AC 2007-1593: LIVE PROBLEM SOLVING VIA COMPUTER IN THE CLASSROOM TO AVOID "DEATH BY POWERPOINT"

Michael Cutlip, University of Connecticut

Mordechai Shacham, Ben-Gurion University of the Negev

Michael Elly, Intel Corp.

Live Problem Solving via Computer in the Classroom to Avoid "Death by PowerPoint"

Introduction

Extensive use of the computer for primarily presentations in class, such as the review of PowerPoint™ notes for example, may have many undesired effects: 1) The attendance in the class session may drop as students have access to copies of the presentation in the course web site. 2) The class may become mind-numbing for students who have studied the material, and the pace may be too fast for students who did not. 3) The students may not appreciate the knowledge of the instructor as he/she only "repeats what is already written on the slides". Felder and Brent[3] have described the undesired effects of the computer use in class as "Death by PowerPoint" and cautioned against the excessive use of pre-prepared PowerPoint visuals for teaching.

Fortunately, there are uses of the computer in teaching in addition to PowerPoint presentations that can be very beneficial in the classroom. A good example is the use of a personal computer for live demonstrations in the classroom for numerical problem solving. One successful scenario is to first discuss the principles and assumptions for a particular problem, and then to develop the mathematical model of the problem on the chalkboard (or a tablet PC). The model can then be entered into the computer in front of the class (or be preprogrammed), and the problem can then be immediately solved using a mathematical software package. Graphical and tabular presentation of the results can serve as the basis to critical analysis and discussion of these results. Questions can then be asked in class regarding the model and expected results when parameters are changed or the model further refined. The resulting model can then be solved in class with the results serving as a basis for further discussion.

In this paper three examples are presented. In these examples a pre-prepared definition of a practical problem is presented and explained. The algorithm and the equations required for the solution are developed in a live demonstration by on a chalkboard or a tablet PC. The problem is then solved using the software packages Polymath© (copyrighted by Polymath Software, http://www.polymath-software.com), MATLAB™ (trademark of MathWorks, Inc., http://www.mathworks.com) or Excel™ (trademark of Microsoft Corporation, http://www.microsoft.com). During the live demonstration, the results are discussed and parametric studies are carried out.

Example 1 – Solving a Two Point Boundary Value Problem with the Newton-Raphson Method (Shooting Method[5])

Mathematical Model of the Problem and Explanation of the Need for Iterative Solution

This problem (Cutlip and Shacham[1]) involves the calculation of the concentration profiles and molar fluxes in simultaneous multi-component diffusion of gases. Gases A and B are diffusing through stagnant gas C. There is multicomponent molecular diffusion between two points where the compositions and distance apart are known. The model of the problem and the special numerical data are shown in Table 1. The problem is specified in a format that is also appropriate as Polymath input file for the solution. The Polymath input coding with the comments (marked by #) provide a compete definition and clear documentation of the model.

Application of the Stefan Maxwell equations to this particular problem yields three differential equations representing the concentrations of components A, B and C (lines 1 through 3 in Table 1). The parameters N_A and N_B (the molar fluxes of components A and B respectively) are unknown. They can be calculated using the boundary conditions at point 2 (z = 0.001m) where $C_A = 0$ and $C_B = 0.002701$. Estimates of N_A and N_B can be obtained from application of the Fick's law assuming simple binary diffusion. An estimate for N_A can be obtained, for example, from

$$N_A = -D_{AC}\frac{\left(C_A|_2 - C_A|_1\right)}{\left(z|_2 - z|_1\right)} = -1.075\times10^{-4}\frac{\left(0 - 2.229\times10^{-4}\right)}{\left(0.001 - 0\right)} = 2.396\times10^{-5}$$

An estimate for N_B can be similarly calculated, yielding N_B = -3.363× 10^{-4}. These estimates for N_A and N_B were introduced into the problem definition of Table 1. Integrating the system of differential equations using these parameter values yields the results shown in Table 2. Observe that the value of C_A at the final point (z = 0.001) is -1.692E-05 rather than 0 and the value C_B is 0.002284 rather than the specified value of 0.002701. Thus, there is a need to develop a method for adjusting the values of N_A and N_B so as to obtain the correct boundary values of the variables.

In this section the Polymath model of Table 1 is *prepared in advance* and used to explain the physical nature of the problem. The *live demonstration* includes the calculation of the initial estimates for N_A and the integration of the model equations using Polymath.

