SANDIA REPORT
SAND2001-8001
Unlimited Release
Printed January 2001

Elements for Real Time Steady Simulation of Flat Glass Lehr Control

Lee A. Bertram

Prepared by
Sandia National Laboratories
Albuquerque, New Mexico 87185 and Livermore, California 94550

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

Issued by Sandia National Laboratories, operated for the United States Department of Energy by Sandia Corporation.

Available to DOE and DOE contractors from
U.S. Department of Energy
Office of Scientific and Technical Information
P.O. Box 62
Oak Ridge, TN 37831

Telephone: (865)576-8401
Facsimile: (865)576-5728
E-Mail: reports@adonis.osti.gov
Online ordering: http://www.doe.gov/bridge

Available to the public from
U.S. Department of Commerce
National Technical Information Service
5285 Port Royal Rd
Springfield, VA 22161

Telephone: (800)553-6847
Facsimile: (703)605-6900
E-Mail: orders@ntis.fedworld.gov
Online order: http://www.ntis.gov/ordering.htm

SAND2001-8001
Unlimited Release
Printed January, 2001

ELEMENTS FOR REAL TIME STEADY SIMULATION OF FLAT GLASS LEHR CONTROL

Lee A. Bertram
Chemical and Materials Process Modeling
Sandia National Laboratories
Livermore, California 94551-0969

ABSTRACT

In partnership with Ford/Visteon, Sandia National Laboratories began work on design and implementation of improved end-to-end control for float glass production, under DOE Office of Industrial Technology sponsorship. The first task undertaken was to provide an intelligent control for the annealing lehr, both for optimum usage of sensors and for the ability to report digitally the state of the lehr to an end-to-end control system. The heat transfer simulation of the lehr enclosure for that purpose is described here. This includes a closed-form solution for the infinitely wide glass ribbon, which allow robust computations of the thermal profile and inverses of this function for controls use. This also allows useful initial temperature estimates for the case with counterflow in the lehr ducts, which should greatly simplify and speed the convergence of more detailed models in which the glass participates in the radiative exchanges. This is supplemented by software to compute radiative viewfactors of lehr elements for use in finite-width versions of the simulation. A brief discussion of how stresses could be computed from such a thermal solution is given, but not implemented in software.

ACKNOWLEDGEMENTS

The author gratefully acknowledges the support of U. S. Department of Energy/Office of Industrial Technology/Advanced Sensors and Controls under the Glass Labs 99 Program and of Sandia National Laboratories under DOE contract DE-AC04-94AL85000. The heat transfer work was developed with the help of Bill Houf of Sandia, with much guidance from Ford/Visteon personnel V. Henry, A. Huber, K. Bhatia, D. Fryz and C. Dodge. Controls issues were discussed with the Visteon personnel and with D. Schaeffer of Sandia and Prof. P. Smith and D. Smith of the University of Utah. P. Walsh of Sandia has provided continuous guidance and encouragement on many fronts during the developments reported here.

TABLE OF CONTENTS

Lehr Model Requirements....7

1. Plane Parallel Solution....8
   1.1. Software Modules....14
      1.1.2 XENDO()....14
      1.1.3 XCOUNT()....14
      1.1.4 Tofx() Speciamens/Mechanical Testing....14
      1.1.5 ALEHR()Speciamens/Mechanical Testing....14
   1.2. Example Calculation....15
2. Finite Widths and Lengths....15
3. Heat Transfer Within the Ribon....18
4. Stresses....19
5. Discussion....20
6. References....22

APPENDIX A....23

APPENDIX B....25

APPENDIX C....45

APPENDIX D....64

APPENDIX E....66

FIGURES

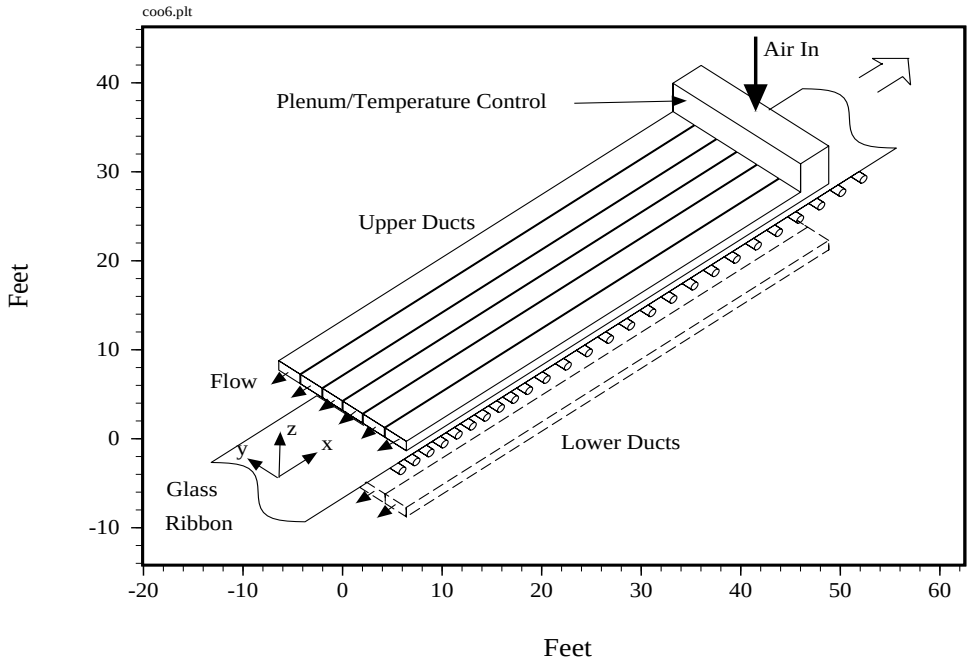
No. **Page**

1. Ribbon/Cooler Temperatures. a) Duct flow is parallel to the glass ribbon motion, and duct fluid is cooling the glass. b) Counterflow solution for same parameters, with no adjustments....12
2. Subdomains of the (a, θ_{do}) plane with distinct solution types for plane parallel infinite glass ribbons and duct surfaces....13
3. Lehr Geometry. 16
4. Some viewfactors for the geometry of Fig.3....17
5 Image for highly-reflective roller....18

LEHR MODEL REQUIREMENTS

The annealing lehr is basically a large insulated enclosure through which a ribbon of flat glass passes at controlled speed (see Schematic). The enclosure contains air ducts through which air flows with controlled initial temperature and flowrate from a plenum. The goal of this arrangement is to subject the ribbon to a suitable temperature history so that the exiting ribbon has a desirable residual stress distribution when it reaches room temperature.

Lehr Heat Exchange Schematic



A glass ribbon carried on supporting rollers enters the lehr in a fluid state at a temperature around 1050 deg F and exits below the glass transition temperature. The complete lehr consists of three sections—conditioner, annealing section, and cooldown section, all enclosed in an insulated box. The first two sections are cooled primarily by radiative exchange with cooling air in the ducts above and below the ribbon. Airflow in the ducts can be parallel or counter to the ribbon motion which defines the positive x-direction here. Electrical heaters may also be installed between the bottom of the ducts and the ribbon to improve control.

To achieve the goal, it is necessary to select flowrates in the individual ducts as well as inflow temperatures, so that the desired temperature distribution is imposed. To do this, we can either specify an array of sensors adequate to the characterization of the temperature distribution, or we can attempt to simulate the glass temperature in real time so that a few robust sensors can provide sufficient information for successful control. We have chosen the second approach.

Taking the Chui model [1] as our starting point, we seek a real time simulation which can track the dynamics of the lehr thermal solution faithfully enough for control development and potentially as an element of a model-based control. The goals are double—to bring the lehr under better control, and to provide a digital set of state variables for the lehr which can be communicated throughout an end-to-end control system for float glass production. The physics incorporated in [1] consists of balancing energy between the glass ribbon and the fluid in an array of ducts which are symmetric about the ribbon's midplane, with radiation flux connecting them.

To begin the discussion, a closed-form solution for a single duct exchanging radiation with a ribbon, both infinitely wide, so there are no viewfactors to consider, (simplified version of the Chui model) is derived, exhaustively analyzed and written into FORTRAN software. Then radiative viewfactors of realistic lehr geometries are generated in the second section, so that the full Chui model can be assembled from very high speed software. This section also includes software to characterize roller radiative exchanges, and discusses the thermal contact between glass ribbon and the supporting rollers as well. Treatment of heaters and coolers which allow more realistic spectral characterization of the glass-heater exchange is also provided in Section 2. In a report to follow, that case will be extended to include radiative participation of the glass ribbon , see Ref. [3].

The approach for estimating stresses is outlined in the fourth section, and a discussion follows in the fifth.

1. PLANE PARALLEL SOLUTION

Consider the case of an infinitely wide glass ribbon moving through an infinitely long and infinitely wide set of ducts in which a fluid flows. Let the ducts be symmetric about the midplane of the ribbon, and suppose that the heat exchange is entirely by radiation between the ribbon surface and the ducts' surfaces. For steady operation of this system, the surface temperature of the glass will be $T_g(x)$, depending only on the position along the direction of motion, and, if the ducts' fluid and surface temperatures are taken to be the same, these will similarly be functions of x alone: $T_d(x)$. If the flux between the duct and ribbon is entirely radiative, the energy balance for bulk temperatures can be written in terms of the mass flux per unit width times the specific heat $\dot{m}_g C_g$, that is, the heat capacity of half the ribbon, simply as:

$$\dot{m}_g C_g \frac{\partial T_g}{\partial x} = -\varepsilon_e \sigma \left({T_g}^4 - {T_d}^4 \right) \qquad [1a]$$

when the x axis is in the direction of motion of the ribbon, and the radiative flux is determined entirely by the temperatures of the duct and ribbon at station x alone—the

temperature gradient along ribbon and duct is ignored here. The "effective emissivity" ε_e is given by:

$$\varepsilon_e = \left(\frac{1}{\varepsilon_g} + \frac{1}{\varepsilon_d} - 1 \right)^{-1}$$

for infinite plane-parallel gray isothermal surfaces. The fluid in the duct then obeys:

$$\pm \dot{m}_d C_d \frac{\partial T_d}{\partial x} = \varepsilon_e \sigma \left(T_g{}^4 - T_d{}^4 \right) \qquad [1b]$$

where the + sign indicates fluid flowing in the same direction as the ribbon moves, while the – sign indicates counterflow. It follows immediately that

$$\frac{\partial T_d}{\partial x} = a \frac{\partial T_g}{\partial x},$$

where the parameter a is simply the ratio of advected heat capacities (per unit width):

$$a = \mp (\dot{m}_g C_g) / (\dot{m}_d C_d).$$

This means that $T_d(x)$ can be eliminated as a first integral; in dimensionless form:

$$\theta_d = \theta_{do} + a(\theta - 1) \qquad [2a]$$

where $\theta_d = T_d(x)/T_{go}$ and $\theta = T_g(x)/T_{go}$ are dimensionless values for the absolute (e.g., K) temperatures, scaled by the initial glass ribbon temperature T_{go}. For convenience, let this be written as $\theta_d = a\theta + b$ where $b = \theta_{do} - a$ is a known constant since a and the initial duct temperature are prescribed .

An inherent length scale $\hat{L}_o$ can be defined from the data as:

$$\hat{L}_o = (\dot{m}_g C_g) / (\varepsilon_e \sigma T_{go}{}^3)$$

Now, the energy balance equation reduces to the ordinary differential equation

$$-\hat{L}_o \frac{d\theta}{dx} = \left(\theta^4 - [\theta_{do} + a(\theta - 1)]^4 \right) = P_4(\theta; \theta_{do}, a)$$

Closed-form quadrature can be carried out on this expression; for three basic cases:

(Case 1: general solution) The result is the cumbersome but explicit form of the solution, including its initial conditions:

$$\frac{(x - x_o)}{\hat{L}_o} = \frac{1 + a^2}{2b^3} \; F(\theta, a, \theta_{do}) \qquad [2b]$$

where

$$F(\theta, a, \theta_{do}) = - \ln\left[\frac{\theta - \theta_d}{1 - \theta_{do}} \frac{1 + \theta_{do}}{\theta + \theta_d}\right]^{1/2} + \left\{\frac{1 - a^2}{1 + a^2} \tan^{-1}\left[\frac{\theta\theta_{do} - \theta_d}{\theta + \theta_{do}\theta_d}\right]\right\} + \frac{a}{1 + a^2} \ln \frac{(1 + \theta_{do}^2)}{(\theta^2 + \theta_d^2)} \frac{(\theta^2 - \theta_d^2)}{(1 - \theta_{do}^2)}$$

The right hand side depends only on the arguments of $F(\theta, a, \theta_{do})$, given the definition of b. That is, the lehr aim point, namely, the glass temperature at the exit, has the dimensionless value θ, while the lehr control settings for (1) flow in the ducts, a, and for (2) the temperature of the duct fluid, θ_{do}, at the station x_o, are the other two arguments.

For counterflow in the ducts, a is positive; it is negative for parallel flow. In the counterflow case, the value of the dimensionless parameter θ_{do} is not usually known; rather, the temperature of the fluid at the inflow end of the duct at station x is known. If we denote the inflow temperature of the duct fluid by θ_{din} regardless of whether it occurs at station x or at station x_o, then the function $F(\theta, a, \theta_{do})$ can be written as $F(\theta, a, \theta_{din})$ with the understanding that θ_{din} means θ_{do} when a is negative, and means θ_d when a is positive.

(Case 2: b=0 in counterflow) Putting b = 0 makes the [2b] singular, but also makes the duct absolute temperature a multiple of the glass temperature, allowing another trivial quadrature:

$$x - x_o = \hat{L}_o \frac{1}{3(1 - a^4)} \left(\frac{1}{\theta^3} - 1\right) \qquad [2c]$$

Note that a cannot be unity in this case, or the duct and ribbon are at the same temperature everywhere, since b =0. Further, because $b = \theta_{do} - a$ = 0 here, the initial duct temperature θ_{do} is simply a.

When b is small, evaluation of [2b] is inaccurate, so it is necessary to expand the solution in powers of b. This is developed in App. 1, where the most natural expansion parameter turns out to be $e = b/(1-a^4)$ rather than simply b.

(Case 3: Minimum ribbon length: a = 0) This case represents infinite heat capacity of the duct flow as compared to the ribbon. The duct fluid remains isothermal at θ_{do} and the quadrature gives:

$$\frac{(x-x_o)}{\hat{L}_o} = \frac{1}{2\theta_{do}^3}\left\{\ln\left[\frac{(1-\theta_{do})(\theta+\theta_{do})}{(\theta-\theta_d)(1+\theta_{do})}\right]^{1/2} + \tan^{-1}\left[\frac{\theta_{do}(\theta-1)}{\theta+\theta_{do}^2}\right]\right\} \qquad [2d]$$

Since this also represents the shortest length for which the glass temperature can be brought to the value θ, it is a useful reference value for calibrating the lehr performance.

The solution given by eqs.[2] above is of the form $\ell = \hat{F}(\theta; a, \theta_{do})$ in terms of the dimensionless distance $\ell = (x-x_o)/\hat{L}_o$ as dependent variable, with independent variable being θ, and the duct settings a and θ_{do} as parameters. Since any value of θ can be used in the right hand sides of eqs.[2], the first step in evaluating the solution must be to select the physically meaningful domain of θ, for the given values of a and θ_{do}.

By inspection of eq.[2b], it is clear that $\theta = \theta_d$ and $\theta = -\theta_d$ are both singular, and produce an infinite value for ℓ. Substitution into eq.[2a] and solution result in two values

$$\theta_c = \frac{b}{1-a} \qquad [3a]$$

$$\theta_o = -\frac{b}{1+a} \qquad [3b]$$

The first of these is the value θ_c at which the ribbon and duct temperature converge, given infinite length to accomplish this asymptotic behavior, as seen in Fig. 1(a) for parallel flow. The second represents an asymptote approached by the glass while the duct approaches $-\theta_o$. Since one temperature ratio or the other must be negative, this infinite length branch must be nonphysical because these are absolute temperatures. Thus, when the value θ_o is positive, the solution must have been cut off when θ_d =0, at $\ell = \ell_{ext}$, an extremum of ℓ. This length corresponds to $\theta^* = -b/a$. Similarly, for a

negative θ_o value, the cutoff must be imposed where θ = 0 (where θ_d = b); in either case, ℓ_{ext} is given by:

$$\ell_{ext} = \frac{1+a^2}{2b^3}\left\{\ln\left|\frac{1-\theta_{do}}{1+\theta_{do}}\right|^{1/2} + \frac{1-a^2}{1+a^2}\tan^{-1}\left(s_o\theta_{do}^{s_o}\right) + \frac{a}{1+a^2}\ln\left|\frac{1+\theta_{do}^2}{1-\theta_{do}^2}\right|\right\} \qquad [3c]$$

where s_o is +1 when θ_o is positive, and is –1 when θ_o is negative, so that the argument of the arctan is θ_{do} and $-1/\theta_{do}$ respectively in these cases. The physical solution branch can now be fully characterized by analysis of the relative sizes of $\theta_c, \theta_o, \theta^*$ in relation to 0, 1, and θ_{do} values.

The usual information sought by solving the energy balance [1] is the variation of temperature with distance from the inflow station x_o. In terms of the variables introduced above, this is the function $\theta = f(\ell; a, \theta_{do})$, with dimensionless distance ℓ as the independent variable and the duct settings a and θ_{do} as parameters, which describes the variation of temperature on ribbon and duct (using eq. [2a]). This function $f(\ell; a, \theta_{do})$ is clearly the inverse of the function $\hat{F}(\theta; a, \theta_{do})$ defined by the solution [2]. Numerical evaluation, using bisection to locate the appropriate value of ℓ for the given a and θ_{do}, is done by a routine called 'Tofx()'. Example plots, using input parameters derived from Chui's work [1], are shown in Fig. 1 below, for both parallel and counterflow in the ducts.

