
c Input initial resin bed temperature in °C.
      read (5,*) tzero
c Input ambient temperature in °C.
      read (5,*) tamb
c Input indicator for air heat transfer (iht=1) or water jacket heat transfer
c (iht=2).
      read (5,*) iht
c Input resin bed depth in centimeters.
      read (5,*) length
c Input height above resin bed in centimeters.
```

```
      read (5,*) lup
c Input depth below resin bed in centimeters.
      read (5,*) ldown
c Input column radius in centimeters.
      read (5,*) radius
c Input particle diameter in centimeters.
      read (5,*) dpart
c Input pore diffusivity in cm2/sec.
      read (5,*) dfpor
c Input particle diffusivity in cm2/sec.
      read (5,*) dfprt
c Input void fraction.
      read (5,*) vfo
c Input Pu feed concentration, gm/cc.
      read (5,*) cfeed
c Input maximum Pu resin bed concentration, gm/cc.
      read (5,*) ffeed
c Input maximum Pu concentration during elution, gm/cc.
      read (5,*) y2max
c Input radiolysis heat in cal/cc/sec.
      read (5,*) qdot
      pi=4.*atan(1.)
      rho=1.23
      cp=0.76
      tzero=tzero+273.0
      tamb=tamb+273.0
      grav=980.
      vfi=0.132
      vfi2=0.160
      an1=1./(vfo+vfi)
      pi=4.*atan(1.)
      tau=dpart**2/(4.*pi**2)/dfprt
      tau2=dpart**2/(24.*0.75*dfpor)
      xg=4.*pi**2*dfpor/dpart**2*(1.-vfo)
c Input # steps in axial, radial directions.
      read (5,*) m,n
c Input # of data points to skip between reading each concentration.
      read (5,*) iskip
      dz=length/m
      dr=radius/n
      vtop=pi*radius**2*lup
      vbot=pi*radius**2*ldown
      vmid=pi*radius**2*length
      m=m+1
      n=n+1
      mm1=m-1
      nn1=n-1
      area(1)=pi*(0.5/nn1*radius)**2
      area(n)=pi*(1.-((nn1-0.5)/nn1)**2)*radius**2
      rra(1)=0.
      i=2
      do while (i.le.nn1)
      r1=(i-1.5)/nn1*radius
      r2=(i-0.5)/nn1*radius
      area(i)=pi*(r2**2-r1**2)
      rra(i)=rra(i-1)+dr
      i=i+1
      enddo
      rra(n)=rra(nn1)+dr
      areat=pi*radius**2
      i=1
      sumf=0.
      dvol=pi*radius**2*dz
```

```
      do while (i.le.mm1)
      sumf=sumf+0.5*(f(i)+f(i+1))
      i=i+1
      enddo
      sumf=sumf*dvol
c Input time increment in seconds.
      read (5,*) dt
      dto=dt
c Input number of time steps.
      read (5,*) stp
c Input loading time in sec.
      read (5,*) tload
c Input washing time in sec.
      read (5,*) twash
c Input elution time in sec.
      read (5,*) telut
c Input time to print out temperature matrix in sec.
      read (5,*) trpt
c Input time interval for printing output.
      read (5,*) tinte
c Input time interval for printing maximum temperatures.
      read (5,*) tinte2
c Calculate material properties.
      stem1=tzero
      stem2=tzero
      cbot=0.
      y2bot=0.
      y2feed=0.
      c1feed=0.
      stem1c=stem1-273.
      stem2c=stem2-273.
      den1t=1.6603-1.9894e-3*stem1c
      den1b=1.6603-1.9894e-3*stem2c
      den2t=0.99683-1.3010e-4*stem1c-2.4358e-6*stem1c**2
      den2b=0.99683-1.3010e-4*stem2c-2.4358e-6*stem2c**2
      xx1t=den1t*cf30*(1.-c1feed)/(den1t*cf30*(1.-c1feed)+den2t
     1*(1.-cf30*(1.-c1feed)))
      xx1b=den1b*cf30*(1.-cbot)/(den1b*cf30*(1.-c1feed)+den2b
     1*(1.-cf30*(1.-cbot)))
      cno3t=(1.-c1feed)*cf30*den1t*1000./63.-4.*y2feed*1000./238.
      cno3b=(1.-cbot)*cf30*den1b*1000./63.-4.*y2bot*1000./238.
      if (cno3t.lt.0.) cno3t=0.
      if (cno3b.lt.0.) cno3b=0.
      denst=den2t+0.4277*y2feed*1000./238./vfo+0.031*cno3t
      densb=den2b+0.4277*y2bot*1000./238./vfo+0.031*cno3b
      capst=1.0104-(1.419-0.3*(stem1c-20.)/80.)*xx1t
     1+(2.005-0.6*(stem1c-20.)/80.)*xx1t**2
     2-(1.147-0.3*(stem1c-20.)/80.)*xx1t**3
      capsb=1.0104-(1.419-0.3*(stem2c-20.)/80.)*xx1b
     1+(2.005-0.6*(stem2c-20.)/80.)*xx1b**2
     2-(1.147-0.3*(stem2c-20.)/80.)*xx1b**3
      rhcp2t=denst*capst
      rhcp2b=densb*capsb
      i=1
      do while (i.le.n)
      j=1
      do while (j.le.m)
      cold(j)=0.
      c(j)=0.
      temc(j,i)=tzero-273.
      den1(j,i)=1.6603-1.9894e-3*temc(j,i)
      den2(j,i)=0.99683-1.3010e-4*temc(j,i)-2.4358e-6*temc(j,i)**2
      xx1=den1(j,i)*cf30*(1.-c(j))/(den1(j,i)*cf30*(1.-c(j))+den2(j,i)
```

```
     1*(1.-cf30*(1.-c(j))))
      cno3=(1.-c(j))*cf30*den1(j,i)*1000./63.-4.*y2(j)*1000.
     1/238./vfo
      if (cno3.lt.0.) cno3=0.
      dens(j,i)=den2(j,i)+0.4277*y2(j)*1000./238./vfo+0.031*cno3
      tcn1=6.1388e-4+1.3951e-6*temc(j,i)
      tcn2=1.3518e-3+2.7903e-6*temc(j,i)
      tcnm(j,i)=tcn1*xx1+tcn2*(1.-xx1)
      tcnm(j,i)=0.5*(5.e-4+tcnm(j,i))
      caps(j,i)=1.0104-(1.419-0.3*(temc(j,i)-20.)/80.)*xx1
     1+(2.005-0.6*(temc(j,i)-20.)/80.)*xx1**2
     2-(1.147-0.3*(temc(j,i)-20.)/80.)*xx1**3
      rhcp(j,i)=0.5*1.25*0.52+0.5*dens(j,i)*caps(j,i)
      rhcp2(j,i)=dens(j,i)*caps(j,i)
      alpha=tcnm(j,i)/rhcp(j,i)
      bz(j,i)=(alpha+2.*dpart*vz*rhcp2(j,i)/rhcp(j,i))*dt/dz**2
      j=j+1
      enddo
      i=i+1
      enddo
      j=1
      do while (j.le.m)
      alpha=tcnm(j,1)/rhcp(j,1)
      br(j,1)=(alpha+0.4*dpart*vz*rhcp2(j,1)/rhcp(j,1))*dt*4./dr**2
      br1(j,1)=0.
      j=j+1
      enddo
      i=2
      rrr=dr
      rro=0.
      do while (i.le.n)
      j=1
      do while (j.le.m)
      alpha=tcnm(j,i)/rhcp(j,i)
      rrn=rrr+dr
      br(j,i)=(alpha+0.4*dpart*vz*rhcp2(j,i)/rhcp(j,i))*dt/dr**2
      br1(j,i)=(alpha+0.4*dpart*vz*rhcp2(j,i)/rhcp(j,i))*dt
     1/2./dr/rra(i)
      j=j+1
      enddo
      rro=rrr
      rrr=rrr+dr
      i=i+1
      enddo
c Input initial volume fraction nitric acid.
      read (5,*) cf30
c Input velocity in cm/sec.
      read (5,*) vz
c Write input variables to output.
      write (1,100) length,radius,m,n
      write (1,121) dpart,vfo
      write (1,101) dt,htc,htc2
      write (1,110) sumf
  100 format ('L (CM) =',f7.3,2x,'R (CM) =',f7.3,2x,'AXIAL NODES =',
     1i3,2x,'RADIAL NODES =',i3)
  101 format ('DT (S) =',f8.4,2x,'HSID (CAL/CM2/S/°C) =',f9.6,
     12x,'HTOP (CAL/CM2/S/°C) =',f9.6)
  110 format ('PU CHARGE (GM) =',f11.4)
      write (2,*) 'TIME (MIN)',tab,'TMAX (°C)',tab,'VTOP (CC)',
     1tab,'VSGMAX (CM/S)'
  121 format ('PART DIAM (CM) =',f8.4,2x,'VOID FRACTION =',f7.4)
c Set all temperatures equal to uniform initial value.  Initialize plutonium
c and nitric acid concentrations.
```

```
      q=1
      do while (q.le.m)
      r=1
      do while (r.le.n)
      temp(q,r)=tzero
      tem1=tzero-273.
      den1(q,r)=1.6603-1.9894e-3*temc(q,r)
      den2(q,r)=0.99683-1.3010e-4*temc(q,r)-2.4358e-6*temc(q,r)**2
      r=r+1
      enddo
      fx(q)=0.
      fold(q)=0.
      yold(q)=0.
      y2old(q)=0.
      y2mx2(q)=0.
      cold(q)=0.
      q=q+1
      enddo
c Perform transient calculations.
      s=1
      time2=0.
      iss=1
      do while (time2.le.tload+twash+telut+1.)
      if (time2.le.tload) ind2=1
      if (time2.gt.tload.and.time2.le.tload+twash) ind2=2
      if (time2.gt.tload+twash) ind2=3
      if (iss.eq.0) then
c Switch between implicit radial, explicit axial and explicit radial, implicit
c axial calculations.
      iss=1
      else
      iss=0
      endif
c Calculate heat transfer coefficients
      sigma=1.356e-12
      emiss=0.6
      q=1
      do while (q.le.m)
      hrad=emiss*sigma*(temp(q,n)**3+temp(q,n)**2*tamb
     1+temp(q,n)*tamb**2+tamb**3)
      cpair=3.5*1.9872*29.
      xkair=0.0262/100*0.2389
      visair=0.00018
      pr=cpair*visair/xkair
      rhoamb=29./82.057/tamb
      rhoair=29./82.057/temp(q,n)
      rayl=980.*(length+lup+ldown)**3*rhoamb*(rhoamb-rhoair)
     1/visair**2*pr
      if (rayl.gt.0.) then
      nu=0.59*rayl**0.25
      endif
      if (nu.lt.1.) nu=1.
      hconv=nu*xkair/(length+lup+ldown)
      vel2=30.0
      re=2.*radius*vel2*rhoamb/visair
      nu3=0.683*re**0.466*pr**(1./3.)
      hconv3=xkair*nu3/2./radius
      if (hconv.lt.hconv3) hconv=hconv3
      htc(q)=hconv+hrad
      q=q+1
      enddo
      hrad=emiss*sigma*(stem1**3+stem1**2*tamb+stem1*tamb**2+tamb**3)
      rhoair=29./82.057/stem1
```

```
      rayl=980.*(length+lup+ldown)**3*rhoamb*(rhoamb-rhoair)
     1/visair**2*pr
      if (rayl.gt.0.) then
      nu=0.59*rayl**0.25
      endif
      if (nu.lt.1.) nu=1.
      hconv=nu*xkair/(length+lup+ldown)
      rayl2=980.*(2.*radius)**3*rhoamb*(rhoamb-rhoair)/visair**2*pr
      if (rayl2.gt.0.) then
      nu2=0.54*rayl2**0.25
      endif
      if (nu2.lt.1.) nu2=1.
      hconv2=nu2*xkair/2./radius
      hconv=(2.*radius*lup*hconv+radius**2*hconv2)/(2.*radius*lup
     1+radius**2)
      vel2=30.0
      re=2.*radius*vel2*rhoamb/visair
      nu3=0.683*re**0.466*pr**(1./3.)
      hconv3=xkair*nu3/2./radius
      if (hconv.lt.hconv3) hconv=hconv3
      htc2t=hrad+hconv
      hrad=emiss*sigma*(stem2**3+stem2**2*tamb+stem2*tamb**2+tamb**3)
      rhoair=29./82.057/stem2
      rayl=980.*(length+lup+ldown)**3*rhoamb*(rhoamb-rhoair)
     1/visair**2*pr
      if (rayl.gt.0.) then
      nu=0.59*rayl**0.25
      endif
      if (nu.lt.1.) nu=1.
      hconv=nu*xkair/(length+lup+ldown)
      rayl2=980.*(2.*radius)**3*rhoamb*(rhoamb-rhoair)/visair**2*pr
      if (rayl2.gt.0.) then
      nu2=0.27*rayl2**0.25
      endif
      if (nu2.lt.1.) nu2=1.
      hconv2=nu2*xkair/2./radius
      hconv=(2.*radius*ldown*hconv+radius**2*hconv2)/(2.*radius*ldown
     1+radius**2)
      vel2=30.0
      re=2.*radius*vel2*rhoamb/visair
      nu3=0.683*re**0.466*pr**(1./3.)
      hconv3=xkair*nu3/2./radius
      if (hconv.lt.hconv3) hconv=hconv3
      htc2b=hrad+hconv
c Calculate temperature changes in the spaces above and below the resin bed.
      stem1a=y2feed*0.8*qdot*dt/rhcp2t
      stem2a=y2bot*0.8*qdot*dt/rhcp2b
      i=1
      do while (i.le.n)
      stem1a=stem1a+bz(1,i)*dz/vtop*area(i)*(temp(2,i)-temp(1,i))
      stem2a=stem2a+bz(m,i)*dz/vbot*area(i)*(temp(mm1,i)-temp(m,i))
      stem2a=stem2a-vz*dt/vbot*area(i)*(stem2-temp(m,i))
      i=i+1
      enddo
      stem1a=stem1a-vz*dt*areat/vtop*(stem1-tzero)
      stem1a=stem1a+htc2t*dt/rhcp(1,1)/(vtop/(areat+2.*pi*rra(n)*lup))
     1*(tamb-stem1)
      stem2a=stem2a+htc2b*dt/rhcp(1,1)/(vbot/
     1(areat+2.*pi*rra(n)*ldown))*(tamb-stem2)
c Calculate concentration changes in the space above the resin bed.
      y2feed=y2feed+2.*dpart*vz*dt/dz/vtop*areat/vfo
     1*(y2old(2)-y2old(1))
      if (ind2.eq.1) then
```

```
      y2feed=y2feed+vz*areat/vtop*dt*(cfeed-y2feed)
      else
      y2feed=y2feed-vz*areat/vtop*dt*y2feed
      endif
      if (ind2.eq.3) then
      c1feed=c1feed+2.*dpart*vz*dt/dz/vtop*areat*(cold(2)-cold(1))
      c1feed=c1feed+vz*areat/vtop*dt*(1.-c1feed)
      endif
c Calculate temperature and concentration changes at top of column.
      phi(1,1)=temp(1,1)+fx(1)*0.8*qdot*dt/rhcp(1,1)
      if (iss.eq.0) then
      rr1(1,1)=phi(1,1)+bz(1,1)*(stem1-temp(1,1))
     1+bz(2,1)*(temp(2,1)-temp(1,1))
      aa1(1,1)=0.
      bb1(1,1)=1.+br(1,1)
      cc1(1,1)=-br(1,1)
      else
      rr1(1,1)=phi(1,1)+br(1,1)*(temp(1,2)-temp(1,1))+bz(1,1)*stem1
      aa1(1,1)=0.
      bb1(1,1)=1.+bz(1,1)+bz(2,1)
      cc1(1,1)=-bz(2,1)
      endif
      rr1(1,1)=rr1(1,1)-vz*dt/dz*(temp(1,1)-stem1)
      if (ind2.eq.1.or.ind2.eq.2) then
      call feeder (1,m,ind2)
      else
      call elut (1,m)
      endif
      phi(1,n)=temp(1,n)+fx(1)*0.8*qdot*dt/rhcp(1,n)
      if (iss.eq.0) then
      rr1(1,n)=phi(1,n)+bz(1,n)*(stem1-temp(1,n))
     1+bz(2,n)*(temp(2,n)-temp(1,n))+htc(1)*dt/rhcp(1,n)
     2*(1./dr+1./2./rra(n))*tamb
      aa1(1,n)=-br(1,nn1)+br1(1,n)
      bb1(1,n)=1.+br(1,nn1)-br1(1,n)+htc(1)*dt/rhcp(1,n)
     1*(1./dr+1./2./rra(n))
      cc1(1,n)=0.
      else
      rr1(1,n)=phi(1,n)-br(1,nn1)*(temp(1,n)-temp(1,nn1))
     1+br1(1,n)*(temp(1,n)-temp(1,nn1))+htc(1)*dt/rhcp(1,n)
     2*(1./dr+1./2./rra(n))*(tamb-temp(1,n))+bz(1,n)*stem1
      aa1(1,n)=0.
      bb1(1,n)=1.+bz(1,n)+bz(2,n)
      cc1(1,n)=-bz(2,n)
      endif
      rr1(1,n)=rr1(1,n)-vz*dt/dz*(temp(1,n)-stem1)
      r=2
      do while (r.le.nn1)
      phi(1,r)=temp(1,r)+fx(1)*0.8*qdot*dt/rhcp(1,r)
      if (iss.eq.0) then
      rr1(1,r)=phi(1,r)+bz(1,r)*(stem1-temp(1,r))
     1+bz(2,r)*(temp(2,r)-temp(1,r))
      aa1(1,r)=-br(1,r-1)+br1(1,r)
      bb1(1,r)=1.+br(1,r)+br(1,r-1)
      cc1(1,r)=-br(1,r)-br1(1,r)
      else
      rr1(1,r)=phi(1,r)-br(1,r-1)*(temp(1,r)-temp(1,r-1))
     1+br(1,r)*(temp(1,r+1)-temp(1,r-1))+br1(1,r)
     2*(temp(1,r)-temp(1,r-1))+bz(1,r)*stem1
      aa1(1,r)=0.
      bb1(1,r)=1.+bz(1,r)+bz(2,r)
      cc1(1,r)=-bz(2,r)
      endif
```