Description of the Application of the Newton-Raphson Method for the Solution of Two Point Boundary Value Problems

Let us define **x** as the vector of unknown parameters (in this particular case $\mathbf{x} = (N_A\ N_B)^T$) and **f** as a vector of functions representing the difference between the desired and calculated concentration values as point 2 , thus

$$\mathbf{f} = \begin{bmatrix} C_A|_2 - 0 \\ C_B|_2 - 2.701\times10^{-3} \end{bmatrix} \tag{1}$$

The Newton-Raphson (NR) method can be written

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \frac{\partial \mathbf{f}}{\partial \mathbf{x}}^{-1}\mathbf{f}(\mathbf{x}_k) \quad k = 0,1,2\text{K} \tag{2}$$

where k is the iteration number, $\mathbf{x}_0$ is the initial estimate and $\partial\mathbf{f}/\partial\mathbf{x}$ is the matrix of partial derivatives at $\mathbf{x} = \mathbf{x}_k$. The matrix of partial derivatives can be calculated using forward differences, thus

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i\left(\mathbf{x}_k + \boldsymbol{\delta}_j\right) - f_i\left(\mathbf{x}_k\right)}{\delta_j} \quad i = 1,\ 2; \quad j = 1,\ 2 \tag{3}$$

where $\boldsymbol{\delta}_j$ is a vector containing the value of δ_j at the j^{th} position and zeroes elsewhere. The iterations of the NR method are stopped when $\left\|\mathbf{f}\left(\mathbf{x}_k\right)\right\| \le \varepsilon_d$ and where ε_d is the desired error tolerance.

The equations in this section are shown and explained in a *live demonstration*. In order to carry out the iterative solution process a programming language (such as MATLAB) should be used.

Translating the Model into a MATLAB Function and Implementing the NR Method to Find the Values of N_A and N_B

An option within Polymath 6.1 can be used to automatically convert the model of the problem into a MATLAB function. The Polymath generated function is shown in Table 3. Note that Polymath reorders the equations and changes the syntax of the model according to the requirements of MATLAB. The "main program" that runs the Polymath generated function (shown in Table 4) is available as a template in the "Help" section of Polymath. Only the function name has to be added (see line 1) and the initial values of the variables had to be copied from the Polymath generated model (see lines 3 and 4). Execution of the complete MATLAB program as specified in Tables 4 and 3 does yield the same results obtained by Polymath (Table 2), thus verifying that the MATLAB representation of the model is correct.

The MATLAB implementation of the NR method using forward differences to calculate the matrix of partial derivatives is shown in Table 5. The modification of the programs shown in Tables 3 and 4 for iterative refinement of the N_A and N_B values involves replacement of lines 8 and 9 in Table 4 by the 24 lines of code shown in Table 5, addition of NA and NB to the function parameter list (line 1 in Table 3) and removing the specification of the values of the same variables (lines 5 and 6 in Table 3) from the function. The modified MATLAB program yields the sequence of N_A, N_B, f_1 and f_2 values shown in Table 6. Note that five NR iterations are required for convergence with error tolerance of $\varepsilon_d = 10^{-10}$. The converged solution values are N_A = 2.1149e-5 and N_B = -4.1425e-4. Using these solution values, the difference between the calculated and desired values of C_A and C_B at point 2 are $<10^{-10}$.

In this part of the solution the MATLAB implementation of the NR method is *prepared in advance*. The export of the model to MATLAB, the assembly of the complete program and the iterative solution can be carried out as *live classroom demonstrations*.

Example 2 – Solving Systems of Linear Equations Arising from Finite Difference Approximation of PDEs Using Both the Jacoby and Gauss-Seidel Methods

Description of the Jacoby and Gauss-Seidel Methods

Let us consider a linear system of equations $\mathbf{Ax} = \mathbf{b}$ where $\mathbf{x}$ is the n-vector of unknowns, $\mathbf{A}$ is an $n \times n$ matrix of coefficients and $\mathbf{b}$ is an n vector of constants. Assuming that $a_{ii} \neq 0$ $i = 1, 2, \ldots, n$, the system of equations can be written

$$x_i = \left(-\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_j + b_i \right) a_{ii}^{-1}, \quad \text{i} = 1, 2, \ldots, n \tag{4}$$

Using this formulation of the system of equations, a sequence of approximate solutions $\mathbf{x}_1$, $\mathbf{x}_2$, … can be calculated using the Jacobi method (Dahlquist *et al.*[2], p. 189)

$$x_{i,k} = \left(-\sum_{\substack{j=1 \\ j \neq i}}^{n} a_{ij} x_{j,k-1} + b_i \right) a_{ii}^{-1} \text{ , i = 1, 2, …, } n\text{; } k = 1, 2, \ldots \quad (5)$$

where k is the iteration number and x_0 is the initial estimate. The iterations are stopped when $\left\| \mathbf{x}_k - \mathbf{x}_{k-1} \right\| \leq \varepsilon_d$.