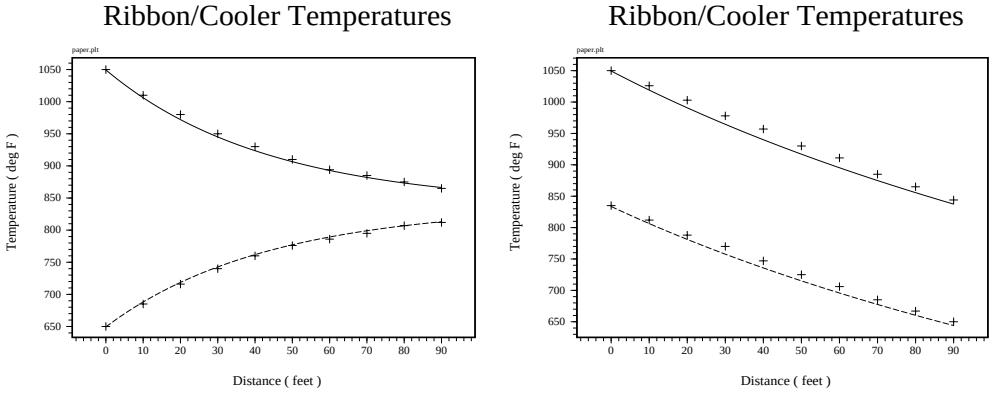


Figure 1. (a, to left) Duct flow is parallel to the glass ribbon motion, and duct fluid is cooling the glass. Parameters (a, θ_{do}) are taken from Ref.[1], Fig. 8(A), and $\hat{L}_o$ is chosen to give the best fit of the closed form solution (continuous curves) to the solution from Ref.[1] (plot symbols +). (b, to right) Counterflow solution for same parameters, with no adjustments. Fig. 8(B) of Ref.[1] is plotted for comparison (plot symbols +). This plot requires the inverse function $f(\ell; a, \theta_{do})$ described below.

The inverse function $f(\ell; a, \theta_{do})$ can only be constructed when an exhaustive classification of the possible solutions is available; see Fig. 2. The physically meaningful part of this plane is its first two quadrants, in which $\theta_{do} > 0$ holds. This halfplane is then subdivided by the line $\theta_{do} = 1$ which separates glass heating (H) solutions from glass cooling (C) ones. Counterflow solutions make up the first quadrant; parallel flows constitute the second quadrant.

The (a, θ_{do}) plane further subdivides because the parameter $b = \theta_{do} - a$ introduced in eq.[2a] vanishes on the 45-degree line through the origin; b is positive above and to the left of this line. Near this line, the power series with leading term eq.[2c] is evaluated whenever (a, θ_{do}) lies between the broken curves which converge at (1,1); outside this region, eq.[2b] is evaluated directly to determine ℓ. At these dashed curves, numerically evaluated ℓ may be discontinuous because of this switch between [2b] and [2c].

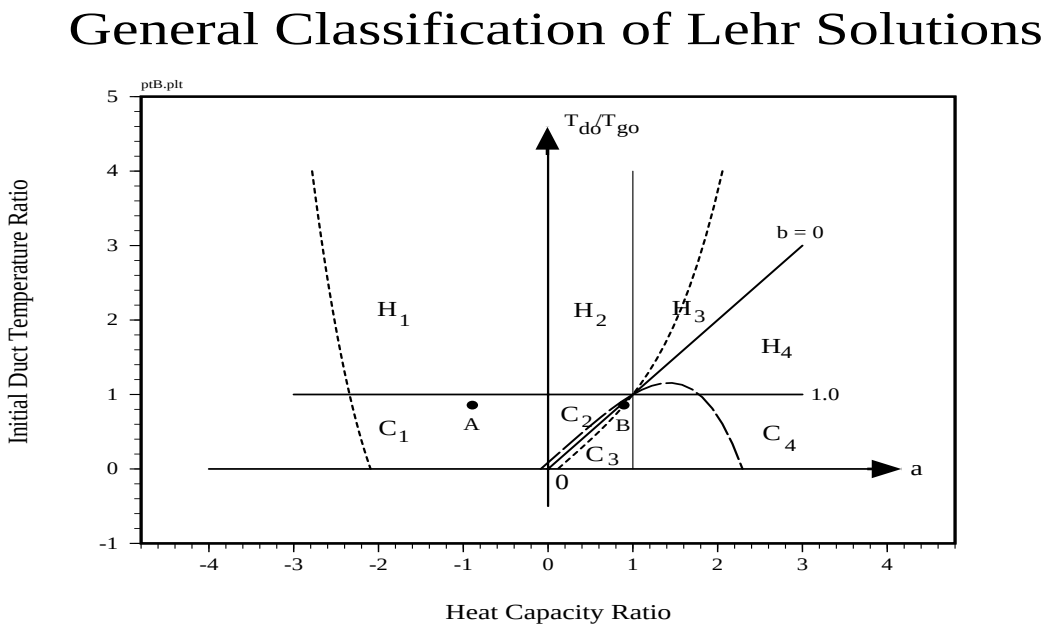


Figure 2. Subdomains of the (a, θ_{do}) plane with distinct solution types for plane parallel infinite glass ribbons and duct surfaces. Solutions in Figs. 1 (a) and (b) correspond to the points A,B with the filled circle plot symbols. All solutions with $\theta_{do} > 1$ have ducts heating the glass, and are labelled "H"; those with $\theta_{do} < 1$ are glass cooling, "C". In each subdomain, the order of the glass temperature values θ given as θ_o, θ_c are different and are in different orders relative to 0, 1 or θ_{do}. Asymptotic form eq. [2c] is used between the dashed curves, which includes all weak duct flow cases (large | a |); values depicted are $e_{\min}$ =-0.115 and $e_{\max}$ =0.086. The infinite duct capacity solution [2d] applies to all points along the vertical axis.

In summary, the solutions given in eqs. [2] have physically meaningful values when the glass temperature aim point θ is between unity and θ_c for the regions C1,C2, H1 and H2; for C3 and C4, θ must lie between $\theta^* = -b/a$ and unity, while H3 and H4 can take on any θ value from unity to infinity.

1.1 Software Modules. To make the results derived above readily available to the user, some basic subroutines have been written. These are enumerated here.

1.1.1 XEND() This function has arguments θ , a , and θ_{din} and returns the overall length ℓ_{oa} as the value of 'xend()'. For the parallel flow case with a < 0, $\theta_{do} = \theta_{din}$ and this is simply the function value of $\hat{F}(\theta; a, \theta_{do})$ discussed above. However, for the counterflow case with $\theta_d = \theta_{din}$, the value returned is ℓ_{oa}, the dimensionless distance to the station x where the duct temperature is θ_{din}. Either the direct solution [2b] or the asymptotic solution [2c] is used in making this evaluation.

1.1.2 XCOUNT() This function has arguments θ , a , and θ_{din}; it returns the length ℓ at which the ribbon temperature ratio is θ , for the counterflow case. Thus, for counterflow, this gives as 'xcount()', the value of $\hat{F}(\theta; a, \theta_{do})$. It is called by the three routines 'Tofx()', 'aLEHR()' and 'Tdin()' described below.

1.1.3 Tofx() This function has arguments a , θ_{din}, ℓ_{oa}, and ℓ; it returns the ribbon temperature θ , found at dimensionless distance ℓ. It is the value of $f(\ell; a, \theta_{do})$, and is found by iterating on θ values until 'xend()' and 'xcount()' return values as near to ℓ as the machine arithmetic allows. The iteration scheme is bisection, so that jumps in ℓ where the asymptotic and direct solutions join are tolerated. This function was used to generate Fig.1.

1.1.4 aLEHR() This function has arguments θ_{din}, ℓ_{oa},and θ aim point; it returns values of the duct heat capacity ratio a which will result in these values, with 'aLEHR()' being the parallel flow value (negative) and the argument 'a2' being the counterflow (positive) value. This is presumed to be useful for model-based control, in that it selects how large an actuator move is required to change the estimated steady solution to the desired one (θ).

1.1.5 Tdin() This function has arguments θ , ℓ_{oa}, and a; it returns the duct temperature θ_{din} needed to achieve this steady solution. Model based control is the application target for this function, which would be used to select the changes

of duct inflow temperature to when estimated glass temperature is not the desired θ value.

1.2. Example calculation. With the function 'Tdin()' above, the calculation of Chui's Fig.9 centerline case can be approximated with input a = 0.8943, θ =0.8678 (850 deg F exit glass temperature), and ℓ_{oa} =90 ft (27.4 m)/ $\hat{L}_o$; the returned duct input temperature is θ_{din} =0.7426 (676 deg F) with θ_{din} =0.8707 (855 deg F). This duct temperature profile lies in the middle of the centerline-to-edge profiles Chui shows, so long as the $\hat{L}_o$ value fitted in Fig. 1(a) is used. Beyond this, the control coefficients he discusses are simple double calls and finite difference formulae. CPU time is completely negligible, well below a millisecond per call on a 266 MHz laptop. Another useful application envisioned for these functions is generation of meshes for more detailed calculations of radiative exchange, to produce near-optimal nodal distributions along the ribbons and ducts for both efficiency and for uniform numerical error control. For the source code of these functions, see App. 2.

2. FINITE WIDTHS AND LENGTHS

The 'infinitely wide' approximation treated in the previous section is the ideal situation which lehr design attempts to approach, by using low headroom between ribbon and ducts, well-insulated sidewalls, etc. However, the ribbon and lehr are not infinitely wide, so the radiative exchange must take account of the enclosure geometry. The first step in this direction was discussed by Chui [1] and later by Gardon [2]. These analyses considered radiative viewfactor effects in the calculation of the radiative flux, and used the same presumption of independence of the flux on axial temperature variations of the ducts as in the previous section. The glass was not transparent to any spectral range of the radiation in these treatments.

When the geometry effects on the viewfactors cannot be ignored, and when the glass is semitransparent, it becomes necessary to calculate them in an economical way. If it is presumed that the glass heat conduction problem will be given a Lagrangian treatment, in order that its internal radiative exchange with the surroundings be conveniently treated (see Sect. 3 below, and Ref. [3]), then the viewfactors need to be provided as a function of arbitrary position of the element of glass being considered. Fortunately, the lehr is predominately made up of plane elements, so its walls, ends, and ducts can all be represented as planar rectangles arranged so as to make up the enclosure.(Fig. 3). Therefore, the ability to calculate the viewfactors of two rectangles at arbitrary distance and arbitrary orientation to each other is the most difficult calculation required. This can be carried out in closed form by application of 'viewfactor algebra' relations [5,6,7]. This has been done in the subroutines attached to this report as App. 3.

Lehr Conditioning Section

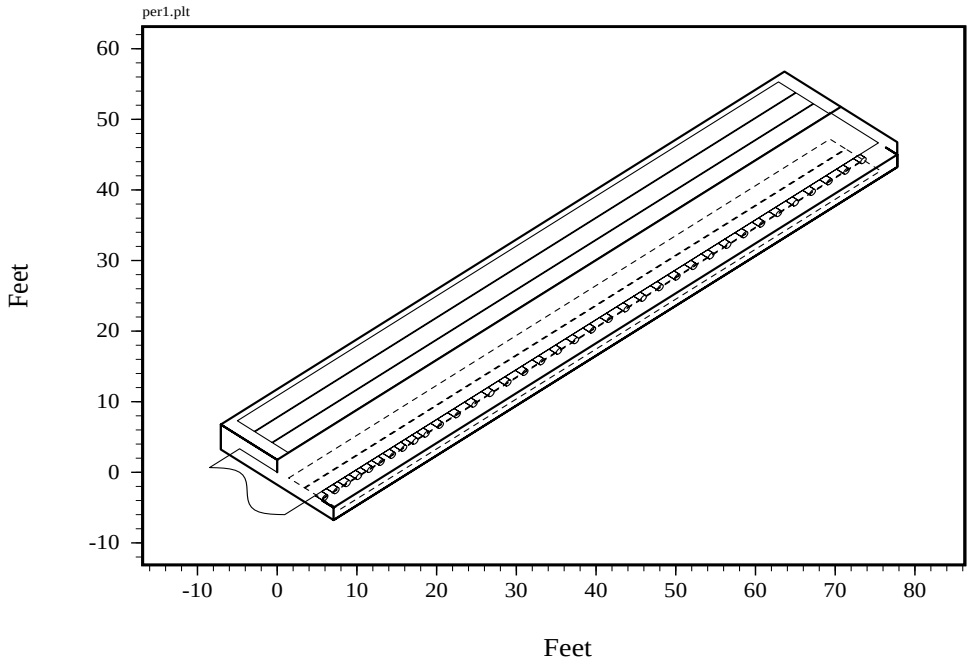


Figure 3. Lehr Geometry. Typical lehr dimensions for flat glass production include a glass ribbon of width around 10 feet (3 m), a lehr width around 15 feet (5 m) and total height between duct surfaces around 3 feet (1 m). Rollers support the glass ribbon as it enters at the lower left and moves at several hundred inches per minute (0.15 m/sec) through the lehr. Separately controlled ducts make up the ceiling (long strips on top with solid lines) and on the floor (dashed lines below ribbon). Glass throughput for numerical examples is about 400 T/day (4.26 kg/sec).

Given the convenience of the viewfactor subroutines, they can also be used for a prismatic approximation to the cylindrical surfaces of the rollers which support the glass ribbon in a physically realistic model of the lehr. This requires only the extension of a one-dimensional quadrature to evaluate the local viewfactors for (prismatic) rollers.

If rollers are included in the radiative calculation, consistency demands that they be given contact conductive boundary conditions as well. However, the possible conditions of contact are so varied that there is no unique model for this effect. Here, we expect to parameterize the contact, compare simulated surface temperatures of the ribbon to measured values, and fix the model parameters for the industrial site at which the lehr models are to be applied.

Lehr Viewfactors

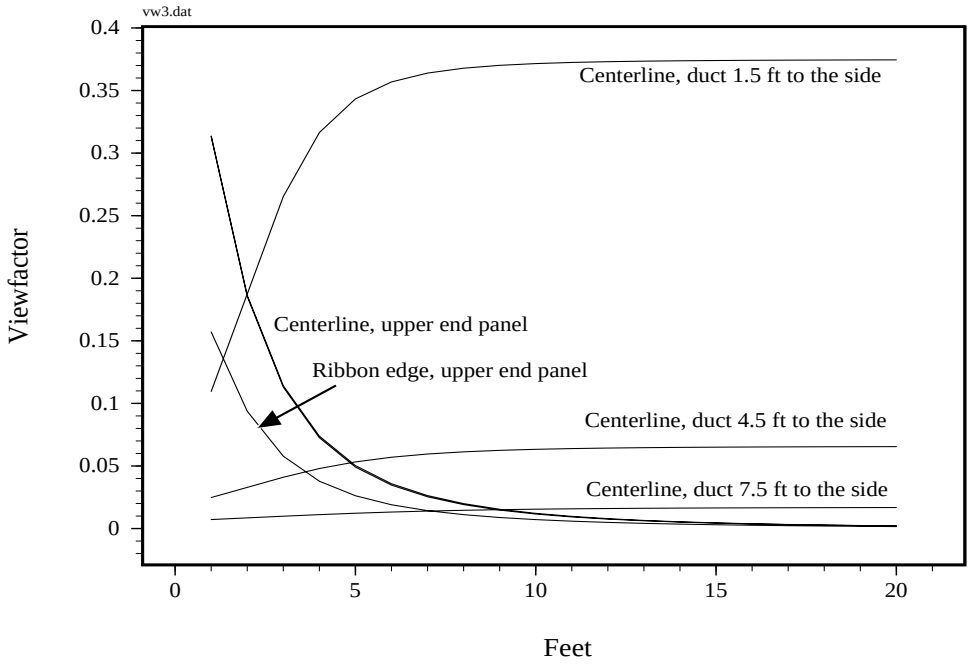


Figure 4. Some viewfactors for the geometry of Fig.3. A point on the ribbon centerline, will see a 3 ft (1 m) wide duct at distance 1 ft (0.2 m) above, with the viewfactor rising from 0.1 to 0.35 as it moves into the lehr, away from the entry end panel. This point will see the next duct to the side with a viewfactor varying from 0.02 to 0.07 ("centerline, duct 4.5 ft to the side"), and the third duct even more weakly ("centerline, duct 7.5 ft to the side"). Were the 0.35 asymptote 0.50 instead, the plane parallel solution would be exact. The sidewall viewfactor, deep in the lehr, would be about 0.14 for this example. The end panel visible above the ribbon has viewfactors as seen by particles on the ribbon centerline and on the ribbon edge which decline as the particles move into the lehr, but the distance required to reduce it by a factor of 10 is about 10 feet, a substantial portion of the conditioning zone length, or of the anneal section.

When models of the detail that includes rollers are handled, it becomes necessary to provide convenient electronic definitions of the model geometry and material properties. This definition has begun, and communicates with the user in convenient units such as throughput in 'tons/day'. Sample files to define the lehr and ribbon are displayed in App. 4.