```
      rr1(1,r)=rr1(1,r)-vz*dt/dz*(temp(1,r)-stem1)
      r=r+1
      enddo
c Calculate temperature and concentration changes in middle of column.
      q=2
      do while (q.le.mm1)
      phi(q,1)=temp(q,1)+fx(q)*0.8*qdot*dt/rhcp(q,1)
      if (iss.eq.0) then
      rr1(q,1)=phi(q,1)+bz(q+1,1)*(temp(q+1,1)-temp(q,1))
     1+bz(q,1)*(temp(q-1,1)-temp(q,1))
      aa1(q,1)=0.
      bb1(q,1)=1.+br(q,1)
      cc1(q,1)=-br(q,1)
      else
      rr1(q,1)=phi(q,1)+br(q,1)*(temp(q,2)-temp(q,1))
      aa1(q,1)=-bz(q,1)
      bb1(q,1)=1.+bz(q,1)+bz(q+1,1)
      cc1(q,1)=-bz(q+1,1)
      endif
      rr1(q,1)=rr1(q,1)-vz*dt/dz*(temp(q,1)-temp(q-1,1))
      phi(q,n)=temp(q,n)+fx(q)*0.8*qdot*dt/rhcp(q,n)
      if (iss.eq.0) then
      rr1(q,n)=phi(q,n)+bz(q+1,n)*(temp(q+1,n)-temp(q,n))
     1+bz(q,n)*(temp(q-1,n)-temp(q,n))+htc(q)*dt/rhcp(q,n)
     2*(1./dr+1./2./rra(n))*tamb
      aa1(q,n)=-br(q,nn1)+br1(q,n)
      bb1(q,n)=1.+br(q,nn1)-br1(q,n)+htc(q)*dt/rhcp(q,n)
     1*(1./dr+1./2./rra(n))
      cc1(q,n)=0.
      else
      rr1(q,n)=phi(q,n)-br(q,nn1)*(temp(q,n)-temp(q,nn1))
     1+br1(q,n)*(temp(q,n)-temp(q,nn1))+htc(q)*dt/rhcp(q,n)
     2*(1./dr+1./2./rra(n))*(tamb-temp(q,n))
      aa1(q,n)=-bz(q,n)
      bb1(q,n)=1.+bz(q,n)+bz(q+1,n)
      cc1(q,n)=-bz(q+1,n)
      endif
      rr1(q,n)=rr1(q,n)-vz*dt/dz*(temp(q,n)-temp(q-1,n))
      q=q+1
      enddo
      q=2
      do while (q.le.mm1)
      r=2
      do while (r.le.nn1)
      phi(q,r)=temp(q,r)+fx(q)*0.8*qdot*dt/rhcp(q,r)
      if (iss.eq.0) then
      rr1(q,r)=phi(q,r)+bz(q+1,r)*(temp(q+1,r)-temp(q,r))
     1+bz(q,r)*(temp(q-1,r)-temp(q,r))
      aa1(q,r)=-br(q,r-1)+br1(q,r)
      bb1(q,r)=1.+br(q,r)+br(q,r-1)
      cc1(q,r)=-br(q,r)-br1(q,r)
      else
      rr1(q,r)=phi(q,r)-br(q,r-1)*(temp(q,r)-temp(q,r-1))
     1+br(q,r)*(temp(q,r+1)-temp(q,r))+br1(q,r)
     2*(temp(q,r+1)-temp(q,r-1))
      aa1(q,r)=-bz(q,r)
      bb1(q,r)=1.+bz(q,r)+bz(q+1,r)
      cc1(q,r)=-bz(q+1,r)
      endif
      rr1(q,r)=rr1(q,r)-vz*dt/dz*(temp(q,r)-temp(q-1,r))
      r=r+1
      enddo
      if (ind2.eq.1.or.ind2.eq.2) then
```

```
      call feeder (q,m,ind2)
      else
      call elut (q,m)
      endif
      q=q+1
      enddo
c Calculate temperature and concentration changes at bottom of column.
      phi(m,i)=temp(m,1)+fx(m)*0.8*qdot*dt/rhcp(m,1)
      if (iss.eq.0) then
      rr1(m,1)=phi(m,1)+bz(m,1)*(stem2-temp(m,1))
     1+bz(m,1)*(temp(mm1,1)-temp(m,1))
      aa1(m,1)=0.
      bb1(m,1)=1.+br(m,1)
      cc1(m,1)=-br(m,1)
      else
      rr1(m,1)=phi(m,1)+br(m,1)*(temp(m,2)-temp(m,1))+bz(m,1)*stem2
      aa1(m,1)=-bz(m,1)
      bb1(m,1)=1.+2.*bz(m,1)
      cc1(m,1)=0.
      endif
      rr1(m,1)=rr1(m,1)-vz*dt/dz*(temp(m,1)-temp(mm1,1))
      phi(m,n)=temp(m,n)+fx(m)*0.8*qdot*dt/rhcp(m,n)
      if (iss.eq.0) then
      rr1(m,n)=phi(m,n)+bz(m,n)*(stem2-temp(m,n))
     1+bz(m,n)*(temp(mm1,n)-temp(m,n))+htc(m)*dt/rhcp(m,n)
     2*(1./dr+1./2./rra(n))*tamb
      aa1(m,n)=-br(m,nn1)+br1(m,n)
      bb1(m,n)=1.+br(m,nn1)-br1(m,n)+htc(m)*dt/rhcp(m,n)
     1*(1./dr+1./2./rra(n))
      cc1(m,n)=0.
      else
      rr1(m,n)=phi(m,n)-br(m,nn1)*(temp(m,n)-temp(m,nn1))
     1+br1(m,n)*(temp(m,n)-temp(m,nn1))+htc(m)*dt/rhcp(m,n)
     2*(1./dr+1./2./rra(n))*(tamb-temp(m,n))+bz(m,n)*stem2
      aa1(m,n)=-bz(m,n)
      bb1(m,n)=1.+2.*bz(m,n)
      cc1(m,n)=0.
      endif
      rr1(m,n)=rr1(m,n)-vz*dt/dz*(temp(m,n)-temp(mm1,n))
      r=2
      do while (r.le.nn1)
      phi(m,r)=temp(m,r)+fx(m)*0.8*qdot*dt/rhcp(m,r)
      if (iss.eq.0) then
      rr1(m,r)=phi(m,r)+bz(m,r)*(stem2-temp(m,r))
     1+bz(m,r)*(temp(mm1,r)-temp(m,r))
      aa1(m,r)=-br(m,r-1)+br1(m,r)
      bb1(m,r)=1.+br(m,r)+br(m,r-1)
      cc1(m,r)=-br(m,r)-br1(m,r)
      else
      rr1(m,r)=phi(m,r)-br(m,r-1)*(temp(m,r)-temp(m,r-1))
     1+br(m,r)*(temp(m,r+1)-temp(m,r))
     2+br1(m,r)*(temp(m,r+1)-temp(m,r-1))+bz(m,r)*stem2
      aa1(m,r)=-bz(m,r)
      bb1(m,r)=1.+2.*bz(m,r)
      cc1(m,r)=0.
      endif
      rr1(m,r)=rr1(m,r)-vz*dt/dz*(temp(m,r)-temp(mm1,r))
      r=r+1
      enddo
      if (ind2.eq.1.or.ind2.eq.2) then
      call feeder (m,m,ind2)
      else
      call elut (m,m)
```

```
      endif
      if (iss.eq.0) then
      j=1
      do while (j.le.m)
      i=1
      do while (i.le.n)
      aa(i)=aa1(j,i)
      bb(i)=bb1(j,i)
      cc(i)=cc1(j,i)
      rr(i)=rr1(j,i)
      i=i+1
      enddo
      call tridag (aa,bb,cc,rr,uu,n)
      i=1
      do while (i.le.n)
      phi(j,i)=uu(i)
      i=i+1
      enddo
      j=j+1
      enddo
      else
      i=1
      do while (i.le.n)
      j=1
      do while (j.le.m)
      aa(j)=aa1(j,i)
      bb(j)=bb1(j,i)
      cc(j)=cc1(j,i)
      rr(j)=rr1(j,i)
      j=j+1
      enddo
      call tridag (aa,bb,cc,rr,uu,m)
      j=1
      do while (j.le.m)
      phi(j,i)=uu(j)
      j=j+1
      enddo
      i=i+1
      enddo
      endif
c Calculate concentration changes in the liquid volume below the resin bed.
      y2bot=y2bot+2.*dpart*vz*dt/dz/vbot*areat/vfo
     1*(y2old(mm1)-y2old(m))
      y2bot=y2bot+vz*areat/vbot*(y2old(m)/vfo-y2bot)
      cbot=cbot+vz*areat/vbot*(cold(m)-cbot)
c Update temperatures and concentrations.  Calculate densities and viscosities.
      stem1=stem1+stem1a
      stem2=stem2+stem2a
      j=1
      do while (j.le.m)
      yold(j)=y(j)
      y2old(j)=y2(j)
      xold(j)=x(j)
      cold(j)=c(j)
      if (ind2.eq.1.or.ind2.eq.2) fold(j)=f(j)
      if (ind2.eq.1.or.ind2.eq.2) fx(j)=f(j)+y2(j)
      if (ind2.eq.3) fx(j)=f(j)+y2(j)+y(j)
      j=j+1
      enddo
      stem1c=stem1-273.
      stem2c=stem2-273.
      den1t=1.6603-1.9894e-3*stem1c
      den1b=1.6603-1.9894e-3*stem2c
```

```
 den2t=0.99683-1.3010e-4*stem1c-2.4358e-6*stem1c**2
 den2b=0.99683-1.3010e-4*stem2c-2.4358e-6*stem2c**2
 xx1t=den1t*cf30*(1.-c1feed)/(den1t*cf30*(1.-c1feed)+den2t
1*(1.-cf30*(1.-c1feed)))
 xx1b=den1b*cf30*(1.-cbot)/(den1b*cf30*(1.-c1feed)+den2b
1*(1.-cf30*(1.-cbot)))
 cno3t=(1.-c1feed)*cf30*den1t*1000./63.-4.*y2feed*1000./238.
 cno3b=(1.-cbot)*cf30*den1b*1000./63.-4.*y2bot*1000./238.
 if (cno3t.lt.0.) cno3t=0.
 if (cno3b.lt.0.) cno3b=0.
 denst=den2t+0.4277*y2feed*1000./238./vfo+0.031*cno3t
 densb=den2b+0.4277*y2bot*1000./238./vfo+0.031*cno3b
 capst=1.0104-(1.419-0.3*(stem1c-20.)/80.)*xx1t
1+(2.005-0.6*(stem1c-20.)/80.)*xx1t**2
2-(1.147-0.3*(stem1c-20.)/80.)*xx1t**3
 capsb=1.0104-(1.419-0.3*(stem2c-20.)/80.)*xx1b
1+(2.005-0.6*(stem2c-20.)/80.)*xx1b**2
2-(1.147-0.3*(stem2c-20.)/80.)*xx1b**3
 rhcp2t=denst*capst
 rhcp2b=densb*capsb
 i=1
 do while (i.le.n)
 j=1
 do while (j.le.m)
 temp(j,i)=phi(j,i)
 if (iht.eq.2.and.i.eq.n) temp(j,i)=tamb
 temc(j,i)=phi(j,i)-273.0
 if (iht.eq.2.and.i.eq.n) temc(j,i)=tamb-273.0
 den1(j,i)=1.6603-1.9894e-3*temc(j,i)
 den2(j,i)=0.99683-1.3010e-4*temc(j,i)-2.4358e-6*temc(j,i)**2
 xx1=den1(j,i)*cf30*(1.-c(j))/(den1(j,i)*cf30*(1.-c(j))
1+den2(j,i)*(1.-cf30*(1.-c(j))))
 cno3=(1.-c(j))*cf30*den1(j,i)*1000./63.-4.*y2(j)*1000.
1/238./vfo
 if (cno3.lt.0.) cno3=0.
 dens(j,i)=den2(j,i)+0.4277*y2(j)*1000./238./vfo+0.031*cno3
 tcn1=6.1388e-4+1.3951e-6*temc(j,i)
 tcn2=1.3518e-3+2.7903e-6*temc(j,i)
 tcnm(j,i)=tcn1*xx1+tcn2*(1.-xx1)
 tcnm(j,i)=0.5*(tcnm(j,i)+5.e-4)
 caps(j,i)=1.0104-(1.419-0.3*(temc(j,i)-20.)/80.)*xx1
1+(2.005-0.6*(temc(j,i)-20.)/80.)*xx1**2
2-(1.147-0.3*(temc(j,i)-20.)/80.)*xx1**3
 rhcp(j,i)=0.5*1.25*0.52+0.5*dens(j,i)*caps(j,i)
 rhcp2(j,i)=dens(j,i)*caps(j,i)
 alpha=tcnm(j,i)/rhcp(j,i)
 bz(j,i)=(alpha+2.*dpart*vz*rhcp2(j,i)/rhcp(j,i))*dt/dz**2
 vis(j,i)=0.01002*10.**((1.3272*(20.-temc(j,i))-0.001053*
1(temc(j,i)-20.)**2)/(temc(j,i)+105.))
 j=j+1
 enddo
 i=i+1
 enddo
 j=1
 do while (j.le.m)
 alpha=tcnm(j,1)/rhcp(j,1)
 br(j,1)=(alpha+0.4*dpart*vz*rhcp2(j,1)/rhcp(j,1))*dt*4./dr**2
 j=j+1
 enddo
 i=2
 rrr=dr
 rro=0.
 do while (i.le.nn1)
```

```
      j=1
      do while (j.le.m)
      alpha=tcnm(j,i)/rhcp(j,i)
      rrn=rrr+dr
      br(j,i)=(alpha+0.4*dpart*vz*rhcp2(j,i)/rhcp(j,i))*dt
     1*4.*(rrn+rrr)/dr/((rrn+rrr)**2-(rrr+rro)**2)
      j=j+1
      enddo
      rro=rrr
      rrr=rrr+dr
      i=i+1
      enddo
c Calculate flow-weighted average effluent temperatures and concentrations at
c top and bottom of column.  These are used as inlet temperatures and
c concentrations for the radial locations where flow enters the column.
      temb=0.
      svz2=0.
      i=1
      do while (i.le.n)
      temb=temb+area(i)*temp(m,i)*vz
      svz2=svz2+area(i)*vz
      i=i+1
      enddo
      stem2=stem2+dt/vbot*(temb-svz2*stem2)
c Calculate maximum plutonium concentrations at each axial location.
      j=1
      do while (j.le.m)
      if (y2mx2(j).lt.y2(j)) y2mx2(j)=y2(j)
      j=j+1
      enddo
c Integrate Pu concentration profile and compare with amount loaded.
      pi=4.*atan(1.)
      xarea=pi*radius**2
      put=xarea*cfeed*vz*tload
      pua=0.
      puf=0.
      j=1
      do while (j.le.mm1)
      pua=pua+0.5*(f(j)+f(j+1))*xarea*dz
      puf=puf+0.5*(fx(j)+fx(j+1))*xarea*dz
      j=j+1
      enddo
      puf=puf+vtop*y2feed+vbot*y2bot
c Write transient temperatures and velocities to output files.
      time2=time2+dt
      if (time2.lt.tload) time=time2/60.
      if (time2.ge.tload.and.time2.lt.tload+twash)
     1time=(time2-tload)/60.
      if (time2.ge.tload+twash)
     1time=(time2-tload-twash)/60.
      if (abs(nint(time/tinte)-time/tinte).lt.0.0001
     1.or.abs((time2-tload)/tinte).lt.0.0001
     2.or.abs((time2-tload-twash)/tinte).lt.0.0001
     3.or.abs((time2-tload-twash-telut)/tinte).lt.0.0001) then
c Write headers for output files.
      write (1,*) 'TIME (MIN)',tab,'POS (CM)',tab,'LOAD',tab,
     1'TCEN (°C)',tab,'TMID (°C)',tab,'TOUT (°C)'
      write (6,*) 'TIME (MIN)',tab,'LENGTH (CM)',tab,'ELU (VF)',tab,
     1'ADS (G/CC)',tab,'LIQ (G/CC)',tab,'POR (G/CC)',tab,'TOT (G/CC)'
      j=1
      stem1m=stem1-273.
      write (1,102) time,tab,-1.,tab,0.,tab,stem1m,tab,stem1m,
     1tab,stem1m
```