In the Gauss-Seidel method, new values of the unknowns are used as soon as they are computed. Thus

$$x_{i,k} = \left(-\sum_{j=1}^{i-1} a_{ij} x_{j,k-1} - \sum_{j=i+1}^{n} a_{ij} x_{j,k-1} + b_i \right) a_{ii}^{-1} \text{ i = 1, 2, …, } n\text{; } k = 1, 2, \ldots \quad (6)$$

The equations of this section are presented and explained in a live demonstration.

Finite Difference Representation of a Steady-State, Two-Dimensional Heat Transfer Problem

A two-dimensional problem is used to demonstrate the application of the Jacobi and Gauss-Seidel methods. Such a problem requires the solution of the Laplace equation.

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \quad (7)$$

Dividing the two dimensional space into squares with edge lengths of $\Delta x = \Delta y = 1$ enables a finite difference representation of Eq. (7)

$$T_{i,j+1} + T_{i,j-1} + T_{i+1,j} + T_{i-1,j} - 4T_{i,j} = 0 \quad (8)$$

In this problem (Geankoplis[4] , p. 340), the steady state temperature profile in a hollow rectangular chamber (see Figure 1) has to be determined. The inside dimensions of the chamber 4×2 m and the outside dimensions 8×8 m. The inside walls are held at 600 K and the outside at 300 K.

Since the chamber is symmetrical, equations in the form of Eq. 8 have to be written only for one fourth of the chamber. The equations, as arranged in the form appropriate for the use of the Jacobi method, are shown in lines 1 through 10 of the 2nd column in Table 7. Note that variable names have been replaced by temperature values at the boundaries and relationships based on symmetry allow replacement of three of the unknown temperatures ($T_{0,2} = T_{2,2}$; $T_{3,6} = T_{3,4}$ and $T_{4,6} = T_{4,4}$). Variable names ending with the letter "i" indicate initial estimates for the temperature and the initial estimates suggested by Geankoplis are introduced in lines 11 through 20. In lines 21 through 30, the elements needed to calculate the norm of the error in the first iteration are computed. Solving this set of equations with Polymath yields the results of the first iteration of the Jacobi method.

Polymath 6.1 can be used to export the problem definition to Excel with a single key-press. A slight modification of the Polymath exported problem enables carrying out additional iterations just by copying the column which contains the formulas for calculating $T_{i,j}$ and $err_{i,j}$ and pasting them into additional columns as shown in Table 8. Note that every column represents one iteration.

Solution of the problem using the Gauss-Seidel method the equations is shown in the third column of Table 7. Observe that the only difference between the Jacobi and the Gauss-Seidel methods is that the newly calculated unknown values are immediately being used in the former method. For example, the calculation of T22 (in line 2) the Jacobi method uses T12i while the Gauss-Seidel method uses T12 which was calculated in the previous line. The equations of the Gauss-Seidel method can be exported to Excel similarly to what was done for the equations of the Jacobi method. The iterations can be carried out and the results compared in one worksheet.

In this case, the equations shown in Table 7 are prepared in advance. The export to Excel, rearrangement and extension of the worksheet along with the iterative solution are carried out as a live demonstration.

Example 3 – Ill-Conditioning in Multiple Linear Regression – Detection and Harmful Effects

Regression Model for Heat of Cement Hardening

Woods *et al.*[7] investigated the integral heat of hardening of cement as a function of composition. Some of the reported results are shown in Table 9. The independent variables represent weight percent of the clinker compounds: x_1 - tricalcium aluminate ($3CaO \cdot Al_2O_3$), x_2 - tricalcium silicate ($3CaO \cdot SiO_2$), x_3 - tetracalcium alumino-ferrite ($4CaO \cdot Al_2O_3 \cdot Fe_2O_3$), and x_4 - β-dicalcium silicate ($3CaO \cdot SiO_2$). The dependent variable y is the total heat evolved (in calories per gram cement) in a 180-day period. Two multiple linear regression models, one with zero y intercept (no free parameter) and one with non-zero intercept, are to be compared.