Cylindrical Mirror Images

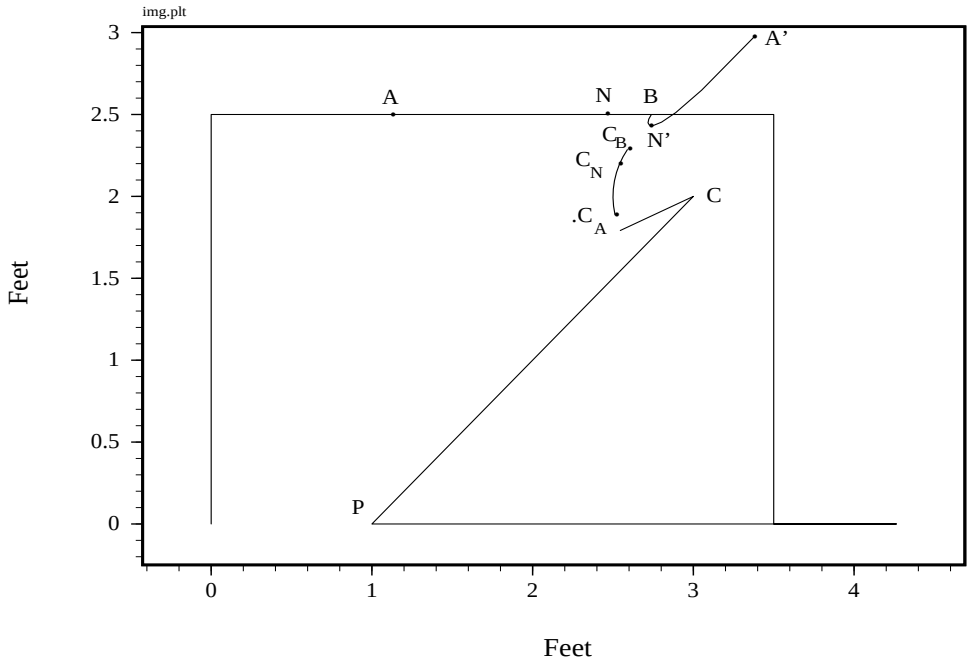


Figure 5. Image for highly-reflective roller. Segment of glass ribbon along line A NB can be viewed directly from point P on the lehr floor. It is also seen at P as a distorted image BN'A' reflected through the segment of the cylindrical roller arc C_B-C_N-C_A The image BN'A' is constructed by ray tracing from P to a point of contact such as C_A, where the ray from A to C_A makes and equal angle to the radius from the roller center C. Then the length of A- C_A is laid out along the direction P- C_A to define point A' on the image. The image can then be meshed to include this reflected energy in the flux received at P. Of course, P also receives the diffuse radiation from roller arc C_B-C_N-C_A . A similar consideration will show that the next segment of the roller below this arc reflects an image of the roller to the left, then the next segment reflects an image of the floor (which trace multiple reflections from glass or rollers, so it is applicable only when such an approximation is accurate. The subroutine 'im.f' in App. 4 lists the source code for this calculation.

3. HEAT TRANSFER WITHIN THE RIBBON

The heat transfer effects discussed above include only radiative transfer at the surface of the ribbon, and at the surface of the duct. This formulation is fully characterized by the dimensionless parameters of Section 1, with the additional dimensionless geometric parameters such as $w_g / \hat{L}_o$, $h / \hat{L}_o$, etc, where w_g is the finite width of the ribbon, h is the height of the lehr sidewall, etc. The viewfactors F_{ij} which connect surfaces 'i' and 'j' are, of course, also dimensionless inputs to the heat transfer calculation in that treatment. This presumes that the glass can conduct whatever flux is needed to accomplish the bulk temperature changes described by variations of the

temperature θ . The issues arising in assuring that the fluxes computed in the enclosure exchange are consistent with the actual ribbon thickness t_g moving at velocity V_g will be treated in a forthcoming description of a calculation with glass as a participating medium in Ref. [3].

4. STRESSES

The lehr is fundamentally an annealing oven, and is meant to control residual stresses in the ribbon. If model based control is to operate in real time on presently practical (i.e., economical) computers, it must assess these stresses in as simple a manner as possible while remaining faithful to the actual sensitivity to changes in process variables. We assume that there is a 'strength of materials' approximation to the relevant stresses, in which changes in fiber length can be computed by one-dimensional expressions, and these strains used to estimate the mechanical stresses needed to maintain continuity of the ribbon, in the same spirit as Adams and Williamson [10].

In its passage through the annealing lehr, a glass ribbon changes its state from a rapidly relaxing viscous liquid medium to a "solid" with long-term residual stress. The change of state is induced by cooling the glass through its annealing temperature Ta down to its strain temperature Ts. Generally, the values of Ta and Ts are selected to agree with the temperature at which viscosity is $10^{13.5}$ and $10^{14.5}$ Poise ($10^{12.5}$ and $10^{13.5}$ Pa-s), respectively. These are heuristically adjusted by ΔT_a computed as

$$\Delta T_a = 8.86 \ln R_{avg} + 65\left(1 - R_{avg}\right)$$

to define upper and lower temperature limits on annealing for finite average cooling rates R_{avg} . Viscosity μ is taken to be a Fulcher value

$$\mu = \exp\left(a_F + b_F \Big/ \left(T - T_F\right) \right).$$

The goal of the anneal process, in general, is to produce a residual stress state which will survive the fabrication processes downstream as well as survive the service environment of the glass part. Our treatment follows Narayanaswamy's tutorial [12]. Both through-thickness and membrane stress is considered.

The governing relations for one-dimensional deformation are written in terms of a reduced time ξ defined in terms of the physical time by

$$\xi = \int_0^t \phi\left(T; T_f\right) dt',$$

i.e., glass is assumed to be 'thermorheologically simple' material, and where

$$\phi(T;T_f) = \mu_B / \mu(T) = \exp\left[\frac{H_g}{R}\left(\frac{1}{T_B}-\frac{1}{T}\right)+\frac{H_f}{R}\left(\frac{1}{T_B}-\frac{1}{T_f}\right)\right]$$

which includes activation energies H_g and H_f, the universal gas constant R, and the 'structural relaxation' effect by including the 'fictive temperature' T_f calculated from

$$T_f = T + \int_0^t M(\xi-\xi')d\xi'$$

for the shift $M(\xi) = e^{-\left(\xi/t_f\right)^{b_f}}$ defined by the two relaxation parameters b_f and t_f. Van Zee & Noritake use an M composed of a sum of two exponentials, rather than a single exponential with a power. The net thermal strain ε_{th} of a fiber at time t will consist of the instantaneous thermal strain and a structural relaxation term r_x

$$\varepsilon_{th} = \alpha_{T,L}\,(T-T_a) - r_x \quad \text{where} \quad r_x = (\alpha_{T,L} - \alpha_{T,g}) \int_0^t M(\xi-\xi')\frac{dT}{d\xi'}d\xi'$$

allows the instantaneous glassy strain with linear coefficient of expansion $\alpha_{T,g}$ to relax toward the relaxed liquid expansion $\alpha_{T,L}$. Then, for a total strain ε, the stress σ in the fiber is given by

$$\sigma = \frac{E(0)}{1-\nu}\int_0^t R(\xi-\xi')\frac{d(\varepsilon-\varepsilon_{th})}{d\xi'}d\xi'$$

where the relaxation function $R(\xi) = e^{-\left(\xi/t_v\right)^{b_v}}$ mirrors the form of M(). E(0) is the instantaneous ('glassy') Young's modulus, and ν is Poisson's ratio. Values for the various parameters appearing here are summarized in App.5 below.

5. DISCUSSION

At this point, the elements of a simple quasisteady model which accepts as its input the high level description of the lehr detailed in App.4 has been described. The first two portions of it, dealing with the simplest possible model of the radiative energy exchange between the ribbon and enclosure elements, have been embodied in FORTRAN code. The validation process has been limited to comparison with Chui's results reported in Ref. [1]. That comparison, it must be stressed, used a fitted value of the length scale $\hat{L}_o$, namely, 6670 cm instead of the value 2985 cm which results from calculating $\hat{L}_o = (\dot{m}_g C_g)/(\varepsilon_e \sigma T_{go}^{\ 3})$ from the values in his Table I. This is a multiplier of 2.2, which

suggests that perhaps the factor 2 in his eq.[1] was omitted from the numerical calculations he reported. The fact that both his Fig. 8(B) and Fig.9 results are essentially reproduced with $\hat{L}_o$ =6670 cm reinforces this likelihood.

The plane parallel thermal solution can be evaluated before the glass ribbon has moved 0.1 mm (i.e., in under a millisecond), so it is quite feasible to use it in a real time control of the lehr. Given sensor data, such as temperatures in an IR strip scanner, it could calculate and return the adjustments required to move the quasisteady solution back to the aim point.

For finer control than the plane parallel solution provides, the viewfactor values could be used in numerically integrating the energy balance eqs.[1-7] of Chui. The parallel flow case is a simple initial value problem quadrature for a system of ordinary differential equations, and is well treated by universally available software. For the counterflow case, it is necessary to solve a boundary value problem because the duct inflow temperatures θ_{din} (one value for each duct) are given at the far end (station x) of the ribbon. While software packages exist which can accomplish this task automatically with controlled error, it is more efficient numerically to use a dedicated code . The most direct method for a robust solution with the known monotonicity properties here would be to iteratively solve initial value problems with estimated θ_{din} values; that is, by "shooting". Not only the initial estimate of these values, but perhaps their subsequent iterations, could be obtained from the plane parallel solution above, very quickly and completely robustly (since those routines make use of the information in Fig. 2). In short, using the software of sections 1 and 2 above, it should be possible to carry out the Chui calculation in real time as well, during the time the ribbon moves around a few cm. As with the plane parallel solution, lehr control adjustments should be possible, based on calculations by such routines.

The next level of sophistication, in which the heat fluxes include radiative participation by the interior of the ribbon, is beyond the scope of discussion here, and will be treated in Ref. [3]. At present it seems that such a calculation may well prove possible in a time useful for update of aim points, and perhaps even in real time.

The actual goal of the lehr is to manage stresses, and so the complete model based control would connect the lehr settings with the residual stresses in the glass as it reaches the cutting station. Simulation of the stresses due to thermal strains can be done in a "Strength of Materials" sense by use of one-dimensional calculations of fiber strains as outlined in Section 4, and these strains used to calculate membrane stresses, perhaps with the help of some plate equations to check for ribbon buckling; see Ref. [13]. This remains for a future effort.

REFERENCES:

1. Chui, G,K., "Heat Transfer and Temperature Control in an Annealing Lehr for Float Glass", *J. Amer. Ceram. Soc. 60,* 477. (1977)

2. Gardon, R. "Modelling Annealing Lehrs for Flat Glass", *J. Amer. Ceram. Soc. 65*,372. (1982)

3. Houf, W. ,"Lagrangian Thermal Model of Annealing Lehr Glass Ribbon as Radiatively Participating Medium (Tentative Title)". Sandia National Laboratories. (2000)

4. *CRC Standard Mathematical Tables*, integrals 87,89,100, 11th ed., CRC, Cleveland. (1957)

5. Brewster, M.Q., *Thermal Radiative Transfer and Properties*, John Wiley & Sons, New York (1991)

6. Siegel, R. and Howell, J.R., Thermal Radiation Heat Transfer, 2nd ed., Hemisphere-McGraw Hill, New York. (1981)

7. Howell, J.R., A Catalog of Radiation Configuration Factors, McGraw Hill, New York. (1982)

8. Trier, W. Glass Furnaces, transl. By K.L. Loewenstein, from Glasschmeltzofen: Konstuktion und Betriebsverhalten, Springer Verlag, 1984. Society of Glass Technology, Sheffield. (1987)

9. Gunther, R. Glass Melting Tank Furnaces, transl. By J. Currie, from Glasschmeltz-Wannenofen, Deutsche Glastechnische Gesellschaft 1954. Society of Glass Technology, Sheffield. (1957)

10. Adams, LH and Williamson, ED, "Annealing of Glass", J. Franklin Inst. 190, 597-631 and 835-870. (1920)

11. Gardon, R and Narayanaswamy, OS, "Stress and Volume Relaxation in Annealing Flat Glass", J. Amer. Ceram. Soc. 53, 380. (1970)

12. Narayanaswamy ,OS,"Annealing of Glass", Chap. 5 in Glass Science and Technology, Vol. 3. Academic Press, New York. (1986)

13. Wang, Chi-The, Applied Elasticity, McGraw-Hill, New York (1953)

APPENDIX A. Derivation of the Plane Parallel Solution and Asymptotic Forms

Separation of variables reduces the energy balance to quadrature:

$$(x - x_o) = -\hat{L}_o \int_1^{\theta} \frac{du}{P_4(u;\theta_{do},a)} .$$

The quartic polynomial $P_4(\theta;\theta_{do},a)$ can be factored, and and expanded into the form:

$$\frac{du}{u^4-(au+b)^4} = \frac{1}{2u^2}\left\{ \frac{du}{u^2-(au+b)^2} + \frac{du}{u^2+(au+b)^2} \right\}$$

The quadrature can now be carried out explicitly [4], to give:

$$\int_1^{\theta} \frac{du}{P_4(u;\theta_{do},a)} = \frac{a}{2b^3}\left(G(\theta)-G(1)\right)$$

in terms of

$$G(\theta) = \ln\left(\frac{\theta^2+\theta_d^2}{\theta^2-\theta_d^2}\right) + \frac{1+a^2}{2a}\ln\left[\frac{(1+a)(\theta-\theta_d)}{(1-a)(\theta+\theta_d)}\right] - \frac{1-a^2}{a}\tan^{-1}\left(\frac{\theta+a\theta_d}{-a\theta+\theta_d}\right)$$

[A1.1]

Evaluating $\left(G(\theta)-G(1)\right)$ and using the trigonometric identity for tan (x + y), [A1.1] becomes eq.[2b].

In cases of counterflow, the first two ln() terms can very nearly cancel one another (which they do exactly when b = 0 produces the Case 1 solution). For computation in that case, the corresponding ln() terms in $\left(G(\theta)-G(1)\right)$ are expanded as

$$\ln(1+\varepsilon) + \frac{1+a^2}{2a}\ln\left[\frac{(1+\theta_{do})(\theta-\theta_d)}{(1-\theta_{do})(\theta+\theta_d)}\right] \approx \varepsilon - \frac{1}{2}\varepsilon^2 + \frac{1+a^2}{2a}\ln\left[\frac{(1+\theta_{do})(\theta-\theta_d)}{(1-\theta_{do})(\theta+\theta_d)}\right]$$

where $\varepsilon = 2\,\dfrac{\theta_{do}(\theta-\theta_d)^2-(1-\theta_{do})^2\theta\theta_d}{(1+\theta_{do}^2)(\theta-\theta_d)^2}$. [A1.2]

Finally, we note that the cases with |a| = 1, b nonzero, can be integrated because the integrand becomes cubic, and quadrature gives:

$$x - x_o \;=\; -\,\mathrm{sgn}(a)\,\hat{L}_o\,\frac{1}{2b^3}\;\ln\left[\frac{(\theta+\mathrm{sgn}(a)\theta_d)^2}{(\theta^2+\theta_d^2)}\,\frac{(1+\theta_{do}^2)}{(1+\mathrm{sgn}(a)\theta_{do})^2}\right]$$

Despite its quite different appearance, this form is a special case of the solution derived above, and need not be treated separately.

In the counterflow case, a > 0 holds and the coefficient $b = \theta_{do} - a$ can be zero or negative. When the solution (1) is evaluated with a small value of the coefficient b, less than about 0.1, the finite precision arithmetic of a computer loses accuracy quite noticeably, so it is necessary to have an asymptotic solution around b=0 to evaluate. This can be constructed by expanding the integrand in powers of

$$w \;=\; \frac{2ab}{u} \;+\; \frac{b^2}{u^2},$$

which gives the expansion

$$(1-a^4)(x-x_o)/\hat{L}_c \;=\; \frac{1}{3}\left(\frac{1}{\theta^3}-1\right) - 2a^2\int_1^\theta \frac{w\,du}{u^4} \;-\; (1+3a^4)\int_1^\theta \frac{w^2\,du}{u^4} \;-$$
$$4a^2(1+a^4)\int_1^\theta \frac{w^3\,du}{u^4} \;-\; (1+10a^4+5a^8)\int_1^\theta \frac{w^4\,du}{u^4} \;-\;\ldots$$

from which the powers of b can be explicitly written. The actual expansion parameter which appears in the power series is $e \;=\; b/(1-a^4)$, and it represents the solution as the fourth order relation:

$$(1-a^4)(x-x_o)/\hat{L}_c \;=\; \frac{1}{3}\left(\frac{1}{\theta^3}-1\right) + a^3\left(\frac{1}{\theta^4}-1\right)e \;+\; \frac{2}{5}a^2(3+5a^4)\left(\frac{1}{\theta^5}-1\right)e^2$$
$$+\;\frac{2}{3}a\,(1+10a^4+5a^8)\left(\frac{1}{\theta^6}-1\right)e^3$$
$$+\;\frac{1}{7}(1+65a^4+155a^8+15a^{12})\left(\frac{1}{\theta^7}-1\right)e^4 \;+\;\ldots[\text{A1.3}]$$

and this is used with heuristically chosen values $-0.115 < e < 0.0827$ in the routine XEND().