```
      do while (j.le.m)
      poszn=length*(j-1)/(m-1)
      write (1,102) time,tab,poszn,tab,f(j),tab,temc(j,1),tab,
     1temc(j,5),tab,temc(j,n)
  102 format (f9.4,a1,f10.5,a1,f7.5,3(a1,f9.4))
      write (6,400) time,tab,poszn,tab,c(j),tab,f(j),tab,y2(j),
     1tab,y(j),tab,fx(j)
  400 format (f9.4,a1,f10.5,5(a1,f9.5))
      j=j+1
      enddo
      write (6,400) time,tab,-1.,tab,c1feed,tab,0.,tab,y2feed,tab,
     10.,tab,0.
      write (6,400) time,tab,-2.,tab,cbot,tab,0.,tab,y2bot,tab,0.,
     1tab,0.
      write (6,*) 'PU FED (GM)',tab,'PU ADS (GM)',tab,'PU TOT (GM)',
     1tab,'FRAC ADS',tab,'FRAC TOT'
      write (6,401) put,tab,pua,tab,puf,tab,pua/put,tab,puf/put
  401 format (f9.4,2(a1,f9.4),2(a1,f9.5))
      stem2m=stem2-273.
      write (1,102) time,tab,-2.,tab,0.,tab,stem2m,tab,stem2m,tab,
     1stem2m
      endif
c Find highest temperature.
      temch=0.
      i=1
      do while (i.le.n)
      j=1
      do while (j.le.m)
      if (temch.lt.temc(j,i)) then
      temch=temc(j,i)
      endif
      j=j+1
      enddo
      i=i+1
      enddo
      if (abs(nint(time/tinte2)-time/tinte2).lt.0.0001)
     1write (2,301) time,tab,temch,tab,vtop,tab,vgim
  301 format (f11.4,a1,f10.5,a1,f11.5,a1,f11.5)
      if (abs(time2-trpt).lt.0.5*dt) then
      write (9,*) stem1c
      write (9,*) stem2c
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      write (9,*) temc(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (9,*) temc(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
      endif
      s=s+1
      enddo
      stop
      end


c This is a subroutine that uses LU decomposition to solve a set of equations
```

```
c that is tridiagonal.  This subroutine was copies from Press, Teukolsky,
c Vetterling, and Flannery, Numerical Recipes in FORTRAN:  The Art of
Scientific
c Computing, 2nd ed., Cambridge University Press (New York), 1986, p. 43.
      subroutine tridag (a,b,c,r,u,n)
      implicit double precision (a-h,o-z)
      dimension a(n),b(n),c(n),r(n),u(n)
      dimension gam(400)
      bet=b(1)
      u(1)=r(1)/bet
      jjj=2
      do while (jjj.le.n)
      gam(jjj)=c(jjj-1)/bet
      bet=b(jjj)-a(jjj)*gam(jjj)
      u(jjj)=(r(jjj)-a(jjj)*u(jjj-1))/bet
      jjj=jjj+1
      enddo
      jjj=n-1
      do while(jjj.ge.1)
      u(jjj)=u(jjj)-gam(jjj+1)*u(jjj+1)
      jjj=jjj-1
      enddo
      return
      end

      subroutine feeder (i,m,ind2)
c Compute changes in the plutonium concentrations during feeding or washing.
      implicit double precision (a-z)
      integer i,m,ind2
      dimension c(200),cold(200),f(200),fold(200),x(200),xold(200),
     1y(200),yold(200),y2(200),y2old(200),y2mx2(200)
      common /elut1/c,cold,f,fold,x,xold,y,yold,y2,y2old,cfeed,
     1dt,tau,tau2,ffeed,an1,dpart,vz,dz,xg,vfi2,vfo,
     2y2feed,y2max,y2mx2,c1feed
      if (i.eq.1.and.ind2.eq.1) dy2=vz/vfo*dt/dz*(vfo*y2feed-y2old(1))
     1+2.*dpart*vz*dt/dz**2*(y2old(2)-y2old(1))
      if (i.eq.1.and.ind2.eq.2) dy2=-(2.*dpart*vz*dt/dz**2
     1+vz/vfo*dt/dz)*y2old(1)+2.*dpart*vz*dt/dz**2*y2old(2)
      if (i.gt.1.and.i.lt.m)
     1dy2=(2.*dpart*vz*dt/dz**2+vz/vfo*dt/dz)*y2old(i-1)
     2-(4.*dpart*vz*dt/dz**2+vz/vfo*dt/dz)*y2old(i)
     3+2.*dpart*vz*dt/dz**2*y2old(i+1)
      if (i.eq.m) dy2=(2.*dpart*vz*dt/dz**2+vz/vfo*dt/dz)
     1*y2old(m-1)-(2.*dpart*vz*dt/dz**2+vz/vfo*dt/dz)*y2old(m)
      if (y2(i)+dy2.lt.0.) dy2=-y2(i)
      xy2=y2old(i)/y2feed
      if (xold(i).lt.1.) then
      if (xold(i).gt.1.d-3)
     1x(i)=xold(i)+dt/tau*xy2*xold(i)**2/(log((1.+xold(i))/
     2(1.-xold(i)))-2.*xold(i))
      if (xold(i).le.1.d-3.and.xold(i).gt.1.d-8)
     1x(i)=xold(i)+dt/tau*xy2*1.5/xold(i)
      if (x(i).gt.1.) x(i)=1.
      if (x(i).lt.xold(i).and.i.gt.1) x(i)=xold(i)
      if (y2mx2(i).gt.1.d-3) then
      df=ffeed*(x(i)-xold(i))*y2old(i)/y2mx2(i)
      else
      df=ffeed*(x(i)-xold(i))
      endif
      if (y2old(i)+dy2.ge.y2mx2(i)) then
      if (df.gt.dy2) then
      df=dy2
      dy2=0.
```

```
      else
      dy2=dy2-df
      endif
      if (xold(i).le.1.d-8) then
      df=dy2
      dy2=0.
      endif
      else
      if (y2(i)+dy2-df.lt.0.) then
      df=y2(i)+dy2
      dy2=-y2(i)
      else
      dy2=dy2-df
      endif
      endif
      endif
      if (ind2.eq.1.and.xold(i).ge.1.) df=0.
      if (ind2.eq.2.and.xold(i).ge.1.) then
      df=0.
      if (y2(i)+dy2.lt.0.) dy2=-y2(i)
      endif
      f(i)=f(i)+df
      y2(i)=y2(i)+dy2
      x(i)=xold(i)+df/ffeed
      return
      end

      subroutine elut (i,m)
c Compute changes in nitric acid and plutonium concentrations for elution.
      implicit double precision (a-z)
      integer i,m
      dimension c(200),cold(200),f(200),fold(200),x(200),xold(200),
     1y(200),yold(200),y2(200),y2old(200),y2mx2(200)
      common /elut1/c,cold,f,fold,x,xold,y,yold,y2,y2old,cfeed,
     1dt,tau,tau2,ffeed,an1,dpart,vz,dz,xg,vfi2,vfo,
     2y2feed,y2max,y2mx2,c1feed
      if (i.eq.1) c(1)=cold(1)+an1*(vz*dt/dz*(c1feed-cold(1))
     1+2.*dpart*vz*dt/dz**2*(cold(2)-cold(1)))
      if (i.gt.1.and.i.lt.m) c(i)=cold(i)+an1*
     1((2.*dpart*vz*dt/dz**2+vz*dt/dz)*cold(i-1)-(4.*dpart*vz*dt/dz**2
     2+vz*dt/dz)*cold(i)+2.*dpart*vz*dt/dz**2*cold(i+1))
      if (i.eq.m) c(m)=cold(m)+an1*(vz*dt/dz
     1*(cold(m-1)-cold(m))+2.*dpart*dt/dz**2*(cold(m-1)-cold(m)))
      if (fold(i).gt.0.) then
      if (y2old(i).gt.1.d-8.and.cold(i).gt.1.d-8) then
      if (y2old(i)/y2max.lt.cold(i)) then
      ffcx=(cold(i)-y2old(i)/y2max)**0.5
      fcx=ffcx/(ffcx+(y2old(i)/y2max)**0.5)
      else
      fcx=1.-cold(i)
      endif
      else
      fcx=1.
      endif
      if (f(i).gt.0..and.y2old(i).lt.y2max) then
      df=-1./(1.-0.8*(fold(i)/f(i))**(1./3.)-0.2*(fold(i)/f(i))**2
     1-1.d-8)*cold(i)*dt/tau2*fold(i)*fcx
      if (df.gt.(3.*cold(i)*dt/tau2*fcx)**0.5*fold(i))
     1then
      df=(3.*cold(i)*dt/tau2*fcx)**0.5*fold(i)
      endif
      else
      df=0.
```

```
 endif
 if (f(i)-df.lt.0.) then
 df=f(i)
 f(i)=0.
 else
 f(i)=f(i)-df
 endif
 else
 df=0.
 endif
 if (y2old(i).lt.y2max) then
 if (fold(i).gt.0.) then
 vfi3=vfi2*(1.-f(i)/fold(i))**0.5
 else
 vfi3=vfi2
 endif
 if (vfi3.lt.1.d-8) vfi3=1.d-8
 if (yold(i).gt.1.d-8) then
 if (y2old(i).gt.0.) then
 y2max1=(y2max*y2old(i))**0.5
 else
 y2max1=0.
 endif
 if (abs(y2max1/vfo-yold(i)/vfi3).gt.1.d-8.and.
1abs(y2max1).gt.1.d-8) then
 dg=xg*dt*(yold(i)/vfi3)**2*((y2max1/vfo)**2
1-(yold(i)/vfi3)**2)/(2.*yold(i)/vfi3*(y2max1/vfo)**2
2-y2max1/vfo*((y2max1/vfo)**2-(yold(i)/vfi3)**2)
3*log((y2max1/vfo+yold(i)/vfi3)/(abs(y2max1/vfo-
4yold(i)/vfi3))))
 else
 dg=xg*dt*(y2max1/vfo-yold(i)/vfi3)
 endif
 else
 if (y2old(i).gt.0.) then
 dg=(1.5*y2max*y2old(i)*xg*dt)**0.5
 else
 dg=0.
 endif
 endif
 if (yold(i)+dg.lt.0.) then
 dg=-yold(i)
 y(i)=0.
 else
 y(i)=yold(i)+dg
 endif
 else
 dg=0.
 endif
 if (i.eq.1) y2(i)=y2old(i)-vz/vfo*dt/dz
1*(y2old(i)-vfo*y2feed)+2.*dpart*vz*dt/dz**2
2*(y2old(i+1)-y2old(i))+df-dg
 if (i.gt.1.and.i.lt.m)
1y2(i)=y2old(i)+(2.*dpart*vz*dt/dz**2
2+vz/vfo*dt/dz)*y2old(i-1)-(4.*dpart*vz*dt/dz**2+2.*vz/vfo*dt/dz)
3*y2old(i)+2.*dpart*vz*dt/dz**2*y2old(i+1)+df-dg
 if (i.eq.m) y2(m)=y2old(m)-vz/vfo*dt/dz
1*(y2old(m)-y2old(m-1))+2.*dpart*vz*dt/dz**2
2*(y2old(m-1)-y2old(m))+df-dg
 if (y2(i).lt.0.) then
 y(i)=y(i)+y2(i)
 y2(i)=0.
 endif
```

```
      return
      end
```

Sample Input Listing for Thermal Analysis during Normal Operation

```
25.
25.
1
34.1
2.54
2.54
3.73
0.04
0.0000012
0.0000012
0.33
0.163
0.062
0.0263
0.13384
48,24
0
0.02
18250
29.
149.
187.
29.
0.25
0.05
0.274
0.458
```

Sample Output Listing for Thermal Analysis during Normal Operation

TIME (MIN)	TMAX (°C)	VTOP (CC)	VSGMAX (CM/S)
0.0500	25.00786	111.02001	0.00000
0.1000	25.02581	111.02001	0.00000
0.1500	25.04540	111.02001	0.00000
0.2000	25.06753	111.02001	0.00000
0.2500	25.09177	111.02001	0.00000
0.3000	25.11744	111.02001	0.00000
0.3500	25.14420	111.02001	0.00000
0.4000	25.17226	111.02001	0.00000
0.4500	25.20083	111.02001	0.00000
0.0500	25.25032	111.02001	0.00000
0.1000	25.28031	111.02001	0.00000
0.1500	25.30669	111.02001	0.00000
0.2000	25.32948	111.02001	0.00000
0.2500	25.34950	111.02001	0.00000
0.3000	25.36851	111.02001	0.00000
0.3500	25.38752	111.02001	0.00000
0.4000	25.40708	111.02001	0.00000
0.4500	25.42714	111.02001	0.00000
0.5000	25.44761	111.02001	0.00000
0.5500	25.46842	111.02001	0.00000
0.6000	25.48939	111.02001	0.00000
0.6500	25.51030	111.02001	0.00000

0.7000	25.53093	111.02001	0.00000
0.7500	25.55119	111.02001	0.00000
0.8000	25.57023	111.02001	0.00000
0.8500	25.58779	111.02001	0.00000
0.9000	25.60306	111.02001	0.00000
0.9500	25.61562	111.02001	0.00000
1.0000	25.62243	111.02001	0.00000
1.0500	25.62660	111.02001	0.00000
1.1000	25.62596	111.02001	0.00000
1.1500	25.62155	111.02001	0.00000
1.2000	25.61445	111.02001	0.00000
1.2500	25.60563	111.02001	0.00000
1.3000	25.59593	111.02001	0.00000
1.3500	25.58602	111.02001	0.00000
1.4000	25.57640	111.02001	0.00000
1.4500	25.56744	111.02001	0.00000
1.5000	25.55936	111.02001	0.00000
1.5500	25.55226	111.02001	0.00000
1.6000	25.54620	111.02001	0.00000
1.6500	25.54113	111.02001	0.00000
1.7000	25.53699	111.02001	0.00000
1.7500	25.53368	111.02001	0.00000
1.8000	25.53107	111.02001	0.00000
1.8500	25.52907	111.02001	0.00000
1.9000	25.52755	111.02001	0.00000
1.9500	25.52642	111.02001	0.00000
2.0000	25.52559	111.02001	0.00000
2.0500	25.52500	111.02001	0.00000
2.1000	25.52457	111.02001	0.00000
2.1500	25.52427	111.02001	0.00000
2.2000	25.52406	111.02001	0.00000
2.2500	25.52392	111.02001	0.00000
2.3000	25.52382	111.02001	0.00000
2.3500	25.52376	111.02001	0.00000
2.4000	25.52371	111.02001	0.00000
2.4500	25.52368	111.02001	0.00000
0.0000	25.52367	111.02001	0.00000
0.0500	25.52366	111.02001	0.00000
0.1000	25.52365	111.02001	0.00000
0.1500	25.52365	111.02001	0.00000
0.2000	25.52365	111.02001	0.00000
0.2500	25.52365	111.02001	0.00000
0.3000	25.52366	111.02001	0.00000
0.3500	25.52364	111.02001	0.00000
0.4000	25.52355	111.02001	0.00000
0.4500	25.52318	111.02001	0.00000
0.5000	25.52231	111.02001	0.00000
0.5500	25.52030	111.02001	0.00000
0.6000	25.51642	111.02001	0.00000
0.6500	25.51000	111.02001	0.00000
0.7000	25.50064	111.02001	0.00000
0.7500	25.48829	111.02001	0.00000
0.8000	25.47319	111.02001	0.00000
0.8500	25.45579	111.02001	0.00000
0.9000	25.43655	111.02001	0.00000
0.9500	25.41591	111.02001	0.00000
1.0000	25.39429	111.02001	0.00000
1.0500	25.37207	111.02001	0.00000
1.1000	25.34961	111.02001	0.00000
1.1500	25.32727	111.02001	0.00000
1.2000	25.30540	111.02001	0.00000
1.2500	25.28431	111.02001	0.00000
1.3000	25.26428	111.02001	0.00000

1.3500	25.24551	111.02001	0.00000
1.4000	25.22817	111.02001	0.00000
1.4500	25.21230	111.02001	0.00000
1.5000	25.19793	111.02001	0.00000
1.5500	25.18499	111.02001	0.00000
1.6000	25.17340	111.02001	0.00000
1.6500	25.16303	111.02001	0.00000
1.7000	25.15376	111.02001	0.00000
1.7500	25.14545	111.02001	0.00000
1.8000	25.13800	111.02001	0.00000
1.8500	25.13128	111.02001	0.00000
1.9000	25.12519	111.02001	0.00000
1.9500	25.11966	111.02001	0.00000
2.0000	25.11460	111.02001	0.00000
2.0500	25.10997	111.02001	0.00000
2.1000	25.10569	111.02001	0.00000
2.1500	25.10174	111.02001	0.00000
2.2000	25.09808	111.02001	0.00000
2.2500	25.09467	111.02001	0.00000
2.3000	25.09148	111.02001	0.00000
2.3500	25.08850	111.02001	0.00000
2.4000	25.08571	111.02001	0.00000
2.4500	25.08308	111.02001	0.00000
2.5000	25.08061	111.02001	0.00000
2.5500	25.07828	111.02001	0.00000
2.6000	25.07608	111.02001	0.00000
2.6500	25.07399	111.02001	0.00000
2.7000	25.07201	111.02001	0.00000
2.7500	25.07013	111.02001	0.00000
2.8000	25.06835	111.02001	0.00000
2.8500	25.06665	111.02001	0.00000
2.9000	25.06503	111.02001	0.00000
2.9500	25.06349	111.02001	0.00000
3.0000	25.06202	111.02001	0.00000
3.0500	25.06061	111.02001	0.00000
3.1000	25.05926	111.02001	0.00000

Source Code Listing for Thermal Analysis during a Flow Stoppage