The regression model for this problem is

$$\mathbf{y} = \hat{\beta}_0\mathbf{x}_0 + \hat{\beta}_1\mathbf{x}_1 + \hat{\beta}_2\mathbf{x}_2 + \hat{\beta}_3\mathbf{x}_3 + \hat{\beta}_4\mathbf{x}_4 \qquad (9)$$

where $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3$ *and* $\hat{\beta}_4$ are the model parameter estimates and $\mathbf{x}_0$ is a vector whose components are unity (included in the model only in case of non-zero y intercept).

The model parameter estimates can be calculated using the least squares method by solving the "normal" equation

$$\left(\mathbf{X}^T\mathbf{X}\right)\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{y} \qquad (10)$$

where $\mathbf{A} = \mathbf{X}^T\mathbf{X}$ is the normal matrix, $\mathbf{X} = [\mathbf{x}_0\ \mathbf{x}_1\ \mathbf{x}_2\ \mathbf{x}_3\ \mathbf{x}_4]$ and $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0\ \hat{\beta}_1\ \hat{\beta}_2\ \hat{\beta}_3\ \hat{\beta}_4]^T$.

The indicators (see, for example, Shacham *et al.* [6]) that can be used to check the appropriateness, accuracy and stability (conditioning) of the model are the residual plots,

variances, linear correlation coefficients, confidence intervals in addition to the eigenvalues and condition numbers of the normal matrices. The harmful effects of ill-conditioning can be demonstrated by carrying out the regression after removing a data point.

In this section the data and the problem definition are *prepared in advance* while the equations are developed in a *live demonstration*.

Analysis of the Two Regression Models

The parameters of the various models and the quality of fit indicators can be calculated with Polymath and MATLAB in a *live demonstration*. Partial results of this analysis are presented in Figure 2 and Table 10. The random distribution of the residuals in Figure 2 shows that the linear model represents the data adequately. However, the results in Table 10 show that the regression model with the non-zero intercept is very unstable (ill-conditioned). The instability is reflected in the values of the 95% confidence intervals all of which are larger than the respective parameter values and the seven order of magnitude difference between the maximal and minimal eigevalues of the normal matrix (condition number =3.64e7). In contrast, the zero-intercept model is stable with much smaller values of the confidence intervals and the condition number.

The harmful effects of the instability can be demonstrated by carrying out the regression when the last data point is removed from the set. In this case the parameter values obtained for the non-zero intercept model are: $\hat{\beta}_0 = 36.2; \hat{\beta}_1 = 1.78; \hat{\beta}_2 = 0.80; \hat{\beta}_3 = 0.328$ *and* $\hat{\beta}_4 = 0.12$. These values are completely different than the parameter values in Table 10. For the case of $\hat{\beta}_4$, even the sign is different. The results for the zero intercept model match the results of Table 10 up to two decimal digits.

Conclusions

A new approach for incorporating the computer in classroom teaching has been demonstrated. In the courses where this approach has been implemented the following educational benefits have been observed:

- The use of real-life problems for demonstration increases considerably the student motivation to study mathematical modeling and numerical methods as they understand better the need for learning these subjects.
- The development of the key algorithms and equations on the chalkboard (or the tablet PC) enables more extensive clarification of unclear points and gives the student more time to absorb and understand the new material.
- Students are very impressed with the live demonstrations that include solution of complex problems with a few key-presses using the various software packages. The conversion of a complex algorithm to a working program and the presentation of graphical and tabular results in a few minutes seems "amazing" to many of them. Consequently, their interest in the course and their appreciation of the instructor's expertise increase considerably.
- The proposed approach helped to retain or bring back the students to the classroom, and it provides many educational benefits in addition to avoiding "death by PowerPoint".