APPENDIX B. Source Code for Evaluating the Plane Parallel Solution

```
program calibrate
c**********************************************************************
c**      Issued by Sandia National Laboratories, operated for the  **
c**      United States Department of Energy by Sandia Corporation. **
c**                         NOTICE                                 **
c**      This program was prepared in the course of work sponsored **
c**      by an agency of the U.S.Government.  Neither the United   **
c**      States Government nor any agency thereof, nor any of their**
c**      employees, makes any warranty, express or implied, or     **
c**      assumes any legal liability or responsibility for the     **
c**      accuracy, completeness, or usefulness of any information, **
c**      apparatus, product, or process disclosed, or represents   **
c**      that its use would not infringe privately owned rights.   **
c**      Reference herein to any specific commercial product,      **
c**      process, or service by trade name, trademark, manufacturer,*
c**      or otherwise, does not necessarily constitute nor imply   **
c**      its endorsement, recommendation, or favoring by the United**
c**      States Government, any agency thereof or any of their     **
c**      contractors or subcontractors.  The views and opinions    **
c**      expressed herein do not necessarily state or reflect      **
c**      those of the United States Government, any agency thereof **
c**      or any of their contractors or subcontractors.            **
c**      This code is supplied for official government use only.   **
c**      No further dissemination is permitted without specific    **
c**      permission from Sandia Laboratories, Livermore, CA 94551. **
c**********************************************************************
c**********************************************************************
c**                                                                  **
c**         Subroutines for Flat Glass Lehr Thermal Simulation       **
c**            written by Lee A. Bertram                             **
c**           Sandia National Laboratories                           **
c**            Livermore, CA  94551                                  **
c**                                                                  **
c**                 April 2000                                       **
c**                                                                  **
c**                                                                  **
c**           Copyright 2000 (c)  Sandia Corporation                 **
c**********************************************************************
C**********************************************************************
c     Evaluate RHS of Apr00 version of closed-form plane-parallel
c     radiative exchange solution for glass (g) ribbon and ducts (d).
c     Input variables are the controlled quantities: heat advection
ratio 'a',
c     and duct inflow dimensionless absolute temperature
thdin=Tdin/Tgo.
c     These are held fixed, and the whole range of the third
independent
c     variable (endpoint ribbon th=Tg/Tgo) is marched through to
display
c     all possible solutions for the given 'a,thdin'.
      open(unit=8,file='soln.dat',status='unknown')
      open(unit=9,file='fort.9',status='unknown')
      open(unit=66,file='cal.log',status='unknown')
      hatL=5481.
```

```
c     Possible emissivity correction of 25%?
c      hatL=1.205*hatL
      hatL=1.25*hatL
c      hatL=1076.
      hatL=2765.
c      hatL=3775.
       hatL= 6670.
      Tgo=838.6
      Tdo=616.3
c      Tdo=916.3
      thdin=Tdo/Tgo
c     To choose parallel flow, set this flag to 1; else, counterflow.
      k8A=21
c     Set an 'a' value, then march 'th' and compare the base solution
c     to the asymptotics of order 0-4, in the counterflow case a > 0.
      if(k8A.eq.1) then
       a=-0.8576
       a=-0.8943
      else
       a=0.8576
       a=0.8943
c       a=2.0
      endif
      nth=51
      write(6,*) 'Initial XEND() calls: a,thdin,hatL=',a,thdin,hatL
      write(66,*) 'Initial case: a,thdin,hatL=',a,thdin,hatL
      bo=(thdin-a)
      if(a.lt.1.) then
       brange=-(thdin-a)+0.99*(1.-a)*thdin
      else
       brange=0.99*(1.-a)-bo
      endif
c     With a,b,thdin all specified, the counterflow solution is fixed.
c     However, when  thdin=thdo, th=1 and (x-xo)=0. This occurs when
the
c     selected b-value is thdin-a, so the minimum b of the range must
be
c     thdin-a.  The maximum b must be 1-a, which requires infinite
length
c     for the ribbon to cool from its degenerate initial condition
thdo=1.
      db=(brange)/float(nth-1)
c
c     For the parallel flow case, the marching must be on th, the
endpoint
c     ribbon temperature, since a, b, thdin are not independent here.
Clearly,
c     the th value goes from 1 (for zero length) to thc (for infinite
length),
c     where thc is the convergence temperature of ribbon and duct.
      if(abs(1.-a**2).le.1.e-4) then
       thc=0.5*(thdin+1.)
      else
       thc=(thdin-a)/(1.-a)
      endif
      dth=(1.-thc)/float(nth)
c
```

```
      write(66,*) 'a,thdin-a,db=',a,thdin-a,db
      do 10 i=1,nth
       if(a.gt.0.) then
        b=bo+db*float(i-1)
        th=(thdin-b)/a
        thdo=a + b
       else
        th=thc+dth*float(i)
        b=thdin-a
       endif
       write(9,*) b,th,thdo,a,thdin
       xth=xend(th,a,thdin,thdo,thd)
       if(abs(th-thdo).gt.2.e-4*amax1(abs(th),abs(thdo))) then
c       Display limit case (a=0) and components of 'xend':
        xzero=xmin(thdin,th)
        xcrunch=xdir(a,thdo,th)
        xseries=xasymp(a,b,th)
       else
        xzero=1.
        xcrunch=1.
        xseries=1.
       endif
       if(a.gt.0.) then
        thd=a*th+b
        write(9,*) xth,th,xseries,xcrunch,xzero
       else
        thd=a*th+b
        write(9,*) xth,th,thd,thdo
       endif
   10 continue
c
c     Second example:  Generate Chui plots for Fig.8:
      thdin=0.734915
      if(a.gt.0.) then
c      Fig. 8B
       a=0.8576
       th=0.856
       a=0.8943
cc       a=2.0
       th=(5.*(844.-32.)/9. + 273.)/Tgo
cc       th=(5.*(1144.-32.)/9. + 273.)/Tgo
cc       th=(5.*(1100.-32.)/9. + 273.)/Tgo
       b=thdin-a*th
       thdo=a+b
      else
c      Fig.8A
       th=0.8764
       th=(5.*(865.-32.)/9. + 273.)/Tgo
       a=-0.8576
       a=-0.8943
       thdo=thdin
      endif
      write(6,*) ' Chui Fig 8 from Tofx() with a,th,thdin=',a,th,thdin
      write(66,*) ' Fig 8 call XEND with a,th,thdin=',a,th,thdin
      oal=xend(th,a,thdin,thdo,thd)
c     Overall length for Chui:  90 ft X 30.48 cm/ft:
      sta=90.*30.48/hatL
```

```
      write(66,*) 'oal,sta=',oal,sta
      npts=101
      ds=sta/float(npts-1)
      do 20 i=1,npts
       sta=amax1(1.e-5,ds*float(i-1))
       write(66,*) ' Enter Tofx with oal=',oal,' and sta=',sta
       tchk=Tofx(a,thdin,oal,sta)
       write(66,*) ' Exit Tofx with i,thdin,tchk=',i,thdin,tchk
       xft=hatL*sta/30.48
       TdegF=tchk*Tgo*9./5.-460.
       if(a.gt.0.) then
        thd=a*tchk+b
       else
        thd=a*(tchk-1.)+thdin
       endif
       Tduct=thd*Tgo*9./5.-460.
       write(8,*) xft,TdegF,Tduct,sta,tchk,thd
c       write(66,*) ' End Tofx with th(oal)=',th,' and th(sta)=',
c     &            tchk,'; oal=',oal
   20 continue

      write(6,*) ' Chui Fig 8 curves Tofx() in file "soln.dat".'
c
      th=0.86745
cc      th=1.031
      oal=xend(th,a,thdin,thdo,thd)
      write(6,*) ' Calling aLEHR with thdin,th,oal; a=',thdin,th,oal,a
      write(66,*) ' aLEHR arguments thdin,th,oal; a=',thdin,th,oal,a
      astar=alehr(thdin,th,oal,thd,thdo,a2)
      write(6,*) ' Roots from aLEHR(): astar,a2=',astar,a2
c     Chui Fig. 9 sets glass exit temperature to 850 deg F,
c     then length=90 ft; seeks Tdin for counterflow duct
c     with a=0.8943 (same as Fig.8) to achieve this aim Tg.
      th=0.8678
      thdin1=0.734915
      a=0.8576
      a=0.8943
cc      a=1.0
      oal=90.*30.48/hatL
c      th=1.031
c      a=2.0
c      oal=9.*30.48/hatL
      write(6,*) ' Tdin() call with th,oal,a=',th,oal,a
      write(66,*) ' Tdin() arguments th,oal,a=',th,oal,a
      thdin=Tdin(th,oal,a,thd,thdo)
      write(6,*) ' Tdin() returns th,a,oal,thdo,thdin='
      write(6,*)  th,a,oal,thdo,thdin
      alen=xend(th,a,thdin,thdo,thd)
      write(6,*) ' For th,a,thdin=',th,a,thdin,', XEND()=',alen
      stop
      end
c
      function alehr(thdin,th,oal,thd,thdo,a2)
c*****************************************************************
c*****************************************************************
c**                                                             **
c**         Subroutines for Flat Glass Lehr Thermal Simulation  **
```

```
c**            written by Lee A. Bertram                          **
c**           Sandia National Laboratories                        **
c**            Livermore, CA  94551                               **
c**                                                               **
c**                 April 2000                                    **
c**                                                               **
c**                                                               **
c**           Copyright 2000 (c)  Sandia Corporation              **
c*******************************************************************
C*******************************************************************
c     Given a dimensionless lehr length 'oal' and a duct input absolute
dimensionless
c     temperature ratio 'thdin' and the target glass temperature
th=Tg/Tgo,
c     find the heat advection ratio 'a' for the duct and return it as
'alehr';
c     also return as an argument the second value 'a2' of opposite
sign.
c     Also return duct temperature ratios at xo (thdo) and at exit oal
(thd).
      k=0
      kmax=50
      eps = 1.e-4
      tol=amax1(eps,eps*oal)
c     Start by seeking parallel flow (a < 0) solution:
      if(abs(1.-th).lt.eps) then
cc       write(66,*) '  Abort ALEHR because ribbon ends with',
cc     &             ' th = Tg/Tgo =',th
       alehr=1.
       a2=-1.
       return
      endif
c     Locate smallest feasible 'a' value, where ribbon and duct
temperatures
c     converge in parallel flow:
      amin=(1.-3.e-6)*(thdin-th)/(1.-th)
      xinf=xend(th,amin,thdin,thdo4,thd4)
cc      write(66,*) 'Parallel flow amin=',amin,' has scaled
length=',xinf
c     Locate largest feasible 'a', which arises when duct temperature
c     at glass inflow station xo is equal to the glass temperature;
ducts
c     are counterflow in this case:
      amax=(1.-3.e-6)*(1.-thdin)/(1.-th)
      xinfc=xend(th,amax,thdin,thdo4,thd4)
cc      write(66,*) 'Counterflow amax=',amax,' has scaled
length=',xinfc
c     Examine length for infinite duct flow (a=0) case:
      a0=0.
      xmin=xend(th,a0,thdin,thdo4,thd4)
cc      write(66,*) ' Isothermal duct flow a=0 case has scaled
length=',
cc     &             xmin
c     Check input arguments for consistency:  can 'oal' be reached for
this
c     thdin,th pair?
cc      write(66,*) ' Feasible range  of lengths is',xmin,' to max of',
```

```
cc     &             xinf,xinfc
      if(oal.lt.xmin.or.oal.gt.amax1(xinf,xinfc)) then
c      Solution is not feasible
cc       write(66,*) 'ALEHR cannot reach the input oal=',oal
cc       write(66,*) ' feasible range  of lengths is',xmin,' to max
of',
cc     &              xinf,xinfc
c      Return error flag alehr=0.
       alehr=0.
       return
      endif
c     Proceed with bisections to locate parallel flow solution:
      aL=amin
      xL=xinf
      sL=1.
c     Set right point as essentially zero:
      aR=-0.00001
      xR=xend(th,aR,thdin,thdo4,thd4)
      targ=oal
      if(abs(xL-oal).le.tol) then
c      Converged; quit:
       alehr=aL
       go to 15
c      else  bisection to right is already set up
      endif
c     Parallel flow:
      thdo=thdin
      thd=a*th+thdo-a
      sL=sign(1.,xL-targ)
      sR=sign(1.,xR-targ)
cc       write(66,*) ' Start ALEHR bisection with
aL,aR,xL,xR,sL,sR,targ='
cc       write(66,*) aL,aR,xL,xR,sL,sR,targ
cc       write(66,*) '  Bisection gives xL, targ, xR values at aM : '
   10 continue
      aM=0.5*(aL+aR)
cc      write(66,*) xL,targ,xR,aM
      b=thdin-aM
      xM=xend(th,aM,thdin,thdo,thd)
      if(xM.lt.0.) then
c      Identify error flag:
       if(xM.gt.-1.5) then
cc        write(66,*) ' Error in ALEHR for counterflow; aM,th,thdin=',
cc     &              aM,th,thdin
        alehr=1.0
        return
cc      stop
       else
cc        write(66,*) ' Error in ALEHR for parallel flow; a,th,thdin=',
cc     &               a,th,thdin
        alehr=-1.0
        return
cc      stop
       endif
      endif
      sM=sign(1.,xM-targ)
      if((sM*sR).gt.0.) then
```

```
c      Left half contains root:
       xR=xM
       sR=sM
       aR=aM
      else
c      Right half contains root:
       xL=xM
       sL=sM
       aL=aM
      endif
      if(abs(aR-aL).le.1.e-6) then
c      Converged
       alehr=0.5*(aL+aR)
      else
       k=k+1
       if(k.le.kmax) then
c       Continue bisection
        go to 10
       else
cc        write(66,*) ' ALEHR() giving up after',k,' bisections.'
        alehr=aM
       endif
      endif
      aL=-0.8576
      xL=xend(th,aL,thdin,thdo,thd)
cc      write(66,*) '    xend(a=',aL,') =',xL,'; thd,thdo=',thd,thdo
cc      write(66,*) 'ALEHR solved parallel flow; a=',aM,' gives
xM=',xM,
cc     &           ' while oal=',oal
c     For parallel flow, the task is finished; for counterflow, the
c     conditions are established, and bisection can proceed.
   15 continue
      write(66,*) '  Start counterflow with th,thdin,oal=',th,thdin,oal
      aR=amax
      sa=1.
      xR=xinfc
cc      write(66,*) ' ALEHR sets xR=',xinfc,' for a=',aR,
cc     &            ' and th,thdin=',th,thdin,'.'
      aL=0.0001
      xL=xend(th,aL,thdin,thdo,thd)
      thdo4=thdo
c     Now have xM consistent with 'a' and 'thdin,th' inputs.  Correct
'a' to
c     get nearer given oal:
      sL=sign(1.,xL-targ)
      sR=sign(1.,xR-targ)
cc      write(66,*) ' For aL=',aL,' get xL,sL=',xL,sL
      k=0
   20 continue
      aM=0.5*(aL+aR)
cc      write(66,*) xL,targ,xR,aM
      b=thdin-aM
      xM=xend(th,aM,thdin,thdo,thd)
      if(xM.lt.0.) then
c      Identify error flag:
       write(6,*) ' Abort in aLEHR().'
       if(xM.gt.-1.5) then
```

```
cc        write(66,*) ' Error in ALEHR for counterflow; aM,th,thdin=',
cc     &              aM,th,thdin
         stop
        else
cc        write(66,*) ' Error in ALEHR for parallel flow; a,th,thdin=',
cc     &               a,th,thdin
         stop
        endif
       endif
       sM=sign(1.,xM-targ)
       if((sM*sR).gt.0.) then
c       Left half contains root:
        xR=xM
        sR=sM
        aR=aM
       else
c       Right half contains root:
        xL=xM
        sL=sM
        aL=aM
       endif
       if(abs(aR-aL).le.1.e-6) then
c       Converged
        a2=0.5*(aL+aR)
       else
        k=k+1
        if(k.le.kmax) then
c        Continue bisection
         go to 20
        else
cc        write(66,*) ' ALEHR() giving up after',k,' bisections.'
         a2=aM
        endif
       endif
cc       write(66,*) 'ALEHR succeeded in counterflow; a2=',a2,
cc     &             ' gives xM=',xM,' while oal=',oal
       aL=0.8576
       xL=xend(th,aL,thdin,thdo,thd)
cc       write(66,*) '    xend(a=',aL,') =',xL
       return
       end
c
       function Tofx(a,thdin,oal,sta)
c**********************************************************************
c**********************************************************************
c**                                                                   **
c**         Subroutines for Flat Glass Lehr Thermal Simulation        **
c**            written by Lee A. Bertram                              **
c**           Sandia National Laboratories                            **
c**            Livermore, CA  94551                                   **
c**                                                                   **
c**                 April 2000                                        **
c**                                                                   **
c**                                                                   **
c**           Copyright 2000 (c)  Sandia Corporation                  **
c**********************************************************************
c**********************************************************************
```

```
c     Given heat advection ratio 'a' and duct inflow 'thdin' at overall
length 'oal',
c     determine the ribbon dimensionless temperature 'Tofx' at
dimensionless
c     disatance 'sta' along the ribbon.  Find 'sta' by bisection, using
the
c     closed form solution 'xend()'; precision of 'sta' is set to six
figures.
c
      k=0
      kmax=50
      eps=1.e-6
      tR=1.
      xR=0.
      xL=1.e11
      if(a.gt.0.) then
c      Counterflow:
       tL=thdin
       targ=oal
       if(thdin.gt.1.) then
        tL=1.
        xL=0.
        if(a.lt.1.) then
         tR=thdin
         xR=1.e11
        else
         tR=1.+(thdin-1.)/a
         xR=1.e11
        endif
       endif
      else
c      Parallel flow; convergence T on left:
       tL=(thdin-a)/(1.-a)
       targ=sta
      endif
      sL=sign(1.,xL-targ)
      sR=sign(1.,xR-targ)
   10 continue
      tM=0.5*(tL+tR)
      th=tM
      xM=xend(th,a,thdin,thdo,thd)
      if(xM.lt.0.) then
c      Identify error flag:
       write(6,*) '  Abort in Tofx().'
       if(xM.gt.-1.5) then
cc        write(66,*) ' Error in Tofx() for parallel flow;
a,th,thdin=',
cc     &              a,th,thdin
        stop
       else
cc        write(66,*) ' Error in Tofx() for counterflow; a,th,thdin=',
cc     &               a,th,thdin
        stop
       endif
      endif
      sM=sign(1.,xM-targ)
      if((sM*sR).gt.0.) then
```

```
c      Left half contains root:
       xR=xM
       sR=sM
       tR=tM
      else
c      Right half contains root:
       xL=xM
       sL=sM
       tL=tM
      endif
      if(abs(tR-tL).le.eps) then
c      Converged
       Tofx=0.5*(tL+tR)
      else
       k=k+1
       if(k.le.kmax) then
c       Continue bisection
        go to 10
       else
cc        write(66,*) ' Tofx() giving up after',k,' bisections.'
        Tofx=tM
       endif
      endif
cc      write(66,*) 'Tofx solved stage 1; th=',tM,' gives xM=',xM
cc      write(66,*) '  inputs were a=',a,', while oal=',oal,
cc     &           ' and sta=',sta
c     For parallel flow, the task is finished; for counterflow, the
initial
c     conditions are established, and the evaluation can proceed.
      if(a.lt.0.) return
c     Set up bisection for th between 1. and Tofx, and find
c     thd value appropriate to 'sta'
      targ=sta
      tL=1.
      tR=Tofx
      xL=0.
      xR=oal
      if(sta.gt.oal) then
       xR=1.e11
       thc=amax1(0.,(Tofx-a)/(1.-a))
       tR=eps+thc
      endif
      sL=sign(1.,xL-targ)
      sR=sign(1.,xR-targ)
      k=0
   20 continue
      tM=0.5*(tL+tR)
      th=tM
      targ=sta
cc      write(66,*) ' Calling XCOUNT with inputs a,thdo,th=',a,thdo,th
      xM=xcount(a,thdo,thdin,th,thd)
      sM=sign(1.,xM-targ)
cc      write(66,*) 'XCOUNT returns tL,tR,xL,xR,sL,sR,xM='
cc      write(66,*) tL,tR,xL,xR,sL,sR,xM