```
c This program calculates the temperature transient following a flow
c interruption during elution of an ion exchange column.  It includes the
c effects of thermal conduction to the column walls and natural convection due
c to temperature and concentration gradients in the bulk solution.  The effect
c of plutonium and nitric acid on the bulk solution density is included.  The
c program solves finite difference energy balance equations with forward time
c differencing, donor cell differencing for convection terms, and centered
space
c differencing for conduction terms.  A list of variable definitions follows.
c
c aa = parameter for calculation of saturated plutonium concentration
c ab = parameter for calculation of saturated nitrate concentration
c af1 = area factor for inward radial flow, 1/cm
c af2 = area factor for outward radial flow, 1/cm
c alpha = thermal diffusivity, cm2/s
c an1 = fraction of void volume that lies between resin beads
c area = partial cross-sectional area of resin bed for calculation node, cm2
c areat = total cross-sectional area of resin bed, cm2
c br = nondimensional radial dispersion coefficient for heat transfer equation
c brc = nondimensional radial dispersion coefficient for acid mass transfer
c       equation
c brc1 = nondimensional radial dispersion coefficient for acid mass transfer
```

```
c        equation, due to area change with radius
c bry = nondimensional radial dispersion coefficient for Pu mass transfer
c       equation
c bry1 = nondimensional radial dispersion coefficient for Pu mass transfer
c        equation, due to area change with radius
c br1 = nondimensional radial dispersion coefficient for heat transfer
equation,
c       due to area change with radius
c bz = nondimensional axial dispersion coefficient for heat transfer equation
c bzc = nondimensional axial dispersion coefficient for acid mass transfer
c       equation
c bzy = nondimensional axial dispersion coefficient for Pu mass transfer
c       equation
c c = volume fraction acid in solution between resin beads
c cold = volume fraction acid in solution between resin beads at previous time
c        step
c cond = thermal conductivity of resin, cal/cm/s/C
c caps = heat capacity of liquid, cal/g/C
c capsb = heat capacity of liquid in volume below resin bed, cal/g/C
c capst = heat capacity of liquid in volume above resin bed, cal/g/C
c cfeed = plutonium concentration in feed solution, g/cc
c cf30 = volume fraction of feed solution occupied by HNO3
c cmax2 = maximum total nitrate concentration for a saturated plutonium nitrate
c         solution, gmole/lit
c cmx2 = 1 - nitric acid concentration/nitric acid concentration at saturation
c cno3 = nitric acid concentration, gmole/lit
c cno3b = nitric acid concentration in liquid volume below resin bed, gmole/lit
c cno3t = nitric acid concentration in liquid volume above resin bed, gmole/lit
c cno32 = saturation nitrate concentration, calculated as a function of
c         plutonium concentration, gmole/lit
c cp = heat capacity, cal/g/C
c cpu = local plutonium concentration in solution between resin beads,
gmole/lit
c cpu2 = saturation plutonium concentration, calculated as a function of acid
c        concentration, gmole/lit
c cpair = heat capacity of ambient air, cal/g/C
c cond = thermal conductivity
c dca = change in volume fraction acid in solution between resin beads at each
c       calculation node
c den1 = density of pure HNO3, g/cc
c den1b = density of pure HNO3 in liquid volume below resin bed, g/cc
c den1t = density of pure HNO3 in liquid volume above resin bed, g/cc
c den2 = density of pure H2O, g/cc
c den2b = density of pure H2O in liquid volume below resin bed, g/cc
c den2t = density of pure H2O in liquid volume above resin bed, g/cc
c dens = liquid density, g/cc
c densb = liquid density in volume below resin bed, g/cc
c denst = liquid density in volume above resin bed, g/cc
c dfpor = pore diffusivity, cm2/s
c dfprt = particle diffusivity, cm2/s
c df = change in adsorbed plutonium concentration during feeding and elution,
c      g/cc
c diffc = molecular diffusivity of HNO3, cm2/s
c diffcb = molecular diffusivity of HNO3 in volume below resin bed, cm2/s
c diffct = molecular diffusivity of HNO3 in volume above resin bed, cm2/s
c diffy = molecular diffusivity of Pu, cm2/s
c diffyb = molecular diffusivity of Pu in volume below resin bed, cm2/s
c diffyt = molecular diffusivity of Pu in volume above resin bed, cm2/s
c dpart = resin bead diameter, cm
c dr = radial discretization step size, cm
c dt = time step, s
c dz = axial discretization step size, cm
c dy2 = change in plutonium concentration in solution between resin beads, g/cc
```

```
c dy2a = change in plutonium concentration in solution between resin beads at
c        each calculation node, g/cc
c emiss = surface emissivity for column
c f = plutonium loading on resin, g/cc
c fact1 = factor for iterative calculation of modified stream function
c fact2 = factor for iterative calculation of modified stream function
c fact3 = factor for iterative calculation of modified stream function
c fact4 = factor for iterative calculation of modified stream function
c fact5 = factor for iterative calculation of modified stream function
c fcx = weighting factor for elution of plutonium by acid dilution
c ffac = multiplying factor for rate of adsorption of plutonium by resin,
c        added to prevent adsorption rate from exceeding rates of convection
c        and dispersion mass transfer and thereby making calculation
numerically
c        unstable
c ffcx = weighting factor for elution of plutonium by acid dilution
c ffeed = maximum plutonium resin bed concentration, g/cc
c fold = plutonium loading on resin at previous time step, g/cc
c fpu = multiplier used to maintain constant plutonium content in column
c fx = total bulk plutonium concentration, g/cc
c grav = gravitational acceleration, cm/s2
c gzn = update value of modified stream function, cc/s
c gzo = value of stream function for previous iteration, cc/s
c hconv = coefficient for natural convection heat transfer to sides of column,
c         also overall convective heat transfer coefficient, cal/cm2/s/C
c hconv2 = coefficient for natural convection heat transfer to top and bottom
c          of column, cal/cm2/s/C
c hconv2b = overall coefficient for natural convection heat transfer to liquid
c           volume at bottom of column, cal/cm2/s/C
c hconv2t = overall coefficient for natural convection heat transfer to liquid
c           volume at top of column, cal/cm2/s/C
c hconv3 = coefficient for forced convection heat transfer to column, assuming
c          an ambient air velocity of 30 cm/s (1 ft/s)
c hrad = radiation heat transfer coefficient, cal/cm2/s/C
c htc = heat transfer coefficient, cal/cm2/s/C
c htc2b = overall heat transfer coefficient for liquid volume below resin bed,
c         cal/cm2/s/C
c htc2t = overall heat transfer coefficient for liquid volume above resin bed,
c         cal/cm2/s/C
c iht = indicator for heat transfer to ambient air (iht=1) or to water-cooled
c       jacket (iht=2) (A heat transfer coefficient is calculated for heat
c       transfer to ambient air, and a surface temperature is specified for
c       cooling by a water jacket.)
c indk = indicator that tells whether rate of adsorption of plutonium exceeds
c        rate of mass transfer due to convection and dispersion.  If true,
c        indk=1 and adsorption rate is halved.  If false, indk=0.
c ind3 = indicator that tells whether solution between the resin beads is
c       saturated with plutonium (If ind=1, solution is saturated and program
c       is directed to loading rate calculations in subroutine feeder; if
ind=0,
c       solution is not saturated and program is directed to elution rate
c       calculations in subroutine elut.)
c ldown = depth of liquid in column below resin bed, cm
c lup = height of liquid in column above resin bed, cm
c length = column length, cm
c m = number of axial discretization nodes
c mlen = effective mixing length for axial dispersion coefficient, ratio of
c        dispersion coefficient to mixing velocity, cm
c mlen2 = mixing cell height for axial dispersion due to Benard cell mixing, cm
c n = number of radial discretization nodes
c nu = Nusselt number for natural convection heat transfer to ambient air from
c      sides of column
c nu2 = Nusselt number for natural convection heat transfer to ambient air from
```

```
c       top and bottom of column
c nu3 = Nusselt number for forced convection heat transfer to ambient air,
c       assuming air velocity of 30 cm/s (1 ft/s)
c phi = updated temperature, K
c poszn = distance from top of resin bed, cm
c pr = Prandtl number for ambient air
c pua = total amount of plutonium adsorbed on the resin beads, g
c puao = initial total amount of plutonium adsorbed on the resin beads, g
c puf = total amount of plutonium in the column, g
c pusn = total amount of plutonium in solution, g
c pusno = initial total amount of plutonium in solution, g
c put = amount of plutonium fed to the column, g
c qdot = heat production rate due to plutonium loading, cal/cc/s
c radius = column radius, cm
c rayl = Rayleigh number for natural convection heat transfer to ambient air
c        from sides of column
c rayl2 = Rayleigh number for natural convection heat transfer to ambient air
c         from top and bottom of column
c re = Reynolds number for forced convection heat transfer to column, assuming
c      an ambient air velocity of 30 cm/s (1 ft/s)
c rhcp = product of density and heat capacity for bulk resin bed, cal/cc/C
c rhcp2 = product of density and heat capacity for solution between
c         resin beads, cal/cc/C
c rhcp2b = product of density and heat capacity for liquid volume below resin
c          bed, cal/cc/C
c rhcp2t = product of density and heat capacity for liquid volume above resin
c          bed, cal/cc/C
c rho = bulk density, g/cc
c rhoair = density of air at resin column surface, g/cc
c rhoamb = density of ambient air, g/cc
c rrn = updated radial position, used in radial integrations of flow rates and
c       temperatures, cm
c rro = second previous radial position, cm
c rrr = previous radial position, cm
c rr1 = radial position of calculation node, cm
c r1 = radial position, cm
c r2 = radial position; cm
c qdot = heat production rate due to plutonium loading, cal/cc/s
c sc1 = volume fraction acid in solution in liquid volume at top of column
c sc1a = change in volume fraction acid in solution in liquid volume at top of
c        column
c sc2 = volume fraction acid in solution in liquid volume at bottom of column
c sc2a = change in volume fraction acid in solution in liquid volume at bottom
c        of column
c sf = stream function for superficial velocities in resin bed, cm2/s
c sigma = Boltzmann's constant, cal/cm2/s/K4
c stem1 = temperature in the liquid volume above the resin bed, K
c stem1a = change in temperature in the liquid volumn above the resin bed, K
c stem1c = temperature in the liquid volumn above the resin bed, C
c stem2 = temperature in the liquid volume below the resin bed, K
c stem2a = change in temperature in the liquid volume below the resin bed, K
c stem2c = temperature in the liquid volume below the resin bed, C
c stp = total number of calculation time steps
c sumsf = integration parameter for stream function calculation, cc/s
c svz2 = integrated volumetric flow rate out column, cc/s
c sy21 = plutonium concentration in liquid volume at top of column, g/cc
c sy21a = change in plutonium concentration in liquid volume at top of column,
c         g/cc
c sy22 = plutonium concentration in liquid volumn at bottom of column, g/cc
c sy22a = change in plutonium concentration in liquid volume at bottom of
c         column, g/cc
c tamb = ambient temperature, K
c tau = time constant for loading plutonium on resin, s
```

```
c tau2 = time constant for eluting plutonium from resin, s
c tcn1 = thermal conductivity for nitric acid, cal/cm/s/C
c tcn1b = thermal conductivity for nitric acid in volume below resin bed,
c         cal/cm/s/C
c tcn1t = thermal conductivity for nitric acid in volume above resin bed,
c         cal/cm/s/C
c tcn2 = thermal conductivity for water, cal/cm/s/C
c tcn2b = thermal conductivity for water in volume below resin bed, cal/cm/s/C
c tcn2t = thermal conductivity for water in volume above resin bed, cal/cm/s/C
c tcnm = thermal conductivity for bulk resin bed, cal/cm/s/C
c tcnmb = thermal conductivity for liquid in volume below resin bed, cal/cm/s/C
c tcnmt = thermal conductivity for liquid in volume above resin bed, cal/cm/s/C
c telut = time for elution, s
c temb = product of volumetric flow rate out of resin bed and temperature at
c        bottom of resin bed, K-cc/s
c temch = maximum temperature in column, C
c temc = temperature, C
c temp = temperature, K
c time = elapsed time after flow is interrupted, m
c time2 = elapsed time after flow is interrupted, s
c tinte = time interval for printing maximum resin bed temperatures, s
c tinte2 = time for printing bed temperature profiles, s
c tload = time for loading, s
c twash = time for washing, s
c tzero = initial temperature, K
c tzero = temperature of solution entering column, K
c vbot = volume of liquid below resin bed, cc
c vel2 = velocity of ambient air, cm/s
c vfi = volume fraction in resin bead pores, prior to adsorption of plutonium
c vfi2 = volume fraction in resin bead pores, after adsorption of plutonium
c vfi3 = volume fraction in resin bead pores, adjusted for actual amount of
c        plutonium adsorbed
c vfo = volume fraction occupied by flowing solution
c vis = liquid dynamic viscosity, g/cm/s
c visb = dynamic viscosity for liquid volume below resin bed, g/cm/s
c vist = dynamic viscosity for liquid volume above resin bed, g/cm/s
c visair = dynamic viscosity of ambient air, g/cm/s .
c vmid = total volume of resin bed, cc
c vr = superficial radial velocity, cm/s
c vtop = volume of liquid above resin bed, cc
c vz = superficial axial velocity, cm/s
c vzsum = integration parameter for integration of axial velocity profile, cc/s
c wf = weighting factor for axial dispersion coefficient, based on distance
c      over which density difference for natural convection velocity is based
c x = fractional loading of plutonium on resin
c xg = elution rate constant, 1/s
c xkair = thermal conductivity of ambient air, cal/cm/s/C
c xold = fractional loading of plutonium on resin at previous time step
c xx1 = mass fraction nitric acid in the solution between resin beads
c xx1b = mass fraction nitric acid in liquid volume below resin bed
c xx1t = mass fraction nitric acid in liquid volume above resin bed
c y = bulk plutonium concentration in pore solution, g/cc
c yold = bulk plutonium concentration in pore solution at previous step, g/cc
c y2 = bulk plutonium concentration in solution between resin beads, g/cc
c y2feed = plutonium concentration in solution entering resin bed, g/cc
c y2max = maximum bulk plutonium concentration in solution between resin beads
c         during elution, g/cc
c y2max2 = maximum plutonium concentration for a saturated plutonium nitrate
c          solution, gmole/lit
c y2mx2 = maximum bulk plutonium concentration in solution between resin beads
c         during elapsed time during elution, g/cc
c y2old = bulk plutonium concentration in solution between resin beads at
c         previous time step, g/cc
```

