

Live Problem Solving Via Computer in the Classroom to Avoid “Death by PowerPoint”

MORDECHAI SHACHAM,¹ MICHAEL B. CUTLIP,² MICHAEL ELLY³

¹*Department of Chemical Engineering, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel*

²*Department of Chemical Engineering, University of Connecticut, Storrs, Connecticut 06269*

³*Intel Corp., Qiryat Gat, Israel*

Received 16 August 2007; accepted 8 October 2007

ABSTRACT: The use of live demonstrations in courses related to numerical problem solving combined with PowerPoint presentations is advocated in order to stimulate student interest and involvement and enhance learning. The proposed approach is demonstrated by solving a system of ordinary differential equations as a boundary value problem, utilizing finite difference approximations for solving partial differential equations, and investigating ill-conditioning in multiple linear regression. The educational benefits of this approach are discussed in detail. © 2009 Wiley Periodicals, Inc. *Comput Appl Eng Educ* 17: 285–294, 2009; Published online in Wiley InterScience (www.interscience.wiley.com); DOI 10.1002/cae.20214

Keywords: live demonstration; numerical methods; programming; mathematical software package

INTRODUCTION

Extensive use of the computer for presentations in class, 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 [1] 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.

Correspondence to: M. Shacham (shacham@bgu.ac.il).
© 2009 Wiley Periodicals Inc.

Live demonstrations in the classroom, on the other hand, have proven to stimulate student interest [2] and also enhance learning if the demonstration is carried out with student involvement [3]. Recent reports on successful use of live demonstrations in various disciplines include, for example, a demonstration of thermodynamics and fluid mechanics by Hill and Chen [4], “Supercriticality” a demonstration of the critical point by Licence et al. [5], and a demonstration of the formation of aerosols from biogenic hydrocarbons by Andino et al. [6].

Live demonstrations have been traditionally used only in the physical sciences (such as chemistry and physics) because of the difficulty in visualizing abstract mathematical principles or computer algorithms. However, recent developments in computer software and hardware enable extension of the applicability of live demonstrations to the more abstract subject areas.

Following this premise, we have developed a technique for including live demonstrations in courses related to numerical problem solving. Such demonstrations are combined with the PowerPoint presentations in order to stimulate student interest and involvement and enhance learning.

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 article, 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 on a chalkboard or by a tablet PC. The problem is then solved using the software packages Polymath (copyrighted by M. Shacham, M. B. Cutlip and M. Elly, <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 this 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 [7])

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

The particular problem[8] that is used to demonstrate this method involves the calculation of the concentration profiles in simultaneous multi-component diffusion of gases. Gases A and B are diffusing through stagnant gas C. This process involves molecular diffusion between two points, where the compositions 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 appropriate as Polymath input. The Polymath equation and the comments (marked by #) provide complete definition and clear documentation of the model.

Applying the Stefan Maxwell equations to this particular problem yields three differential equations representing the concentrations of components A, B, and C (lines 1–3 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 1 ($z = 0.0$ m) $C_A = 2.229 \times 10^{-4}$ and $C_B = 0$; at point 2 ($z = 0.001$ m) $C_A = 0$ and $C_B = 2.701 \times 10^{-3}$. Estimates of N_A and N_B can be obtained from application of Fick’s law assuming simple binary diffusion. An estimate for N_A can be obtained, for example, from

$$N_A = -D_{AC} \frac{(C_A|_2 - C_A|_1)}{(z_2 - z_1)}$$

$$= -1.075 \times 10^{-4} \frac{(0 - 2.229 \times 10^{-4})}{(0.001 - 0)} = 2.396 \times 10^{-5}$$

An estimate for N_B can be similarly calculated, yielding $N_B = -3.363 \times 10^{-4}$. 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 of 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