      if((sM*sR).gt.0.) then
c      Left half contains root:
       xR=xM
```

```
       sR=sM
       tR=tM
      else
c      Right half contains root:
       xL=xM
       sL=sM
       tL=tM
      endif
      if(abs(tR-tL).le.eps) then
c      Converged
       Tofx=0.5*(tL+tR)
      else
       k=k+1
       if(k.le.kmax) then
c       Continue bisection
        go to 20
       else
cc        write(66,*) ' Tofx() giving up after',k,' bisections.'
        Tofx=tM
        return
       endif
      endif
cc      write(66,*) 'Tofx succeeded in counterflow; th=',tM,
cc     &           ' gives xM=',xM,' while sta=',sta
      return
      end
c
      function xend(th,a,thdin,thdo,thd)
c**********************************************************************
c**********************************************************************
c**                                                                   **
c**         Subroutines for Flat Glass Lehr Thermal Simulation        **
c**            written by Lee A. Bertram                              **
c**           Sandia National Laboratories                            **
c**            Livermore, CA  94551                                   **
c**                                                                   **
c**                 April 2000                                        **
c**                                                                   **
c**                                                                   **
c**           Copyright 2000 (c)  Sandia Corporation                  **
c**********************************************************************
c**********************************************************************
c     Evaluate closed-form solution for glass length 'xend' at which
the
c     dimensionless absolute temperature Tg/Tgo = th, when given the
mass flow
c     ratio a = [(dm/dt)*Cp]g/[(dm/dt)Cp]d and duct inflow
thdin=Td/Tgo.
c     In counterflow case, a smooth asymptotic approximation is
provided when
c     Td is nearly a*Tg. Returns duct temperatures at xo (thdo) and at
x (thd),
c     as well as dimensionless length xend=(x-xo)/Lc.
c     Error flags are xend=-1. (parallel flow error) and -2.
(counterflow error).
      if(abs(a).le.2.e-4) then
c      Isothermal ducts with infinite heat capacity:
```

```
       thdo=thdin
       val=xmin(thdo,th)
       xend=val
       thd=thdo
       return
      endif
      if(a.gt.0.) then
c      Counterflow:
       thd=thdin
       b=thd-a*th
       thdo=b+a
       thc=(thdo-a)/(1.-a)
c      Checking input 'th' consistency: on same side of 'thc' as 1?
       sc=sign(1.,(th-thc)*(1.-thc))
      if(sc.lt.0.) then
cc        write(66,*) 'XEND inputs physically unrealistic:'
cc        write(66,*) '  With a,b,th,thdin,thdo=',a,b,th,thdin,thdo
cc        write(66,*) 'Return xend=-2. error flag.'
        xend=-2.
        return
       endif
c      Select heuristic join between asymptotics and direct evaluation.
       blow=amin1((-0.118*(1.-a**4)),(0.0827*(1.-a**4)))
       bhigh=amax1((-0.118*(1.-a**4)),(0.0827*(1.-a**4)))
       blow=(-0.118*(1.-a**4))
       bhigh=(0.0827*(1.-a**4))
cc       write(66,*) 'XEND blow=',blow,' and bhigh=',bhigh,', b=',b
       if(abs(1.-a**2).le.1.e-4) then
cc        write(66,*) 'Degenerate XEND case with a=1.'
c       Requires direct evaluation to avoid NaNs, Infs.
        bhigh=0.5*blow
        if(abs(b).le.(1.e-3)) then
cc          write(66,*) ' XEND has degenerate case a=b=0 with a,b=',a,b
          xend=1.e10
          return
        endif
       endif
      else
c      Parallel flow:
       thdo=thdin
c      Check for physically possible solution by computing converging
c      temperature value for duct and ribbon:
       thc=(thdo-a)/(1.-a)
       if((th-thc)*(1.-thc).lt.0.) then
cc        write(66,*) 'Inputs to XEND are not physically realizable:'
cc        write(66,*) ' The given ratio of mass flows was a=',a
cc        write(66,*) ' and the initial temperatures,  th=1 and
thdo=',thdo
cc        write(66,*) ' Convergence T/Tgo =', thc,'.'
cc        write(66,*) 'Abort XEND, return xend=-1.'
        xend=-1.
        return
       endif
       b=thdo-a
       thd=a*th+b
c      Suppress calls to power series in 'b':
       blow=1.1
```

```
       bhigh=1.0
      endif
      if(b.gt.blow.and.b.lt.bhigh) then
       xth=xasymp(a,b,th)
      else
c      Direct evaluation of general solution:
       xth=xdir(a,thdo,th)
      endif
      xend=xth
      return
      end
c
      function xcount(a,thdo,thdin,th,thd)
c**********************************************************************
c**********************************************************************
c**                                                                   **
c**         Subroutines for Flat Glass Lehr Thermal Simulation        **
c**            written by Lee A. Bertram                              **
c**           Sandia National Laboratories                            **
c**            Livermore, CA  94551                                   **
c**                                                                   **
c**                 April 2000                                        **
c**                                                                   **
c**                                                                   **
c**           Copyright 2000 (c)  Sandia Corporation                  **
c**********************************************************************
c**********************************************************************
c     Evaluate closed-form solution for glass length 'xcount' in
counterflow.
c     Inputs are dimensionless heat advection ratio 'a', the duct
initial
c     dimensionless absolute temperature Tdo/Tgo = thdo, and a
c     glass temperature th=Tg/Tgo which is not at the station x
c     where the duct inflow temperature 'thdin' occurs; 'xcount' is
c     the station at which 'th' occurs.
c     From these, evaluate the closed-form solution for distance
c     'xcount', and duct temperature thd=Td/Tgo at 'xcount'.
c     The counterflow case uses a smooth asymptotic approximation when
c     Td is nearly a*Tg.
c     Error flag is xend=-2. (counterflow error).
      if(abs(a).le.2.e-4) then
c      Isothermal ducts with infinite heat capacity:
       val=xmin(thdo,th)
       xcount=val
       thd=thdo
       return
      endif
      if(a.gt.0.) then
c      Counterflow:
       b=thdo-a
       thd=a*th+b
       thc=(thdo-a)/(1.-a)
       sc=(th-thc)*(1.-thc)
       if(sc.lt.0.) then
cc        write(66,*) 'XCOUNT inputs physically unrealistic:'
cc        write(66,*) '  Inputs were a,th,thdin=',a,th,thdin
cc        write(66,*) 'Return xcount=-2. error flag.'
```

```
        xcount=-2.
        return
       endif
c      Select heuristic joint between asymptotics and direct
evaluation.
       blow=(-0.118*(1.-a**4))
       bhigh=(0.0827*(1.-a**4))
      else
cc       write(66,*) 'XCOUNT has parallel flow input a=',a
       xcount=-2.
       return
      endif
      if(b.gt.blow.and.b.lt.bhigh) then
       xth=xasymp(a,b,th)
      else
c      Direct evaluation of general solution:
       xth=xdir(a,thdo,th)
      endif
      xcount=xth
      return
      end
c
      function Tdin(th,oal,a,thd,thdo)
c**********************************************************************
c**********************************************************************
c**                                                                  **
c**         Subroutines for Flat Glass Lehr Thermal Simulation       **
c**            written by Lee A. Bertram                             **
c**           Sandia National Laboratories                           **
c**            Livermore, CA  94551                                  **
c**                                                                  **
c**                 April 2000                                       **
c**                                                                  **
c**                                                                  **
c**           Copyright 2000 (c)  Sandia Corporation                 **
c**********************************************************************
c**********************************************************************
c     Given a dimensionless lehr length 'oal', the heat advection ratio
'a'
c     and the target glass temperature th=Tg/Tgo, find and return as
'Tdin'
c     the duct input absolute dimensionless temperature ratio 'thdin'
c     Also return duct temperature ratio at exit oal (thd) and at glass
c     entry xo (thdo), unambiguously labelled.
      k=0
      kmax=50
      eps = 1.e-6
      tol=amax1(eps,eps*oal)
c     Start by seeking parallel flow (a < 0) solution:
      if(abs(1.-th).lt.eps) then
cc       write(66,*) '  Abort Tdin() because ribbon ends with th =',
cc     &               ' Tg/Tgo =',th
       Tdin=-1.
       return
      endif
      if(a.lt.0.) then
```

```
c      Locate smallest feasible 'oal' value, with thdo=0 in parallel
flow:
       thdin=0.
       tL=0.
       aminl=xend(th,a,thdin,thdo,thd)
cc       write(66,*) 'Parallel flow aminl=',aminl,' for thdo=0 in
Tdin().'
c      Now locate infinite 'oal' value, with th=convergence in parallel
flow:
       thdin=a+(1.-a)*th -1.e-6
       tR=thdin
       ainfl=xend(th,a,thdin,thdo,thd)
cc       write(66,*) '    Max oal=ainfl=',ainfl,' for thdo=thc in
Tdin().'
      else
c      Range of thdo for counterflow (a > 0)
cc       write(66,*) ' Set up bisection for counterflow.'
c      Lower limit for thdo in cooling:
       tL=a*(1.-th)
c      Upper limit for thdo in heating with a > 1:
       tR=100.
       if(th.gt.1.) then
c       Heating:
        tL=1.+10.*eps
        if(a.lt.1.) then
         tL=a+(1.-a)*th
        endif
       else
c       Cooling:
        tR=1.-10.*eps
        tR=amin1(tR,0.999*(th+a*(1.-th)))
       endif
       thdo=tL
       thdin=a*th+thdo-a
       aminl= xcount(a,thdo,thdin,th,thd)
c      Infinite length when thdo=tR:
       amaxl= xcount(a,tR,thdin,th,thd)
c      Because these may interchange roles, check to assure
c      that tL is associated with the smaller x-xo:
       if(amaxl.lt.aminl) then
        xL=amaxl
        amaxl=tL
        tL=tR
        tR=amaxl
        xR=aminl
       else
        xL=aminl
        xR=amaxl
       endif
       ainfl=xR
cc       write(66,*) ' Counterflow: ',tL,'.lt.thdo.lt.',tR,'.'
cc       write(66,*) '              ',xL,'.lt.oal.lt.',xR,'.'
       thdo=0.5*(tL+tR)
       if(thdo.ne.123.) go to 6
       do 5 i=1,kmax
        thdinx=thdinx+0.1*((0.5)**i)
        thdo=thdinx
```

```
        thdin=a*th+thdo-a
        ainflx= xcount(a,thdo,thdin,th,thd)
        if(ainflx.gt.0.) then
         tR=thdo
         ainfl=ainflx
        else
         go to 6
        endif
    5  continue
    6  continue
       thdin=a*th+thdo-a
       ainfl= xcount(a,tR,thdin,th,thd)
       aminl=xL
cc       write(66,*) ' Counterflow range for oal is',aminl,' to',
cc     &               ainfl,'.'
cc       write(66,*) '  obtained for thdo=',tL,tR
      endif
      if(oal.gt.ainfl) then
c      Singular solution:
       Tdin=thdin
       if(a.gt.0.) then
        thd=thdin
        thdo=thd-a*th+a
       else
        thdo=thdin
        thd=a*th+thdo-a
       endif
       return
      elseif(oal.lt.aminl) then
c      Physically inconsistent inputs to Tdin():
cc       write(66,*) ' Inconsistent inputs to Tdin(): oal=',oal
cc       write(66,*) ' but aminl=',aminl,' is minimum possible',
cc     &             ' ribbon length'
cc       write(66,*) ' for input th,a=',th,a
       Tdin=-1.
       thdo=-1.
       thd=-1.
       return
      endif
    7 continue
c     Bisect interval:
      xR=ainfl
      xL=aminl
      targ=oal
      sL=sign(1.,xL-targ)
      sR=sign(1.,xR-targ)
cc      write(66,*) '  Bisections for Tdin(): xL,oal,xR,tM'
   10 continue
      tM=0.5*(tL+tR)
cc      write(66,*)  xL,oal,xR,tM
      thdin=tM
      if(a.lt.0.) then
       xM=xend(th,a,thdin,thdo,thd)
      else
       thdo=tM
       thdin=a*th+thdo-a
       xM= xcount(a,thdo,thdin,th,thd)
```

```
      endif
      if(xM.lt.0.) then
c      Identify error flag:
       write(6,*) ' Abort in Tdin().'
       if(xM.gt.-1.5) then
cc        write(66,*) ' Error in XEND for parallel flow; a,th,thdin=',
cc     &              a,th,thdin
        stop
       else
        write(6,*) ' Error in XEND for counterflow; a,th,thdin=',
     &               a,th,thdin
        stop
       endif
      endif
      sM=sign(1.,xM-targ)
      if((sM*sR).gt.0.) then
c      Left half contains root:
       xR=xM
       sR=sM
       tR=tM
      else
c      Right half contains root:
       xL=xM
       sL=sM
       tL=tM
      endif
      if(abs(tR-tL).le.eps) then
c      Converged
       Tdin=0.5*(tL+tR)
      else
       k=k+1
       if(k.le.kmax) then
c       Continue bisection
        go to 10
       else
cc        write(66,*) ' Tdin() giving up after',k,' bisections.'
        Tdin=tM
       endif
      endif
      if(a.gt.0.) then
c      Counterflow:
       thdo=Tdin
       thd=a*th+thdo-a
       Tdin=thd
      endif
cc      write(66,*) 'Tdin solved for thdin=',Tdin,', giving xM=',xM
cc      write(66,*) '  inputs were a=',a,', and oal=',oal
      return
      end
c
      function xdir(a,thdo,th)
c**********************************************************************
c**********************************************************************
c**                                                                  **
c**         Subroutines for Flat Glass Lehr Thermal Simulation       **
c**            written by Lee A. Bertram                             **
c**           Sandia National Laboratories                           **
```

```
c**            Livermore, CA  94551                                    **
c**                                                                    **
c**                 April 2000                                         **
c**                                                                    **
c**                                                                    **
c**           Copyright 2000 (c)  Sandia Corporation                   **
c**********************************************************************
c**********************************************************************
c     Evaluating full solution directly to get dimensionless
c     distance 'xdir' for given heat advection ratio 'a', given
c     duct temperature at station xo 'thdo', and final temperature
c     ratio 'th'.  Calling program must have checked that a.ne.0.,
c     that a.ne.1, and that th.gt.(thdo-a)/(1-a) so that no
c     operation below is singular.
c       write(66,*) ' Direct XEND with th,thd,a,thdo=',
c     &             th,thd,a,thdo
cc      if(abs(b).gt.1.e-3) then
c      Nonzero b; full solution:
       b=(thdo-a)
       thd=a*th+b
        thsq=th*th
        thdsq=thd*thd
        f1=(thsq+thdsq)/(1.+thdo**2)
        f2=(1.-thdo**2)/(thsq-thdsq)
        f12=f1*f2
        fp3=(th-thd)/(1.-thdo)
        fp4=(1.+thdo)/(th+thd)
        arg=fp3*fp4
        exp1=(1.+a**2)/(2.*a)
        f3=arg**exp1
        val=(th*thdo-thd)/(th+thdo*thd)
        val2=atan(val)
        val3=-(1.-a**2)*val2/a
        if((f1*f2*fp3*fp4-1.).gt.2.e-3) then
         xth=-a*(val3+alog(f1*f2*f3))/(2.*b**3)
        else
c        Expand ln(1-eps) as -eps + 0.5*eps**2 :
         f14=2.*(thdo*((th-thd)**2)-((1-thdo)**2)*th*thd)
         f14=f14/((1.+thdo**2)*((th+thd)**2))
         f12=f14+0.5*(f14**2)+((1.-a)**2)*(alog(fp3*fp4))/(2.*a)
         xth=-a*(val3+f12)/(2.*b**3)
        endif
       xdir=xth
       return
       end
c
       function xmin(thdo,th)
c**********************************************************************
c**********************************************************************
c**                                                                    **
c**         Subroutines for Flat Glass Lehr Thermal Simulation         **
c**            written by Lee A. Bertram                               **
c**           Sandia National Laboratories                             **
c**            Livermore, CA  94551                                    **
c**                                                                    **
c**                 April 2000                                         **
c**                                                                    **
```

```
c**                                                                   **
c**           Copyright 2000 (c)  Sandia Corporation                  **
c***********************************************************************
c***********************************************************************
c     For the maximum cooling rate, with a=0, the duct remains
c     isothermal at 'thdo' and the distance to reach 'th' is
c     the minimum possible.  Distance 'xmin' is that dimensionless
c     value.  Calling program must have checked that a =0.,
c     and that 'th' and 'thdo' are such that no operation below
c     is singular.
       f1=thdo*(th-1.)/(th+thdo**2)
       f1=atan(f1)
       f2=(th-thdo)/(1.-thdo)
       f2=f2*(1.+thdo)/(th+thdo)
       if(f2.gt.0.) then
        f2=0.5*alog(f2)
       else
        xmin=1.
        return
       endif
c      xmin=f2
      xmin = (f1-f2)/(2.*(thdo**3))
      return
      end
c
      function xasymp(a,b,th)
c***********************************************************************
c***********************************************************************
c**                                                                   **
c**         Subroutines for Flat Glass Lehr Thermal Simulation        **
c**            written by Lee A. Bertram                              **
c**           Sandia National Laboratories                            **
c**            Livermore, CA  94551                                   **
c**                                                                   **
c**                 April 2000                                        **
c**                                                                   **
c**                                                                   **
c**           Copyright 2000 (c)  Sandia Corporation                  **
c***********************************************************************
c***********************************************************************
      b4=1.e-4
      b3=1.e-4
      b2=1.e-6
      b1=1.e-12
c      Degenerate case; b=0 solution
       a4m1=1.-a**4
       e=b/a4m1
       xth=((1./(th**3))-1.)/(3.*a4m1)
       x0=xth
       if(abs(b).gt.b1) then
c       Asymptotic linear correction:
        xth=xth-e*(a**3)*(1.-1./(th**4))/(a4m1)
       endif
       x1=xth
       if(abs(b).gt.b2) then
c       Asymptotic quadratic correction:
        xth=xth-0.4*((e*a)**2)*(3.+5.*a**4)*
```

```
     &              (1.-1./(th**5))/(a4m1)
       endif
       x2=xth
       if(abs(b).gt.b3) then
c       Asymptotic cubic correction:
       xth=xth-(e**3)*(2.*a/3.)*(1.+10.*a**4+5.*a**8)*
     &              (1.-1./(th**6))/(a4m1)
       endif
       x3=xth
       if(abs(b).gt.b4) then
        pofa=1.+65.*(a**4)+155.*(a**8)+15.*(a**12)
        xth=xth+pofa*(e**4)*((1./(th**7))-1.)/(7.*a4m1)
       endif
       x4=xth
      xasymp=xth
      return
      end
c
```