```
c zden = area-average liquid density at a given level, g/cc
c zvz = area-average superficial axial velocity at a given level, cm/s
      implicit double precision (a-z)
      integer i,j,m,mm1,n,nn1,q,r,s,stp,stp2,iskip,is1,mm2,mm3,jlev,
     1ind,istt,istc,ind3,k,nct,nct2,nct3,nlast,j4,jmax,jmin,mm4,mm5,
     2indk,k2
      character*10 ch1
      dimension temc(200,100),temp(200,100),phi(200,100)
      dimension vr(200,100),vz(200,100),dens(200,100),zden(200)
      dimension vis(200,100),zvz(200)
      dimension area(100),den2(200,100),br1(200,100),brc1(200,100)
      dimension rhcp(200,100),tcnm(200,100)
      dimension bz(200,100),br(200,100),fx(200,100)
      dimension rr1(100),bzc(200,100),brc(200,100)
      dimension bzy(200,100),bry(200,100),bry1(200,100)
      dimension rhcp2(200,100),diffy(200,100),diffc(200,100)
      dimension alpha(200,100),gzo(200,100),gzn(200,100)
      dimension dy2a(200,100),dca(200,100)
      dimension af1(100),af2(100),vzm(200)
      dimension c(200,100),cold(200,100),f(200,100),fold(200,100),
     1x(200,100),xold(200,100),y2(200,100),y2old(200,100),
     2den1(200,100),ind3(200,100),cmx2(200,100),
     3y2max(200,100)
      character tab /9/
      common /elut1/c,cold,f,fold,x,xold,y2,y2old,den1,
     2y2max,y2max2,cmax2,ind3,cmx2,cfeed,dt,tau,tau2,ffeed,an1,
     3dpart,dz,xg,vfi2,vfo,ind
c Input ambient temperature in °C.
      read (5,*) tamb
c Input indicator for air heat transfer (iht=1) or water jacket heat transfer
c (iht=2).
      read (5,*) iht
c Input resin bed depth in centimeters.
      read (5,*) length
c Input height above resin bed in centimeters.
      read (5,*) lup
c Input depth below resin bed in centimeters.
      read (5,*) ldown
c Input column radius in centimeters.
      read (5,*) radius
c Input particle diameter in centimeters.
      read (5,*) dpart
c Input pore diffusivity in cm2/sec.
      read (5,*) dfpor
c Input particle diffusivity in cm2/sec.
      read (5,*) dfprt
c Input void fraction.
      read (5,*) vfo
c Input maximum resin bed concentration.
      read (5,*) ffeed
c Input radiolysis heat in cal/cc/sec.
      read (5,*) qdot
c Input initial volume fraction nitric acid.
      read (5,*) cf30
      pi=4.*atan(1.)
c Input parameters for mixing model.
      rho=1.23
      cp=0.76
      tzero=tzero+273.0
      tamb=tamb+273.0
      grav=980.
      vfi=0.132
      vfi2=0.160
```

```
      an1=1./(vfo+vfi)
      mlen=0.48*radius
      mlen2=1.6*radius
      tau=dpart**2/(4.*pi**2)/dfprt
      tau2=dpart**2/(24.*0.75*dfpor)
      xg=4.*pi**2*dfpor/dpart**2*(1.-vfo)
      aa=1.
      k=1
      do while (k.le.8)
      aa=aa-(aa**5-2.9**4*0.26*(4.-aa)*(1.-aa)**4)
     1/(4.*aa**4+2.9**4*0.26*((1.-aa)**4+4.*(4.-aa)*(1.-aa)**3))
      k=k+1
      enddo
      y2max2=aa/2.9/(4.-aa)/(1.-aa)
      cmax2=4.*y2max2
c Input # steps in axial, radial directions.
      read (5,*) m,n
c Input # of data points to skip between reading each concentration.
      read (5,*) iskip
c Input starting line for reading concentration profile.
      read (5,*) istc
      m=m+1
      n=n+1
      mm1=m-1
      nn1=n-1
      mm2=m+1
      mm3=m+2
      mm4=m-2
      mm5=m-3
c Read in temperatures.
      read (9,*) stem1c
      read (9,*) stem2c
      stem1=stem1c+273.
      stem2=stem2c+273.
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      read (9,*) temp(j,i)
      temp(j,i)=temp(j,i)+273.
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      read (9,*) temc(j,i)
      temp(j,i)=temc(j,i)+273.
      i=i+1
      enddo
      j=j-1
      enddo
c Read in concentrations.
      istc=istc-1
      k=1
      do while (k.le.istc)
      read (8,*)
      k=k+1
      enddo
      j=1
      do while (j.le.m)
      read (8,*) tim,ln,xc1,xc2,xc3,xc4,xc5
      i=1
      do while (i.le.n)
      c(j,i)=cf30*(1.-xc1)
```

```
      cold(j,i)=c(j,i)
      f(j,i)=xc2
      fold(j,i)=xc2
      y2(j,i)=xc3+xc4
      y2old(j,i)=xc3+xc4
      xold(j,i)=xc2/ffeed
      fx(j,i)=xc5
      i=i+1
      enddo
      j=j+1
      enddo
      read (8,*) tim,ln,xc1,xc2,xc3,xc4,xc5
      sc1=cf30*(1.-xc1)
      sy21=xc3
      read (8,*) tim,ln,xc1,xc2,xc3,xc4,xc5
      sc2=cf30*(1.-xc1)
      sy22=xc3
      dz=length/dfloat(mm1)
      dr=radius/dfloat(nn1)
      vtop=pi*radius**2*lup
      vbot=pi*radius**2*ldown
      vmid=pi*radius**2*length
      area(1)=pi*(0.5/dfloat(nn1)*radius)**2
      area(n)=pi*(1.-((dfloat(nn1)-0.5)/dfloat(nn1))**2)*radius**2
      i=2
      do while (i.le.nn1)
      r1=(dfloat(i)-1.5)/dfloat(nn1)*radius
      r2=(dfloat(i)-0.5)/dfloat(nn1)*radius
      area(i)=pi*(r2**2-r1**2)
      i=i+1
      enddo
      areat=pi*radius**2
c Input starting time increment in seconds.
      read (5,*) dts
c Input later time increment in seconds.
      read (5,*) dtf
c Input number of time steps for starting time increment.
      read (5,*) stp2
c Input total number of time steps.
      read (5,*) stp
c Initialize radial positions.
      rr1(1)=0.
      i=2
      do while (i.le.n)
      rr1(i)=rr1(i-1)+dr
      i=i+1
      enddo
      af1(1)=0.
      af2(1)=0.
      i=2
      do while (i.le.nn1)
      af1(i)=2.*rr1(i)/(rr1(i)**2-rr1(i-1)**2)
      af2(i)=2.*rr1(i)/(rr1(i+1)**2-rr1(i)**2)
      i=i+1
      enddo
c Initialize physical properties.
      stem1c=stem1-273.
      stem2c=stem2-273.
      den1t=1.6603-1.9894e-3*stem1c
      den1b=1.6603-1.9894e-3*stem2c
      den2t=0.99683-1.3010e-4*stem1c-2.4358e-6*stem1c**2
      den2b=0.99683-1.3010e-4*stem2c-2.4358e-6*stem2c**2
      xx1t=den1t*sc1/(den1t*sc1+den2t*(1.-sc1))
```

```
 xx1b=den1b*sc2/(den1b*sc2+den2b*(1.-sc2))
 cno3t=sc1*den1t*1000./63.-4.*sy21*1000./238.
 cno3b=sc2*den1b*1000./63.-4.*sy22*1000./238.
 if (cno3t.lt.0.) cno3t=0.
 if (cno3b.lt.0.) cno3b=0.
 denst=den2t+0.4277*sy21*1000./238.+0.031*cno3t
 densb=den2b+0.4277*sy22*1000./238.+0.031*cno3b
 capst=1.0104-(1.419-0.3*(stem1c-20.)/80.)*xx1t
1+(2.005-0.6*(stem1c-20.)/80.)*xx1t**2
2-(1.147-0.3*(stem1c-20.)/80.)*xx1t**3
 capsb=1.0104-(1.419-0.3*(stem2c-20.)/80.)*xx1b
1+(2.005-0.6*(stem2c-20.)/80.)*xx1b**2
2-(1.147-0.3*(stem2c-20.)/80.)*xx1b**3
 rhcp2t=denst*capst
 rhcp2b=densb*capsb
 tcn1t=6.1388e-4+1.3951e-6*stem1c
 tcn2t=1.3518e-3+2.7903e-6*stem1c
 tcn1b=6.1388e-4+1.3951e-6*stem2c
 tcn2b=1.3518e-3+2.7903e-6*stem2c
 tcnmt=tcn1t*xx1t+tcn2t*(1.-xx1t)
 tcnmb=tcn1b*xx1t+tcn2b*(1.-xx1t)
 vist=0.01002*10.**((1.3272*(20.-stem1c)-0.001053*
1(stem1c-20.)**2)/(stem1c+105.))
 visb=0.01002*10.**((1.3272*(20.-stem2c)-0.001053*
1(stem2c-20.)**2)/(stem2c+105.))
 diffyt=8.314/96500**2*stem1*(1./1.+1./3.)/(1./69.5
1+1./71.4)
 diffct=8.314/96500**2*stem1*(1./1.+1./1.)/(1./349.8
1+1./71.4)
 diffyb=8.314/96500**2*stem2*(1./1.+1./3.)/(1./69.5
1+1./71.4)
 diffcb=8.314/96500**2*stem2*(1./1.+1./1.)/(1./349.8
1+1./71.4)
 i=1
 do while (i.le.n)
 j=1
 do while (j.le.m)
 temc(j,i)=temp(j,i)-273.
 den1(j,i)=1.6603-1.9894e-3*temc(j,i)
 den2(j,i)=0.99683-1.3010e-4*temc(j,i)-2.4358e-6*temc(j,i)**2
 xx1=den1(j,i)*c(j,i)/(den1(j,i)*c(j,i)+den2(j,i)*(1.-c(j,i)))
 cno3=c(j,i)*den1(j,i)*1000./63.-4.*y2(j,i)*1000./238./vfo
 if (cno3.lt.0.) cno3=0.
 dens(j,i)=den2(j,i)+0.4277*y2(j,i)*1000./238./vfo+0.031*cno3
 tcn1=6.1388e-4+1.3951e-6*temc(j,i)
 tcn2=1.3518e-3+2.7903e-6*temc(j,i)
 tcnm(j,i)=tcn1*xx1+tcn2*(1.-xx1)
 tcnm(j,i)=0.5*(5.e-4+tcnm(j,i))
 caps=1.0104-(1.419-0.3*(temc(j,i)-20.)/80.)*xx1
1+(2.005-0.6*(temc(j,i)-20.)/80.)*xx1**2
2-(1.147-0.3*(temc(j,i)-20.)/80.)*xx1**3
 rhcp(j,i)=0.5*1.25*0.52+0.5*dens(j,i)*caps
 rhcp2(j,i)=dens(j,i)*caps
 vis(j,i)=0.01002*10.**((1.3272*(20.-temc(j,i))-0.001053*
1(temc(j,i)-20.)**2)/(temc(j,i)+105.))
 alpha(j,i)=tcnm(j,i)/rhcp(j,i)
 diffy(j,i)=8.314/96500**2*temp(j,i)*(1./1.+1./3.)/(1./69.5
1+1./71.4)
 diffc(j,i)=8.314/96500**2*temp(j,i)*(1./1.+1./1.)/(1./349.8
1+1./71.4)
 bz(j,i)=(alpha(j,i)+2.*dpart*abs(vz(j,i))*rhcp2(j,i)/rhcp(j,i))
1*dt/dz**2
 bzy(j,i)=(diffy(j,i)+2.*dpart*abs(vz(j,i)))*dt/dz**2
```

```
      bzc(j,i)=(diffc(j,i)+2.*dpart*abs(vz(j,i)))*dt/dz**2
      j=j+1
      enddo
      i=i+1
      enddo
      j=1
      do while (j.le.m)
      br(j,1)=(alpha(j,1)+0.4*dpart*abs(vz(j,1))*rhcp2(j,1)/rhcp(j,1))
     1*dt*4./dr**2
      br1(j,1)=0.
      bry(j,1)=(diffy(j,1)+0.4*dpart*abs(vz(j,1)))*dt*4./dr**2
      bry1(j,1)=0.
      brc(j,1)=(diffc(j,1)+0.4*dpart*abs(vz(j,1)))*dt*4./dr**2
      brc1(j,1)=0.
      j=j+1
      enddo
      i=2
      do while (i.le.n)
      j=1
      do while (j.le.m)
      br(j,i)=(alpha(j,i)+0.4*dpart*abs(vz(j,i))*rhcp2(j,i)
     1/rhcp(j,i))*dt/dr**2
      br1(j,i)=(alpha(j,i)+0.4*dpart*abs(vz(j,i))*rhcp2(j,i)
     1/rhcp(j,i))*dt/dr/rr1(i)/2.
      bry(j,i)=(diffy(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr**2
      bry1(j,i)=(diffy(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr/rr1(i)/2.
      brc(j,i)=(diffc(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr**2
      brc1(j,i)=(diffc(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr/rr1(i)/2.
      j=j+1
      enddo
      i=i+1
      enddo
c Write input variables to output.
      write (1,100) length,radius,m,n
      write (1,121) dpart,vfo
      write (1,101) dt,htc,htc2
c      write (1,110) sumf
  100 format ('L (CM) =',f7.3,2x,'R (CM) =',f7.3,2x,'AXIAL NODES =',
     1i3,2x,'RADIAL NODES =',i3)
  101 format ('DT (S) =',f8.4,2x,'HSID (CAL/CM2/S/°C) =',f9.6,
     12x,'HTOP (CAL/CM2/S/°C) =',f9.6)
c  110 format ('PU CHARGE (GM) =',f11.4)
      write (2,*) 'TIME (MIN)',tab,'TMAX (°C)',tab,'PUA (GM)',
     1tab,'PUT (GM)'
  121 format ('PART DIAM (CM) =',f8.4,2x,'VOID FRACTION =',f7.4)
c Find highest temperature.
      temch=0.
      j=1
      do while (j.le.m)
      i=1
      do while (i.le.n)
      if (temch.lt.temc(j,i)) then
      temch=temc(j,i)
      endif
      i=i+1
      enddo
      j=j+1
      enddo
c Calculate total amounts of plutonium and plutonium adsorbed.
      pua=0.
      puf=areat*(lup*sy21+ldown*sy22)
      pusn=areat*(lup*sy21+ldown*sy22)
      i=1
```

```
      do while (i.le.n)
      pua=pua+0.5*dz*area(i)*(f(1,i)+f(m,i))
      puf=puf+0.5*dz*area(i)*(fx(1,i)+fx(m,i))
      pusn=pusn+0.5*dz*area(i)*(y2(1,i)+y2(m,i))
      i=i+1
      enddo
      j=2
      do while (j.le.mm1)
      i=1
      do while (i.le.n)
      pua=pua+dz*f(j,i)*area(i)
      puf=puf+dz*fx(j,i)*area(i)
      pusn=pusn+dz*y2(j,i)*area(i)
      i=i+1
      enddo
      j=j+1
      enddo
      puao=pua
      pusno=pusn
      time=0.
      if (abs(nint(time/2)-time/2).lt.0.001)
     1write (2,301) time,tab,temch,tab,pua,tab,puf
c Set initial velocities equal to zero.
      j=1
      do while (j.le.m)
      vzm(j)=0.
      i=1
      do while (i.le.n)
      vz(j,i)=0.
      vr(j,i)=0.
      i=i+1
      enddo
      j=j+1
      enddo
      i=1
c Write initial concentrations and temperatures.
      write (1,*) 'TIME (MIN)',tab,'POS (CM)',tab,'LOAD',tab,
     1'TCEN (°C)',tab,'TMID (°C)',tab,'TOUT (°C)'
      write (6,*) 'TIME (MIN)',tab,'LENGTH (CM)',tab,'ELU (VF)',tab,
     1'ADS (G/CC)',tab,'LIQ (G/CC)',tab,'TOT (G/CC)'
      j=1
      write (1,102) time,tab,-1.,tab,0.,tab,stem1c,tab,stem1c,
     1tab,stem1c
      do while (j.le.m)
      poszn=length*(j-1)/(m-1)
      write (1,102) time,tab,poszn,tab,f(j,1),tab,temc(j,1),tab,
     1temc(j,5),tab,temc(j,n)
      write (6,400) time,tab,poszn,tab,c(j,1),tab,f(j,1),tab,y2(j,1),
     1tab,fx(j,1)
      j=j+1
      enddo
      write (1,102) time,tab,-2.,tab,0.,tab,stem2c,tab,stem2c,tab,
     1stem2c
c Perform transient calculations.
      indk=1
      s=1
      time2=0.
      dt=dts
      do while (s.le.stp.and.vtop.gt.5.)
      if (s.gt.stp2) dt=dtf
c Calculate heat transfer coefficients
      sigma=1.356e-12
      emiss=0.6
```