Table 1 Polymath Input for the Multi-Component Diffusion Problem

No.	Equation # comment
1	$d(C_A)/d(z) = (x_A \times N_B - x_B \times N_A)/D_{AB} + (x_A \times N_C - x_C \times N_A)/D_{AC}$ # concentration of A (g mol/L)
2	$d(C_B)/d(z) = (x_B \times N_A - x_A \times N_B)/D_{AB} + (x_B \times N_C - x_C \times N_B)/D_{BC}$ # concentration of B (g mol/L)
3	$d(C_C)/d(z) = (x_C \times N_A - x_A \times N_C)/D_{AC} + (x_C \times N_B - x_B \times N_C)/D_{BC}$ # concentration of C (g mol/L)
4	$N_B = -0.0003363$ # molal flux of component B (kg mol/m ² s)
5	$N_A = 2.396E-5$ # molal flux of component A (kg mol/m ² s)
6	$D_{AB} = 1.47E-4$ # diffusivity of A through B (m ² /s)
7	$N_C = 0$ # molal flux of stagnant component C (kg mol/m ² s)
8	$D_{AC} = 1.075E-4$ # diffusivity of A through C (m ² /s)
9	$D_{BC} = 1.245E-4$ # diffusivity of B through C (m ² /s)
10	$C_T = 0.2/(82.057E-3 \times 328)$ # gas concentration (kg mol/m ³)
11	$x_A = C_A/C_T$ # mole fraction of A
12	$x_B = C_B/C_T$ # mole fraction of B
13	$x_C = C_C/C_T$ # mole fraction of C
14	$z(0) = 0$ # length coordinate at point 1
15	$C_{B(0)} = 0$ # concentration of B at point 1
16	$C_{A(0)} = 0.0002229$ # concentration of A at point 1
17	$C_{C(0)} = 0.007208$ # concentration of C at point 1
18	$z(f) = 0.001$ # length coordinate at point 2

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

$$\mathbf{f} = \begin{bmatrix} C_A|_2 - 0 \\ C_B|_2 - 2.701 \times 10^{-3} \end{bmatrix} \quad (1)$$

The Newton–Raphson (NR) method can be written as

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

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

$$\frac{\partial f_i}{\partial x_j} = \frac{f_i(\mathbf{x}_k + \delta_j) - f_i(\mathbf{x}_k)}{\delta_j}, \quad i = 1, 2, j = 1, 2 \quad (3)$$

where δ_j is a vector containing the value of δ_j at the j th position and zeros elsewhere. The iterations of the NR method are stopped when $\|\mathbf{f}(\mathbf{x}_k)\| \leq \varepsilon_d$, 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

Polymath 6.1 can be used to 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

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

No.	Equation % comment
1	function dY _{funcvec} dz = ODEfun (z, Yfuncvec)
2	C _A = Y _{funcvec} (1);
3	C _B = Y _{funcvec} (2);
4	C _C = Y _{funcvec} (3);
5	N _B = -0.0003363; % molal flux of component B (kg mol/m ² s)
6	N _A = 0.00002396; % molal flux of component A (kg mol/m ² s)
7	D _{AB} = 0.000147; % diffusivity of A through B (m ² /s)
8	N _C = 0; % molal flux of stagnant component A (kg mol/m ² s)
9	D _{AC} = 0.0001075; % diffusivity of A through C (m ² /s)
10	D _{BC} = 0.0001245; % diffusivity of B through C (m ² /s)
11	C _T = 0.2/(0.082057 × 328); % gas concentration (kg mol/m ³)
12	x _A = C _A /C _T ; % mole fraction of A
13	x _B = C _B /C _T ; % mole fraction of B
14	x _C = C _C /C _T ; % mole fraction of C
15	dC _A dz = (x _A × N _B - (x _B × N _A))/D _{AB} + (x _A × N _C - (x _C × N _A))/D _{AC} ; % concentration of A (g mol/L)
16	dC _B dz = (x _B × N _A - (x _A × N _B))/D _{AB} + (x _B × N _C - (x _C × N _B))/D _{BC} ; % concentration of B (g mol/L)
17	dC _C dz = (x _C × N _A - (x _A × N _C))/D _{AC} + (x _C × N _B - (x _B × N _C))/D _{BC} ; % concentration of C (g mol/L)
18	dY _{funcvec} dz = [dC _A dz; dC _B dz; dC _C dz]

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). The complete MATLAB program specified in Tables 3 and 4, when executed, should yield the same results obtained by Polymath (Table 2), 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 N_A and N_B to the function parameter list (line 1 in Table 3),

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

No.	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

No.	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	[typ]=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,:)']);

and removal of 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 were required for convergence with error tolerance of $\epsilon_d = 10^{-10}$. The converged solution values are $N_A = 2.1149E-5$ and $N_B = -4.1425E-4$. Using these parameter 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 demonstrations*.