Bibliography

1. Cutlip, M. B. and M. Shacham, *Problem Solving in Chemical and Biochemical Engineering with Polymath, Excel and MATLAB*, 2nd Ed, Prentice-Hall, Upper Saddle River, New-Jersey (2007)
2. Dahlquist, G., Björck, Å., and Anderson, N., *Numerical Methods*, Prentice-Hall, Englewood Cliffs, New-Jersey (1974)
3. Felder, R. M. and Brent, R., "Death by PowerPoint", *Chemical Engineering Education*, **39** (1), 28-29 (2005)
4. Geankoplis, C. J., *Transport Processes and Separation Process Principles*, 4th Ed, Prentice-Hall, Upper Saddle River, New-Jersey (2003)
5. Press, W. H., Flannery, B. F., Teukolsky, S. A., & Vetterling, W. T., *Numerical Recipes in FORTRAN: The Art of Scientific Computing* , 2nd Ed, Cambridge University Press, Cambridge [England] (1992)
6. Shacham, M., N. Brauner and M. B. Cutlip, "Replacing the Graph Paper by Interactive Software in Modeling and Analysis of Experimental Data", *Comput. Appl. Eng. Educ*.,**4**(3), 241-251(1996)
7. Woods, H., Steinour, H. H. and H.R. Starke, "Effect of Composition of Portland Cement on Heat Evolved during Hardening", *Ind. Eng. Chem*., **24**(11), 1207 (1932)

Table 1. Polymath Input File for the Multi-Component Diffusion Problem

Line	Equation # Comment
1	d(CA)/d(z) = (xA * NB - xB * NA) / DAB + (xA * NC - xC * NA) / DAC # Concentration of A (g-mol/L)
2	d(CB)/d(z) = (xB * NA - xA * NB) / DAB + (xB * NC - xC * NB) / DBC # Concentration of B (g-mol/L)
3	d(CC)/d(z) = (xC * NA - xA * NC) / DAC + (xC * NB - xB * NC) / DBC # Concentration of C (g-mol/L)
4	NB = -0.0003363 # Molal flux of component B (kg-mol/m^2*s)
5	NA = 2.396e-5 # Molal flux of component A (kg-mol/m^2*s)
6	DAB = 1.47e-4 # Diffusivity of A through B (m^2/s)
7	NC = 0 # Molal flux of stagnant component C (kg-mol/m^2*s)
8	DAC = 1.075e-4# Diffusivity of A through C (m^2/s)
9	DBC = 1.245e-4 # Diffusivity of B through C (m^2/s)
10	CT = 0.2 / (82.057e-3 * 328) # Gas concentration g-mol/L
11	xA = CA / CT # Mole fraction of A
12	xB = CB / CT # Mole fraction of B
13	xC = CC / CT # Mole fraction of C
14	z(0) = 0 # Length coordinate at point 1
15	CB(0) = 0 # Concentration of B at point 1
16	CA(0) = 0.0002229 # Concentration of A at point 1
17	CC(0) = 0.007208 # Concentration of C at point 1
18	z(f) = 0.001 # Length coordinate at point 2 where at solution CA2 = 0 and CB2 = 0.002701

Table 2. Concentration Values Obtained Using the Estimates: $N_A = -2.396\times 10^{-5}$ and $N_B = -3.363\times 10^{-4}$

Variable	Initial value	Minimal value	Maximal value	Final value
z	0	0	0.001	0.001
C_A	0.0002229	-1.69E-05	0.0002229	-1.69E-05
C_B	0	0	0.002284	0.002284
C_C	0.007208	0.0051638	0.007208	0.0051638

Table 3. MATLAB Function for the Multi-Component Diffusion Problem.

Line	Equation % Comment
1	function dYfuncvecdz = ODEfun(z,Yfuncvec);
2	CA = Yfuncvec(1);
3	CB = Yfuncvec(2);
4	CC = Yfuncvec(3);
5	NB = -.0003363; % Molal flux of component B (kg-mol/m^2*s)
6	NA = .00002396; % Molal flux of component A (kg-mol/m^2*s)
7	DAB = .000147; % Diffusivity of A through B (m^2/s)
8	NC = 0; % Molal flux of stagnant component A (kg-mol/m^2*s)
9	DAC = .0001075; % Diffusivity of A through C (m^2/s)
10	DBC = .0001245; % Diffusivity of B through C (m^2/s)
11	CT = .2 / (.082057 * 328);% Gas concentration g-mol/L
12	xA = CA / CT; % Mole fraction of A
13	xB = CB / CT; % Mole fraction of B
14	xC = CC / CT; % Mole fraction of C
15	dCAdz = (xA * NB - (xB * NA)) / DAB + (xA * NC - (xC * NA)) / DAC; % Concentration of A (g-mol/L)
16	dCBdz = (xB * NA - (xA * NB)) / DAB + (xB * NC - (xC * NB)) / DBC; % Concentration of B (g-mol/L)
17	dCCdz = (xC * NA - (xA * NC)) / DAC + (xC * NB - (xB * NC)) / DBC; % Concentration of C (g-mol/L)
18	dYfuncvecdz = [dCAdz; dCBdz; dCCdz];