```
 tavg=0.
 q=1
 do while (q.le.m)
 tavg=tavg+temp(q,n)
 q=q+1
 enddo
 tavg=tavg/m
 hrad=emiss*sigma*(tavg**3+tavg**2*tamb
1+tavg*tamb**2+tamb**3)
 cpair=3.5*1.9872*29.
 xkair=0.0262/100*0.2389
 visair=0.00018
 pr=cpair*visair/xkair
 rhoamb=29./82.057/tamb
 rhoair=29./82.057/tavg
 rayl=grav*(length+lup+ldown)**3*rhoamb*(rhoamb-rhoair)
1/visair**2*pr
 if (rayl.gt.0.) then
 nu=0.59*rayl**0.25
 endif
 if (nu.lt.1.) nu=1.
 hconv=nu*xkair/(length+lup+ldown)
 vel2=30.0
 re=2.*radius*vel2*rhoamb/visair
 nu3=0.683*re**0.466*pr**(1./3.)
 hconv3=xkair*nu3/2./radius
 if (hconv.lt.hconv3) hconv=hconv3
 htc=hconv+hrad
 hrad=emiss*sigma*(stem1**3+stem1**2*tamb+stem1*tamb**2+tamb**3)
 rhoair=29./82.057/stem1
 rayl=grav*(length+lup+ldown)**3*rhoamb*(rhoamb-rhoair)
1/visair**2*pr
 if (rayl.gt.0.) then
 nu=0.59*rayl**0.25
 endif
 if (nu.lt.1.) nu=1.
 hconv=nu*xkair/(length+lup+ldown)
 rayl2=grav*(2.*radius)**3*rhoamb*(rhoamb-rhoair)/visair**2*pr
 if (rayl2.gt.0.) then
 nu2=0.54*rayl2**0.25
 endif
 if (nu2.lt.1.) nu2=1.
 hconv2=nu2*xkair/2./radius
 hconv2t=(2.*radius*lup*hconv+radius**2*hconv2)/(2.*radius*lup
1+radius**2)
 vel2=30.0
 re=2.*radius*vel2*rhoamb/visair
 nu3=0.683*re**0.466*pr**(1./3.)
 hconv3=xkair*nu3/2./radius
 if (hconv2t.lt.hconv3) hconv2t=hconv3
 htc2t=hrad+hconv2t
 hrad=emiss*sigma*(stem2**3+stem2**2*tamb+stem2*tamb**2+tamb**3)
 rhoair=29./82.057/stem2
 rayl=grav*(length+lup+ldown)**3*rhoamb*(rhoamb-rhoair)
1/visair**2*pr
 if (rayl.gt.0.) then
 nu=0.59*rayl**0.25
 endif
 if (nu.lt.1.) nu=1.
 hconv=nu*xkair/(length+lup+ldown)
 rayl2=grav*(2.*radius)**3*rhoamb*(rhoamb-rhoair)/visair**2*pr
 if (rayl2.gt.0.) then
 nu2=0.27*rayl2**0.25
```

```
      endif
      if (nu2.lt.1.) nu2=1.
      hconv2=nu2*xkair/2./radius
      hconv2b=(2.*radius*ldown*hconv+radius**2*hconv2)/(2.*radius*ldown
     1+radius**2)
      vel2=30.0
      re=2.*radius*vel2*rhoamb/visair
      nu3=0.683*re**0.466*pr**(1./3.)
      hconv3=xkair*nu3/2./radius
      if (hconv2b.lt.hconv3) hconv2b=hconv3
      htc2b=hrad+hconv2b
c Calculate changes in temperatures and concentrations in the liquid volumes
c above and below the resin bed.
      sy21a=0.
      sy22a=0.
      sc1a=0.
      sc2a=0.
      stem1a=sy21*0.8*qdot*dt/rhcp2t
      stem2a=sy22*0.8*qdot*dt/rhcp2b
      i=1
      do while (i.le.n)
      stem1a=stem1a+(bz(1,i)*dz/lup+vzm(1)*dt/lup/dz)
     1*area(i)/areat*(temp(2,i)-stem1)
      stem2a=stem2a+(bz(m,i)*dz/ldown+vzm(m)*dt/ldown/dz)
     1*area(i)/areat*(temp(mm1,i)-stem2)
      sy21a=sy21a+(bzy(1,i)*dz/lup+vzm(1)*dt/lup/dz)
     1*area(i)/areat*(y2old(2,i)/vfo-sy21)
      sy22a=sy22a+(bzy(m,i)*dz/ldown+vzm(m)*dt/ldown/dz)
     1*area(i)/areat*(y2old(mm1,i)/vfo-sy22)
      sc1a=sc1a+(bzc(1,i)*dz/lup+vzm(1)*dt/lup/dz)
     1*area(i)/areat*(cold(2,i)-sc1)
      sc2a=sc2a+(bzc(m,i)*dz/ldown+vzm(m)*dt/ldown/dz)
     1*area(i)/areat*(cold(mm1,i)-sc2)
      i=i+1
      enddo
      stem1a=stem1a+htc2t*(radius**2+2.*lup*radius)/lup/radius**2
     1*dt/rhcp2t*(tamb-stem1)
      stem2a=stem2a+htc2b*(radius**2+2.*ldown*radius)/ldown/radius**2
     1*dt/rhcp2b*(tamb-stem2)
c Initialize changes in concentrations.
      j=1
      do while (j.le.m)
      i=1
      do while (i.le.n)
      dy2a(j,i)=0.
      dca(j,i)=0.
      i=i+1
      enddo
      j=j+1
      enddo
c Calculate temperature and concentration changes in middle of column.
      q=2
      do while (q.le.mm1)
      phi(q,n)=temp(q,n)+fx(q,n)*0.8*qdot*dt/rhcp(q,n)
      phi(q,n)=phi(q,n)+bz(q,n)*(temp(q+1,n)+temp(q-1,n)-2.*temp(q,n))
     1+htc*dt/rhcp(q,n)*(1./dr+1./2./rr1(n))*(tamb-temp(q,n))
     2+br(q,nn1)*(temp(q,nn1)-temp(q,n))
     3+2.*br1(q,n)*(temp(q,n)-temp(q,nn1))
     4+vzm(q)*rhcp2(q,n)/rhcp(q,n)*dt/dz**2*(temp(q+1,n)+temp(q-1,n)
     5-2.*temp(q,n))
      dy2a(q,n)=dy2a(q,n)+bzy(q,n)/vfo*(y2old(q+1,n)+y2old(q-1,n)
     1-2.*y2old(q,n))
     2+bry(q,nn1)/vfo*(y2old(q,nn1)-y2old(q,n))
```

```
3+2.*bry1(q,n)/vfo*(y2old(q,n)-y2old(q,nn1))
4+vzm(q)/vfo*dt/dz**2*(y2old(q+1,n)+y2old(q-1,n)-2.*y2old(q,n))
 dca(q,n)=dca(q,n)+an1*(bzc(q,n)*(cold(q+1,n)+cold(q-1,n)
1-2.*cold(q,n))
2+brc(q,nn1)*(cold(q,nn1)-cold(q,n))
3+2.*brc1(q,n)*(cold(q,n)-cold(q,nn1))
4+vzm(q)*dt/dz**2*(cold(q+1,n)+cold(q-1,n)-2.*cold(q,n)))
 if (vz(q,n).gt.0.) then
 phi(q,n)=phi(q,n)-vz(q,n)*dt/dz*rhcp2(q,n)/rhcp(q,n)
1*(temp(q,n)-temp(q-1,n))
 dy2a(q,n)=dy2a(q,n)-vz(q,n)*dt/dz/vfo*(y2old(q,n)-y2old(q-1,n))
 dca(q,n)=dca(q,n)-an1*vz(q,n)*dt/dz*(cold(q,n)-cold(q-1,n))
 endif
 if (vz(q,n).lt.0.) then
 phi(q,n)=phi(q,n)-vz(q,n)*dt/dz*rhcp2(q,n)/rhcp(q,n)
1*(temp(q+1,n)-temp(q,n))
 dy2a(q,n)=dy2a(q,n)-vz(q,n)*dt/dz/vfo*(y2old(q+1,n)-y2old(q,n))
 dca(q,n)=dca(q,n)-an1*vz(q,n)*dt/dz*(cold(q+1,n)-cold(q,n))
 endif
 q=q+1
 enddo
 q=2
 do while (q.le.mm1)
 r=2
 do while (r.le.nn1)
 phi(q,r)=temp(q,r)+fx(q,r)*0.8*qdot*dt/rhcp(q,r)
 phi(q,r)=phi(q,r)+bz(q,r)*(temp(q+1,r)+temp(q-1,r)-2.*temp(q,r))
1+br(q,r-1)*(temp(q,r-1)-temp(q,r))
2+br(q,r)*(temp(q,r+1)-temp(q,r))
3+br1(q,r)*(temp(q,r+1)-temp(q,r-1))
4+vzm(q)*rhcp2(q,r)/rhcp(q,r)*dt/dz**2*(temp(q+1,r)+temp(q-1,r)
5-2.*temp(q,r))
 dy2a(q,r)=dy2a(q,r)+bzc(q,r)/vfo*(y2old(q+1,r)+y2old(q-1,r)
1-2.*y2old(q,r))
2+bry(q,r-1)/vfo*(y2old(q,r-1)-y2old(q,r))
3+bry(q,r)/vfo*(y2old(q,r+1)-y2old(q,r))
4+bry1(q,r)/vfo*(y2old(q,r+1)-y2old(q,r-1))
5+vzm(q)/vfo*dt/dz**2*(y2old(q+1,r)+y2old(q-1,r)-2.*y2old(q,r))
 dca(q,r)=dca(q,r)+an1*(bzc(q,r)*(cold(q+1,r)+cold(q-1,r)
1-2.*cold(q,r))
2+brc(q,r-1)*(cold(q,r-1)-cold(q,r))
3+brc(q,r)*(cold(q,r+1)-cold(q,r))
4+brc1(q,r)*(cold(q,r+1)-cold(q,r-1))
5+vzm(q)*dt/dz**2*(cold(q+1,r)+cold(q-1,r)-2.*cold(q,r)))
 if (vz(q,r).gt.0.) then
 phi(q,r)=phi(q,r)-vz(q,r)*dt/dz*rhcp2(q,r)/rhcp(q,r)
1*(temp(q,r)-temp(q-1,r))
 dy2a(q,r)=dy2a(q,r)-vz(q,r)*dt/dz/vfo*(y2old(q,r)-y2old(q-1,r))
 dca(q,r)=dca(q,r)-an1*vz(q,r)*dt/dz*(cold(q,r)-cold(q-1,r))
 endif
 if (vz(q,r).lt.0.) then
 phi(q,r)=phi(q,r)-vz(q,r)*dt/dz*rhcp2(q,r)/rhcp(q,r)
1*(temp(q+1,r)-temp(q,r))
 dy2a(q,r)=dy2a(q,r)-vz(q,r)*dt/dz/vfo*(y2old(q+1,r)-y2old(q,r))
 dca(q,r)=dca(q,r)-an1*vz(q,r)*dt/dz*(cold(q+1,r)-cold(q,r))
 endif
 if (vr(q,r).gt.0.) then
 phi(q,r+1)=phi(q,r+1)-vr(q,r)*dt*af2(r)*rhcp2(q,r)/rhcp(q,r)
1*(temp(q,r+1)-temp(q,r))
 dy2a(q,r+1)=dy2a(q,r+1)-vr(q,r)*dt*af2(r)/vfo
1*(y2old(q,r+1)-y2old(q,r))
 dca(q,r+1)=dca(q,r+1)-an1*vr(q,r)*dt*af2(r)
1*(cold(q,r+1)-cold(q,r))
```

```
      endif
      if (vr(q,r).lt.0.) then
      phi(q,r-1)=phi(q,r-1)-vr(q,r)*dt*af1(r)*rhcp2(q,r)/rhcp(q,r)
     1*(temp(q,r)-temp(q,r-1))
      dy2a(q,r-1)=dy2a(q,r-1)-vr(q,r)*dt*af1(r)/vfo
     1*(y2old(q,r)-y2old(q,r-1))
      dca(q,r-1)=dca(q,r-1)-an1*vr(q,r)*dt*af1(r)
     1*(cold(q,r)-cold(q,r-1))
      endif
      r=r+1
      enddo
      q=q+1
      enddo
      j=1
      do while (j.le.m)
      i=1
      do while (i.le.n)
      if (y2old(j,i)+dy2a(j,i).lt.0.) then
      dy2a(j,i)=-y2old(j,i)
      endif
      i=i+1
      enddo
      j=j+1
      enddo
c Calculate changes in plutonium concentrations due to adsorption.
c This is accomplished by calling the subroutine crite to determine if the bulk
c solution is saturated with plutonium.  If the solution is saturated, the
c subroutine feeder is called to calculate the adsorption rate.
      q=2
      do while (q.le.mm1)
      r=2
      do while (r.le.n)
      call crite (q,r,m)
      r=r+1
      enddo
      q=q+1
      enddo
c Calculate the amount of plutonium adsorbed by calling subroutine feeder.  A
c check is added to ensure that the rate of adsorption does not exceed the
c rate of mass transfer by convection and dispersion.  If this should occur,
c numerical instabilities would result.
      indk=1
      ffac=1.
      k2=1
      do while (k2.le.20.and.indk.eq.1)
      q=2
      do while (q.le.mm1)
      r=2
      do while (r.le.n)
      dy2=dy2a(q,r)
      dc=dca(q,r)
      if (ind3(q,r).eq.1) then
      call feeder (q,r,m,dy2,dc,df,ffac)
      endif
      f(q,r)=fold(q,r)+df
      y2(q,r)=y2old(q,r)+dy2
      c(q,r)=cold(q,r)+dc
      x(q,r)=xold(q,r)+df/ffeed
      r=r+1
      enddo
      q=q+1
      enddo
      indk=0
```

```
      i=2
      do while (i.le.n)
      j=1
      do while (j.le.m-3)
      if (y2(j,i).lt.y2(j+1,i).and.y2(j+1,i).gt.y2(j+2,i).and.
     1y2(j+2,i).lt.y2(j+3,i)) indk=1
      if (y2(j,i).gt.y2(j+1,i).and.y2(j+1,i).lt.y2(j+2,i).and.
     1y2(j+2,i).gt.y2(j+3,i)) indk=1
      if (c(j,i).lt.c(j+1,i).and.c(j+1,i).gt.c(j+2,i).and.
     1c(j+2,i).lt.c(j+3,i)) indk=1
      if (c(j,i).gt.c(j+1,i).and.c(j+1,i).lt.c(j+2,i).and.
     1c(j+2,i).gt.c(j+3,i)) indk=1
      j=j+1
      enddo
      i=i+1
      enddo
      if (indk.eq.1) ffac=ffac/2.
      k2=k2+1
      enddo
c Update conditions in liquid layers at top and bottom of resin bed.
      stem1=stem1+stem1a
      sy21=sy21+sy21a
      sc1=sc1+sc1a
      stem2=stem2+stem2a
      sy22=sy22+sy22a
      sc2=sc2+sc2a
c Specify symmetry condition at column center.
      j=1
      do while (j.le.m)
      phi(j,1)=phi(j,2)
      temp(j,1)=temp(j,2)
      temc(j,1)=temc(j,2)
      c(j,1)=c(j,2)
      cold(j,1)=cold(j,2)
      y2(j,1)=y2(j,2)
      y2old(j,1)=y2old(j,2)
      f(j,1)=f(j,2)
      fold(j,1)=fold(j,2)
      fx(j,1)=fx(j,2)
      xold(j,1)=xold(j,2)
      j=j+1
      enddo
c Specify top and bottom conditions.
      i=1
      do while (i.le.n)
      phi(1,i)=stem1
      y2(1,i)=sy21*vfo
      c(1,i)=sc1
      phi(m,i)=stem2
      y2(m,i)=sy22*vfo
      c(m,i)=sc2
      i=i+1
      enddo
c Update temperatures and concentrations.  Calculate densities and viscosities.
      stem1c=stem1-273.
      stem2c=stem2-273.
      den1t=1.6603-1.9894e-3*stem1c
      den1b=1.6603-1.9894e-3*stem2c
      den2t=0.99683-1.3010e-4*stem1c-2.4358e-6*stem1c**2
      den2b=0.99683-1.3010e-4*stem2c-2.4358e-6*stem2c**2
      xx1t=den1t*sc1/(den1t*sc1+den2t*(1.-sc1))
      xx1b=den1b*sc2/(den1b*sc2+den2b*(1.-sc2))
      cno3t=sc1*den1t*1000./63.-4.*sy21*1000./238.
```