EXAMPLE 2: SOLVING SYSTEMS OF LINEAR EQUATIONS, ARISING FROM FINITE DIFFERENCE APPROXIMATION OF PDEs, USING THE JACOBI AND GAUSS–SEIDEL METHODS

Description of the Jacobi 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 $a_{ii} \neq 0$, $i = 1, 2, \dots, n$, the system of equations can be written as

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

$$i = 1, 2, \dots, n$$

Using this formulation of the system of equations, a sequence of approximate solutions $\mathbf{x}_1, \mathbf{x}_2, \dots$ can be calculated using the Jacobi method [[9], p. 189]

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

$$i = 1, 2, \dots, n, k = 1, 2, \dots$$

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

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

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

$$i = 1, 2, \dots, n, k = 1, 2, \dots$$

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

Table 6 Shooting Method Iterations for N_A and N_B

Iteration 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

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:

$$\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} = 0 \tag{7}$$

Dividing the two-dimensional space into squares with edge lengths of $\Delta x = \Delta y = 1$ enables finite difference representation of Equation 7

$$T_{i,j+1} + T_{i,j-1} + T_{i+1,j} + T_{i-1,j} - 4T_{i,j} = 0 \tag{8}$$

In the particular problem solved[10], the steady-state temperature profile in a hollow rectangular chamber (see Fig. 1) has to be found. The inside dimension of the chamber is 4 m × 2 m and the outside dimension is 8 m × 8 m. The inside walls are held at 600 K and the outside at 300 K.

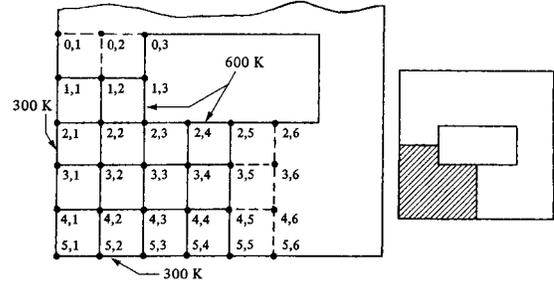


Figure 1 Cross section of hollow chamber with square grid pattern for Example 2 [10].

Since the chamber is symmetrical, equations in the form of Equation 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 1–10 of the second column in Table 7. Note that variable names have been replaced by temperature values at the boundaries and relationships based on symmetry were used to remove three of the unknown temperatures ($T_{0,2} = T_{2,2}$; $T_{3,6} = T_{3,4}$;

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

No.	Jacobi	Gauss–Seidel
1	$T_{12} = (300 + T_{22i} + 600 + T_{22i})/4$	$T_{12} = (300 + T_{22i} + 600 + T_{22i})/4$
2	$T_{22} = (300 + T_{12i} + 600 + T_{32i})/4$	$T_{22} = (300 + T_{12} + 600 + T_{32i})/4$
3	$T_{32} = (300 + T_{22i} + T_{33i} + T_{42i})/4$	$T_{32} = (300 + T_{22} + T_{33i} + T_{42i})/4$
4	$T_{42} = (300 + T_{32i} + T_{43i} + 300)/4$	$T_{42} = (300 + T_{32} + T_{43i} + 300)/4$
5	$T_{33} = (T_{32i} + 600 + T_{34i} + T_{43})/4$	$T_{33} = (T_{32} + 600 + T_{34i} + T_{43i})/4$
6	$T_{43} = (T_{42i} + T_{33i} + T_{44i} + 300)/4$	$T_{43} = (T_{42} + T_{33} + T_{44i} + 300)/4$
7	$T_{34} = (T_{33i} + 600 + T_{35i} + T_{44i})/4$	$T_{34} = (T_{33} + 600 + T_{35i} + T_{44i})/4$
8	$T_{44} = (T_{43i} + T_{34i} + T_{45i} + 300)/4$	$T_{44} = (T_{43} + T_{34} + T_{45i} + 300)/4$
9	$T_{35} = (T_{34i} + 600 + T_{34i} + T_{45i})/4$	$T_{35} = (T_{34} + 600 + T_{34} + T_{45i})/4$
10	$T_{45} = (T_{44i} + T_{35} + T_{44i} + 300)/4$	$T_{45} = (T_{44} + T_{35} + T_{44} + 300)/4$
11	$T_{12i} = 450$	$T_{12i} = 450$
12	$T_{22i} = 400$	$T_{22i} = 400$
13	$T_{32i} = 400$	$T_{32i} = 400$
14	$T_{42i} = 325$	$T_{42i} = 325$
15	$T_{33i} = 400$	$T_{33i} = 400$
16	$T_{43i} = 350$	$T_{43i} = 350$
17	$T_{34i} = 450$	$T_{34i} = 450$
18	$T_{44i} = 375$	$T_{44i} = 375$
19	$T_{35i} = 500$	$T_{35i} = 500$
20	$T_{45i} = 400$	$T_{45i} = 400$
21	$err_{12} = (T_{12} - T_{12i})^2$	$err_{12} = (T_{12} - T_{12i})^2$
22	$err_{22} = (T_{22} - T_{22i})^2$	$err_{22} = (T_{22} - T_{22i})^2$
23	$err_{32} = (T_{32} - T_{32i})^2$	$err_{32} = (T_{32} - T_{32i})^2$
24	$err_{42} = (T_{42} - T_{42i})^2$	$err_{42} = (T_{42} - T_{42i})^2$
25	$err_{33} = (T_{33} - T_{33i})^2$	$err_{33} = (T_{33} - T_{33i})^2$
26	$err_{43} = (T_{43} - T_{43i})^2$	$err_{43} = (T_{43} - T_{43i})^2$
27	$err_{34} = (T_{34} - T_{34i})^2$	$err_{34} = (T_{34} - T_{34i})^2$
28	$err_{44} = (T_{44} - T_{44i})^2$	$err_{44} = (T_{44} - T_{44i})^2$
29	$err_{35} = (T_{35} - T_{35i})^2$	$err_{35} = (T_{35} - T_{35i})^2$
30	$err_{45} = (T_{45} - T_{45i})^2$	$err_{45} = (T_{45} - T_{45i})^2$