APPENDIX C. Source Code for Radiative Viewfactors

```
program ex_vf_cyl
      dimension Fkl(101)
c     Calculates single heater element viewfactors.
c     Fkl is an array of the viewfactor between a sample point
c     point (xi,yj,0) in the (x,y) plane with area dA1, and a
c     cylindrical element of length cL, diameter 2r, with its
c     centroid over the origin at height h1.  Each row at constant
c     xi is computed as an array Fkl(j), and written to the
c     output file 'Fkl.dat'.
c**********************************************************************
c**      Issued by Sandia National Laboratories, operated for the  **
c**      United States Department of Energy by Sandia Corporation. **
c**                          NOTICE                                **
c**      This program was prepared in the course of work sponsored **
c**      by an agency of the U.S.Government.  Neither the United   **
c**      States Government nor any agency thereof, nor any of their**
c**      employees, makes any warranty, express or implied, or     **
c**      assumes any legal liability or responsibility for the     **
c**      accuracy, completeness, or usefulness of any information, **
c**      apparatus, product, or process disclosed, or represents   **
c**      that its use would not infringe privately owned rights.   **
c**      Reference herein to any specific commercial product,      **
c**      process, or service by trade name, trademark, manufacturer,*
c**      or otherwise, does not necessarily constitute nor imply   **
c**      its endorsement, recommendation, or favoring by the United**
c**      States Government, any agency thereof or any of their     **
c**      contractors or subcontractors.  The views and opinions    **
c**      expressed herein do not necessarily state or reflect      **
c**      those of the United States Government, any agency thereof **
c**      or any of their contractors or subcontractors.            **
c**      This code is supplied for official government use only.   **
c**      No further dissemination is permitted without specific    **
c**      permission from Sandia Laboratories, Livermore, CA 94551. **
c**********************************************************************
c**********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c**********************************************************************
c**********************************************************************
      open(unit=10,file='Fkl.dat',status='unknown')
      open(unit=11,file='Fkl.log',status='unknown')
      open(unit=12,file='Fkl.plt',status='unknown')
      open(unit=13,file='vwx.plt',status='unknown')
      pi=4.*atan(1.)
      cL=10./12.
      r=1./24.
c     Quadrature to get viewfactor of cylinder of radius r,
c     length cL.  Sample point dA1 is distance yc perpendicular
c     to cylinder axis, and height hc from axis.  Cylinder's near
c     face is xf away from dA1; its far end is at xf+cL. Ends are
c     not included in viewfactor.  Sample point dA1 is at height
c     h1 above origin: h1=hc+r.
c     Start at element centroid:
```

```
      hc=0.5
c      hc=2.
      xf=-0.5*cL
      dx=1./12.
      dy=1./12.
      jmax=51
      imax=51
      write(6,*) '    '
      write(6,*) 'Computing cylinder viewfactor from dA1.'
      write(6,*) 'Writing ',imax,' X ',jmax,' array of viewfactors.'
      write(6,*) 'Viewfactors written to file "Fkl.dat" for axial'
      write(6,*) 'and transverse spacings dx,dy=', dx,dy,'.'
      write(10,*) imax,jmax,dx,dy,cL,r,hc
      sum=0.
      rect=0.
      dsum=2.*hc*dx/(r*cL)
      drect=2.*dx*dy/(pi*r*cL)
      zc=2.0
      h1=hc+zc
      do 30 i=1,imax
      yc=0.0
      do 20 j=1,jmax
       ycn=yc
       xarg=xf
       Fkl(j)=cyl2(ycn,zc,xarg,r,cL,h1)
       write(12,*) yc,Fkl(j)
       rfac=1.
       if(j.eq.1.or.j.eq.jmax) then
        qfac=1.
        rfac=0.5
        if(i.eq.1.or.i.eq.imax) then
         qfac=0.5
         rfac=0.25
        endif
        if(j.eq.1) then
         sum=sum+qfac*dsum*Fkl(j)
        endif
        if(j.ne.jmax) then
         write(13,*) xf,Fkl(j),sum,rect
        endif
       endif
       rect=rect+rfac*drect*Fkl(j)
       yc=yc+dy
   20 continue
      xf=xf+dx
      write(10,101) (Fkl(j),j=1,jmax)
   30 continue
      close(10)
      close(11)
      close(12)
  101 format(31e13.4)
      write(6,*) '           Viewfactors done.'
      write(6,*) '     Enclosing cylinder P1/P2=',sum
      write(6,*) '         Rectangle P1/P2=',rect
      write(6,*) '  '
      stop
      end
```

```
c
c
      function vert(a,b,c,g,h)
c     Compute viewfactor of rectangle with sides a and b, in a plane
c     perpendicular to the plane of a sample point with area dA1.
c     Let lower left corner of the rectangle have coordinates (g,h)
c     (displaced by g along side a, and by h along
c     and side b) relative to normal dropped from dA1 onto
c     rectangle's plane. (Both g and h can be negative).
c     See Spiegel & Howell Appendix C: 'Selected Configuration
c     Factors'.
c*********************************************************************
c*********************************************************************
c**      Issued by Sandia National Laboratories, operated for the  **
c**      United States Department of Energy by Sandia Corporation. **
c**                         NOTICE                                 **
c**      This program was prepared in the course of work sponsored **
c**      by an agency of the U.S.Government.  Neither the United   **
c**      States Government nor any agency thereof, nor any of their**
c**      employees, makes any warranty, express or implied, or     **
c**      assumes any legal liability or responsibility for the     **
c**      accuracy, completeness, or usefulness of any information, **
c**      apparatus, product, or process disclosed, or represents   **
c**      that its use would not infringe privately owned rights.   **
c**      Reference herein to any specific commercial product,      **
c**      process, or service by trade name, trademark, manufacturer,*
c**      or otherwise, does not necessarily constitute nor imply   **
c**      its endorsement, recommendation, or favoring by the United**
c**      States Government, any agency thereof or any of their     **
c**      contractors or subcontractors.  The views and opinions    **
c**      expressed herein do not necessarily state or reflect      **
c**      those of the United States Government, any agency thereof **
c**      or any of their contractors or subcontractors.            **
c**      This code is supplied for official government use only.   **
c**      No further dissemination is permitted without specific    **
c**      permission from Sandia Laboratories, Livermore, CA 94551. **
c*********************************************************************
c*********************************************************************
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*********************************************************************
c*********************************************************************
      pi=4.*atan(1.)
      tol=1.e-4
c      write(13,*) 'VERT gets',a,' X',b,' rectangle at',c,
c     &' from dA1; offsets vert,horiz=',g,h
      r=sqrt((a**2)+(b**2))
      if(c.lt.tol*r) then
c      dA1 is in contact with rectangle's plane;
       if((g).lt.r*tol.and.(h).lt.r*tol) then
        vert=0.50
       else
        vert=0.
       endif
       if(abs(g).lt.r*tol.and.abs(h).lt.r*tol) then
```

```
        vert=0.25
       endif
       return
      endif
      disp=sqrt(g**2+h**2)
      if( (r.lt.c*tol).or.
     &    (g.gt.tol.and.h.gt.tol.and.r.lt.tol*disp) ) then
c      Sample point at infinity:
       vert=0.
       return
      endif
c     First step: rectangle (a+g) X (h+b), with dA1 on its edge (from
c     Handbook tables):
      sg=sign(1.,g+0.1*tol)
      x=(a+g)/(b+h)
      y=c/(b+h)
      rxy=sqrt(x**2+y**2)
      vert=atan(1./y)-(y/rxy)*atan(1./rxy)
c      write(11,*) c,x,y,rxy,vert
      v1=vert
      if(abs(h).lt.tol*r) go to 10
c     Subtract rectangle  (a+g)Xh with dA1 on its edge:
      sh=sign(1.,h+0.1*tol)
      x=(a+g)/h
      y=c/(h*sh)
      rxy=sqrt(x**2+y**2)
      v2=sh*(atan(1./y)-(y/rxy)*atan(1./rxy))
      vert=vert-v2
   10 continue
      if(abs(g).lt.tol*r) go to 20
c     Subtract rectangle  gX(b+h) with dA1 on its edge:
      x=(g*sg)/(b+h)
      y=c/(b+h)
      rxy=sqrt(x**2+y**2)
      v3=sg*(atan(1./y)-(y/rxy)*atan(1./rxy))
      vert=vert-v3
   20 continue
      if((abs(g).lt.tol*r).or.(abs(h).lt.tol*r)) go to 30
c     Add back double-subtracted rectangle gXh with dA1 on its edge:
      sf=sg*sh
      x=g*sg/h*sh
      y=c/(h*sh)
      rxy=sqrt(x**2+y**2)
      v4=sf*(atan(1./y)-(y/rxy)*atan(1./rxy))
      vert=vert+v4
   30 continue
c      write(13,*) h,v1,v2,v3,v4,sg,sh,sf
      vert=vert/(2.*pi)
      return
      end
c
      function horiz(a,b,c,g,h)
c     Compute viewfactor of rectangle with sides a and b, in a plane
c     parallel to the plane of a sample point with area dA1.
c     Let lower left corner of the rectangle have coordinates (g,h)
c     (displaced by g along side a, and by h along
c     and side b) relative to normal dropped from dA1 onto
```

```
c     rectangle's plane. (Both g and h can be negative)
c     See Spiegel & Howell Appendix C: 'Selected Configuration
c     Factors'.
c*****************************************************************
c*****************************************************************
c**      Issued by Sandia National Laboratories, operated for the  **
c**      United States Department of Energy by Sandia Corporation. **
c**                         NOTICE                                 **
c**      This program was prepared in the course of work sponsored **
c**      by an agency of the U.S.Government.  Neither the United   **
c**      States Government nor any agency thereof, nor any of their**
c**      employees, makes any warranty, express or implied, or     **
c**      assumes any legal liability or responsibility for the     **
c**      accuracy, completeness, or usefulness of any information, **
c**      apparatus, product, or process disclosed, or represents   **
c**      that its use would not infringe privately owned rights.   **
c**      Reference herein to any specific commercial product,      **
c**      process, or service by trade name, trademark, manufacturer,*
c**      or otherwise, does not necessarily constitute nor imply   **
c**      its endorsement, recommendation, or favoring by the United**
c**      States Government, any agency thereof or any of their     **
c**      contractors or subcontractors.  The views and opinions    **
c**      expressed herein do not necessarily state or reflect      **
c**      those of the United States Government, any agency thereof **
c**      or any of their contractors or subcontractors.            **
c**      This code is supplied for official government use only.   **
c**      No further dissemination is permitted without specific    **
c**      permission from Sandia Laboratories, Livermore, CA 94551. **
c*****************************************************************
c*****************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*****************************************************************
c*****************************************************************
      pi=4.*atan(1.)
      tol=1.e-4
      r=sqrt((a**2)+(b**2))
      if(c.lt.tol*r) then
c      dA1 is in contact with rectangle's plane;
       if((g).lt.r*tol.and.(h).lt.r*tol) then
        vert=1.0
       else
        vert=0.
       endif
       if(abs(g).lt.r*tol.and.abs(h).lt.r*tol) then
        vert=0.25
       endif
       return
      endif
      disp=sqrt(g**2+h**2)
      if( (r.lt.c*tol).or.
     &    (g.gt.tol.and.h.gt.tol.and.r.lt.tol*disp) ) then
c      Sample point at infinity:
       vert=0.
       return
      endif
```

```
c     First step: rectangle (a+g) X (h+b), with dA1 on its edge (from
c     Handbook tables):
      sg=sign(1.,g)
      sh=sign(1.,h)
      x=(a+g)/c
      y=(b+h)/c
      rx=sqrt(1.+x**2)
      ry=sqrt(1.+y**2)
      horiz=((x/rx)*atan(y/rx)+(y/ry)*atan(x/ry))
      if(abs(h).lt.tol*r) go to 10
c     Subtract rectangle  (a+g)Xh with dA1 on its edge:
      x=(a+g)/c
      y=h*sh/c
      rx=sqrt(1.+x**2)
      ry=sqrt(1.+y**2)
      horiz=horiz-sh*((x/rx)*atan(y/rx)+(y/ry)*atan(x/ry))
   10 continue
      if(abs(g).lt.tol*r) go to 20
c     Subtract rectangle  gX(b+h) with dA1 on its edge:
      x=(g*sg)/c
      y=(b+h)/c
      rx=sqrt(1.+x**2)
      ry=sqrt(1.+y**2)
      horiz=horiz-sg*((x/rx)*atan(y/rx)+(y/ry)*atan(x/ry))
   20 continue
      if((abs(g).lt.tol*r).or.(abs(h).lt.tol*r)) go to 30
c     Add back double-subtracted rectangle gXh with dA1 on its edge:
      sf=sg*sh
      x=g*sg/c
      y=h*sh/c
      rx=sqrt(1.+x**2)
      ry=sqrt(1.+y**2)
      horiz=horiz+sf*((x/rx)*atan(y/rx)+(y/ry)*atan(x/ry))
   30 continue
      horiz=horiz/(2.*pi)
      return
      end
c
      function cyl(dc,hc,yc,r,cL)
c*********************************************************************
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                    *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*********************************************************************
c*********************************************************************
c     Compute the viewfactor of a cylinder of length cL, radius r as
c     seen from dA1 which is hc below and dc to the right of the
c     cylinder axis; the axis lies parallel to +y. Use the viewfactor
c     of a strip of inclination alpha, width dw,  and displacements
c     (d,h), lying from yc to y2; obviously, cL=y2-yc.
c     RESTRICTION:  y2 must be positive, nonzero. Argument yc is
c     modified to  conform with this rule.
      pi=4.*atan(1.)
      tol=1.e-4
      rtol=tol*sqrt(dc**2+hc**2+(yc+cL)**2)
      if(yc.lt.0..and.(yc+cL).lt.rtol) then
```

```
       yc=-(yc+cL)
      endif
      sdc=sign(1.,dc)
      dc=sdc*dc
      if(dc.lt.rtol) then
c       write(11,*) ' CYL sees dc,hc,yc,rtol=',dc,hc,yc,rtol,
c     &             ' in closed form evaluation.'
c      dA1 is on cylinder axis: check that it is outside cylinder:
       if(hc.gt.r-rtol) then
c       Closed form solution is:
        sc=1.
        cyl1=0.
        if(abs(yc).gt.rtol) then
c        Needs two evaluations:
         sc=sign(1.,yc)
         aL=sc*yc/r
         aH=hc/r
         aX=(1.+aH)**2+aL**2
         aY=(1.-aH)**2+aL**2
         cyl1=atan(sqrt((aH-1.)/(aH+1.)))
         cyl2=atan(sqrt(aX*(aH-1.)/(aY*(aH+1.))))
         cyl1=((aX-2.*aH)*cyl2/sqrt(aX*aY))-cyl1
         cyl1= (atan(aL/sqrt(aH**2-1.))+aL*cyl1)/aH
        endif
        aL=(yc+cL)/r
        aH=hc/r
        aX=(1.+aH)**2+aL**2
        aY=(1.-aH)**2+aL**2
        cyl2=atan(sqrt((aH-1.)/(aH+1.)))
        cyl3=atan(sqrt(aX*(aH-1.)/(aY*(aH+1.))))
        cyl2=((aX-2.*aH)*cyl3/sqrt(aX*aY))-cyl2
        cyl2= (atan(aL/sqrt(aH**2-1.))+aL*cyl2)/aH
        cyl=(cyl2-sc*cyl1)/pi
       else
c       Inside (or on) cylinder:
        cyl=0.
        if(abs(hc-r).lt.rtol) cyl=1.
       endif
       dc=sdc*dc
       return
      endif
c     dA1 has definite offset from cylinder axis: quadrature required.
c     dA1 Can be inside or on cylinder; check (dc**2+hc**2)>r**2:
      dist=sqrt(dc**2 + hc**2)
      if(abs(dist-r).lt.rtol) then
c      On cylinder at angle atan(hc/dc):
       cyl=(atan(hc/dc))/pi
       return
      elseif(dist.lt.r-rtol) then
       cyl=0.
       return
      endif
c     Compute limits of integration; 'ang' measured from -y going ccw:
      dphi=atan(r/sqrt(dc**2+hc**2))
      angc=atan(hc/dc)
      ang1=dphi-angc
      ang2=pi-(dphi+angc)
```

```
c     Choose number of evaluation points in quadrature (minimum
allowed=3):
      iq=51
      iq=max1(3,iq)
      dang=(ang2-ang1)/float(iq-1)
c     End integration limits
c     Perform trapezoidal rule quadrature:
      ang=ang1+0.5*dang
      d=dc-r*sin(ang)
      h=hc-r*cos(ang)
      y1=yc
      y2=yc+cL
      dw=r*dang
c      write(11,*) ' CYL quadrature with ang1,ang,dang,d,h,y1,y2,dw='
c      write(11,*)  ang1,ang,dang,d,h,y1,y2,dw
      angs=0.5*pi-ang
      dF=0.5*strip(d,h,y1,y2,angs,dw)
c      write(11,*) ' CYL quadrature with ang1,ang2,dang,iq,dF1='
c      write(11,*)  ang1,ang2,dang,iq,dF
      do 10 i=2,iq-1
       ang=ang+dang
       d=dc-r*sin(ang)
       h=hc-r*cos(ang)
       y1=yc
       y2=yc+cL
       dw=r*dang
       angs=0.5*pi-ang
       dFq=strip(d,h,y1,y2,angs,dw)
c       write(11,*)  ang,angs,dang,iq,dFq
       dF=dFq+dF
   10 continue
      ang=ang2-0.5*dang
      d=dc-r*sin(ang)
      h=hc-r*cos(ang)
      y1=yc
      y2=yc+cL
      dw=r*dang
      angs=0.5*pi-ang
      dF=dF+0.5*strip(d,h,y1,y2,angs,dw)
      cyl=dF
      dc=sdc*dc
      return
      end
c
      function strip(d,h,y1,y2,alpha,dw)
c     Let dA1 be on the x-axis, with normal in +z direction, and
calculate the viewfactor
c     of a strip of width dw, with its normal at alpha radians ccw from
+x.  The strip is
c     parallel to the y-axis, and is at height h above the (x,y) plane;
x-displacement
c     between dA1 and strip is d.  Strip lies between y1 and y2
c*********************************************************************
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
```

```
c*********************************************************************
c*********************************************************************
      pi=4.*atan(1.)
      tol=1.e-4
      rtol=tol*sqrt(d**2 + h**2 + y1**2)
c     Start with (0,y1) value:
      strip=0.
      sp=1.
      if(abs(y1).gt.rtol) then
       sp=sign(1.,y1)
       p=sp*y1/sqrt(h**2 + d**2 )
       strip=atan(p)+p/(1.+(p**2))
       strip=strip*(d*cos(alpha)+h*sin(alpha))
     &              *(h/((sqrt(d**2+h**2))**3))
       strip1=sp*strip*dw
c       write(13,*) strip1,p,y1,d,h,alpha,dw
      endif
      if(abs(y2-y1).gt.rtol) then
       p=y2/sqrt(h**2 + d**2 )
       strip2=atan(p)+p/(1.+(p**2))
       strip2=strip2*(d*cos(alpha)+h*sin(alpha))
     &              *(h/((sqrt(d**2+h**2))**3))
       strip2=strip2*dw
c       write(13,*) strip2,p,y2,d,h,alpha,dw
       strip=strip2-(strip1)
      else
       strip=0.
      endif
c      write(13,*) y1,strip,strip1,strip2,y2
      strip=strip/(2.*pi)
      return
      end
c
      function cyl2(yc,zc,xf,r,cL,h1)
c     Compute the viewfactor of a cylinder of length cL, radius r as
c     seen from sample area dA1.  Sample point is at z=h1, cylinder
c     axis is at (yc,zc).  Thus, dA1 is hc=h1-zc above and dc=yc to
c     the right of the cylinder axis; the axis lies parallel to +x.
c     Using the viewfactor of a strip of inclination alpha, width dw,
c     to assemble the integrand for Trapezoidal Rule quadrature, we
c     compute the viewfactor 'cyl2'.  The cylinder face
c     is distance 'xf' from the (y,z) plane.
c     RESTRICTION:  x2=xf+cL must be positive, nonzero. Argument xf
c     is modified to conform with this rule, but returned intact.