```
 cno3b=sc2*den1b*1000./63.-4.*sy22*1000./238.
 if (cno3t.lt.0.) cno3t=0.
 if (cno3b.lt.0.) cno3b=0.
 denst=den2t+0.4277*sy21*1000./238.+0.031*cno3t
 densb=den2b+0.4277*sy22*1000./238.+0.031*cno3b
 capst=1.0104-(1.419-0.3*(stem1c-20.)/80.)*xx1t
1+(2.005-0.6*(stem1c-20.)/80.)*xx1t**2
2-(1.147-0.3*(stem1c-20.)/80.)*xx1t**3
 capsb=1.0104-(1.419-0.3*(stem2c-20.)/80.)*xx1b
1+(2.005-0.6*(stem2c-20.)/80.)*xx1b**2
2-(1.147-0.3*(stem2c-20.)/80.)*xx1b**3
 rhcp2t=denst*capst
 rhcp2b=densb*capsb
 tcn1t=6.1388e-4+1.3951e-6*stem1c
 tcn2t=1.3518e-3+2.7903e-6*stem1c
 tcn1b=6.1388e-4+1.3951e-6*stem2c
 tcn2b=1.3518e-3+2.7903e-6*stem2c
 tcnmt=tcn1t*xx1t+tcn2t*(1.-xx1t)
 tcnmb=tcn1b*xx1t+tcn2b*(1.-xx1t)
 vist=0.01002*10.**((1.3272*(20.-stem1c)-0.001053*
1(stem1c-20.)**2)/(stem1c+105.))
 visb=0.01002*10.**((1.3272*(20.-stem2c)-0.001053*
1(stem2c-20.)**2)/(stem2c+105.))
 diffyt=8.314/96500**2*stem1*(1./1.+1./3.)/(1./69.5
1+1./71.4)
 diffct=8.314/96500**2*stem1*(1./1.+1./1.)/(1./349.8
1+1./71.4)
 diffyb=8.314/96500**2*stem2*(1./1.+1./3.)/(1./69.5
1+1./71.4)
 diffcb=8.314/96500**2*stem2*(1./1.+1./1.)/(1./349.8
1+1./71.4)
 i=1
 do while (i.le.n)
 j=1
 do while (j.le.m)
 temp(j,i)=phi(j,i)
 if (iht.eq.2.and.i.eq.n) temp(j,i)=tamb
 temc(j,i)=phi(j,i)-273.0
 if (iht.eq.2.and.i.eq.n) temc(j,i)=tamb-273.0
 cold(j,i)=c(j,i)
 fold(j,i)=f(j,i)
 y2old(j,i)=y2(j,i)
 fx(j,i)=f(j,i)+y2(j,i)
 xold(j,i)=x(j,i)
 den1(j,i)=1.6603-1.9894e-3*temc(j,i)
 den2(j,i)=0.99683-1.3010e-4*temc(j,i)-2.4358e-6*temc(j,i)**2
 xx1=den1(j,i)*c(j,i)/(den1(j,i)*c(j,i)+den2(j,i)*(1.-c(j,i)))
 cno3=c(j,i)*den1(j,i)*1000./63.-4.*y2(j,i)*1000./238./vfo
 if (cno3.lt.0.) cno3=0.
 dens(j,i)=den2(j,i)+0.4277*y2(j,i)*1000./238./vfo+0.031*cno3
 tcn1=6.1388e-4+1.3951e-6*temc(j,i)
 tcn2=1.3518e-3+2.7903e-6*temc(j,i)
 tcnm(j,i)=tcn1*xx1+tcn2*(1.-xx1)
 tcnm(j,i)=0.5*(tcnm(j,i)+5.e-4)
 caps=1.0104-(1.419-0.3*(temc(j,i)-20.)/80.)*xx1
1+(2.005-0.6*(temc(j,i)-20.)/80.)*xx1**2
2-(1.147-0.3*(temc(j,i)-20.)/80.)*xx1**3
 rhcp(j,i)=0.5*1.25*0.52+0.5*dens(j,i)*caps
 rhcp2(j,i)=dens(j,i)*caps
 vis(j,i)=0.01002*10.**((1.3272*(20.-temc(j,i))-0.001053*
1(temc(j,i)-20.)**2)/(temc(j,i)+105.))
 alpha(j,i)=tcnm(j,i)/rhcp(j,i)
 diffy(j,i)=8.314/96500**2*temp(j,i)*(1./1.+1./3.)/(1./69.5
```

```
     1+1./71.4)
      diffc(j,i)=8.314/96500**2*temp(j,i)*(1./1.+1./1.)/(1./349.8
     1+1./71.4)
      bz(j,i)=(alpha(j,i)+2.*dpart*abs(vz(j,i))*rhcp2(j,i)/rhcp(j,i))
     1*dt/dz**2
      bzy(j,i)=(diffy(j,i)+2.*dpart*abs(vz(j,i)))*dt/dz**2.
      bzc(j,i)=(diffc(j,i)+2.*dpart*abs(vz(j,i)))*dt/dz**2
      j=j+1
      enddo
      i=i+1
      enddo
      j=1
      do while (j.le.m)
      br(j,1)=(alpha(j,1)+0.4*dpart*abs(vz(j,1))*rhcp2(j,1)/rhcp(j,1))
     1*dt*4./dr**2
      br1(j,1)=0.
      bry(j,1)=(diffy(j,1)+0.4*dpart*abs(vz(j,1)))*dt*4./dr**2
      bry1(j,1)=0.
      brc(j,1)=(diffc(j,1)+0.4*dpart*abs(vz(j,1)))*dt*4./dr**2
      brc1(j,1)=0.
      j=j+1
      enddo
      i=2
      do while (i.le.n)
      j=1
      do while (j.le.m)
      br(j,i)=(alpha(j,i)+0.4*dpart*abs(vz(j,i))*rhcp2(j,i)
     1/rhcp(j,i))*dt/dr**2
      br1(j,i)=(alpha(j,i)+0.4*dpart*abs(vz(j,i))*rhcp2(j,i)
     1/rhcp(j,i))*dt/dr/rr1(i)/2.
      bry(j,i)=(diffy(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr**2
      bry1(j,i)=(diffy(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr/rr1(i)/2.
      brc(j,i)=(diffc(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr**2
      brc1(j,i)=(diffc(j,i)+0.4*dpart*abs(vz(j,i)))*dt/dr/rr1(i)/2.
      j=j+1
      enddo
      i=i+1
      enddo
c Calculate the average density at each level.
      j=1
      do while (j.le.m)
      rrr=dr/2.
      dsum=rrr**2*dens(j,1)
      i=2
      do while (i.le.nn1)
      rro=rrr
      rrr=rrr+dr
      dsum=dsum+(rrr**2-rro**2)*dens(j,i)
      i=i+1
      enddo
      rro=rrr
      rrr=rrr+dr/2.
      dsum=dsum+(rrr**2-rro**2)*dens(j,n)
      zden(j)=dsum/rrr**2
      j=j+1
      enddo
c Calculate superficial velocities.  This calculation is performed using a
c Richardson iteration.
      j=1
      do while (j.le.m)
      gzo(j,1)=0.
      gzo(j,n)=0.
      j=j+1
```

```
      enddo
      fact1=1./(2./dr**2+2./dz**2)
      fact2=1./dz**2
      k=1
      do while (k.le.200)
      i=2
      do while (i.le.nn1)
      fact3=1./dr**2-0.5/rr1(i)/dr
      fact4=1./dr**2+0.5/rr1(i)/dr
      fact5=dpart**2*vfo**4/((1.-vfo)**2*200.)*grav*rr1(i)/2./dr
      gzn(1,i)=fact1*(fact2*(gzo(3,i)-2.*gzo(2,i)+3.*gzo(1,i))
     1+fact3*gzo(1,i+1)+fact4*gzo(1,i-1)
     2+fact5/vis(1,i)*(dens(1,i+1)-dens(1,i-1)))
      gzn(m,i)=fact1*(fact2*(gzo(mm4,i)-2.*gzo(mm1,i)+3.*gzo(m,i))
     1+fact3*gzo(m,i+1)+fact4*gzo(m,i-1)
     2+fact5/vis(m,i)*(dens(m,i+1)-dens(m,i-1)))
      j=2
      do while (j.le.mm1)
      gzn(j,i)=fact1*(fact2*(gzo(j+1,i)+gzo(j-1,i))+fact3*gzo(j,i+1)
     1+fact4*gzo(j,i-1)+fact5/vis(j,i)*(dens(j,i+1)-dens(j,i-1)))
      j=j+1
      enddo
      i=i+1
      enddo
      i=2
      do while (i.le.nn1)
      j=2
      do while (j.le.mm1)
      gzo(j,i)=gzn(j,i)
      j=j+1
      enddo
      i=i+1
      enddo
      k=k+1
      enddo
      i=2
      do while (i.le.nn1)
      vr(1,i)=(gzo(2,i)-gzo(1,i))/dz/rr1(i)
      vr(m,i)=(gzo(m,i)-gzo(mm1,i))/dz/rr1(i)
      vz(1,i)=(gzo(1,i-1)-gzo(1,i+1))/2./dr/rr1(i)
      vz(m,i)=(gzo(m,i-1)-gzo(m,i+1))/2./dr/rr1(i)
      j=2
      do while (j.le.mm1)
      vr(j,i)=(gzo(j+1,i)-gzo(j-1,i))/2./dz/rr1(i)
      vz(j,i)=(gzo(j,i-1)-gzo(j,i+1))/2./dr/rr1(i)
      j=j+1
      enddo
      i=i+1
      enddo
      j=1
      do while (j.le.m)
      vz(j,1)=vz(j,2)
      vz(j,n)=(gzo(j,nn1)-gzo(j,n))/dr/rr1(n)
      j=j+1
      enddo
c Calculate mixing in the resin bed. This is modeled using an axial dispersion
c model for bubble column mixing.
      j=1
      do while (j.le.m)
      vzm(j)=0.
      k=j-1
      do while (k.ge.1)
      wf=exp(-(dfloat(j-k)+dfloat(j-k-1))*dz/mlen2)
```

```
      if (zden(k).gt.zden(j)) then
      vzm(j)=vzm(j)+dpart**2*vfo**4/((1.-vfo)**2*400.*vis(k,1))
     1*(zden(k)-zden(j))*grav*mlen*wf
      endif
      k=k-1
      enddo
      k=j+1
      do while (k.le.m)
      wf=exp(-(dfloat(k-j)+dfloat(k-j-1))*dz/mlen2)
      if (zden(k).lt.zden(j)) then
      vzm(j)=vzm(j)+dpart**2*vfo**4/((1.-vfo)**2*400.*vis(k,1))
     1*(zden(j)-zden(k))*grav*mlen*wf
      endif
      k=k+1
      enddo
      j=j+1
      enddo
c Calculate total amounts of plutonium and plutonium adsorbed.  Renormalize to
c ensure that the total amount of plutonium is conserved.
      pua=0.
      puf=areat*(lup*sy21+ldown*sy22)
      pusn=areat*(lup*sy21+ldown*sy22)
      i=1
      do while (i.le.n)
      pua=pua+0.5*dz*area(i)*(f(1,i)+f(m,i))
      puf=puf+0.5*dz*area(i)*(fx(1,i)+fx(m,i))
      pusn=pusn+0.5*dz*area(i)*(y2(1,i)+y2(m,i))
      i=i+1
      enddo
      j=2
      do while (j.le.mm1)
      i=1
      do while (i.le.n)
      pua=pua+dz*f(j,i)*area(i)
      puf=puf+dz*fx(j,i)*area(i)
      pusn=pusn+dz*y2(j,i)*area(i)
      i=i+1
      enddo
      j=j+1
      enddo
      fpu=(pusno-pua+puao)/pusn
      sy21=sy21*fpu
      sy22=sy22*fpu
      i=1
      do while (i.le.n)
      j=1
      do while (j.le.m)
      y2old(j,i)=y2old(j,i)*fpu
      y2(j,i)=y2(j,i)*fpu
      j=j+1
      enddo
      i=i+1
      enddo
c Write transient temperatures and velocities to output files.
      time2=time2+dt
      time=time2/60.
      if (abs(nint(time/30.)-time/30.).lt.dt/2./1800.) then
c Write headers for output files.
      write (1,*) 'TIME (MIN)',tab,'POS (CM)',tab,'LOAD',tab,
     1'TCEN (°C)',tab,'TMID (°C)',tab,'TOUT (°C)'
      write (6,*) 'TIME (MIN)',tab,'LENGTH (CM)',tab,'ELU (VF)',tab,
     1'ADS (G/CC)',tab,'LIQ (G/CC)',tab,'TOT (G/CC)'
      write (10,*) 'TIME (MIN)',tab,'LENGTH (CM)',tab,'IND',tab,
```

```
     1'VFA',tab,'VFAM'
      j=1
      write (1,102) time,tab,-1.,tab,0.,tab,stem1c,tab,stem1c,
     1tab,stem1c
      do while (j.le.m)
      poszn=length*(j-1)/(m-1)
      write (1,102) time,tab,poszn,tab,f(j,1),tab,temc(j,1),tab,
     1temc(j,5),tab,temc(j,n)
  102 format (f9.4,a1,f10.5,a1,f7.5,3(a1,f9.4))
      write (6,400) time,tab,poszn,tab,c(j,1),tab,f(j,1),tab,y2(j,1),
     1tab,fx(j,1)
  400 format (f9.4,a1,f10.5,4(a1,f9.5))
      write (10,500) time,tab,poszn,tab,ind3(j,1),tab,
     1y2(j,1),tab,y2max(j,i),tab,cmx2(j,1)
  500 format (f9.4,a1,f10.5,a1,i2,3(a1,f9.5))
      j=j+1
      enddo
      write (1,102) time,tab,-2.,tab,0.,tab,stem2c,tab,stem2c,tab,
     1stem2c
      endif
c Find highest temperature.
      temch=0.
      j=1
      do while (j.le.m)
      i=1
      do while (i.le.n)
      if (temch.lt.temc(j,i)) then
      temch=temc(j,i)
      endif
      i=i+1
      enddo
      j=j+1
      enddo
      if (abs(nint(time/2.)-time/2.).lt.dt/4./60.)
     1write (2,301) time,tab,temch,tab,pua,tab,puf
  301 format (f11.4,a1,f10.5,a1,f11.5,a1,f11.5)
c Increment transient calculation loop.
      s=s+1
      enddo
c Write temperature profile to output file.
      write (7,*) stem1c
      write (7,*) stem2c
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      write (7,*) temc(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (7,*) temc(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
c Write total plutonium concentrations to output file.
      write (21,*) 1000.*sy21
      write (21,*) 1000.*sy22
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
```

```
      write (21,*) 1000.*fx(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (21,*) 1000.*fx(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
c Write solution plutonium concentrations to output file.
      write (22,*) 1000.*sy21
      write (22,*) 1000.*sy22
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      write (22,*) 1000.*y2(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (22,*) 1000.*y2(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
c Write densities to output file.
      write (23,*) denst
      write (23,*) densb
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      write (23,*) dens(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (23,*) dens(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
c Write axial velocities to output file.
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      write (16,*) 10000.*vz(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (16,*) 10000.*vz(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
c Write radial velocities to output file.
      j=m
      do while (j.ge.1)
      i=n
```

```
      do while (i.ge.1)
      write (15,*) 10000.*vr(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (15,*) 10000.*vr(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
c Write modified stream function profile to output file.  The stream function
c is modified by multiplying by the radius.  This modified function gives a
c profile of the velocity multiplied by the radius.
      j=m
      do while (j.ge.1)
      i=n
      do while (i.ge.1)
      write (17,*) 10000.*gzo(j,i)
      i=i-1
      enddo
      i=2
      do while (i.le.n)
      write (17,*) 10000.*gzo(j,i)
      i=i+1
      enddo
      j=j-1
      enddo
      stop
      end

      subroutine feeder (j,i,m,dy2,dc,df,ffac)
c Compute changes in the plutonium concentrations during feeding or washing.
      implicit double precision (a-z)
      integer i,j,m,ind,ind3
      dimension c(200,100),cold(200,100),f(200,100),fold(200,100),
     1x(200,100),xold(200,100),y2(200,100),y2old(200,100),
     2den1(200,100),ind3(200,100),cmx2(200,100),
     3y2max(200,100)
      common /elut1/c,cold,f,fold,x,xold,y2,y2old,den1,
     2y2max,y2max2,cmax2,ind3,cmx2,cfeed,dt,tau,tau2,ffeed,an1,
     3dpart,dz,xg,vfi2,vfo,ind
      if (xold(j,i).lt.1.) then
      if (xold(j,i).gt.1.d-3)
     1xte=xold(j,i)+dt/tau*xold(j,i)**2/(log((1.+xold(j,i))/
     2(1.-xold(j,i)))-2.*xold(j,i))
      if (xold(j,i).le.1.d-3.and.xold(j,i).gt.1.d-8)
     1xte=xold(j,i)+dt/tau*1.5/xold(j,i)
      if (xte.gt.1.) xte=1.
      if (xte.lt.xold(j,i)) xte=xold(j,i)
      df=ffeed*(xte-xold(j,i))
      endif
      if (xold(j,i).ge.1.) then
      df=0.
      endif
      df=ffac*df
      dy2=dy2-df
      if (y2(j,i)+dy2.lt.0.) then
      dy2=-y2(j,i)
      df=dy2+y2(j,i)
      endif
      dc=dc+4.*dy2*63./238./vfo/den1(j,i)
      if (cold(j,i)+dc.lt.0.) dc=-cold(j,i)
```