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 [10] are introduced in lines 11–20. In lines 21–30, 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 export the problem definition to Excel on 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.

To solve the problem using the Gauss–Seidel method, the equations shown in the third column of Table 7 should be used. Observe that the difference between the Jacobi and the Gauss–Seidel method is that the newly calculated unknown values are immediately being used in the later. For example, for 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 while the export to Excel, rearrangement and extension of the worksheet and the iterative solution are carried out as a live demonstration.

Example 3: III-CONDITIONING IN MULTIPLE LINEAR REGRESSION—DETECTION AND HARMFUL EFFECTS

Problem Definition

Woods et al. [11] investigated the integral heat of hardening of cement as a function of composition. Some of the results reported by them are shown in Table 9. The independent variables represent weight percent of the clinker compounds: x_1 , tricalcium aluminate ($3\text{CaO}\cdot\text{Al}_2\text{O}_3$); x_2 , tricalcium silicate ($3\text{CaO}\cdot\text{SiO}_2$); x_3 , tetracalcium alumino-ferrite ($4\text{CaO}\cdot\text{Al}_2\text{O}_3\cdot\text{Fe}_2\text{O}_3$); and x_4 , β -dicalcium silicate ($3\text{CaO}\cdot\text{SiO}_2$). The dependent variable y is the total heat evolved (in calories per gram cement) in a 180-day period. Two multiple linear regression models,

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

	Iteration 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		1,406.25	295.4102	119.6289	16.42466	9.746561
err32		1,914.063	625	58.02155	49.26696	3.162749
err42		156.25	119.6289	103.1494	7.476807	10.15082
err33		2,500	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
$\ x_k - x_{k-1}\ $		89.04879	41.27522	21.45657	12.72036	7.24675

Table 9 Data for the Multiple Linear Regression Example [11]

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

one with zero y intercept (no free parameter) and one with non-zero intercept have to be compared.

The regression model for this problem is

$$y = \hat{\beta}_0 x_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \hat{\beta}_3 x_3 + \hat{\beta}_4 x_4 \quad (9)$$

where $\hat{\beta}_0, \hat{\beta}_1, \hat{\beta}_2, \hat{\beta}_3,$ and $\hat{\beta}_4$ are the model parameter estimates and 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

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

where $\mathbf{A} = \mathbf{X}^T \mathbf{X}$ is the normal matrix, $\mathbf{X} = [x_0 \ x_1 \ x_2 \ x_3 \ x_4]$, and $\hat{\boldsymbol{\beta}} = [\hat{\beta}_0 \ \hat{\beta}_1 \ \hat{\beta}_2 \ \hat{\beta}_3 \ \hat{\beta}_4]^T$.