c*********************************************************************
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*********************************************************************
c*********************************************************************
      pi=4.*atan(1.)
      tol=1.e-4
      hc=h1-zc
      dc=yc
      rtol=tol*sqrt(dc**2+hc**2+(xf+cL)**2)
      xflag=0.
```

```
      if(xf.lt.0..and.(xf+cL).lt.rtol) then
       xf=-(xf+cL)
       xflag=1.
      endif
      sdc=sign(1.,dc)
      dc=sdc*dc
      if(dc.lt.rtol) then
c       write(11,*) ' CYL2 sees dc,hc,xf,rtol=',dc,hc,xf,rtol,
c     &             ' dA1 is on-axis.'
c      dA1 is on cylinder axis: check that it is outside cylinder:
       if(hc.gt.r-rtol) then
c       Closed form solution is:
        sc=1.
        cyl1=0.
        if(abs(xf).gt.rtol) then
c        Needs two evaluations:
         sc=sign(1.,xf)
         aL=sc*xf/r
         aH=hc/r
         aX=(1.+aH)**2+aL**2
         aY=(1.-aH)**2+aL**2
         cyl1=atan(sqrt((aH-1.)/(aH+1.)))
         cyl2=atan(sqrt(aX*(aH-1.)/(aY*(aH+1.))))
         cyl1=((aX-2.*aH)*cyl2/sqrt(aX*aY))-cyl1
         cyl1= (atan(aL/sqrt(aH**2-1.))+aL*cyl1)/aH
        endif
        aL=(xf+cL)/r
        aH=hc/r
        aX=(1.+aH)**2+aL**2
        aY=(1.-aH)**2+aL**2
        cyl2=atan(sqrt((aH-1.)/(aH+1.)))
        cyl3=atan(sqrt(aX*(aH-1.)/(aY*(aH+1.))))
        cyl2=((aX-2.*aH)*cyl3/sqrt(aX*aY))-cyl2
        cyl2= (atan(aL/sqrt(aH**2-1.))+aL*cyl2)/aH
        cyl2=(cyl2-sc*cyl1)/pi
c       Quadrature check on analytical value:
c       Integration limits for on-axis dA1:
c        if(r.gt.abs(h1-zc)) then
c         cyl2=0.
c         return
c        else
c         ang1=asin(r/(h1-zc))
c        endif
c        ang2=0.5*pi
c        iq=26
c        iq=max1(3,iq)
c       End integration limits
c       Perform trapezoidal rule quadrature:
c        x1=xf
c        x2=xf+cL
c        cyl2=2.*cylq(iq,ang1,ang2,yc,zc,h1,r,x1,x2)
c        cyl2=cyl2a-cyl2x
c        write(11,*) cyl2,cyl2a,h1,x1,x2
       else
c       Inside (or on) cylinder:
        cyl=0.
        if(abs(hc-r).lt.rtol) cyl=1.
```

```
       endif
       if(xflag.gt.0.1) then
        xf=-(xf+cL)
       endif
       return
      endif
c     dA1 has definite offset from cylinder axis: quadrature required.
c     dA1 Can be inside or on cylinder; check (dc**2+hc**2)>r**2:
      dist=sqrt(dc**2 + hc**2)
      if(abs(dist-r).lt.rtol) then
c      On cylinder at angle atan(hc/dc):
       cyl2=(atan(hc/dc))/pi
       return
      elseif(dist.lt.r-rtol) then
       cyl2=0.
       return
      endif
c     Compute limits of integration; 'ang' measured from -y going ccw:
      rc=sqrt(dc**2+hc**2)
      if(r.gt.rc) then
       cyl2=0.
       return
      else
       dphi=asin(r/rc)
      endif
      angc=atan(hc/dc)
      ang1=angc-0.5*pi+dphi
      ang2=angc+0.5*pi-dphi
      if(hc.lt.r) then
c      Horizon of dA1 intersects cylinder:
       ang2=asin(hc/r)
      endif
c     Choose number of quadrature points (minimum allowed=3):
      iq=51
      iq=max1(3,iq)
c     End integration limits
c     Perform trapezoidal rule quadrature:
      x1=xf
      x2=xf+cL
c      write(11,*) r,rc,angc,dphi,ang1,ang2
      cyl2=cylq(iq,ang1,ang2,yc,zc,h1,r,x1,x2)
      if(xflag.gt.0.1) then
       xf=-(xf+cL)
      endif
      return
      end
c
      function strip2(d,h,x1,x2,alpha,dw)
c     Let dA1 be on the z-axis at height h above the origin, with
c     normal at angle 'alpha' ccw from the  -z direction.
c     Calculate the viewfactor of a strip of width dw, length
c     (x2-x1), with its normal in +z direction.  The strip is
c     parallel to the x-direction, and is distance 'd' from it.
c*********************************************************************
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
```

```
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c**********************************************************************
c**********************************************************************
      pi=4.*atan(1.)
      tol=1.e-4
      rtol=tol*sqrt(d**2 + h**2 + x1**2)
c     Start with (0,x1) value:
      strip=0.
      r2sq=d**2+h**2
      rsq1=r2sq+(x1)**2
      rsq2=r2sq+(x2)**2
      v2=1./sqrt(r2sq)
      sf1=h*((x1/rsq1)+v2*atan(x1*v2))
      sf2=h*((x2/rsq2)+v2*atan(x2*v2))
      sf=(sf2-sf1)*dw/(2.*pi*r2sq)
c      write(11,*) x1,x2,alpha,sf1,sf2
      strip2=sf*(d*sin(alpha)+h*cos(alpha))
c      strip2=sf*(-d*sin(alpha)+h*cos(alpha))
      return
      end
c
      subroutine image(d,r,h,x,ang1,ang2,xcm,rcm,kimage)
c**********************************************************************
c**********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c**********************************************************************
c**********************************************************************
      dimension ang1(*),ang2(*),xcm(*),rcm(*)
c     For a point at distance x beyond the midpoint between
c     rollers of radius r, spacing d, with top of rollers at
c     height h above the plane containing x, compute the angles
c     ang1(k),ang2(k), k=1,2,...,kimage  of the reflected image
c     of the glass between the rollers (at height h).
c
c             0 < x < d/2    and  2r < d     required
c
c     Angles are measured ccw from horizontal at point x. The
c     images are insides of cylinders of radius rcm(k), with
c     centers at xcm(k), tangent to glass above.
c
      pi=4.*atan(1.)
      delta=d/r
      k=1
      dk=0.5*d+d*float(k-1)
      dic=sqrt((h-r)**2+(dk-x)**2)
      thic=atan((h-r)/(dk-x))
      return
      end
c
      function cylq(iq,ang1,ang2,yc,zc,h1,r,x1,x2)
c     Sets up Trapezoidal Rule quadrature of analytical view
c     factor for a strip, in order to build up a cylindrical
c     surface with axis parallel to +x, center at (yc,zc).
c     Cylinder extends from x1 to x2.
c     Sample point dA1 is on z-axis at h1 above the origin,
```

```
c     with normal in -z direction.  It can view that portion
c     of the cylinder between angles ang1<ang<ang2 where
c     ang=0 is the horizontal -y direction from cylinder
c     center.  Quadrature evaluates strip function at 'iq'
c     points, including endpoints at ang1,ang2.
c*****************************************************************
c**      Issued by Sandia National Laboratories, operated for the  **
c**      United States Department of Energy by Sandia Corporation. **
c**                         NOTICE                                 **
c**      This program was prepared in the course of work sponsored **
c**      by an agency of the U.S.Government.  Neither the United   **
c**      States Government nor any agency thereof, nor any of their**
c**      employees, makes any warranty, express or implied, or     **
c**      assumes any legal liability or responsibility for the     **
c**      accuracy, completeness, or usefulness of any information, **
c**      apparatus, product, or process disclosed, or represents   **
c**      that its use would not infringe privately owned rights.   **
c**      Reference herein to any specific commercial product,      **
c**      process, or service by trade name, trademark, manufacturer,*
c**      or otherwise, does not necessarily constitute nor imply   **
c**      its endorsement, recommendation, or favoring by the United**
c**      States Government, any agency thereof or any of their     **
c**      contractors or subcontractors.  The views and opinions    **
c**      expressed herein do not necessarily state or reflect      **
c**      those of the United States Government, any agency thereof **
c**      or any of their contractors or subcontractors.            **
c**      This code is supplied for official government use only.   **
c**      No further dissemination is permitted without specific    **
c**      permission from Sandia Laboratories, Livermore, CA 94551. **
c*****************************************************************
c*****************************************************************
c
c*****************************************************************
c*****************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*****************************************************************
c*****************************************************************
      pi=4.*atan(1.)
      tol=1.e-5
      iq=max1(4,iq)
      dang=(ang2-ang1)/float(iq-1)
c     Perform trapezoidal rule quadrature from ang1 to ang2:
      ang=ang1
      d=yc-r*sin(ang)
      zt=zc*sin(ang)-yc*cos(ang)+r
      h=h1*sin(ang)-zt
      d=yc*sin(ang)-(h1-zc)*cos(ang)
      rtol=tol*sqrt(h**2+d**2)
      rot=ang-0.5*pi
c     End new variables...
      dw=r*dang
      if(abs(d).gt.rtol) then
       vwang=atan(abs(h/d))
      else
       vwang=0.5*pi
```

```
      endif
cc      if(rot.le.-vwang.or.rot.ge.(0.5*pi+vwang)) then
cc       dF=0.
cc      else
       dF=0.5*strip2(d,h,x1,x2,rot,dw)
cc      endif
c      write(11,*) ' CYLQ quadrature with ang1,rot,d,h,x1,x2,dw='
c      write(11,*)  ang1,rot,d,h,dF,vwang
c      write(11,*) ' CYLQ quadrature with ang1,ang2,dang,iq,dF1='
c      write(11,*)  ang1,ang2,dang,iq,dF
      do 10 i=2,iq-1
       ang=ang+dang
       d=yc-r*sin(ang)
       zt=zc*sin(ang)-yc*cos(ang)+r
       h=h1*sin(ang)-zt
       d=yc*sin(ang)-(h1-zc)*cos(ang)
       rot=ang-0.5*pi
       if(abs(d).gt.rtol) then
        vwang=atan(abs(h/d))
       else
        vwang=0.5*pi
       endif
cc       if(rot.gt.-vwang.and.rot.lt.(0.5*pi+vwang)) then
        dFq=strip2(d,h,x1,x2,rot,dw)
cc       endif
c       write(11,*)  ang,rot,d,h,dFq,vwang
       dF=dFq+dF
   10 continue
      ang=ang2
      d=yc-r*sin(ang)
      zt=zc*sin(ang)-yc*cos(ang)+r
      h=h1*sin(ang)-zt
      d=yc*sin(ang)-(h1-zc)*cos(ang)
      rot=ang-0.5*pi
      if(abs(d).gt.rtol) then
       vwang=atan(abs(h/d))
      else
       vwang=0.5*pi
      endif
cc      if(rot.gt.-vwang.and.rot.lt.(0.5*pi+vwang)) then
       dF=dF+0.5*strip2(d,h,x1,x2,rot,dw)
cc      endif
c      write(11,*)  ang,rot,d,h,dF,vwang
      cylq=dF
      return
      end

program sum_vf_cyl
c*****************************************************************
c*****************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*****************************************************************
c*****************************************************************
      parameter(mk=101)
      dimension Fkl(mk,mk),xfkl(mk),yfkl(mk)
```

```
c     Outputs heater assembly viewfactors of a sample point
c     point (x1,y1,0) in the (x,y) plane with area dA1.  Heater is
c     array of cylindrical elements, each of which has viewfactors
c     Fkl(k,l) on a grid of imax,jmax points with uniform spacing
c     dx,dy.  Array of elements has offset xoff,yoff from entry
c     to section, spacing Lx in ribbon motion direction and Ly in
c     the transverse direction.  Each element is over the glass
c     at height h.  Input file 'Fkl.dat' contains single-element
c     viewfactors; output file 'Fht.dat' contains summed viewfactors
c     of all the elements at the xi,yj point corresponding to Fij.
      open(unit=7,file='Fkl.dat',status='old')
      open(unit=10,file='Fht.log',status='unknown')
      open(unit=11,file='Fht.plt',status='unknown')
      open(unit=12,file='chk.plt',status='unknown')
      pi=4.*atan(1.)
      read(7,*) imax,jmax,dx,dy,cL,r,hc
c     Sample point dA1 is at (x1,y1) with origin on centerline
c     of glass ribbon, at entry to the section of the lehr
c     being heated.  The ribbon moves along +x.
c     Height hc above  ribbon is already specified in 'Fkl.dat';
c     new version of that file is required to change distance--
c     rerun 'vwelem.dat'.  This value is used in 'hvf()' to
c     extrapolate viewfactor beyond the range covered in 'Fkl.dat'.
      xoff=1.5
      yoff=0.5
      xL=2.0
      yL=1.0
c     Footprint rectangle (usually cooler boundary):
      xf=36.
      yf=3.5
      nx=ifix((xf-xoff)/xL)
      ny=ifix((yf-yoff)/yL)
cc      imax=imax-1
      write(6,*) '  SUM has imax,jmax,nx,ny=',imax,jmax,nx,ny
      do 10 j=1,jmax
        yfkl(j)=dy*float(j-1)
   10  continue
      do 20 i=1,imax
        read(7,101) (Fkl(i,j),j=1,jmax)
        xfkl(i)=dx*float(i-1)
c        write(12,*) xfkl(i),Fkl(i,1)
   20  continue
c     Echo first row of array just read:
c      i1=1
c      write(6,101) (Fkl(i1,j),j=1,jmax)
c     Plot file for element positions:
      xe=0.
      ye=0.001
      do 25 i=1,nx
       write(12,*) xe,yo
       xe=xoff+xL*float(i-1)
       write(12,*) xe,yo
       write(12,*) xe,ye
       xe=xoff+xL*float(i-1)+cL
       write(12,*) xe,ye
   25 continue
      wy=0.
```

```
      x=2.
      y=3.
      call ptdst(cL,wy,x,y,xoff,yoff,xL,yL,nx,ny,xr,yr,n1,n2)
      write(6,*) 'Position (x,y)=(',x,',',y,'); n1,n2=',n1,n2
c
c     Sample points (xs,ys) at which viewfactor is computed and
c     written into 'Fht.plt'
      dxs=2./12.
      dys=2./12.
      xs=-4.
      ys=0.5
      isam=91
      jsam=1
      do 60 kjs=1,jsam
      y=ys+dys*float(kjs-1)
c      write(6,*) '  Sampling points in column kjs=',kjs,
c     &           '; y=',y
      do 50 ks=1,isam
       x=xs+dxs*float(ks-1)
c       write(6,*) '  Sampling x=',x
c     For this position, sum the elements' contributions to
c     viewfactor:
      Fxy=0.
      do 40 j=1,ny
      ye=abs(y-yoff-yL*float(j-1)-0.5*wy)
c      write(6,*) '  Summing elements with ye=',ye
       do 30 i=1,nx
        xe=abs(x-xoff-xL*float(i-1)-0.5*cL)
        call hvf(mk,imax,jmax,hc,xfkl,yfkl,Fkl,xe,ye,Fe)
c        write(6,*) '  Summing element xe=',xe,' with Fe=',Fe
        Fxy=Fxy+Fe
   30  continue
   40 continue
      xout=x-xoff-0.5*cL
      write(11,*) x,Fxy,xout,xe,ye
   50 continue
   60 continue
      close(7)
      close(10)
      close(11)
  101 format(31e13.4)
      stop
      end
c
      subroutine ptdst(cL,wy,x,y,xoff,yoff,xL,yL,nx,ny,xr,yr,n1,n2)
c*****************************************************************
c*****************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*****************************************************************
c*****************************************************************
c     Given position (x,y) in array of elements with spacing
c     xL and yL along x,y respectively, find number of
c     columns nx and number of rows ny of elements.  Each
c     element begins at xoff, xoff+xL,..., and is cL long.
c     Transverse start at yoff, then yoff+yL,...  wy wide.
```

```
c     Return the indices (n1,n2) of the element nearest
c     point (x,y), and the distance (xr,yr) to that
c     element's centroid.
c
c     Ficticious y-width for generality:
c      wy=0.
      i1=ifix((x-xoff)/xL)
      j1=ifix((y-yoff)/yL)
      i1=amin1(nx,amax1(i1,0))
      j1=amin1(ny,amax1(j1,0))
c      write(6,*) ' x,y,i1,j1=',x,y,i1,j1
      call near(i1,nx,x,xL,cL,xoff,n1,xr)
      call near(j1,ny,y,yL,wy,yoff,n2,yr)
c      write(6,*) ' xr,yr,n1,n2=',xr,yr,n1,n2
      return
      end
c
      subroutine near(i1,nx,x,xL,cL,xoff,n1,xr)
c*********************************************************************
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*********************************************************************
c*********************************************************************
c     These index i1 is the element to the left of
c     point x. Now determine if this is the nearest
c     element.
      n1=i1
      x1=xoff+xL*float(i1)+0.5*cL
      if(i1.eq.0) then
       if((x-xoff).gt.0.) then
c       Point x is within first row:
        xr=x-x1
       else
c       Point (x,y) is to left of xoff; first row is nearest:
        xr=x1-x
       endif
      elseif(i1.eq.nx) then
c      Point (x,y) is to right of whole array; nx row nearest:
       xr=x-x1+cL
      else
c      Point (x,y) is inside array; check row i1+1 distance:
       xr1=x-x1
       xr2=x1+xL-x
       if(xr2.lt.xr1) then
        n1=n1+1
        xr=xr2
       else
        xr=xr1
       endif
      endif
      return
      end
c
      subroutine hvf(mk,imax,jmax,hc,xfkl,yfkl,Fkl,xe,ye,fe)
c*********************************************************************
```

```
c*********************************************************************
c*****   Written May 1999  for OIT/Glass/Sensors and Controls   *****
c*****                    by Lee A. Bertram                     *****
c*****    Copyright (c) May 1999  Sandia National Laboratories  *****
c*********************************************************************
c*********************************************************************
      dimension Fkl(mk,*),xfkl(*),yfkl(*)
c     Positive quadrant is covered by (xfkl,yfkl) mesh,
c     starting at (0,0).  These values are interpolated
c     to (xe,ye) inside the mesh, but extrapolated by
c     a 1/dist decay function outside the mesh.
      ie=imax
      tol=1.e-5
      do 10 i=1,imax
       if(xfkl(i)+tol.lt.xe) then
        go to 10
       else
        ie=max1(1,i-1)
        go to 15
       endif
   10 continue
      ie=imax
   15 continue
      do 20 j=1,jmax
       if(yfkl(j)+tol.lt.ye) then
        go to 20
       else
        je=max1(1,j-1)
        go to 25
       endif
   20 continue
      je=jmax
   25 continue
c     Indices (ie,je) are to L and below (xe,ye) point.
      if(ie.eq.imax) then
c      xe is outside  mesh (xfkl,yfkl)
       if(je.eq.jmax) then
c       Point (xe,ye) is to UR of mesh:
        rmesh=sqrt(hc**2+xfkl(imax)**2 + yfkl(jmax)**2)
        re=sqrt(hc**2+xe**2 + ye**2)
        fe=Fkl(imax,jmax)*((rmesh/re)**3)
        return
       elseif(je.ge.2) then
c       Point is to R of mesh but within y-range:
        fR=fkl(imax,je)+(fkl(imax,je+1)-fkl(imax,je))*
     &      (ye-yfkl(je))/(yfkl(je+1)-yfkl(je))
        rmesh=sqrt(hc**2+xfkl(imax)**2 + ye**2)
        re=sqrt(hc**2+xe**2 + ye**2)
        fe=FR*((rmesh/re)**3)
        return
       else
c       Point is to R of and below mesh:
        rmesh=sqrt(hc**2+xfkl(imax)**2 + yfkl(1)**2)
        re=sqrt(hc**2+xe**2 + ye**2)
        fe=Fkl(imax,1)*((rmesh/re)**3)
        return
       endif
```

```
       endif
       if(je.eq.jmax) then
c       ye is outside  mesh (xfkl,yfkl)
        if(ie.ge.2) then
c        Point is above mesh but within x-range:
         fR=fkl(ie,jmax)+(fkl(ie+1,jmax)-fkl(ie,jmax))*
     &       (xe-xfkl(ie))/(xfkl(ie+1)-xfkl(ie))
         rmesh=sqrt(hc**2+yfkl(jmax)**2 + xe**2)
         re=sqrt(hc**2+xe**2 + ye**2)
         fe=FR*((rmesh/re)**3)
         return
        endif
       endif
       f1=Fkl(ie,je)
       f2=Fkl(ie+1,je)
       f3=Fkl(ie+1,je+1)
       f4=Fkl(ie,je+1)
       x1=xfkl(ie)
       x2=xfkl(ie+1)
       y1=yfkl(je)
       y2=yfkl(je+1)
       xfrac=(xe-x1)/(x2-x1)
       yfrac=(ye-y1)/(y2-y1)
c       write(12,*) ' Interpolating with x1,x2,y1,y2=',
c      &           x1,x2,y1,y2,' and ie,je=',ie,je
c       write(12,*) ' Corner values f1,f2,f3,f4=',
c      &           f1,f2,f3,f4,' with xfrac,yfrac=',
c      &           xfrac,yfrac
       fe1=(f1+(f2-f1)*xfrac)
       fe2=(f4+(f3-f4)*xfrac)
       fe=(fe1+(fe2-fe1)*yfrac)
       return
       end
c
```