Table 4. MATLAB "Main Program" for the Multi-Component Diffusion Problem

Line	Command % Comment
1	**function MultDiffusB**
2	clear, clc, format short g, format compact
3	**tspan = [0 0.001]; % Range for the independent variable**
4	**y0 = [0.0002229; 0; 0.007208]; % Initial values for the dependent variables function**
5	disp(' Variable values at the initial point ');
6	disp([' t = ' num2str(tspan(1))]);
7	disp(' y dy/dt ');
8	disp([y0 ODEfun(tspan(1),y0)]);
9	[t,y]=ode45(@ODEfun,tspan,y0);
10	for i=1:size(y,2)
11	disp([' Solution for dependent variable y' int2str(i)]);
12	disp([' t y' int2str(i)]);
13	disp([t y(:,i)]);
14	plot(t,y(:,i));
15	title([' Plot of dependent variable y' int2str(i)]);
16	xlabel(' Independent variable (t)');
17	ylabel([' Dependent variable y' int2str(i)]);
18	pause
19	end

Table 5. MATLAB Implementation of the NR Method for the Multi-Component Diffusion Problem

Line	Command
1	NAB(:,1)=[2.396e-5; 3.363e-4];
2	disp([y0 ODEfun(tspan(1),y0,NAB(1,1),NAB(2,1))]);
3	err=1;
4	it=0;
5	while (err>1e-10) & (it<20)
6	it=it+1;
7	itno(it)=it;
8	[t,y]=ode45(@ODEfun,tspan,y0,[],NAB(1,it),NAB(2,it));
9	f(:,it)=[y(end,1); y(end,2)-2.701e-3];
10	err=sqrt(f(:,it)'*f(:,it));
11	for j=1:2
12	delj=abs(NAB(j,it))*0.01;
13	NAB(j,it)=NAB(j,it)+delj;
14	[t,yp]=ode45(@ODEfun,tspan,y0,[],NAB(1,it),NAB(2,it));
15	fp=[yp(end,1); yp(end,2)-2.701e-3];
16	for k=1:2
17	DF(k,j)=(fp(k)-f(k,it))/delj;
18	end
19	NAB(j,it)=NAB(j,it)-delj;
20	end
21	NAB(:,it+1)=NAB(:,it)-inv(DF)*f(:,it);
22	end
23	disp(' Iter. No. NA NB f1 f2 ');
24	disp([itno' NAB(1,1:it)' NAB(2,1:it)' f(1,:)' f(2,:)']);

Table 6. Shooting Method Iterations for N_A and N_B

Iter. No.	N_A	N_B	f_1	f_2
0	2.3960E-05	3.3630E-04	3.35E-05	-5.99E-03
1	2.2076E-05	-1.7614E-04	7.53E-06	-1.40E-03
2	2.1252E-05	-3.8575E-04	8.52E-07	-1.49E-04
3	2.1150E-05	-4.1375E-04	1.77E-08	-2.56E-06
4	2.1149E-05	-4.1424E-04	7.05E-11	-6.41E-09
5	2.1149E-05	-4.1425E-04	1.67E-13	-1.43E-11

Table 7. Polymath Inputs for Carrying Out the First Jacobi and Gauss-Seidel Iterations for the Steady State Heat Conduction Problem