The indicators (see, e.g., [12]) 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 and 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 eigenvalues 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:

$$\begin{aligned} \hat{\beta}_0 &= 36.2; \hat{\beta}_1 = 1.78; \hat{\beta}_2 \\ &= 0.80; \hat{\beta}_3 = 0.328; \text{ and } \hat{\beta}_4 = 0.12 \end{aligned}$$

These values are completely different than the ones 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

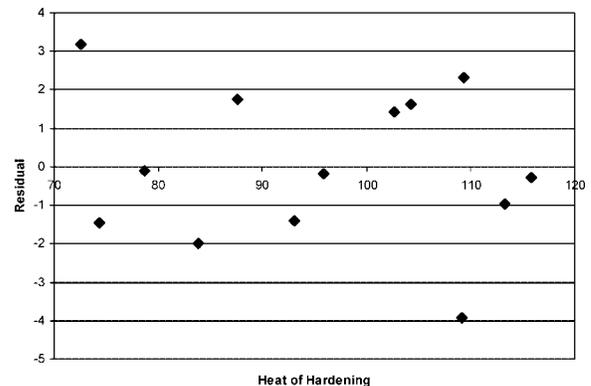


Figure 2 Residual plot for the “heat of hardening” problem.

Table 10 Results of the Multiple Linear Regression Example

Variable	Non-zero intercept		Zero intercept	
	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	40,402.00		40,402.00	
Min. eigenvalue	0.0011		101.35	
Condition number	3.64E+07		398.64	

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 seem to many of them as a “magic.” Consequently, their interest in the course and their appreciation of the instructor’s expertise increase considerably.

The proposed approach helped to bring back the students to the classroom and it provides many educational benefits in addition to avoiding “death by PowerPoint.”

REFERENCES

- [1] R. M. Felder and R. Brent, Death by PowerPoint, *Chem Eng Educ* 39 (2005), 28–29.
- [2] R. Di Stefano, Preliminary IUPP results: Student reactions to in-class demonstrations and to the presentation of coherent themes, *Am J Phys* 64 (1996), 58–68.
- [3] C. H. Crouch, A. P. Fagen, J. P. Callan, and E. Mazur, Classroom demonstrations: Learning tools or entertainment?, *Am J Phys* 72 (2004), 835–838.
- [4] R. C. Hill and W. Chen, A laboratory demonstration: Thermodynamics and fluid mechanics, *J Eng Technol* 17 (2000), 52–56.
- [5] P. Licence, D. Litchfield, M. P. Dellar, and M. Poliakoff, Supercriticality; a dramatic but safe demonstration of the critical point, *Green Chem* 6 (2004), 352–354.
- [6] J. M. Andino, T. J. Wallington, M. D. Hurley, and R. P. Wayne, A classroom demonstration of the formation of aerosols from biogenic hydrocarbons, *J Chem Educ* 77 (2000), 1584–1586.
- [7] W. H. Press, B. F. Flannery, S. A. Teukolsky, and W. T. Vetterling, *Numerical recipes in FORTRAN: The art of scientific computing*, 2nd edition, Cambridge University Press, Cambridge, England, 1992.
- [8] M. B. Cutlip and M. Shacham, *Problem solving in chemical and biochemical engineering with Polymath, Excel and MATLAB*, 2nd edition, Prentice-Hall, Upper Saddle River, New Jersey, 2008.
- [9] G. Dahlquist, Å. Björck, and N. Anderson, *Numerical methods*, Prentice-Hall, Englewood Cliffs, New Jersey, 1974.
- [10] C. J. Geankoplis, *Transport processes: Momentum, heat and mass*, Allyn and Bacon, Boston, MA, 1983.
- [11] H. Woods, H. H. Steinour, and H. R. Starke, Effect of composition of portland cement on heat evolved during hardening, *Ind Eng Chem* 24 (1932), 1207.
- [12] M. Shacham, 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 (1996), 241–251.

BIOGRAPHIES



Mordechai Shacham is the Benjamin H. Swig professor and head of the Department of Chemical Engineering at the Ben-Gurion University of the Negev in Israel. Since 2007 he serves as the chairman of the Israeli Inter-University Center for e-Learning (IUCEL). He received his BSc and DSc degrees from the Technion, Israel Institute of Technology. His research interest includes analysis, modeling and regression of data, applied numerical methods, and prediction and consistency analysis of physical properties.



Michael B. Cutlip is professor emeritus of the Chemical Engineering Department at the University of Connecticut and has served as department head and director of the University's Honors Program. He has BChE and MS degrees from Ohio State and a PhD from the University of Colorado. His current interests include the development of general software for numerical problem solving and application to chemical and biochemical engineering.



Michael Elly holds BSc and MBA degrees from the Ben-Gurion University of the Negev. As a part of his graduation work, he developed a software application for calculating Thermodynamic Vapor Liquid Equilibrium models for which he received the Dean's Prize of Excellence. He joined Intel Corporation in 1996, serving several senior IT/Automation positions in Israel and in the US.