APPENDIX D. Inputs to Define Lehr Geometry and Operating Conditions

INPUT FILES FOR ENCLOSURE

GEOM.DAT

```
 FL1    Lehr Geometry: Enclosures by section.
 Section  L(ft)   W (ft)   H (ft; to tubes)   D (ft; to tubes)
Name/Function
   1     100.       20.       2.5       2.5   A   Conditioning
   2     150.       20.       2.5       2.5   B   Anneal
   3     150.       20.       2.5       2.5   C1  1st Cooldown
   4     150.       20.       2.5       2.5   C2  2nd Cooldown
   5      50.       20.        10.      2.5    --  Air Chill
 End Section Definitions
  6              Section A Coolers:  From CL to N wall
  Index x gap(ft) y gap(in) height(ft) L      W        Comment
   1     2.00     0.75     2.50    96.00     2.75     TC3 sensor
   2     2.00     1.50     2.50    96.00     3.00     TC4 sensor
   3     2.00     1.50     2.50    96.00     3.00     TC5 sensor
   4     2.00     0.75    -2.50    96.00     2.75     TC13 sensor*
   5     2.00     1.50    -2.50    96.00     3.00     TC14 sensor*
   6     2.00     1.50    -2.50    96.00     3.00     TC15 sensor*
  6              Section A Heaters   From CL to N wall
 Index nx  ny xinit  xspace yinit  yspace Le(in) De(in) Ht(ft)   kW
Comment
   1  48   5   1.00   2.00   0.00   0.50   4.00   0.50   2.00   1.00
4A3
   2  48   5   1.00   2.00   2.00   0.50   4.00   0.50   2.00   1.00
4A4
   3  48   5   1.00   2.00   4.00   0.50   4.00   0.50   2.00   1.00
4A5
   4  48   5   1.00   2.00   0.00   0.50   4.00   0.50  -2.00   1.00
4A13
   5  48   5   1.00   2.00   2.00   0.50   4.00   0.50  -2.00   1.00
4A14
   6  48   5   1.00   2.00   4.00   0.50   4.00   0.50  -2.00   1.00
4A15
  2              Section A Rollers
 Material Dia(in) Space (C-C, in) L (in) no.  Drive Group   Comments
   1     12.00     24.00    180.00  10  1          Stainless; inside
lehr
   2     12.00     36.00    180.00  26  2          Xyolite; inside lehr
```

OPNL.DAT

```
ribbon width (in) pull rate (ton/day)
  0.1600e+03  0.5500e+03
Conditioner center line T, edge dT (deg F)         [SETPOINT]
  0.1070e+04  0.2000e+02
Incoming center line T, edge dT (deg F)     [IR Strip Sensor]
  0.1120e+04 -0.4000e+02
```

```
Anneal rate R (deg C / sec )
  0.3000e+01
Peak stresses in C1/C2 (psi)
     min         max      Anneal stresses
  0.5000e+02  0.2000e+03
   transverse   axial     Membrane stresses      (split/tear)
 0.12500e+04 0.1500e+04
```

APPENDIX E. Thermophysical and Constitutive Properties for Glass

```
1   27  Thermophysical properties for glass ktype; nlines of data
  0.2400e+01  density, room temperature                          (gm/cc)
  0.1030e+01  specific heat, room temperature                  (J/gm-K)
  0.1030e+01  specific heat, anneal temperature                (J/gm-K)
  0.3000e-01  thermal conductivity, room temperature           (W/cm-K)
  0.4000e-01  thermal conductivity, anneal temperature         (W/cm-K)
  0.5750e+03  Anneal temperature (1070 deg F)                   (deg C)
  0.5100e+03  Glass temperature  (950 deg F)                    (deg C)
  0.5200e+03  Fictive residual temperature , Tfr                (deg C)
  0.1000e-07  Glass linear coefficient of thermal expansion,  (1/deg C)
  0.3000e-07  Liquid linear coefficient of thermal expansion, (1/deg C)
  0.8000e+00  Emissivity at glass temperature, 5 micron IR          (1)
  0.2511e+03  Fulcher To                                        (deg C)
  0.1000e+05  Fulcher b                                         (deg C)
 -0.6120e+01  Fulcher a                                            (1)
  0.1000e+08  Young's modulus, E(0), room T                      (psi)
  0.1000e+08  Young's modulus, E(0), glass T                     (psi)
  0.2500e+00  Poisson's ratio, nu, room T                          (1)
  0.4675e+05  GN structural relax activation energy Hs       (cal/mole)
  0.7560e+03  GN structural relax time,tf                         (sec)
  0.1300e+01  GN structural relax exponent, bf                     (1)
  0.9350e+05  GN viscous relax activation energy Hv          (cal/mole)
  0.1410e+03  GN viscous relax time,tvisc                         (sec)
  0.1300e+01  GN viscous relax exponent, bv                        (1)
  0.5380e+03  viscosity base temperature,TB                     (deg C)
  0.3000e+13  base viscosity at TB: visc,B                     (Pa-sec)
  0.2600e+03  radiation/conduction parity temperature           (deg C)
  0.2700e+01  upper wavelength bound for transparency (4.5?)   (micron)
Notes: Anneal is presumed to start at Narayanaswamy's TU and end at
       his TL, given by

            TU = Ta + dTa
            TL = Ts + dTa
       where
            dTa = 8.86 ln Ravg + 65 (1 - Ravg)
       in terms of average cooling rate Ravg (deg C/sec).

       Fulcher refers to viscosity-temperature relation

                visc(T) = exp { a + b / (T - To) }

       The given coefficients do not give visc=10**(12.5) Pa-sec at
       Ta, nor is visc=10**(13.5) at Ts, as in classical definitions.

       GN is Gardon-Narayanaswamy viscoelastic formulation

                                              bf
                       M (ksi) =  exp{-(ksi/tf)  }
       where

       ksi = reduced time = int(0,t) [ phi(T,Tf) dt' ] , in terms of

                    phi =   visc,B/visc(T)
```

with

$$\ln(\phi) = [\, Hg(1/TB-1/T)+Hs(1/TB-1/Tf)]/Rg$$

in which

$$Tf = T + \int_0^t [\, M(\xi-\xi')\,(dT/d\xi')\, d\xi' \,].$$

Then stress is given by

$$\sigma = \frac{E(0)}{(1-\nu)} \int_0^t [\, R(\xi') \frac{d\,(e-eth)}{d\xi'} \, d\xi' \,]$$

$$eth = a_{T,L}\,(T- Ta) - rx, \text{ where}$$

$$rx=(a_{T,L}-a_{T,g}) \int_0^t [\, M(\xi-\xi')\,(dT/d\xi')\, d\xi' \,]$$

with $a_{T,L}$ = liquid linear coefficient of thermal expansion and $a_{T,g}$ = glass linear coefficient of thermal expansion; and the stress relaxation function R for a slab is:

$$R(\xi) = \exp\{-[\, \xi/tvisc]^{bv} \} .$$

Notation

"int(0,t) [ ]" denotes "integral from 0 to t, of integrand [ ]"
GN use of VZN seems to have been supplanted by Rekhson/Mazurin terms for viscosity, M and R in Narayanaswamy 1981. This also used consistent phi = visc,B/visc relationship.

VZN is Van Zee & Noritake viscoelastic formulation

$$M(\xi) = c1 \exp\{-\xi/tfast\} + (1-c1) \exp\{-\xi/tslow\}, \text{ with}$$

ξ = reduced time = $\int_0^t [\, \phi(T,Tf)\, dt' \,]$, in terms of

$$\phi = \exp\{\, c2\,(T - Tref)\,\}$$

DISTRIBUTION:

1 Henry Technology Solutions, L.L.C.
Attn: Vincent I. Henry, President
603 Florence Road
Ann Arbor, MI 48103

5 PPG Industries, Inc.
Attn: Andrew Calabrese, Director, Glass Research & Development
Rajiv Tiwary
Ray Mayer
William Krall
William Siskos
Guys Run Road
P.O. Box 11472
Pittsburgh, PA 15238-0472

1 University of Texas
Mechanical Engineering
Attn: Professor Joseph Beaman
Austin, TX 78712

2 University of Utah
Department of Chemical Engineering
Attn: Professor Philip J. Smith
Dr. Dale Smith
Salt Lake City, UT 84112

2 U.S. Department of Energy
Office of Industrial Technologies, EE-20
Attn: Theodore Johnson
Elliott Levine
1000 Industrial Ave. SW
Washington, DC 20585

1 MS0501 M. K. Lau, 2338
1 MS0501 T. F. Owen, 2338
1 MS0834 R. A. Roach, 9114
1 MS1134 J. A. Van Den Avyle, 1835
1 MS1349 W. F. Hammetter, 1846
1 MS9056 P. M. Walsh, 8356
1 MS9102 D. A. Sheafer, 8416
1 MS9103 R. G. Hillaire, 8120
1 MS9102 K. L. Schroeder, 8120
5 MS9042 L.A. Bertram, 8728

DISTRIBUTION (Con't)

1	MS9042	D.R. Chenoweth, 8728
1	MS9042	G.N. Evans, 8728
1	MS9042	S.K. Griffiths, 8728
1	MS9042	W.G. Houf, 8728
1	MS9042	M. P. Kanouff, 8728
1	MS9042	R. S. Larson, 8728
1	MS9042	C.D. Moen, 8728
1	MS9042	A. H. Nilson, 8728
1	MS9042	P.A. Spence, 8728
1	MS9042	A. Ting, 8728
1	MS9042	W.S. Winters, 8728
1	MS9051	L. A. Rahn, 8351
1	MS9052	D.R. Hardesty, 8361
1	MS9053	J.O. Keller¸ 8362
1	MS9054	F.P. Tully, 8350
1	MS9054	R. W. Carling, 8360
1	MS9055	T. T. Bramlette, 8365
1	MS9055	J. E. Goldsmith, 8353
1	MS9056	R. E. Palmer, 8305
1	MS9056	R. J. Gallagher, 8366
1	MS9161	W. Bauer, 8302
1	MS9951	A.E. Pontau, 8358
3	MS 9018	Central Technical Files, 8940-2
1	MS 0899	Technical Library, 4916
1	MS 9021	Classification Office, 8511/ Technical Library, MS 0899, 4916
1	MS 9021	Classification Office, 8511 For DOE/OSTI