Line	Jacobi	Gauss-Seidel
1	T12 = (300 + T22i + 600 + T22i) / 4	T12 = (300 + T22i + 600 + T22i) / 4
2	T22 = (300 + T12i + 600 + T32i) / 4	T22 = (300 + T12 + 600 + T32i) / 4
3	T32 = (300 + T22i + T33i + T42i) / 4	T32 = (300 + T22 + T33i + T42i) / 4
4	T42 = (300 + T32i + T43i + 300) / 4	T42 = (300 + T32 + T43i + 300) / 4
5	T33 = (T32i + 600 + T34i + T43) / 4	T33 = (T32 + 600 + T34i + T43i) / 4
6	T43 = (T42i + T33i + T44i + 300) / 4	T43 = (T42 + T33 + T44i + 300) / 4
7	T34 = (T33i + 600 + T35i + T44i) / 4	T34 = (T33 + 600 + T35i + T44i) / 4
8	T44 = (T43i + T34i + T45i + 300) / 4	T44 = (T43 + T34 + T45i + 300) / 4
9	T35 = (T34i + 600 + T34i + T45i) / 4	T35 = (T34 + 600 + T34 + T45i) / 4
10	T45 = (T44i + T35 + T44i + 300) / 4	T45 = (T44 + T35 + T44 + 300) / 4
11	T12i = 450	T12i = 450
12	T22i = 400	T22i = 400
13	T32i = 400	T32i = 400
14	T42i = 325	T42i = 325
15	T33i = 400	T33i = 400
16	T43i = 350	T43i = 350
17	T34i = 450	T34i = 450
18	T44i = 375	T44i = 375
19	T35i = 500	T35i = 500
20	T45i = 400	T45i = 400
21	err12=(T12-T12i)^2	err12=(T12-T12i)^2
22	err22=(T22-T22i)^2	err22=(T22-T22i)^2
23	err32=(T32-T32i)^2	err32=(T32-T32i)^2
24	err42=(T42-T42i)^2	err42=(T42-T42i)^2
25	err33=(T33-T33i)^2	err33=(T33-T33i)^2
26	err43=(T43-T43i)^2	err43=(T43-T43i)^2
27	err34=(T34-T34i)^2	err34=(T34-T34i)^2
28	err44=(T44-T44i)^2	err44=(T44-T44i)^2
29	err35=(T35-T35i)^2	err35=(T35-T35i)^2
30	err45=(T45-T45i)^2	err45=(T45-T45i)^2

Table 8. Jacobi Method Iterations for the Steady State Heat Conduction Problem

Iter. No.	0	1	2	3	4	5
T12	450	425	443.75	435.1563	440.625	438.5986
T22	400	437.5	420.3125	431.25	427.1973	430.3192
T32	400	356.25	381.25	373.6328	380.6519	378.8734
T42	325	337.5	326.5625	336.7188	333.9844	337.1704
T33	400	450	447.6563	454.6387	454.3121	456.8563
T43	350	350	365.625	362.3047	368.0298	367.1471
T34	450	468.75	475	475.5859	479.6265	479.554
T44	375	375	375	380.7617	380.2917	383.485
T35	500	475	479.6875	483.1055	483.6121	486.3842
T45	400	381.25	382.4219	383.2764	386.2839	386.7419
err12		625	351.5625	73.85254	29.90723	4.106164
err22		1406.25	295.4102	119.6289	16.42466	9.746561
err32		1914.063	625	58.02155	49.26696	3.162749
err42		156.25	119.6289	103.1494	7.476807	10.15082
err33		2500	5.493164	48.75422	0.106627	6.473016
err43		0	244.1406	11.02448	32.77674	0.779196
err34		351.5625	39.0625	0.343323	16.32586	0.005253
err44		0	0	33.1974	0.220872	10.19706
err35		625	21.97266	11.68251	0.256635	7.684763
err45		351.5625	1.373291	0.730157	9.0451	0.20981
$\|\mathbf{x}_k-\mathbf{x}_{k-1}\|$		89.04879	41.27522	21.45657	12.72036	7.24675

Table 9. Data for the Multiple Linear Regression Example (Woods *et al.*1 above [7])

No.	x_1	x_2	x_3	x_4	y
1	7	26	6	60	78.7
2	1	29	15	52	74.3
3	11	56	8	20	104.3
4	11	31	8	47	87.6
5	7	52	6	33	95.9
6	11	55	9	22	109.2
7	3	71	17	6	102.7
8	1	31	22	44	72.5
9	2	54	18	22	93.1
10	21	47	4	26	115.9
11	1	40	23	34	83.8
12	11	66	9	12	113.3
13	10	68	8	12	109.4