```
      return
      end

      subroutine crite (j,i,m)
c Determine if solution is saturated with plutonium.
      implicit double precision (a-z)
      integer i,j,m,ind,ind3
      dimension c(200,100),cold(200,100),f(200,100),fold(200,100),
     1x(200,100),xold(200,100),y2(200,100),y2old(200,100),
     2den1(200,100),ind3(200,100),cmx2(200,100),
     3y2max(200,100)
      common /elut1/c,cold,f,fold,x,xold,y2,y2old,den1,
     2y2max,y2max2,cmax2,ind3,cmx2,cfeed,dt,tau,tau2,ffeed,an1,
     3dpart,dz,xg,vfi2,vfo,ind
      cno3=cold(j,i)*1000./63.*den1(j,i)
      cpu=y2old(j,i)*1000./238./vfo
      cpu2=0.
      if (cno3.le.cmax2) then
      cpu2=y2max2
      else
      aa=1.
      k=1
      do while (k.le.8)
      aa=aa-(2.9*aa**3*(1.-aa)*cno3-2.9**4*0.26*(1.-aa)**4-aa**4)
     1/(3.*2.9*aa**2*(1.-aa)*cno3-2.9*aa**3*cno3
     2+4.*2.9**4*0.26*(1.-aa)**3-4.*aa**3)
      k=k+1
      enddo
      cpu2=cno3/aa-1./2.9/(1.-aa)
      endif
      y2max(j,i)=vfo*0.238*cpu2
      if (cpu.ge.cpu2) then
      ind3(j,i)=1
      else
      ind3(j,i)=0
      endif
      ab=1.
      if (cpu.gt.1.d-8) then
      k=1
      do while (k.le.5)
      ab=(1/3*ab**(4./3.)*cpu**(1./3.)+0.26**(1./3.)*2.9)/
     1(4./3.*ab**(1./3.)*cpu**(1./3.)+0.26**(1./3.)*2.9)
      k=k+1
      enddo
      endif
      if (1.-ab.gt.1.d-8)
     1cno32=ab*cpu+ab/2.9/(1.-ab)
      if (cno32.gt.cmax2.and.cpu.gt.y2max2) cno32=cmax2
      cmx2(j,i)=1.-cno3/cno32
      if (cmx2(j,i).lt.0.) cmx2(j,i)=0.
      if (cmx2(j,i).gt.1.) cmx2(j,i)=1.
      return
      end
```

Sample Input Listing for Thermal Analysis during a Flow Stoppage

```
25.
1
34.1
2.54
2.54
```

```
3.73
0.04
0.0000012
0.0000012
0.33
0.05
0.13384
0.274
48,24
0
56
1.0
1.0
36000
36000
```

Sample Output Listing for Thermal Analysis during a Flow Stoppage

TIME (MIN)	TMAX (°C)	PUA (GM)	PUT (GM)
0.0000	25.22047	32.37012	92.77193
2.0000	26.93949	52.49574	92.77060
4.0000	28.46193	52.49574	92.77031
6.0000	29.86569	52.49574	92.77018
8.0000	31.28179	52.49584	92.77034
10.0000	32.68002	52.49584	92.77021
12.0000	34.04205	52.49584	92.77013
14.0000	35.36350	52.49584	92.77006
16.0000	36.65562	52.49584	92.76999
18.0000	37.89207	52.49584	92.76994
20.0000	39.08942	52.49584	92.76988
22.0000	40.23867	52.49584	92.76983
24.0000	41.32405	52.49584	92.76979
26.0000	42.37786	52.49584	92.76975
28.0000	43.37710	52.49584	92.76970
30.0000	44.31755	52.49584	92.76967
32.0000	45.20343	52.49584	92.76963
34.0000	46.06065	52.49584	92.76960
36.0000	46.86528	52.49584	92.76956
38.0000	47.61947	52.49584	92.76953
40.0000	48.32548	52.49584	92.76950
42.0000	48.98560	52.49584	92.76947
44.0000	49.60221	52.49584	92.76944
46.0000	50.19265	52.49584	92.76941
48.0000	50.74232	52.49584	92.76938
50.0000	51.25312	52.49584	92.76935
52.0000	51.72684	52.49584	92.76932
54.0000	52.16518	52.49584	92.76929
56.0000	52.56972	52.49584	92.76925
58.0000	52.94194	52.49584	92.76921
60.0000	53.28325	52.49584	92.76916
62.0000	53.59492	52.49584	92.76911
64.0000	53.87821	52.49584	92.76905
66.0000	54.13431	52.49584	92.76899
68.0000	54.36438	52.49584	92.76891
70.0000	54.56974	52.50657	92.77111
72.0000	54.75289	52.77818	92.77528
74.0000	54.91620	53.35178	92.77915
76.0000	55.07611	54.18079	92.77872
78.0000	55.22506	54.94720	92.77121
80.0000	55.35980	55.06963	92.77535
82.0000	55.48226	56.00477	92.77889

84.0000	55.59502	56.97047	92.77800
86.0000	55.70022	58.04878	92.77972
88.0000	55.79999	59.11180	92.77893
90.0000	55.89599	60.11682	92.77780
92.0000	55.98986	61.15192	92.77700
94.0000	56.08296	62.01746	92.77119
96.0000	56.18058	62.24486	92.77112
98.0000	56.28321	62.79764	92.77110
100.0000	56.37576	62.79794	92.77104
102.0000	56.45868	62.79827	92.77099
104.0000	56.53382	62.89013	92.77522
106.0000	56.60308	63.24992	92.77102
108.0000	56.66752	63.75953	92.77515
110.0000	56.72871	64.34706	92.77488
112.0000	56.78715	64.99140	92.77465
114.0000	56.84271	65.61835	92.77118
116.0000	56.89453	65.73035	92.77019
118.0000	56.94245	65.73035	92.76997
120.0000	56.98685	65.73044	92.77122
122.0000	57.02833	65.88174	92.77423
124.0000	57.07485	66.18198	92.77272
126.0000	57.12701	66.49636	92.77272
128.0000	57.17706	66.82122	92.77393
130.0000	57.22449	67.15398	92.77268
132.0000	57.26885	67.47528	92.77380
134.0000	57.30976	67.82819	92.77365
136.0000	57.34691	68.18432	92.77355
138.0000	57.38007	68.53554	92.77350
140.0000	57.40907	68.89245	92.77339
142.0000	57.43374	69.00623	92.77239
144.0000	57.45418	69.17869	92.77236
146.0000	57.47221	69.35574	92.77234
148.0000	57.50065	69.53382	92.77232
150.0000	57.52513	69.70978	92.77291
152.0000	57.54589	69.88289	92.77226
154.0000	57.56310	69.91868	92.77167
156.0000	57.57700	69.91882	92.77167
158.0000	57.58789	69.92985	92.77168
160.0000	57.59607	69.92999	92.77168
162.0000	57.60179	69.93014	92.77168
164.0000	57.60531	69.93028	92.77169
166.0000	57.60684	69.94465	92.77169
168.0000	57.61338	69.98084	92.77171
170.0000	57.62201	70.00483	92.77172
172.0000	57.62887	70.02252	92.77173
174.0000	57.63411	70.04470	92.77172
176.0000	57.63785	70.04484	92.77173
178.0000	57.64022	70.06215	92.77174
180.0000	57.64134	70.06229	92.77175
182.0000	57.64131	70.07895	92.77177
184.0000	57.64022	70.10754	92.77178
186.0000	57.63816	70.13509	92.77179
188.0000	57.63522	70.15646	92.77180
190.0000	57.63370	70.17413	92.77181
192.0000	57.63642	70.19333	92.77183
194.0000	57.63826	70.21095	92.77184
196.0000	57.63929	70.22822	92.77185
198.0000	57.63958	70.23483	92.77186
200.0000	57.63918	70.25028	92.77187
202.0000	57.63814	70.25861	92.77188
204.0000	57.63650	70.26763	92.77189
206.0000	57.63431	70.28337	92.77190
208.0000	57.63161	70.30200	92.77190

210.0000	57.62841	70.31383	92.77191
212.0000	57.62475	70.31397	92.77192
214.0000	57.62352	70.31410	92.77192
216.0000	57.62399	70.31423	92.77193
218.0000	57.62391	70.31436	92.77193
220.0000	57.62331	70.31450	92.77193
222.0000	57.62221	70.31463	92.77194
224.0000	57.62064	70.31476	92.77194
226.0000	57.61860	70.31489	92.77194
228.0000	57.61612	70.31503	92.77194
230.0000	57.61323	70.31516	92.77193
232.0000	57.60993	70.32303	92.77193
234.0000	57.60625	70.32470	92.77193
236.0000	57.60220	70.35917	92.77191
238.0000	57.59782	70.36313	92.77191
240.0000	57.59536	70.36642	92.77191
242.0000	57.59334	70.40199	92.77189
244.0000	57.59092	70.40212	92.77189
246.0000	57.58815	70.40225	92.77189
248.0000	57.58503	70.40238	92.77189
250.0000	57.58162	70.40251	92.77189
252.0000	57.57792	70.43023	92.77188
254.0000	57.57398	70.43652	92.77188
256.0000	57.56981	70.44252	92.77187
258.0000	57.56546	70.44831	92.77187
260.0000	57.56095	70.45388	92.77187
262.0000	57.55630	70.45931	92.77186
264.0000	57.55155	70.50665	92.77185
266.0000	57.54673	70.51685	92.77184
268.0000	57.54187	70.52716	92.77184
270.0000	57.53757	70.52883	92.77184
272.0000	57.53473	70.52896	92.77184
274.0000	57.53188	70.52909	92.77184
276.0000	57.52900	70.52921	92.77184
278.0000	57.52606	70.52934	92.77184
280.0000	57.52308	70.52947	92.77184
282.0000	57.52007	70.52959	92.77184
284.0000	57.51702	70.52972	92.77184
286.0000	57.51394	70.52985	92.77184
288.0000	57.51085	70.52997	92.77183
290.0000	57.50774	70.53010	92.77183
292.0000	57.50462	70.53023	92.77183
294.0000	57.50150	70.53036	92.77183
296.0000	57.49841	70.53048	92.77183
298.0000	57.49652	70.53212	92.77182
300.0000	57.49536	70.53306	92.77182
302.0000	57.49427	70.53390	92.77182
304.0000	57.49326	70.53462	92.77182
306.0000	57.49234	70.53525	92.77182
308.0000	57.49154	70.53557	92.77181
310.0000	57.49086	70.53570	92.77181
312.0000	57.49033	70.53583	92.77181
314.0000	57.49019	70.53596	92.77181
316.0000	57.49182	70.53665	92.77181
318.0000	57.49360	70.53677	92.77181
320.0000	57.49553	70.53690	92.77181
322.0000	57.49762	70.53703	92.77181
324.0000	57.49988	70.53715	92.77180
326.0000	57.50332	70.53728	92.77180
328.0000	57.50765	70.53741	92.77180
330.0000	57.51211	70.53753	92.77180
332.0000	57.51671	70.53766	92.77180
334.0000	57.52243	70.53779	92.77180

336.0000	57.52889	70.53792	92.77180
338.0000	57.53545	70.53804	92.77180
340.0000	57.54289	70.53817	92.77180
342.0000	57.55110	70.53830	92.77179
344.0000	57.55936	70.53842	92.77179
346.0000	57.56892	70.53855	92.77179
348.0000	57.57863	70.53868	92.77179
350.0000	57.58882	70.53880	92.77179
352.0000	57.59985	70.54088	92.77179
354.0000	57.61083	70.54525	92.77179
356.0000	57.62252	70.54621	92.77179
358.0000	57.63467	70.54633	92.77179
360.0000	57.64674	70.54646	92.77179
362.0000	57.65905	70.54659	92.77179
364.0000	57.67219	70.54671	92.77179
366.0000	57.68521	70.54684	92.77179
368.0000	57.69813	70.54697	92.77178
370.0000	57.71093	70.54709	92.77178
372.0000	57.72469	70.54722	92.77178
374.0000	57.73842	70.54735	92.77178
376.0000	57.75202	70.54747	92.77178
378.0000	57.76549	70.54760	92.77178
380.0000	57.77883	70.54773	92.77178
382.0000	57.79204	70.54785	92.77178
384.0000	57.80587	70.54798	92.77178
386.0000	57.81985	70.54811	92.77178
388.0000	57.83368	70.54823	92.77178
390.0000	57.84735	70.54836	92.77178
392.0000	57.86088	70.54848	92.77178
394.0000	57.87426	70.54861	92.77178
396.0000	57.88750	70.54874	92.77178
398.0000	57.90059	70.54886	92.77178
400.0000	57.91354	70.54899	92.77178
402.0000	57.92636	70.54912	92.77178
404.0000	57.93939	70.54924	92.77178
406.0000	57.95270	70.54937	92.77178
408.0000	57.96585	70.54950	92.77178
410.0000	57.97884	70.54962	92.77178
412.0000	57.99168	70.54975	92.77178
414.0000	58.00436	70.54988	92.77178
416.0000	58.01689	70.55000	92.77178
418.0000	58.02927	70.55013	92.77178
420.0000	58.04151	70.55026	92.77178
422.0000	58.05359	70.55038	92.77178
424.0000	58.06554	70.55051	92.77178
426.0000	58.07734	70.55064	92.77178
428.0000	58.08901	70.55400	92.77178
430.0000	58.10054	70.55413	92.77178
432.0000	58.11194	70.55425	92.77178
434.0000	58.12320	70.55438	92.77178
436.0000	58.13434	70.55451	92.77178
438.0000	58.14535	70.55463	92.77178
440.0000	58.15623	70.55476	92.77178
442.0000	58.16699	70.55489	92.77178
444.0000	58.17762	70.55501	92.77178
446.0000	58.18813	70.55514	92.77177
448.0000	58.19853	70.55526	92.77177
450.0000	58.20880	70.55539	92.77177
452.0000	58.21896	70.55552	92.77177
454.0000	58.22900	70.55564	92.77177
456.0000	58.23897	70.55577	92.77177
458.0000	58.24907	70.55590	92.77177
460.0000	58.25906	70.55602	92.77177

462.0000	58.26892	70.55615	92.77177
464.0000	58.27867	70.55628	92.77177
466.0000	58.28830	70.55640	92.77177
468.0000	58.29781	70.55653	92.77177
470.0000	58.30722	70.55665	92.77177
472.0000	58.31651	70.55678	92.77177
474.0000	58.32569	70.55691	92.77177
476.0000	58.33477	70.55703	92.77177
478.0000	58.34375	70.55716	92.77177
480.0000	58.35262	70.55729	92.77177
482.0000	58.36139	70.55741	92.77177
484.0000	58.37006	70.55754	92.77177
486.0000	58.37864	70.55767	92.77177
488.0000	58.38711	70.55779	92.77177
490.0000	58.39550	70.55792	92.77177
492.0000	58.40379	70.55804	92.77177
494.0000	58.41199	70.55817	92.77177
496.0000	58.42011	70.55830	92.77177
498.0000	58.42813	70.55842	92.77177
500.0000	58.43607	70.55855	92.77177
502.0000	58.44392	70.55868	92.77177
504.0000	58.45169	70.55880	92.77177
506.0000	58.45938	70.55893	92.77177
508.0000	58.46699	70.55905	92.77177
510.0000	58.47451	70.55918	92.77177
512.0000	58.48196	70.55931	92.77177
514.0000	58.48934	70.55943	92.77177
516.0000	58.49664	70.55956	92.77177
518.0000	58.50386	70.55969	92.77177
520.0000	58.51101	70.55981	92.77177
522.0000	58.51809	70.55994	92.77177
524.0000	58.52510	70.56006	92.77177
526.0000	58.53204	70.56019	92.77177
528.0000	58.53891	70.56032	92.77177
530.0000	58.54571	70.56044	92.77177
532.0000	58.55245	70.56057	92.77177
534.0000	58.55912	70.56070	92.77177
536.0000	58.56573	70.56082	92.77177
538.0000	58.57228	70.56095	92.77177
540.0000	58.57876	70.56107	92.77177
542.0000	58.58519	70.56120	92.77177
544.0000	58.59155	70.56133	92.77177
546.0000	58.59785	70.56145	92.77177
548.0000	58.60410	70.56158	92.77177
550.0000	58.61029	70.56171	92.77177
552.0000	58.61643	70.56183	92.77177
554.0000	58.62250	70.56196	92.77177
556.0000	58.62853	70.56208	92.77177
558.0000	58.63450	70.56221	92.77177
560.0000	58.64041	70.56234	92.77177
562.0000	58.64627	70.56246	92.77177
564.0000	58.65209	70.56259	92.77177
566.0000	58.65785	70.56271	92.77177
568.0000	58.66355	70.56284	92.77177
570.0000	58.66921	70.56297	92.77177
572.0000	58.67482	70.56309	92.77177
574.0000	58.68038	70.56322	92.77177
576.0000	58.68590	70.56335	92.77177
578.0000	58.69136	70.56347	92.77177
580.0000	58.69678	70.56360	92.77177
582.0000	58.70215	70.56372	92.77177
584.0000	58.70748	70.56385	92.77177
586.0000	58.71276	70.56398	92.77177

588.0000	58.71799	70.56410	92.77177
590.0000	58.72318	70.56423	92.77177
592.0000	58.72833	70.56435	92.77177
594.0000	58.73343	70.56448	92.77177
596.0000	58.73850	70.56461	92.77177
598.0000	58.74351	70.56473	92.77177
600.0000	58.74849	70.56486	92.77177

WSRC INTERNAL DISTRIBUTION

Savannah River Technology Center

M. L. Crowder, 773-24A
F. R. Graham, 773-A
J. R. Knight, 773-A
J. E. Laurinat, 773-A
M. C. Thompson, 773-A
C. R. Wolfe, 773-A
SRTC Records, 773-52A

LANL INTERNAL DISTRIBUTION

Nuclear Materials Technology Division
Power Source Technologies

M. E. Pansoy-Hjelvik, MS E502
K. B. Ramsey, MS E502