Table 10. Results of the Multiple Linear Regression Example

	Non-zero Intercept		Zero Intercept	
Variable	**Value**	**95% confidence**	**Value**	**95% confidence**
$\hat{\beta}_0$	60.899	161.62	-	-
$\hat{\beta}_1$	1.563	1.72	2.189	0.42
$\hat{\beta}_2$	0.527	1.67	1.154	0.11
$\hat{\beta}_3$	0.113	1.74	0.753	0.36
$\hat{\beta}_4$	-0.127	1.64	0.489	0.09
R^2	0.98		0.98	
Variance	5.99		5.82	
Max. eigenvalue	40402.00		40402.00	
Min. eigenvalue	0.0011		101.35	
Condition number	3.64E+07		398.64	

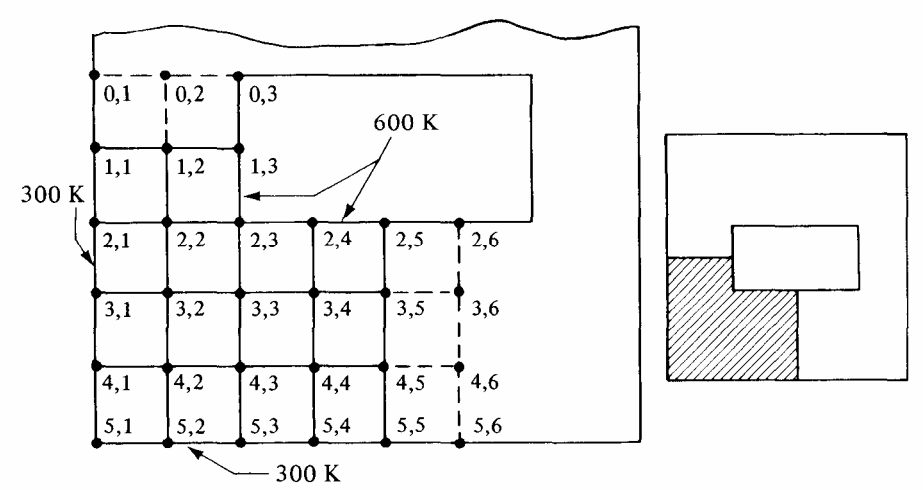

Figure 1 - Cross section of hollow chamber with square grid pattern for Example 2 (from Geankoplis[4])

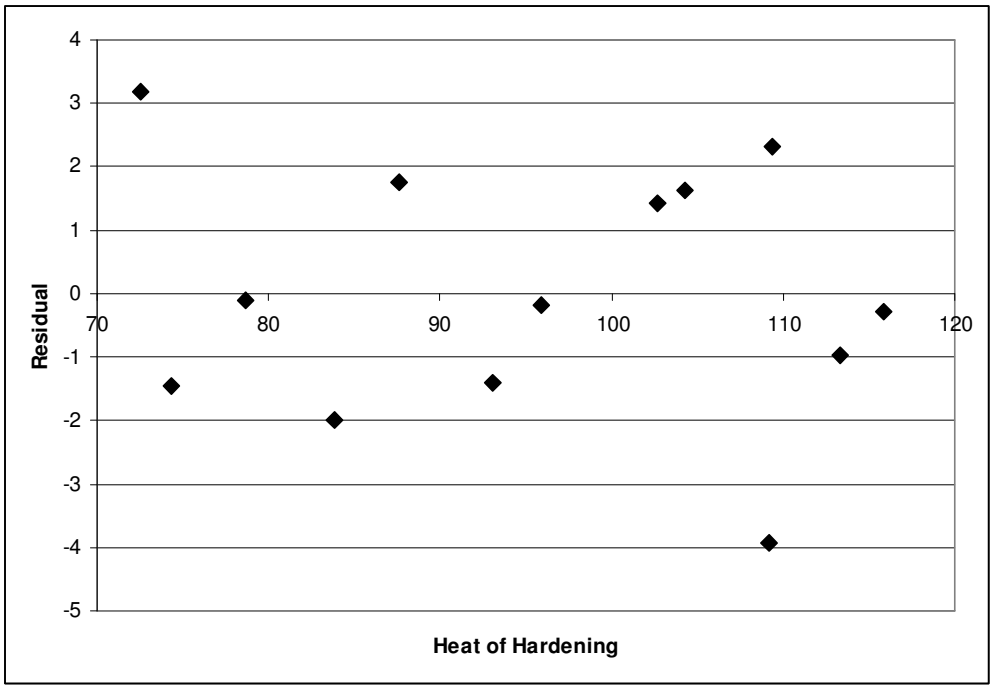

Figure 2 - Residual Plot for the "Heat of Hardening" Example 3